Guía para desarrolladores, versión 2

AWS IoT Greengrass

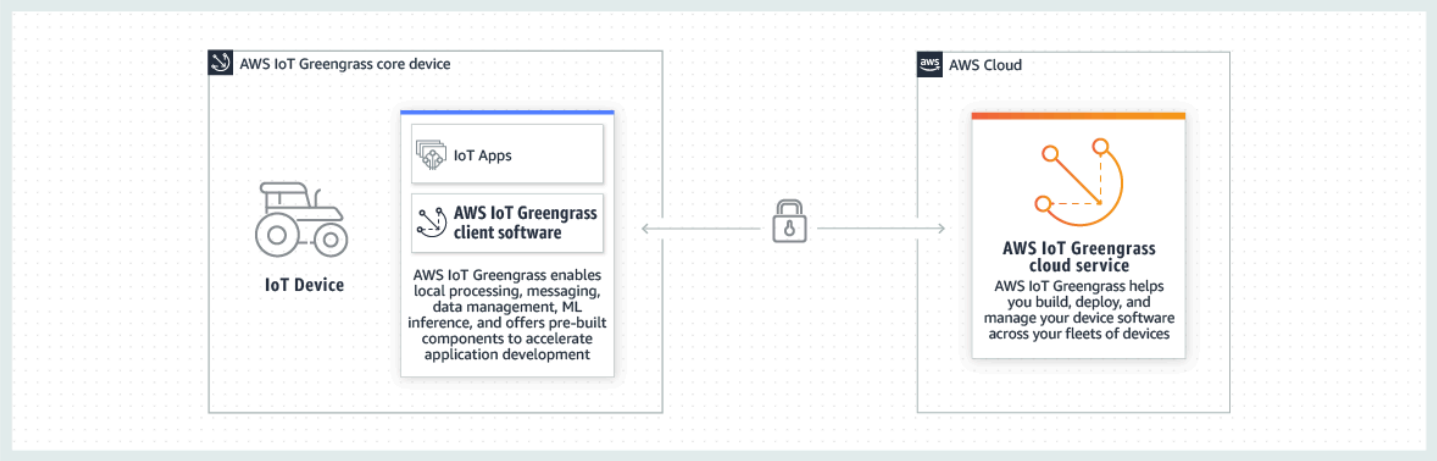
AWS IoT Greengrass: Guía para desarrolladores, versión 2

Table of Contents

¿Qué es AWS IoT Greengrass?

AWS IoT Greengrasses un servicio en la nube y de tiempo de ejecución perimetral del Internet de las cosas (IoT) de código abierto que le ayuda a crear, implementar y administrar aplicaciones de IoT en sus dispositivos. Puede utilizarlo AWS IoT Greengrass para crear software que permita a sus dispositivos actuar de forma local a partir de los datos que generan, ejecutar predicciones basadas en modelos de aprendizaje automático y filtrar y agregar los datos de los dispositivos. AWS IoT Greengrasspermite que sus dispositivos recopilen y analicen datos más cerca de donde se generan, reaccionen de forma autónoma ante los eventos locales y se comuniquen de forma segura con otros dispositivos de la red local. Los dispositivos Greengrass también pueden comunicarse de forma segura con los datos de IoT AWS IoT Core y exportarlos a. Nube de AWS Puede utilizar AWS IoT Greengrass para crear aplicaciones en la periferia mediante módulos de software prediseñados, denominados componentes, que puedan conectar sus dispositivos periféricos a servicios AWS o servicios externos. También puede utilizarlo AWS IoT Greengrass para empaquetar y ejecutar el software mediante funciones Lambda, contenedores de Docker, procesos nativos del sistema operativo o tiempos de ejecución personalizados de su elección.

El siguiente ejemplo muestra cómo interactúa un AWS IoT Greengrass dispositivo con. Nube de AWS



Nuevas características

AWS IoT Greengrass V2presenta nuevas funciones y mejoras. A continuación, se incluye más información sobre las nuevas funciones que se ofrecen en la versión 2.

- Qué hay de nuevo en AWS IoT Greengrass Version 2

Para usuarios primerizos de AWS IoT Greengrass

Si es la primera vez que lo usaAWS IoT Greengrass, le recomendamos que consulte la siguiente sección:

- Cómo funciona AWS IoT Greengrass

A continuación, sigue el tutorial de introducción para probar las funciones básicas deAWS IoT Greengrass. En este tutorial, instalará el software AWS IoT Greengrass principal en un dispositivo, desarrollará un componente de Hello World y empaquetará ese componente para su implementación.

Para los usuarios actuales de AWS IoT Greengrass

Para los usuarios actuales deAWS IoT Greengrass V1, recomendamos los siguientes temas para ayudarles a entender las diferencias entre Greengrass versión 1 y Greengrass versión 2, y aprender cómo pasar de la versión 1 a la versión 2:

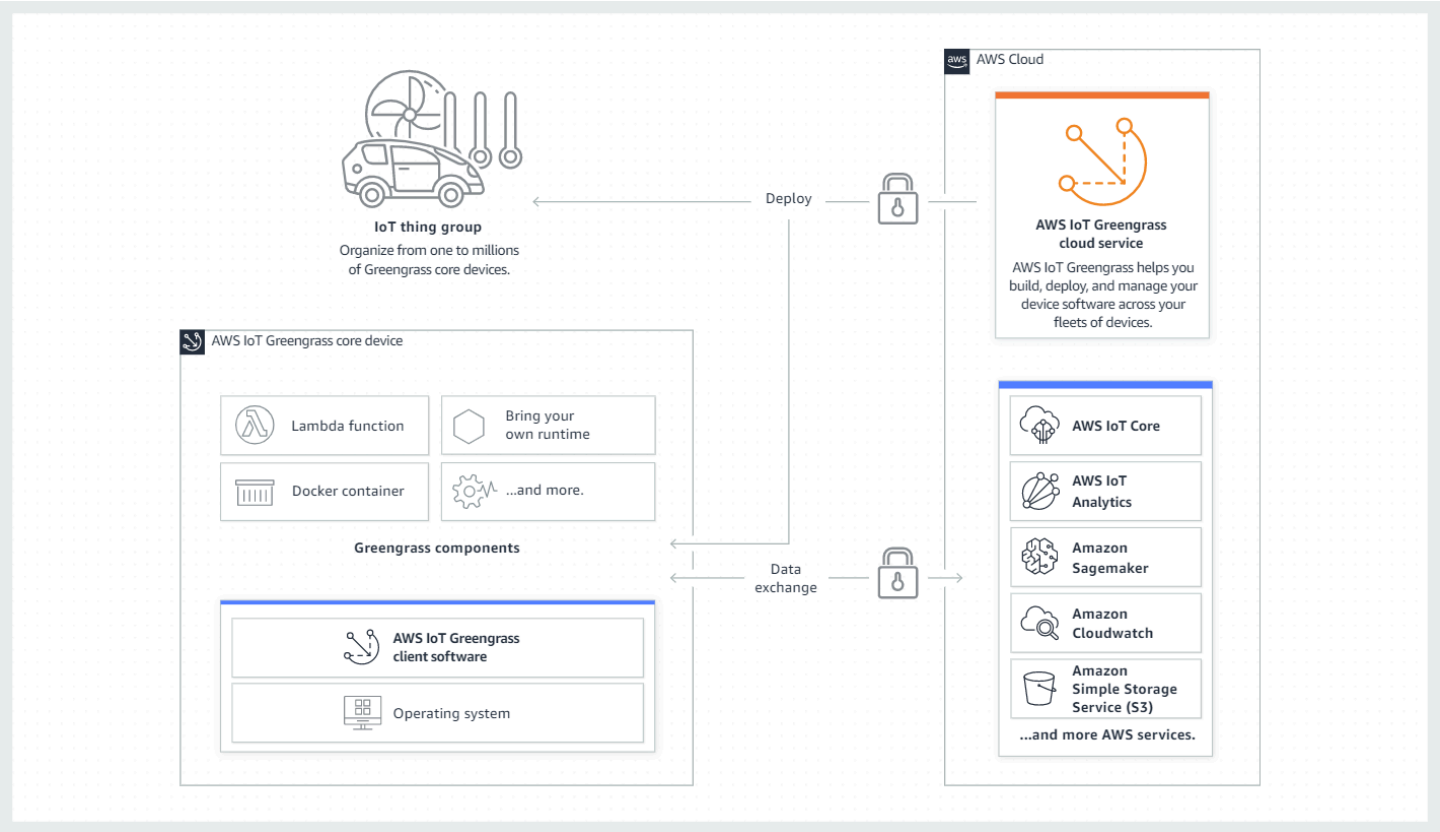
- Migrar desde AWS IoT Greengrass la versión 1

Cómo funciona AWS IoT Greengrass

El software AWS IoT Greengrass cliente, también denominado software AWS IoT Greengrass Core, se ejecuta en distribuciones basadas en Windows y Linux, como Ubuntu o Raspberry Pi OS, para dispositivos con arquitecturas ARM o x86. Con élAWS IoT Greengrass, puede programar los dispositivos para que actúen localmente a partir de los datos que generan, ejecutar predicciones basadas en modelos de aprendizaje automático y filtrar y agregar los datos de los dispositivos. AWS IoT Greengrasspermite la ejecución local de AWS Lambda funciones, contenedores de Docker, procesos nativos del sistema operativo o tiempos de ejecución personalizados de su elección.

AWS IoT Greengrassproporciona módulos de software prediseñados denominados componentes que le permiten ampliar fácilmente la funcionalidad de los dispositivos periféricos. AWS IoT Greengrasslos componentes le permiten conectarse a AWS servicios y aplicaciones de terceros en la periferia. Tras desarrollar sus aplicaciones de IoT, AWS IoT Greengrass le permite implementar, configurar y gestionar de forma remota esas aplicaciones en su flota de dispositivos sobre el terreno.

El siguiente ejemplo muestra cómo interactúa un AWS IoT Greengrass dispositivo con el servicio en la AWS IoT Greengrass nube y otros AWS servicios del Nube de AWS.



Conceptos clave de AWS IoT Greengrass

Los siguientes son conceptos esenciales para su comprensión y usoAWS IoT Greengrass:

AWS IoTcosa

Una AWS IoT cosa es una representación de un dispositivo específico o entidad lógica. La información sobre una cosa se almacena en el AWS IoT registro.

Dispositivo central Greengrass

Un dispositivo que ejecuta el software AWS IoT Greengrass Core. Un dispositivo central de Greengrass es cosa del AWS IoT. Puede añadir varios dispositivos principales a grupos de AWS IoT cosas para crear y gestionar grupos de dispositivos principales de Greengrass. Para obtener más información, consulte Configuración de los dispositivos AWS IoT Greengrass principales.

Dispositivo cliente Greengrass

Un dispositivo que se conecta y se comunica con un dispositivo principal de Greengrass a través de MQTT. Un dispositivo cliente de Greengrass existe. AWS IoT El dispositivo principal puede procesar, filtrar y agregar datos de los dispositivos cliente que se conectan a él. Puede configurar el dispositivo principal para retransmitir mensajes MQTT entre los dispositivos cliente, el servicio

AWS IoT Core en la nube y los componentes de Greengrass. Para obtener más información, consulte Interactúa con dispositivos IoT locales.

Los dispositivos cliente pueden ejecutar FreeRTOS o usar la API de descubrimiento o SDK para dispositivos con AWS IoTGreengrass para obtener información sobre los dispositivos principales a los que se pueden conectar.

Componente Greengrass

Módulo de software que se implementa y se ejecuta en un dispositivo central de Greengrass. Todo el software que se desarrolla e implementa AWS IoT Greengrass se modela como un componente. AWS IoT Greengrassproporciona componentes públicos prediseñados que proporcionan características y funcionalidades que puede utilizar en sus aplicaciones. También puede desarrollar sus propios componentes personalizados, en su dispositivo local o en la nube. Después de desarrollar un componente personalizado, puede usar el servicio AWS IoT Greengrass en la nube para implementarlo en dispositivos de uno o varios núcleos. Puede crear un componente personalizado e implementarlo en un dispositivo principal. Al hacerlo, el dispositivo principal descarga los siguientes recursos para ejecutar el componente:

- Receta: un archivo JSON o YAML que describe el módulo de software definiendo los detalles, la configuración y los parámetros de los componentes.
- Artefacto: el código fuente, los binarios o los scripts que definen el software que se ejecutará en el dispositivo. Puede crear artefactos desde cero o puede crear un componente mediante una función Lambda, un contenedor de Docker o un entorno de ejecución personalizado.
- Dependencia: la relación entre los componentes que permite aplicar actualizaciones o reinicios automáticos de los componentes dependientes. Por ejemplo, puede hacer que un componente de procesamiento seguro de mensajes dependa de un componente de cifrado. Esto garantiza que cualquier actualización del componente de cifrado actualice y reinicie automáticamente el componente de procesamiento de mensajes.

Para obtener más información, consulte AWS-componentes proporcionados y Desarrolle AWS IoT Greengrass componentes.

Implementación

El proceso para enviar componentes y aplicar la configuración de componentes deseada a un dispositivo objetivo de destino, que puede ser un único dispositivo central de Greengrass o un grupo de dispositivos principales de Greengrass. Las implementaciones aplican automáticamente cualquier configuración de componentes actualizada al destino e incluyen cualquier otro componente que se defina como dependencias. También puede clonar una implementación

existente para crear una nueva que utilice los mismos componentes pero que se despliegue en un destino diferente. Las implementaciones son continuas, lo que significa que cualquier actualización que realice en los componentes o en la configuración de los componentes de una implementación se envía automáticamente a todos los destinos de destino. Para obtener más información, consulte Implemente AWS IoT Greengrass componentes en los dispositivos.

AWS IoT GreengrassSoftware básico

El conjunto de todo el AWS IoT Greengrass software que se instala en un dispositivo principal. AWS IoT Greengrass El software principal comprende lo siguiente:

- Nucleus: este componente obligatorio proporciona la funcionalidad mínima del software AWS IoT Greengrass Core. El núcleo gestiona las implementaciones, la organización y la gestión del ciclo de vida de otros componentes. También facilita la comunicación entre AWS IoT Greengrass los componentes de forma local en un dispositivo individual. Para obtener más información, consulte Núcleo de Greengrass.
- Componentes opcionales: estos componentes configurables los proporcionan AWS IoT Greengrass los dispositivos periféricos, que habilitan funciones adicionales. En función de sus requisitos, puede elegir los componentes opcionales que desee implementar en su dispositivo, como la transmisión de datos, la inferencia de aprendizaje automático local o una interfaz de línea de comandos local. Para obtener más información, consulte AWS-componentes proporcionados.

Puede actualizar su software AWS IoT Greengrass principal implementando nuevas versiones de sus componentes en su dispositivo.

Características de AWS IoT Greengrass

AWS IoT Greengrass Version 2consta de los siguientes elementos:

- Distribuciones de software
  - El componente núcleo de Greengrass, que es la instalación mínima del software AWS IoT Greengrass Core. Este componente gestiona las implementaciones, la organización y la gestión del ciclo de vida de los componentes de Greengrass.
  - Componentes adicionales opcionales AWS proporcionados que se integran con los servicios, protocolos y software.
  - Herramientas de desarrollo de Greengrass, que puede utilizar para crear, probar, compilar, publicar e implementar componentes personalizados de Greengrass.

  - TheSDK para dispositivos con AWS IoT, que contiene la biblioteca de comunicación entre procesos (IPC) para componentes personalizados de Greengrass y la biblioteca de descubrimiento de Greengrass para dispositivos cliente.
  - El SDK de Stream Manager, que puede utilizar para gestionar los flujos de datos en los dispositivos principales.
- Servicio en la nube
  - API de AWS IoT Greengrass V2
  - Consola de AWS IoT Greengrass V2

Software de AWS IoT Greengrass Core

Puede usar el software AWS IoT Greengrass Core que se ejecuta en sus dispositivos perimetrales para hacer lo siguiente:

- Procesa los flujos de datos en el dispositivo local con exportaciones automáticas a la AWS nube. Para obtener más información, consulte Gestione los flujos de datos en los dispositivos principales de Greengrass.
- Support MQTT mensajería entre componentes AWS IoT y. Para obtener más información, consulte Publicar/suscribir mensajes MQTT AWS IoT Core.
- Interactúe con los dispositivos locales que se conectan y se comunican a través de MQTT. Para obtener más información, consulte Interactúa con dispositivos IoT locales.
- Support los mensajes de publicación y suscripción locales entre componentes. Para obtener más información, consulte Publicar/suscribir mensajes locales.
- Implemente e invoque componentes y funciones Lambda. Para obtener más información, consulte Implemente AWS IoT Greengrass componentes en los dispositivos.
- Gestione los ciclos de vida de los componentes, por ejemplo, mediante la compatibilidad con scripts de instalación y ejecución. Para obtener más información, consulte AWS IoT Greengrass referencia de recetas de componentes.
- Realice actualizaciones de software seguras over-the-air (OTA) del software AWS IoT Greengrass principal y de los componentes personalizados. Para obtener más información, consulte Actualice el software AWS IoT Greengrass principal (OTA) y Implemente AWS IoT Greengrass componentes en los dispositivos.
- Proporcione un almacenamiento seguro y cifrado de los secretos locales y un acceso controlado por parte de los componentes. Para obtener más información, consulte Gestor secreto.

- Proteja las conexiones entre los dispositivos y la AWS nube con autenticación y autorización de los dispositivos. Para obtener más información, consulte Autenticación y autorización de dispositivos en AWS IoT Greengrass.

Los dispositivos principales de Greengrass se configuran y administran a través de AWS IoT Greengrass API en las que se crean despliegues de software continuos. Para obtener más información, consulte Implemente AWS IoT Greengrass componentes en los dispositivos.

Algunas funciones solo son compatibles con determinadas plataformas. Para obtener más información, consulte Compatibilidad de funciones de Greengrass por sistema operativo.

Para obtener más información sobre las plataformas compatibles, los requisitos y las descargas, consulteConfiguración de los dispositivos AWS IoT Greengrass principales.

Al descargar este software, acepta el acuerdo de licencia del software de Greengrass Core.

Compatibilidad de funciones de Greengrass por sistema operativo

AWS IoT Greengrass admite dispositivos que ejecutan varios sistemas operativos. Algunas funciones solo son compatibles con algunos sistemas operativos. Utilice las tablas siguientes para saber qué funciones están disponibles para cada sistema operativo compatible. Para obtener más información sobre los sistemas operativos compatibles, los requisitos y cómo configurar los dispositivos principales de Greengrass, consulte. Configuración de los dispositivos AWS IoT Greengrass principales

Mensajería

Característica	Linux	Windows
Intercambie mensajes MQTT entre los componentes AWS IoT	 Sí	 Sí
Intercambie mensajes locales de publicación/suscripción entre componentes	 Sí	 Sí

Característica	Linux	Windows
Interactúa con dispositivos IoT locales a través de MQTT	 Sí	 Sí
Interactúe con los dispositivos Modbus-RTU locales mediante el componente Modbus-RTU	 Sí	 No

Seguridad

Característica	Linux	Windows
Conexiones seguras con autenticación y autorización de dispositivos	 Sí	 Sí
Implemente secretos seguros y cifrados y acceda a ellos desde AWS Secrets Manager	 Sí	 Sí
Utilice un módulo de seguridad de hardware (HSM) para almacenar de forma segura la clave privada y el certificado del dispositivo	 Sí	 No
Audite los dispositivos principales con AWS IoT Device Defender	 Sí	 Sí

Característica	Linux	Windows
Utilice AWS las credenciales para interactuar con AWS los servicios	 Sí	 Sí

Instalación

Característica	Linux	Windows
Instale AWS IoT Greengras s con aprovisionamiento automático	 Sí	 Sí
Instale AWS IoT Greengrass con aprovisionamiento manual	 Sí	 Sí
Instale AWS IoT Greengrass con el aprovisionamiento AWS IoT de flotas	 Sí	 Sí
Realice la instalación AWS IoT Greengrass con complemen tos de aprovisionamiento personalizados	 Sí	 Sí
Se ejecuta AWS IoT Greengrass en un contenedor de Docker con una imagen de Docker prediseñada	 Sí	 No

Mantenimiento y actualizaciones remotos

Característica	Linux	Windows
Realice actualizaciones de software seguras over-the-air (OTA)	 Sí	 Sí
Administre los dispositivos principales con AWS Systems Manager	 Sí	 No
Conéctese a los dispositivos principales con una tunelizac ión AWS IoT segura	 Sí	 No

Machine learning

Característica	Linux	Windows
Realice inferencias de aprendizaje automático con Amazon SageMaker Edge Manager	 Sí	 Sí
Realice inferencias de aprendizaje automático con Amazon Lookout for Vision	 Sí	 No
Realice inferencias de aprendizaje automático mediante DLR	 Sí	 Sí

Característica	Linux	Windows
Realice una inferencia de aprendizaje automático mediante TensorFlow	 Sí	 Sí

Características de los componentes

Característica	Linux	Windows
Implementación e invocación de funciones Lambda	 Sí	 No
Ejecute contenedores de Docker en los componentes	 Sí	 Sí
Procese y exporte flujos de datos de gran volumen mediante el administrador de flujos	 Sí	 Sí
Gestione los ciclos de vida de los componentes con scripts de ciclo de vida	 Sí	 Sí
Interactúe con las sombras de los dispositivos	 Sí	 Sí

Característica	Linux	Windows
Subir registros a Amazon CloudWatch Logs	 Sí	 Sí
Sube datos a CloudWatch las métricas de Amazon mediante el componente de CloudWatch métricas	 Sí	 Sí
Publique mensajes en Amazon Simple Notificat ion Service mediante el componente Amazon SNS	 Sí	 No
Publique datos en las transmisiones de entrega de Amazon Data Firehose mediante el administrador de transmisiones	 Sí	 Sí
Publique datos en las transmisiones de entrega de Amazon Data Firehose mediante el componente Firehose	 Sí	 No
Recopile las métricas de telemetría del sistema en tiempo real y actúe en función de ellas	 Sí	 Sí

Característica	Linux	Windows
Configure los límites de recursos del sistema para los procesos de los componentes	 Sí	 No
Pausa y reanuda los procesos de los componentes	 Sí	 No
Intégrelo con el AWS IoT SiteWise uso de los AWS IoT SiteWise componentes	 Sí	 Sí
Publique transmisiones de vídeo en Amazon Kinesis Video Streams mediante el conector perimetral para el componente Kinesis Video Streams	 Sí	 No

Desarrollo de componentes

Característica	Linux	Windows
Desarrolle componentes localmente en los dispositivos principales	 Sí	 Sí

Característica	Linux	Windows
Interactúe con un dispositivo principal mediante la AWS IoT Greengrass CLI	 Sí	 Sí
Interactúe con un dispositivo principal mediante la consola de depuración local	 Sí	 Sí
Utilice SDK para dispositivos con AWS IoT para Python en componentes personalizados	 Sí	 Sí
Utilice el SDK para dispositivos con AWS IoT para C++ en componentes personalizados	 Sí	 Sí
Utilice el SDK para dispositivos con AWS IoT para Java en componentes personalizados	 Sí	 Sí

Certificación de dispositivos

Característica	Linux	Windows
AWS IoT Device Tester Úselo AWS IoT Greengrass V2 para validar dispositivos de IoT	 Sí	 Sí

Qué hay de nuevo en AWS IoT Greengrass Version 2

AWS IoT Greengrass Version 2 es una versión principal AWS IoT Greengrass que presenta las siguientes funciones:

- Componentes compatibles con Publisher: AWS IoT Greengrass ahora ofrece componentes compatibles con Publisher. Estos componentes son desarrollados, ofrecidos y mantenidos por proveedores externos. Para obtener más información, consulte Componentes compatibles con Publisher.
- Operar un dispositivo Greengrass en VPC: ahora está disponible el funcionamiento de un dispositivo central Greengrass en VPC. Esto le permite realizar despliegues en una VPC sin acceso público a Internet. Para obtener más información, consulte Opere un dispositivo AWS IoT Greengrass central en VPC.
- Greengrass Testing Framework (GTF) — GTF for AWS IoT Greengrass Version 2 ya está disponible. El GTF es un conjunto de componentes básicos que respaldan la automatización. end-to-end Permite a los clientes AWS IoT Greengrass Version 2 internos utilizar el mismo marco de pruebas que utiliza el equipo de servicio para calificar los cambios de software, aceptarlos automáticamente y garantizar la calidad. Para obtener más información, consulte Greengrass Testing Framework en Github.
- Certificado por PSA: las versiones 2.7.0 y posteriores de AWS IoT Greengrass Nucleus ahora cuentan con la certificación Platform Security Architecture (PSA). Para obtener más información, consulte Cuenta con la certificación PSA.AWS IoT Greengrass

AWS IoT Greengrass las notas de la versión proporcionan detalles sobre AWS IoT Greengrass las versiones: nuevas funciones, actualizaciones y mejoras, y correcciones generales. AWS IoT Greengrass tiene los siguientes tipos de versiones:

- Lanzamientos de nuevas funciones para AWS IoT Greengrass
- AWS IoT Greengrass Actualizaciones de software principales

Esta sección contiene todas las notas de la AWS IoT Greengrass V2 versión, primero las más recientes, e incluye los principales cambios en las funciones y correcciones de errores importantes. Para obtener información sobre otras correcciones menores, consulte la organización de aws-greengrass en. GitHub

Notas de la versión

- Lanzamiento: actualización del software AWS IoT Greengrass Core v2.12.6 el 24 de mayo de 2024
- Lanzamiento: actualización del software AWS IoT Greengrass Core v2.12.5 el 25 de abril de 2024
- Lanzamiento: actualización del software AWS IoT Greengrass Core v2.12.4 el 2 de abril de 2024
- Lanzamiento: actualización del software AWS IoT Greengrass Core v2.12.3 el 27 de marzo de 2024
- Lanzamiento: actualización del software AWS IoT Greengrass Core v2.12.2 el 15 de febrero de 2024
- Lanzamiento: actualización del software AWS IoT Greengrass Core v2.12.1 el 8 de diciembre de 2023
- Lanzamiento: actualización del software AWS IoT Greengrass Core v2.12.0 el 7 de noviembre de 2023
- Lanzamiento: actualización del software AWS IoT Greengrass Core v2.11.3 el 18 de octubre de 2023
- Lanzamiento: actualización del software AWS IoT Greengrass Core v2.11.2 el 9 de agosto de 2023
- Lanzamiento: actualización del software AWS IoT Greengrass Core v2.11.1 el 21 de julio de 2023
- Lanzamiento: actualización del software AWS IoT Greengrass Core v2.11.0 el 28 de junio de 2023
- Lanzamiento: actualización del software AWS IoT Greengrass Core v2.10.3 el 21 de junio de 2023
- Lanzamiento: actualización del software AWS IoT Greengrass Core v2.10.2 el 5 de junio de 2023
- Lanzamiento: actualización del software AWS IoT Greengrass Core v2.10.1 el 11 de mayo de 2023
- Lanzamiento: actualización del software AWS IoT Greengrass Core v2.10.0 el 9 de mayo de 2023
- Lanzamiento: actualización del software AWS IoT Greengrass Core v2.9.6 el 20 de abril de 2023
- Lanzamiento: actualización del software AWS IoT Greengrass Core v2.9.5 el 30 de marzo de 2023
- Lanzamiento: actualización del software AWS IoT Greengrass Core v2.9.4 el 24 de febrero de 2023
- Lanzamiento: actualización del software AWS IoT Greengrass Core v2.9.3 el 1 de febrero de 2023
- Lanzamiento: actualización del software AWS IoT Greengrass Core v2.9.2 el 22 de diciembre de 2022
- Lanzamiento: actualización del software AWS IoT Greengrass Core v2.9.1 el 18 de noviembre de 2022

Lanzamiento: actualización del software AWS IoT Greengrass Core v2.12.6 el 24 de mayo de 2024

Esta versión incluye la versión 2.12.6 del componente núcleo de Greengrass y actualizaciones de los componentes proporcionados. AWS

Fecha de lanzamiento: 24 de mayo de 2024

Detalles del lanzamiento

Actualizaciones de componentes públicos

En la siguiente tabla se enumeran los componentes proporcionados AWS que incluyen funciones nuevas y actualizadas.

> **Important**
>
> Al implementar un componente, AWS IoT Greengrass instala las últimas versiones compatibles de todas las dependencias de ese componente. Por este motivo, es posible que las nuevas versiones con parches de los componentes públicos AWS proporcionados se implementen automáticamente en sus dispositivos principales si agrega nuevos dispositivos a un grupo de cosas o si actualiza la implementación destinada a esos dispositivos. Algunas actualizaciones automáticas, como las actualizaciones de Nucleus, pueden provocar que los dispositivos se reinicien de forma inesperada.
> Para evitar actualizaciones no deseadas de un componente que se ejecuta en su dispositivo, le recomendamos que incluya directamente la versión que prefiera de ese componente al crear una implementación. Para obtener más información sobre el comportamiento de las actualizaciones AWS IoT Greengrass del software principal, consulteActualice el software AWS IoT Greengrass principal (OTA).

Componente	Detalles
Núcleo de Greengrass	Está disponible la versión 2.12.6 del núcleo de Greengrass. Mejoras y correcciones de errores • Corrige un problema que provocaba un bloqueo al inicio en algunos procesadores ARMv8, incluido el Jetson Nano.
Greengrass CLI	Está disponible la versión 2.12.6 de la CLI de Greengrass. Mejoras y correcciones de errores • Versión actualizada para la versión 2.12.6 de Greengrass Nucleus.
Gestor secreto	Está disponible la versión 2.1.8 del administrador secreto.

Componente	Detalles
	Correcciones de errores y mejoras • Soluciona un problema por el que el administrador secreto no aceptaba un arn parcial.

Lanzamiento: actualización del software AWS IoT Greengrass Core v2.12.5 el 25 de abril de 2024

Esta versión incluye la versión 2.12.5 del componente núcleo de Greengrass y actualizaciones AWS de los componentes proporcionados.

Fecha de lanzamiento: 25 de abril de 2024

Detalles del lanzamiento

- Actualizaciones de componentes públicos

Actualizaciones de componentes públicos

En la siguiente tabla se enumeran los componentes proporcionados AWS que incluyen funciones nuevas y actualizadas.

Important

Al implementar un componente, AWS IoT Greengrass instala las últimas versiones compatibles de todas las dependencias de ese componente. Por este motivo, es posible que las nuevas versiones con parches de los componentes públicos AWS proporcionados se implementen automáticamente en sus dispositivos principales si agrega nuevos dispositivos a un grupo de cosas o si actualiza la implementación destinada a esos dispositivos. Algunas actualizaciones automáticas, como las actualizaciones de Nucleus, pueden provocar que los dispositivos se reinicien de forma inesperada.
Para evitar actualizaciones no deseadas de un componente que se ejecuta en su dispositivo, le recomendamos que incluya directamente la versión que prefiera de ese componente al crear una implementación. Para obtener más información sobre el comportamiento de las actualizaciones AWS IoT Greengrass del software principal, consulteActualice el software AWS IoT Greengrass principal (OTA).

Componente	Detalles
Núcleo de Greengrass	Está disponible la versión 2.12.5 del núcleo de Greengrass. Mejoras y correcciones de errores • Soluciona un problema que provocaba que, en ocasiones, la reversión de la implementación se bloqueara al revertir un componente previamente dañado con dependencias sólidas. • Soluciona un problema por el que el núcleo no publicaba actualizaciones de estado tras el aprovisionamiento de la flota. • Añade reintentos a la `GetDeploymentConfiguration` API después de recibir 404 errores.

Lanzamiento: actualización del software AWS IoT Greengrass Core v2.12.4 el 2 de abril de 2024

Esta versión incluye la versión 2.12.4 del componente núcleo de Greengrass y actualizaciones de los componentes proporcionados. AWS

Fecha de lanzamiento: 2 de abril de 2024

Detalles del lanzamiento

- Actualizaciones de componentes públicos

Actualizaciones de componentes públicos

En la siguiente tabla se enumeran los componentes proporcionados AWS que incluyen funciones nuevas y actualizadas.

> **Important**
>
> Al implementar un componente, AWS IoT Greengrass instala las últimas versiones compatibles de todas las dependencias de ese componente. Por este motivo, es posible que las nuevas versiones con parches de los componentes públicos AWS proporcionados se implementen automáticamente en sus dispositivos principales si agrega nuevos dispositivos

a un grupo de cosas o si actualiza la implementación destinada a esos dispositivos. Algunas actualizaciones automáticas, como las actualizaciones de Nucleus, pueden provocar que los dispositivos se reinicien de forma inesperada.
Para evitar actualizaciones no deseadas de un componente que se ejecuta en su dispositivo, le recomendamos que incluya directamente la versión que prefiera de ese componente al crear una implementación. Para obtener más información sobre el comportamiento de las actualizaciones AWS IoT Greengrass del software principal, consulteActualice el software AWS IoT Greengrass principal (OTA).

Componente	Detalles
Núcleo de Greengrass	Está disponible la versión 2.12.4 del núcleo de Greengrass. Mejoras y correcciones de errores • Soluciona un problema por el que el núcleo entra en un punto muerto durante el arranque en algunos dispositivos Linux.

Lanzamiento: actualización del software AWS IoT Greengrass Core v2.12.3 el 27 de marzo de 2024

Esta versión incluye la versión 2.12.3 del componente núcleo de Greengrass y actualizaciones de los componentes proporcionados. AWS

Fecha de lanzamiento: 27 de marzo de 2024

Detalles del lanzamiento

- Actualizaciones de componentes públicos

Actualizaciones de componentes públicos

En la siguiente tabla se enumeran los componentes proporcionados AWS que incluyen funciones nuevas y actualizadas.

> **Important**
>
> Al implementar un componente, AWS IoT Greengrass instala las últimas versiones compatibles de todas las dependencias de ese componente. Por este motivo, es posible que las nuevas versiones con parches de los componentes públicos AWS proporcionados se implementen automáticamente en sus dispositivos principales si agrega nuevos dispositivos a un grupo de cosas o si actualiza la implementación destinada a esos dispositivos. Algunas actualizaciones automáticas, como las actualizaciones de Nucleus, pueden provocar que los dispositivos se reinicien de forma inesperada.
> Para evitar actualizaciones no deseadas de un componente que se ejecuta en su dispositivo, le recomendamos que incluya directamente la versión que prefiera de ese componente al crear una implementación. Para obtener más información sobre el comportamiento de las actualizaciones AWS IoT Greengrass del software principal, consulteActualice el software AWS IoT Greengrass principal (OTA).

Componente	Detalles
Núcleo de Greengrass	Está disponible la versión 2.12.3 del núcleo de Greengrass. Mejoras y correcciones de errores • Soluciona un problema que provocaba que el núcleo no mostrara el estado correcto de los componentes después del relanzamiento del núcleo y durante la recuperación de los componentes. • Corrección de errores y mejoras generales.
Gestor oculto	Está disponible la versión 2.3.7 del componente administrador de sombras. Mejoras y correcciones de errores Soluciona un problema por el que el administrador oculto registraba periódicamente un `NullPointerException` error durante la sincroniz ación del administrador oculto.
Aprovisio namiento de flota	Está disponible la versión 1.2.1 del complemento de aprovisionamiento de AWS IoT flotas.

Componente	Detalles
	Mejoras y correcciones de errores Soluciona un problema por el que el complemento de aprovisionamiento de flotas estaba fuera de línea durante el inicio de Greengrass Nucleus. El complemento de aprovisionamiento de flotas ahora reintenta indefinid amente las llamadas de conexión MQTT.
Detector de IP	Está disponible la versión 2.1.9 del componente disk spooler. Mejoras y correcciones de errores Ajusta el paso de IP adquirida para enviar solo los registros a nivel del registro de depuración.
Component e de broker MQTT 3.1.1 de Moquette	Está disponible la versión 2.3.6 del componente de broker Moquette MQTT 3.1.1. Mejoras y correcciones de errores Corrección de errores y mejoras generales.
Gestor Lambda	Está disponible la versión 2.3.3 del componente Lambda Manager. Mejoras y correcciones de errores Corrección de errores y mejoras generales.
Consola de depuración local	Está disponible la versión 2.4.2 del componente de la consola de depuración local. Mejoras y correcciones de errores Corrección de errores y mejoras generales.

Lanzamiento: actualización del software AWS IoT Greengrass Core v2.12.2 el 15 de febrero de 2024

Esta versión incluye la versión 2.12.2 del componente núcleo de Greengrass y actualizaciones de los componentes proporcionados. AWS

Fecha de lanzamiento: 15 de febrero de 2024

Detalles del lanzamiento

- Actualizaciones de componentes públicos

Actualizaciones de componentes públicos

En la siguiente tabla se enumeran los componentes proporcionados AWS que incluyen funciones nuevas y actualizadas.

> **Important**
>
> Al implementar un componente, AWS IoT Greengrass instala las últimas versiones compatibles de todas las dependencias de ese componente. Por este motivo, es posible que las nuevas versiones con parches de los componentes públicos AWS proporcionados se implementen automáticamente en sus dispositivos principales si agrega nuevos dispositivos a un grupo de cosas o si actualiza la implementación destinada a esos dispositivos. Algunas actualizaciones automáticas, como las actualizaciones de Nucleus, pueden provocar que los dispositivos se reinicien de forma inesperada.
> Para evitar actualizaciones no deseadas de un componente que se ejecuta en su dispositivo, le recomendamos que incluya directamente la versión que prefiera de ese componente al crear una implementación. Para obtener más información sobre el comportamiento de las actualizaciones AWS IoT Greengrass del software principal, consulteActualice el software AWS IoT Greengrass principal (OTA).

Componente	Detalles
Núcleo de Greengrass	Está disponible la versión 2.12.2 del núcleo de Greengrass.

Componente	Detalles
	Mejoras y correcciones de errores • Soluciona un problema por el que los registros antiguos no se limpiaban correctamente. • Corrección de errores y mejoras generales.
Gestor en la sombra	Está disponible la versión 2.3.6 del componente administrador de sombras. Mejoras y correcciones de errores Soluciona un problema que provocaba que las propiedades ocultas que se Nube de AWS eliminaran mediante las actualizaciones mientras el dispositivo estaba desconectado siguieran existiendo en la sombra local tras recuperar la conectividad.
Lanzador Lambda	Está disponible la versión 2.0.13 del componente lambda launcher. Mejoras y correcciones de errores Corrección de errores y mejoras generales.
Spooler de discos	Está disponible la versión 1.0.3 del componente disk spooler. Mejoras y correcciones de errores Mejora el rendimiento al reutilizar las conexiones de bases de datos.

Lanzamiento: actualización del software AWS IoT Greengrass Core v2.12.1 el 8 de diciembre de 2023

Esta versión incluye la versión 2.12.1 del componente núcleo de Greengrass y actualizaciones de los componentes proporcionados. AWS

Fecha de lanzamiento: 8 de diciembre de 2023

Detalles del lanzamiento

- Actualizaciones de componentes públicos

Actualizaciones de componentes públicos

En la siguiente tabla se enumeran los componentes proporcionados AWS que incluyen funciones nuevas y actualizadas.

> **Important**
>
> Al implementar un componente, AWS IoT Greengrass instala las últimas versiones compatibles de todas las dependencias de ese componente. Por este motivo, es posible que las nuevas versiones con parches de los componentes públicos AWS proporcionados se implementen automáticamente en sus dispositivos principales si agrega nuevos dispositivos a un grupo de cosas o si actualiza la implementación destinada a esos dispositivos. Algunas actualizaciones automáticas, como las actualizaciones de Nucleus, pueden provocar que los dispositivos se reinicien de forma inesperada.
> Para evitar actualizaciones no deseadas de un componente que se ejecuta en su dispositivo, le recomendamos que incluya directamente la versión que prefiera de ese componente al crear una implementación. Para obtener más información sobre el comportamiento de las actualizaciones AWS IoT Greengrass del software principal, consulteActualice el software AWS IoT Greengrass principal (OTA).

Componente	Detalles
Núcleo de Greengrass	Está disponible la versión 2.12.1 del núcleo de Greengrass. Mejoras y correcciones de errores • Soluciona un problema por el que el núcleo podía duplicar las suscripci ones de MQTT por temas de despliegue, lo que provocaba más registros y publicaciones de MQTT.
Autenticación del dispositivo cliente	Está disponible la versión 2.4.5 del componente de autenticación del dispositi vo cliente. Nuevas características Añade compatibilidad con prefijos comodín para seleccionar los nombres de las cosas con el parámetro. `selectionRule`

Componente	Detalles
	Mejoras y correcciones de errores Soluciona un problema por el que, en algunos casos, los certificados no se actualizaban con nueva información de conectividad.
Spooler de discos	Está disponible la versión 1.0.2 del componente disk spooler. Mejoras y correcciones de errores Soluciona un problema por el que el campo de formato de mensaje MQTT no se conserva en algunos casos.
Puente MQTT	Está disponible la versión 2.3.1 del componente disk spooler. Mejoras y correcciones de errores Soluciona un problema por el que el cliente MQTT local entra en un bucle de desconexión.
Gestor de transmisiones	Está disponible la versión 2.1.12 del componente administrador de transmisiones. Mejoras y correcciones de errores Actualiza el orden en que se utilizan las credenciales, de modo que las credenciales de Greengrass son las preferidas para las solicitudes de AWS servicio.

Lanzamiento: actualización del software AWS IoT Greengrass Core v2.12.0 el 7 de noviembre de 2023

Esta versión incluye la versión 2.12.0 del componente núcleo de Greengrass y actualizaciones de los componentes proporcionados. AWS

Fecha de lanzamiento: 7 de noviembre de 2023

Aspectos destacados del lanzamiento

- Bootstrap en fase de reversión: AWS IoT Greengrass ahora proporciona un parámetro de configuración del núcleo de Greengrass llamado. `BootstrapOnRollback` Esta función le permite ejecutar los pasos del ciclo de vida de bootstrap como parte de una implementación de reversión.

Detalles de la versión

- Actualizaciones de componentes públicos

Actualizaciones de componentes públicos

En la siguiente tabla se enumeran los componentes proporcionados AWS que incluyen funciones nuevas y actualizadas.

> **Important**
>
> Al implementar un componente, AWS IoT Greengrass instala las últimas versiones compatibles de todas las dependencias de ese componente. Por este motivo, es posible que las nuevas versiones con parches de los componentes públicos AWS proporcionados se implementen automáticamente en sus dispositivos principales si agrega nuevos dispositivos a un grupo de cosas o si actualiza la implementación destinada a esos dispositivos. Algunas actualizaciones automáticas, como las actualizaciones de Nucleus, pueden provocar que los dispositivos se reinicien de forma inesperada.
> Para evitar actualizaciones no deseadas de un componente que se ejecuta en su dispositivo, le recomendamos que incluya directamente la versión que prefiera de ese componente al crear una implementación. Para obtener más información sobre el comportamiento de las actualizaciones AWS IoT Greengrass del software principal, consulteActualice el software AWS IoT Greengrass principal (OTA).

Componente	Detalles
Núcleo de Greengrass	Está disponible la versión 2.12.0 del núcleo de Greengrass. Nuevas características • Le permite ejecutar los pasos del ciclo de vida del arranque como parte de una implementación de reversión.

Lanzamiento: actualización del software AWS IoT Greengrass Core v2.11.3 el 18 de octubre de 2023

Esta versión incluye la versión 2.11.3 del componente núcleo de Greengrass.

Fecha de lanzamiento: 18 de octubre de 2023

Detalles de la versión

- Actualizaciones de componentes públicos

Actualizaciones de componentes públicos

En la siguiente tabla se enumeran los componentes proporcionados AWS que incluyen funciones nuevas y actualizadas.

> **Important**
>
> Al implementar un componente, AWS IoT Greengrass instala las últimas versiones compatibles de todas las dependencias de ese componente. Por este motivo, es posible que las nuevas versiones con parches de los componentes públicos AWS proporcionados se implementen automáticamente en sus dispositivos principales si agrega nuevos dispositivos a un grupo de cosas o si actualiza la implementación destinada a esos dispositivos. Algunas actualizaciones automáticas, como las actualizaciones de Nucleus, pueden provocar que los dispositivos se reinicien de forma inesperada.
> Para evitar actualizaciones no deseadas de un componente que se ejecuta en su dispositivo, le recomendamos que incluya directamente la versión que prefiera de ese componente al crear una implementación. Para obtener más información sobre el comportamiento de las actualizaciones AWS IoT Greengrass del software principal, consulteActualice el software AWS IoT Greengrass principal (OTA).

Componente	Detalles
Núcleo de Greengrass	Está disponible la versión 2.11.3 del núcleo de Greengrass.

Componente	Detalles
	Mejoras y correcciones de errores • Soluciona un problema en el núcleo que podía iniciar incorrectamente un componente cuando fallaban sus dependencias. Nuevas características • Añade un tipo de punto final s3 configurable.
Gestor Lambda	Está disponible la versión 2.3.1 del componente Lambda Manager. Mejoras y correcciones de errores • Ajusta los niveles de registro para detectar determinados errores.
Consola de depuración local	Está disponible la versión 2.4.0 del componente Lambda Manager. Nuevas características • Añade la consola de depuración de Stream Manager.
Gestor de registros	Está disponible la versión 2.3.6 del componente gestor de registros. Mejoras y correcciones de errores • Ajusta los niveles de registro para detectar determinados errores.
Gestor oculto	Está disponible la versión 2.3.4 del componente Shadow Manager. Mejoras y correcciones de errores • Añade compatibilidad con documentos de estado oculto nulos y vacíos.

Lanzamiento: actualización del software AWS IoT Greengrass Core v2.11.2 el 9 de agosto de 2023

Esta versión incluye la versión 2.11.2 del componente núcleo de Greengrass.

Fecha de lanzamiento: 9 de agosto de 2023

Detalles del lanzamiento

- Actualizaciones de componentes públicos

Actualizaciones de componentes públicos

En la siguiente tabla se enumeran los componentes proporcionados AWS que incluyen funciones nuevas y actualizadas.

> **Important**
>
> Al implementar un componente, AWS IoT Greengrass instala las últimas versiones compatibles de todas las dependencias de ese componente. Por este motivo, es posible que las nuevas versiones con parches de los componentes públicos AWS proporcionados se implementen automáticamente en sus dispositivos principales si agrega nuevos dispositivos a un grupo de cosas o si actualiza la implementación destinada a esos dispositivos. Algunas actualizaciones automáticas, como las actualizaciones de Nucleus, pueden provocar que los dispositivos se reinicien de forma inesperada.
> Para evitar actualizaciones no deseadas de un componente que se ejecuta en su dispositivo, le recomendamos que incluya directamente la versión que prefiera de ese componente al crear una implementación. Para obtener más información sobre el comportamiento de las actualizaciones AWS IoT Greengrass del software principal, consulteActualice el software AWS IoT Greengrass principal (OTA).

Componente	Detalles
Núcleo de Greengrass	Está disponible la versión 2.11.2 del núcleo de Greengrass. Mejoras y correcciones de errores • Soluciona un problema en el cliente Core MQTT 5 que podía aparecer desconectado cuando se utilizaban un gran número de suscripciones (> 50). • Añade un reintento por el error TCP del docker dial.

Lanzamiento: actualización del software AWS IoT Greengrass Core v2.11.1 el 21 de julio de 2023

Esta versión incluye la versión 2.11.1 del componente núcleo de Greengrass.

Fecha de lanzamiento: 21 de julio de 2023

Detalles del lanzamiento

- Actualizaciones de componentes públicos

Actualizaciones de componentes públicos

En la siguiente tabla se enumeran los componentes proporcionados AWS que incluyen funciones nuevas y actualizadas.

> **Important**
>
> Al implementar un componente, AWS IoT Greengrass instala las últimas versiones compatibles de todas las dependencias de ese componente. Por este motivo, es posible que las nuevas versiones con parches de los componentes públicos AWS proporcionados se implementen automáticamente en sus dispositivos principales si agrega nuevos dispositivos a un grupo de cosas o si actualiza la implementación destinada a esos dispositivos. Algunas actualizaciones automáticas, como las actualizaciones de Nucleus, pueden provocar que los dispositivos se reinicien de forma inesperada.
> Para evitar actualizaciones no deseadas de un componente que se ejecuta en su dispositivo, le recomendamos que incluya directamente la versión que prefiera de ese componente al crear una implementación. Para obtener más información sobre el comportamiento de las actualizaciones AWS IoT Greengrass del software principal, consulteActualice el software AWS IoT Greengrass principal (OTA).

Componente	Detalles
Núcleo de Greengrass	Está disponible la versión 2.11.1 del núcleo de Greengrass. Mejoras y correcciones de errores • Soluciona un problema por el que el núcleo no se iniciaba si se producía un error en una tarea de arranque y el archivo de metadatos de despliegue estaba dañado. • Soluciona un problema por el que los componentes Lambda bajo demanda no se incluyen en las actualizaciones del estado de la implementación.

Componente	Detalles
	• Añade compatibilidad con los identificadores de política de autorización duplicados.
Gestor Lambda	Está disponible la versión 2.2.11 del gestor Lambda. Mejoras y correcciones de errores • Soluciona un problema por el que la LegacySubscriptionRouter configuración no se actualiza cuando cambia la configuración de Lambda.

Lanzamiento: actualización del software AWS IoT Greengrass Core v2.11.0 el 28 de junio de 2023

Esta versión proporciona la versión 2.11.0 del componente núcleo de Greengrass.

Fecha de lanzamiento: 28 de junio de 2023

Aspectos destacados del lanzamiento

- Spooler de disco persistente: AWS IoT Greengrass ahora proporciona una implementación de spooler persistente para los mensajes enviados desde los dispositivos principales de Greengrass a. AWS IoT Core Este componente almacenará estos mensajes salientes en el disco. Para obtener más información, consulte Bobina de disco.
- Mejoras en la implementación local: ahora puede cancelar las implementaciones locales, establecer políticas de gestión de los problemas de implementación y obtener información detallada sobre el estado de la implementación.
- Mejoras en la velocidad de registro: se han mejorado las velocidades de carga de registros para el componente de administrador de registros.

Detalles de la versión

- Actualizaciones de componentes públicos

Actualizaciones de componentes públicos

En la siguiente tabla se enumeran los componentes proporcionados AWS que incluyen funciones nuevas y actualizadas.

> **Important**
>
> Al implementar un componente, AWS IoT Greengrass instala las últimas versiones compatibles de todas las dependencias de ese componente. Por este motivo, es posible que las nuevas versiones con parches de los componentes públicos AWS proporcionados se implementen automáticamente en sus dispositivos principales si agrega nuevos dispositivos a un grupo de cosas o si actualiza la implementación destinada a esos dispositivos. Algunas actualizaciones automáticas, como las actualizaciones de Nucleus, pueden provocar que los dispositivos se reinicien de forma inesperada.
> Para evitar actualizaciones no deseadas de un componente que se ejecuta en su dispositivo, le recomendamos que incluya directamente la versión que prefiera de ese componente al crear una implementación. Para obtener más información sobre el comportamiento de las actualizaciones AWS IoT Greengrass del software principal, consulteActualice el software AWS IoT Greengrass principal (OTA).

Componente	Detalles
Núcleo de Greengrass	Está disponible la versión 2.11.0 del núcleo de Greengrass. Nuevas características • Permite cancelar una implementación local. • Le permite configurar una política de gestión de errores para una implementación local. • Añade compatibilidad con un complemento de spooler de discos.
Greengrass CLI	Está disponible la versión 2.11.0 de la CLI de Greengrass. Nuevas características • Le permite cancelar una implementación local. • Le permite configurar una política de gestión de errores para una implementación local.

Componente	Detalles
	• Mejora los informes detallados del estado de la implementación.
Disk Spooler	Está disponible la versión 1.0.0 del componente disk spooler. • El componente disk spooler proporciona un almacenamiento persistente de los mensajes enviados desde los dispositivos principales de Greengrass a. AWS IoT Core
Gestor de registros	Está disponible la versión 2.3.5 del componente gestor de registros. Mejoras Mejora la velocidad de carga de los registros.

Lanzamiento: actualización del software AWS IoT Greengrass Core v2.10.3 el 21 de junio de 2023

Esta versión incluye la versión 2.10.3 del componente núcleo de Greengrass.

Fecha de lanzamiento: 21 de junio de 2023

Detalles del lanzamiento

- Actualizaciones de componentes públicos

Actualizaciones de componentes públicos

En la siguiente tabla se enumeran los componentes proporcionados AWS que incluyen funciones nuevas y actualizadas.

> **Important**
>
> Al implementar un componente, AWS IoT Greengrass instala las últimas versiones compatibles de todas las dependencias de ese componente. Por este motivo, es posible que las nuevas versiones con parches de los componentes públicos AWS proporcionados se implementen automáticamente en sus dispositivos principales si agrega nuevos dispositivos a un grupo de cosas o si actualiza la implementación destinada a esos dispositivos. Algunas

actualizaciones automáticas, como las actualizaciones de Nucleus, pueden provocar que los dispositivos se reinicien de forma inesperada.
Para evitar actualizaciones no deseadas de un componente que se ejecuta en su dispositivo, le recomendamos que incluya directamente la versión que prefiera de ese componente al crear una implementación. Para obtener más información sobre el comportamiento de las actualizaciones AWS IoT Greengrass del software principal, consulteActualice el software AWS IoT Greengrass principal (OTA).

Componente	Detalles
Núcleo de Greengrass	Está disponible la versión 2.10.3 del núcleo de Greengrass. Mejoras y correcciones de errores • Soluciona un problema por el que Greengrass no se suscribe a las notificaciones de despliegue cuando utiliza el proveedor PKCS #11.

Lanzamiento: actualización del software AWS IoT Greengrass Core v2.10.2 el 5 de junio de 2023

Esta versión incluye la versión 2.10.2 del componente núcleo de Greengrass.

Fecha de lanzamiento: 5 de junio de 2023

Detalles del lanzamiento

- Actualizaciones de componentes públicos

Actualizaciones de componentes públicos

En la siguiente tabla se enumeran los componentes proporcionados AWS que incluyen funciones nuevas y actualizadas.

Important

Al implementar un componente, AWS IoT Greengrass instala las últimas versiones compatibles de todas las dependencias de ese componente. Por este motivo, es posible que

las nuevas versiones con parches de los componentes públicos AWS proporcionados se implementen automáticamente en sus dispositivos principales si agrega nuevos dispositivos a un grupo de cosas o si actualiza la implementación destinada a esos dispositivos. Algunas actualizaciones automáticas, como las actualizaciones de Nucleus, pueden provocar que los dispositivos se reinicien de forma inesperada.
Para evitar actualizaciones no deseadas de un componente que se ejecuta en su dispositivo, le recomendamos que incluya directamente la versión que prefiera de ese componente al crear una implementación. Para obtener más información sobre el comportamiento de las actualizaciones AWS IoT Greengrass del software principal, consulteActualice el software AWS IoT Greengrass principal (OTA).

Componente	Detalles
Núcleo de Greengrass	Está disponible la versión 2.10.2 del núcleo de Greengrass. Mejoras y correcciones de errores • Permite analizar los ciclos de vida de los componentes sin distinguir entre mayúsculas y minúsculas. • Soluciona un problema por el que la variable PATH del entorno no se recreaba correctamente. • Corrige la codificación URI del proxy para los componentes, incluido el administrador de flujos para los nombres de usuario con caracteres especiales.
Autenticación del dispositivo cliente	Está disponible la versión 2.4.2 del componente de autenticación del dispositivo cliente. Nuevas características Añade una nueva opción de `startupTimeoutSeconds` configuración.
Gestor Lambda	Está disponible la versión 2.2.9 del componente Lambda Manager. Mejoras y correcciones de errores Soluciona un problema por el que el número de puerto estaba dañado debido a un reloj sesgado.

Componente	Detalles
Gestor de registros	Está disponible la versión 2.3.4 del componente gestor de registros. Mejoras y correcciones de errores • Añade soporte para configurar el `periodicUploadIntervalSec` parámetro en valores fraccionarios. El mínimo es de 1 microsegundo. • Soluciona un problema por el que el administrador de registros no respetaba los CloudWatch `putLogEvents` límites.
Broker MQTT 3.1 (Moquette)	Está disponible la versión 2.3.3 del componente broker MQTT 3.1 (Moquette). Nuevas características Añade una nueva opción de configuración. `startupTimeoutSeconds`
Puente MQTT	Está disponible la versión 2.2.6 del componente de puente MQTT. Nuevas características Añade una nueva opción de `startupTimeoutSeconds` configuración.
Gestor de transmisiones	Está disponible la versión 2.1.7 del componente administrador de transmisiones. Mejoras y correcciones de errores Soluciona un problema por el que el administrador de transmisiones no podía leer correctamente la configuración del proxy.

Lanzamiento: actualización del software AWS IoT Greengrass Core v2.10.1 el 11 de mayo de 2023

Esta versión incluye la versión 2.10.1 del componente núcleo de Greengrass.

Fecha de lanzamiento: 11 de mayo de 2023

Detalles del lanzamiento

- Actualizaciones de componentes públicos

Actualizaciones de componentes públicos

En la siguiente tabla se enumeran los componentes proporcionados AWS que incluyen funciones nuevas y actualizadas.

> **Important**
>
> Al implementar un componente, AWS IoT Greengrass instala las últimas versiones compatibles de todas las dependencias de ese componente. Por este motivo, es posible que las nuevas versiones con parches de los componentes públicos AWS proporcionados se implementen automáticamente en sus dispositivos principales si agrega nuevos dispositivos a un grupo de cosas o si actualiza la implementación destinada a esos dispositivos. Algunas actualizaciones automáticas, como las actualizaciones de Nucleus, pueden provocar que los dispositivos se reinicien de forma inesperada.
> Para evitar actualizaciones no deseadas de un componente que se ejecuta en su dispositivo, le recomendamos que incluya directamente la versión que prefiera de ese componente al crear una implementación. Para obtener más información sobre el comportamiento de las actualizaciones AWS IoT Greengrass del software principal, consulteActualice el software AWS IoT Greengrass principal (OTA).

Componente	Detalles
Núcleo de Greengrass	Está disponible la versión 2.10.1 del núcleo de Greengrass. Mejoras y correcciones de errores • Corrige un problema que podía provocar un bloqueo al arrancar algunos procesadores ARMv8, incluido el Jetson Nano. • Greengrass ya no cierra el estándar de un componente, lo que revierte el comportamiento anterior a la versión 2.10.0
Gestor de transmisiones	Ya está disponible la versión 2.1.6 del nuevo administrador de transmisiones. Mejoras y correcciones de errores Corrige un problema que podía provocar un bloqueo al arrancar algunos procesadores ARMv8, incluido el Jetson Nano.

Lanzamiento: actualización del software AWS IoT Greengrass Core v2.10.0 el 9 de mayo de 2023

Esta versión incluye la versión 2.10.0 del componente núcleo de Greengrass y actualizaciones de los componentes proporcionados. AWS

Fecha de lanzamiento: 9 de mayo de 2023

Aspectos destacados del lanzamiento

- Compatibilidad con MQTT5: AWS IoT Greengrass ahora es compatible con el envío y la recepción de mensajes AWS IoT Core mediante el uso de MQTT5. Para obtener más información, consulte Publicar mensajes MQTT. AWS IoT Core

Detalles de la versión

- Actualizaciones de componentes públicos

Actualizaciones de componentes públicos

En la siguiente tabla se enumeran los componentes proporcionados AWS que incluyen funciones nuevas y actualizadas.

> **Important**
>
> Al implementar un componente, AWS IoT Greengrass instala las últimas versiones compatibles de todas las dependencias de ese componente. Por este motivo, es posible que las nuevas versiones con parches de los componentes públicos AWS proporcionados se implementen automáticamente en sus dispositivos principales si agrega nuevos dispositivos a un grupo de cosas o si actualiza la implementación destinada a esos dispositivos. Algunas actualizaciones automáticas, como las actualizaciones de Nucleus, pueden provocar que los dispositivos se reinicien de forma inesperada.
>
> Para evitar actualizaciones no deseadas de un componente que se ejecuta en su dispositivo, le recomendamos que incluya directamente la versión que prefiera de ese componente al crear una implementación. Para obtener más información sobre el comportamiento de las actualizaciones AWS IoT Greengrass del software principal, consulteActualice el software AWS IoT Greengrass principal (OTA).

Componente	Detalles
Núcleo de Greengrass	Está disponible la versión 2.10.0 del núcleo de Greengrass. Nuevas características • Añade `interpolateComponentConfiguration` compatibilidad con la expresión regular vacía. Greengrass ahora interpola desde el objeto de configuración raíz. • Añade compatibilidad con MQTT5. • Añade un mecanismo para cargar los componentes del plugin rápidamente sin necesidad de escanearlos. • Permite a Greengrass ahorrar espacio en disco al eliminar las imágenes de Docker no utilizadas. Mejoras y correcciones de errores • Soluciona un problema por el que la reversión dejaba determinados valores de configuración de una implementación en su lugar. • Soluciona un problema por el que el núcleo de Greengrass valida una secuencia de AWS dominios en puntos finales de AWS datos y sin credenciales personalizados. • Actualiza la resolución de dependencias multigrupo para volver a resolver todas las dependencias de los grupos mediante la Nube de AWS negociación, en lugar de limitarlas a la versión activa. Esta actualización también elimina el código de error de implementación. `INSTALLED_COMPONENT_NOT_FOUND` • Actualiza el núcleo de Greengrass para omitir la descarga de imágenes de Docker cuando ya existen localmente. • Actualiza el núcleo de Greengrass para reiniciar el paso de instalación de un componente antes de que se agote el tiempo de espera. • Correcciones y mejoras menores adicionales.
Gestor de sombras	Ya está disponible la versión 2.3.2 del nuevo gestor de sombras.

Componente	Detalles
	Mejoras y correcciones de errores Soluciona un problema por el que el administrador oculto entra en el `BROKEN` estado cuando la base de datos oculta local está dañada.

Lanzamiento: actualización del software AWS IoT Greengrass Core v2.9.6 el 20 de abril de 2023

Esta versión incluye la versión 2.9.6 del componente núcleo de Greengrass.

Fecha de lanzamiento: 20 de abril de 2023

Detalles de la versión

- Actualizaciones de componentes públicos

Actualizaciones de componentes públicos

En la siguiente tabla se enumeran los componentes proporcionados AWS que incluyen funciones nuevas y actualizadas.

Important

Al implementar un componente, AWS IoT Greengrass instala las últimas versiones compatibles de todas las dependencias de ese componente. Por este motivo, es posible que las nuevas versiones con parches de los componentes públicos AWS proporcionados se implementen automáticamente en sus dispositivos principales si agrega nuevos dispositivos a un grupo de cosas o si actualiza la implementación destinada a esos dispositivos. Algunas actualizaciones automáticas, como las actualizaciones de Nucleus, pueden provocar que los dispositivos se reinicien de forma inesperada.
Para evitar actualizaciones no deseadas de un componente que se ejecuta en su dispositivo, le recomendamos que incluya directamente la versión que prefiera de ese componente al crear una implementación. Para obtener más información sobre el comportamiento de las actualizaciones AWS IoT Greengrass del software principal, consulteActualice el software AWS IoT Greengrass principal (OTA).

Componente	Detalles
Núcleo de Greengrass	Está disponible la versión 2.9.6 del núcleo de Greengrass. Mejoras y correcciones de errores • Corrige un problema por el que una implementación de Greengras s fallaba con el error LAUNCH_DIRECTORY_CORRUPTED y un posterior reinicio del dispositivo no podía iniciar Greengrass. Este error puede producirse al mover el dispositivo Greengrass entre varios grupos de cosas con implementaciones que requieren el reinicio de Greengras s.

Lanzamiento: actualización del software AWS IoT Greengrass Core v2.9.5 el 30 de marzo de 2023

Esta versión incluye la versión 2.9.5 del componente núcleo de Greengrass.

Fecha de lanzamiento: 30 de marzo de 2023

Detalles del lanzamiento

- Actualizaciones de componentes públicos

Actualizaciones de componentes públicos

En la siguiente tabla se enumeran los componentes proporcionados AWS que incluyen funciones nuevas y actualizadas.

> **Important**
>
> Al implementar un componente, AWS IoT Greengrass instala las últimas versiones compatibles de todas las dependencias de ese componente. Por este motivo, es posible que las nuevas versiones con parches de los componentes públicos AWS proporcionados se implementen automáticamente en sus dispositivos principales si agrega nuevos dispositivos a un grupo de cosas o si actualiza la implementación destinada a esos dispositivos. Algunas actualizaciones automáticas, como las actualizaciones de Nucleus, pueden provocar que los dispositivos se reinicien de forma inesperada.

Para evitar actualizaciones no deseadas de un componente que se ejecuta en su dispositivo, le recomendamos que incluya directamente la versión que prefiera de ese componente al crear una implementación. Para obtener más información sobre el comportamiento de las actualizaciones AWS IoT Greengrass del software principal, consulteActualice el software AWS IoT Greengrass principal (OTA).

Componente	Detalles
Núcleo de Greengrass	Está disponible la versión 2.9.5 del núcleo de Greengrass. Nuevas características • Añade compatibilidad con la verificación de firmas del software Greengrass Nucleus. Mejoras y correcciones de errores • Soluciona un problema por el que una implementación falla cuando la región de metadatos de la receta local no coincide con la región de lanzamiento del núcleo de Greengrass. El núcleo de Greengrass ahora renegocia con la nube cuando esto ocurre. • Soluciona un problema por el que el spooler de mensajes de MQTT se llenaba y nunca eliminaba los mensajes. • Correcciones y mejoras menores adicionales.

Lanzamiento: actualización del software AWS IoT Greengrass Core v2.9.4 el 24 de febrero de 2023

Esta versión incluye la versión 2.9.4 del componente núcleo de Greengrass.

Fecha de lanzamiento: 24 de febrero de 2023

Detalles del lanzamiento

- Actualizaciones de componentes públicos

Actualizaciones de componentes públicos

En la siguiente tabla se enumeran los componentes proporcionados AWS que incluyen funciones nuevas y actualizadas.

> **Important**
>
> Al implementar un componente, AWS IoT Greengrass instala las últimas versiones compatibles de todas las dependencias de ese componente. Por este motivo, es posible que las nuevas versiones con parches de los componentes públicos AWS proporcionados se implementen automáticamente en sus dispositivos principales si agrega nuevos dispositivos a un grupo de cosas o si actualiza la implementación destinada a esos dispositivos. Algunas actualizaciones automáticas, como las actualizaciones de Nucleus, pueden provocar que los dispositivos se reinicien de forma inesperada.
> Para evitar actualizaciones no deseadas de un componente que se ejecuta en su dispositivo, le recomendamos que incluya directamente la versión que prefiera de ese componente al crear una implementación. Para obtener más información sobre el comportamiento de las actualizaciones AWS IoT Greengrass del software principal, consulteActualice el software AWS IoT Greengrass principal (OTA).

Componente	Detalles
Núcleo de Greengrass	Está disponible la versión 2.9.4 del núcleo de Greengrass. Mejoras y correcciones de errores • Comprueba si hay un mensaje nulo antes de eliminar los mensajes de QOS 0. • Trunca los valores de detalle del estado del trabajo si superan el límite de 1024 caracteres. • Actualiza el script de arranque para Windows para que lea correctam ente la ruta raíz de Greengrass si esa ruta incluye espacios. • Actualiza la suscripción para AWS IoT Core que se eliminen los mensajes de los clientes si no se envió la respuesta de la suscripción.

Componente	Detalles
	• Garantiza que el núcleo cargue su configuración a partir de archivos de respaldo cuando el archivo de configuración principal esté dañado o falte.

Lanzamiento: actualización del software AWS IoT Greengrass Core v2.9.3 el 1 de febrero de 2023

Esta versión incluye la versión 2.9.3 del componente núcleo de Greengrass.

Fecha de lanzamiento: 1 de febrero de 2023

Detalles del lanzamiento

- Actualizaciones de componentes públicos

Actualizaciones de componentes públicos

En la siguiente tabla se enumeran los componentes proporcionados AWS que incluyen funciones nuevas y actualizadas.

Important

Al implementar un componente, AWS IoT Greengrass instala las últimas versiones compatibles de todas las dependencias de ese componente. Por este motivo, es posible que las nuevas versiones con parches de los componentes públicos AWS proporcionados se implementen automáticamente en sus dispositivos principales si agrega nuevos dispositivos a un grupo de cosas o si actualiza la implementación destinada a esos dispositivos. Algunas actualizaciones automáticas, como las actualizaciones de Nucleus, pueden provocar que los dispositivos se reinicien de forma inesperada.
Para evitar actualizaciones no deseadas de un componente que se ejecuta en su dispositivo, le recomendamos que incluya directamente la versión que prefiera de ese componente al crear una implementación. Para obtener más información sobre el comportamiento de las actualizaciones AWS IoT Greengrass del software principal, consulteActualice el software AWS IoT Greengrass principal (OTA).

Componente	Detalles
Núcleo de Greengrass	Está disponible la versión 2.9.3 del núcleo de Greengrass. Mejoras y correcciones de errores • Garantiza que los ID de los clientes de MQTT no estén duplicados. • Añade una lectura y escritura de archivos más robustas para evitar la corrupción y recuperarse de ella. • Reintenta extraer la imagen del docker en caso de errores específicos relacionados con la red. • Añade la `noProxyAddresses` opción de conexión MQTT.

Lanzamiento: actualización del software AWS IoT Greengrass Core v2.9.2 el 22 de diciembre de 2022

Esta versión incluye la versión 2.9.2 del componente núcleo de Greengrass.

Fecha de lanzamiento: 22 de diciembre de 2022

Detalles del lanzamiento

- Actualizaciones de componentes públicos

Actualizaciones de componentes públicos

En la siguiente tabla se enumeran los componentes AWS proporcionados que incluyen funciones nuevas y actualizadas.

> **Important**
>
> Al implementar un componente, AWS IoT Greengrass instala las últimas versiones compatibles de todas las dependencias de ese componente. Por este motivo, es posible que las nuevas versiones con parches de los componentes públicos AWS proporcionados se implementen automáticamente en sus dispositivos principales si agrega nuevos dispositivos a un grupo de cosas o si actualiza la implementación destinada a esos dispositivos. Algunas

actualizaciones automáticas, como las actualizaciones de Nucleus, pueden provocar que los dispositivos se reinicien de forma inesperada.
Para evitar actualizaciones no deseadas de un componente que se ejecuta en su dispositivo, le recomendamos que incluya directamente la versión que prefiera de ese componente al crear una implementación. Para obtener más información sobre el comportamiento de las actualizaciones AWS IoT Greengrass del software principal, consulteActualice el software AWS IoT Greengrass principal (OTA).

Componente	Detalles
Núcleo de Greengrass	Está disponible la versión 2.9.2 del núcleo de Greengrass. Mejoras y correcciones de errores • Soluciona un problema por el que la configuración `interpolateComponentConfiguration` no se aplicaba a una implementación en curso. • Utiliza OSHI para enumerar todos los procesos secundarios.

Lanzamiento: actualización del software AWS IoT Greengrass Core v2.9.1 el 18 de noviembre de 2022

Esta versión incluye la versión 2.9.1 del componente núcleo de Greengrass y actualizaciones AWS de los componentes proporcionados.

Fecha de lanzamiento: 18 de noviembre de 2022

Características destacadas de la versión

- Administrador de registros: el administrador de registros ahora procesa y carga directamente los archivos de registro activos en lugar de esperar a que se roten los archivos nuevos. Esta mejora reduce considerablemente los retrasos en los registros. Para obtener más información, consulte Administrador de registros

Detalles de la versión

- Actualizaciones de componentes públicos

Actualizaciones de componentes públicos

En la siguiente tabla se enumeran los componentes AWS proporcionados que incluyen funciones nuevas y actualizadas.

> **Important**
>
> Al implementar un componente, AWS IoT Greengrass instala las últimas versiones compatibles de todas las dependencias de ese componente. Por este motivo, es posible que las nuevas versiones con parches de los componentes públicos AWS proporcionados se implementen automáticamente en sus dispositivos principales si agrega nuevos dispositivos a un grupo de cosas o si actualiza la implementación destinada a esos dispositivos. Algunas actualizaciones automáticas, como las actualizaciones de Nucleus, pueden provocar que los dispositivos se reinicien de forma inesperada.
> Para evitar actualizaciones no deseadas de un componente que se ejecuta en su dispositivo, le recomendamos que incluya directamente la versión que prefiera de ese componente al crear una implementación. Para obtener más información sobre el comportamiento de las actualizaciones AWS IoT Greengrass del software principal, consulteActualice el software AWS IoT Greengrass principal (OTA).

Componente	Detalles
Núcleo de Greengrass	Está disponible la versión 2.9.1 del núcleo de Greengrass. Mejoras y correcciones de errores • Añade una corrección por la que Greengrass se reinicia si una implementación elimina un componente del complemento.
Gestor de registros	Está disponible la versión 2.3.0 del nuevo administrador de registros. **Note** Le recomendamos que actualice a Greengrass nucleus 2.9.1 cuando actualice a log manager 2.3.0.

Componente	Detalles
	Nuevas características • Reduce las demoras en el registro al procesar y cargar directamente los archivos de registro activos en lugar de esperar a que se roten los archivos nuevos. Mejoras y correcciones de errores • Mejora la compatibilidad con la rotación de registros al rotar archivos con un nombre único. • Correcciones y mejoras menores adicionales.

Lanzamiento: actualización del software AWS IoT Greengrass Core v2.9.0 el 15 de noviembre de 2022

Esta versión incluye la versión 2.9.0 del componente núcleo de Greengrass y actualizaciones AWS de los componentes proporcionados.

Fecha de lanzamiento: 15 de noviembre de 2022

Aspectos destacados del lanzamiento

- Autenticación sin conexión: AWS IoT Greengrass ahora es compatible con la autenticación sin conexión. Puede configurar su dispositivo AWS IoT Greengrass principal para que los dispositivos cliente puedan conectarse a un dispositivo principal, incluso cuando el dispositivo principal no esté conectado a la nube. Para obtener más información, consulte Autenticación sin conexión.
- Subdespliegues: ahora puede crear subdespliegues. Puede usar un subdespliegue para resolver los despliegues fallidos. Cada subimplementación puede probar una configuración diferente de una implementación fallida en un subconjunto de dispositivos más pequeño. Para obtener más información, consulte Crear subdespliegues.

Detalles de la versión

- Actualizaciones de componentes públicos

Actualizaciones de componentes públicos

En la siguiente tabla se enumeran los componentes AWS proporcionados que incluyen funciones nuevas y actualizadas.

> **Important**
>
> Al implementar un componente, AWS IoT Greengrass instala las últimas versiones compatibles de todas las dependencias de ese componente. Por este motivo, es posible que las nuevas versiones con parches de los componentes públicos AWS proporcionados se implementen automáticamente en sus dispositivos principales si agrega nuevos dispositivos a un grupo de cosas o si actualiza la implementación destinada a esos dispositivos. Algunas actualizaciones automáticas, como las actualizaciones de Nucleus, pueden provocar que los dispositivos se reinicien de forma inesperada.
> Para evitar actualizaciones no deseadas de un componente que se ejecuta en su dispositivo, le recomendamos que incluya directamente la versión que prefiera de ese componente al crear una implementación. Para obtener más información sobre el comportamiento de las actualizaciones AWS IoT Greengrass del software principal, consulteActualice el software AWS IoT Greengrass principal (OTA).

Componente	Detalles
Núcleo de Greengrass	Está disponible la versión 2.9.0 del núcleo de Greengrass. Nuevas características • Añade la posibilidad de crear subdespliegues que reintenten los despliegues con un subconjunto de dispositivos más pequeño. Esta función crea una forma más eficiente de probar y resolver las implement aciones fallidas. Mejoras y correcciones de errores • Mejora la compatibilidad con los sistemas que no tienen `useraddgroupadd`, `yusermod`. • Mejoras y correcciones menores adicionales.

Componente	Detalles
Autenticación del dispositivo cliente	Está disponible la versión 2.3.0 del componente de autenticación del dispositivo cliente. Nuevas características • Añade compatibilidad con la autenticación sin conexión de los dispositivos cliente. Con esta función, los dispositivos cliente pueden seguir conectándose al dispositivo principal cuando este no esté conectado a Internet. • Añade compatibilidad con las autoridades de certificación (CA) proporcionadas por el cliente. Su dispositivo principal utiliza una CA proporcionada por el cliente como certificado raíz para generar los certificados de intermediario de MQTT.
Broker MQTT 5 (EMQX)	Está disponible la versión 1.2.0 del Bróker MQTT 5 (EMQX) componente. Nuevas características Añade soporte para cadenas de certificados.
Broker MQTT de Moquette	Está disponible la versión 2.3.0 del nuevo componente de broker MQTT de Moquette. Nuevas características Añade soporte para cadenas de certificados.
Gestor secreto	Ya está disponible la versión 2.1.4 del nuevo gestor secreto. Correcciones de errores y mejoras Soluciona un problema que provocaba que los secretos guardados en caché se eliminaran cuando se desplegaba el administrador de secretos y se reiniciaba Greengrass Nucleus.

Componente	Detalles
Gestor de transmisiones	Ya está disponible la versión 2.1.2 del nuevo administrador de transmisiones. Mejoras y correcciones de errores Soluciona un problema en los sistemas operativos Windows que utilizan un idioma distinto del inglés.

Lanzamiento: actualización del software AWS IoT Greengrass Core v2.8.1 el 13 de octubre de 2022

Esta versión incluye la versión 2.8.1 del componente núcleo de Greengrass.

Fecha de lanzamiento: 13 de octubre de 2022

> **Note**
>
> Si utiliza la versión 2.8.0 del núcleo de Greengrass, le recomendamos encarecidamente que actualice a la versión 2.8.1 del núcleo de Greengrass.

Detalles de la versión

- Actualizaciones de componentes públicos

Actualizaciones de componentes públicos

En la siguiente tabla se enumeran los componentes AWS proporcionados que incluyen funciones nuevas y actualizadas.

> **Important**
>
> Al implementar un componente, AWS IoT Greengrass instala las últimas versiones compatibles de todas las dependencias de ese componente. Por este motivo, es posible que las nuevas versiones con parches de los componentes públicos AWS proporcionados se implementen automáticamente en sus dispositivos principales si agrega nuevos dispositivos a un grupo de cosas o si actualiza la implementación destinada a esos dispositivos. Algunas

actualizaciones automáticas, como las actualizaciones de Nucleus, pueden provocar que los dispositivos se reinicien de forma inesperada.
Para evitar actualizaciones no deseadas de un componente que se ejecuta en su dispositivo, le recomendamos que incluya directamente la versión que prefiera de ese componente al crear una implementación. Para obtener más información sobre el comportamiento de las actualizaciones AWS IoT Greengrass del software principal, consulteActualice el software AWS IoT Greengrass principal (OTA).

Componente	Detalles
Núcleo de Greengrass	Está disponible la versión 2.8.1 del núcleo de Greengrass. Mejoras y correcciones de errores • Soluciona un problema por el que los códigos de error de despliegue no se generaban correctamente a partir de errores de la API de Greengras s. • Soluciona un problema que provocaba que las actualizaciones del estado de la flota enviaran información inexacta cuando un componente alcanzaba un `ERRORED` estado determinado durante un despliegue. • Soluciona un problema por el que las implementaciones no se podían completar cuando Greengrass tenía más de 50 suscripciones existente s.

Lanzamiento: actualización del software AWS IoT Greengrass Core v2.8.0 el 7 de octubre de 2022

Esta versión incluye la versión 2.8.0 del componente núcleo de Greengrass y la versión 1.1.0 del componente MQTT 5 broker (EMQX).

Fecha de lanzamiento: 7 de octubre de 2022

Aspectos destacados del lanzamiento

- Códigos de error de despliegue: el núcleo de Greengrass ahora informa de una respuesta sobre el estado del despliegue que incluye códigos de error detallados cuando no se puede completar

el despliegue de un componente. Para obtener más información, consulte Códigos de error de implementación detallados.

- Estados de error de los componentes: el núcleo de Greengrass ahora informa de una respuesta al estado de funcionamiento de los componentes que incluye estados de error detallados cuando un componente entra en `BROKEN` el estado o. `ERRORED` Para obtener más información, consulte Códigos de estado detallados de los componentes.

Detalles de la versión

- Actualizaciones de componentes públicos

Actualizaciones de componentes públicos

En la siguiente tabla se enumeran los componentes AWS proporcionados que incluyen funciones nuevas y actualizadas.

> **Important**
>
> Al implementar un componente, AWS IoT Greengrass instala las últimas versiones compatibles de todas las dependencias de ese componente. Por este motivo, es posible que las nuevas versiones con parches de los componentes públicos AWS proporcionados se implementen automáticamente en sus dispositivos principales si agrega nuevos dispositivos a un grupo de cosas o si actualiza la implementación destinada a esos dispositivos. Algunas actualizaciones automáticas, como las actualizaciones de Nucleus, pueden provocar que los dispositivos se reinicien de forma inesperada.
>
> Para evitar actualizaciones no deseadas de un componente que se ejecuta en su dispositivo, le recomendamos que incluya directamente la versión que prefiera de ese componente al crear una implementación. Para obtener más información sobre el comportamiento de las actualizaciones AWS IoT Greengrass del software principal, consulteActualice el software AWS IoT Greengrass principal (OTA).

Componente	Detalles
Núcleo de Greengrass	Está disponible la versión 2.8.0 del núcleo de Greengrass.

Componente	Detalles
	Nuevas características • Actualiza el núcleo de Greengrass para informar sobre una respuesta sobre el estado del despliegue que incluye códigos de error detallados cuando se produce un problema al implementar los componentes en un dispositivo central. Para obtener más información, consulte Códigos de error de implementación detallados. • Actualiza el núcleo de Greengrass para informar de una respuesta al estado de salud de un componente que incluye códigos de error detallados cuando un componente entra en el estado `BROKEN` o`ERRORED`. Para obtener más información, consulte Códigos de estado detallados de los componentes. • Amplía los campos de los mensajes de estado para mejorar la informaci ón de disponibilidad de los dispositivos en la nube. • Mejora la solidez del servicio de estado de la flota. Mejoras y correcciones de errores • Permite que un componente dañado se vuelva a instalar cuando se modifique su configuración. • Soluciona un problema por el que el reinicio del núcleo durante la implementación de bootstrap provoca un error en la implementación. • Soluciona un problema en Windows por el que la instalación fallaba cuando una ruta raíz contiene espacios. • Soluciona un problema por el que un componente que se apagaba durante una implementación utilizaba el script de apagado de la nueva versión. • Varias mejoras de apagado. • Correcciones y mejoras menores adicionales.

Componente	Detalles
Broker MQTT 5 (EMQX)	Está disponible la versión 1.1.0 del Bróker MQTT 5 (EMQX) componente. Nuevas características • Añade compatibilidad con las configuraciones de EMQX, incluidas las opciones de intermediación y los complementos. Mejoras y correcciones de errores • Actualiza EMQX a la versión 4.4.9.

Lanzamiento: actualización del software AWS IoT Greengrass Core v2.7.0 el 28 de julio de 2022

Esta versión incluye la versión 2.7.0 del componente núcleo de Greengrass, la versión 2.1.0 del componente administrador de flujos y la versión 2.2.5 del componente administrador Lambda.

Fecha de lanzamiento: 28 de julio de 2022

Características destacadas de la versión

- Métricas de telemetría de Stream Manager: ahora, Stream Manager envía automáticamente las métricas de telemetría a Amazon EventBridge, para que puedas crear aplicaciones en la nube que supervisen y analicen el volumen de datos que cargan tus dispositivos principales. Para obtener más información, consulte Recopile datos de telemetría del estado del sistema de los dispositivos principales AWS IoT Greengrass.
- Autoridad de certificación (CA) personalizada: ahora se admiten los certificados de cliente firmados por una CA de certificados personalizada, en la que la CA no esté registrada. AWS IoT Para obtener más información, consulte Utilice un certificado de dispositivo firmado por una entidad emisora de certificados privada.

Detalles de la versión

- Actualizaciones de componentes públicos

Actualizaciones de componentes públicos

En la siguiente tabla se enumeran los componentes AWS proporcionados que incluyen funciones nuevas y actualizadas.

> **Important**
>
> Al implementar un componente, AWS IoT Greengrass instala las últimas versiones compatibles de todas las dependencias de ese componente. Por este motivo, es posible que las nuevas versiones con parches de los componentes públicos AWS proporcionados se implementen automáticamente en sus dispositivos principales si agrega nuevos dispositivos a un grupo de cosas o si actualiza la implementación destinada a esos dispositivos. Algunas actualizaciones automáticas, como las actualizaciones de Nucleus, pueden provocar que los dispositivos se reinicien de forma inesperada.
> Para evitar actualizaciones no deseadas de un componente que se ejecuta en su dispositivo, le recomendamos que incluya directamente la versión que prefiera de ese componente al crear una implementación. Para obtener más información sobre el comportamiento de las actualizaciones AWS IoT Greengrass del software principal, consulteActualice el software AWS IoT Greengrass principal (OTA).

Componente	Detalles
Núcleo de Greengrass	Está disponible la versión 2.7.0 del núcleo de Greengrass. Nuevas características • Actualiza el núcleo de Greengrass para enviar actualizaciones de estado a la AWS IoT Greengrass nube cuando el dispositivo principal aplica un despliegue local. • Añade compatibilidad con los certificados de cliente firmados por una autoridad de certificación (CA) personalizada, en la que la CA no esté registrada. AWS IoT Para utilizar esta función, puede establecer la nueva opción `greengrassDataPlaneEndpoint` de configura ción `eniotdata`. Para obtener más información, consulte Utilice un certificado de dispositivo firmado por una entidad emisora de certifica dos privada.

Componente	Detalles
	Mejoras y correcciones de errores • Soluciona un problema por el que el núcleo de Greengrass retrasa un despliegue en determinadas situaciones cuando el núcleo se detiene o se reinicia. El núcleo ahora reanuda su despliegue después de que se reinicie. • Actualiza el instalador de Greengrass para que respete el `--start` argumento cuando se especifica configurar el software como un servicio del sistema. • Actualiza el comportamiento SubscribeToComponentUpdates para establecer el ID de despliegue en los casos en los que el núcleo actualiza un componente. • Correcciones y mejoras menores adicionales.
Gestor de transmisiones	Está disponible la versión 2.1.0 del componente administrador de transmisiones. Nuevas características • Actualiza este componente para enviar automáticamente las métricas de telemetría a Amazon. EventBridge Para obtener más información, consulte Recopile datos de telemetría del estado del sistema de los dispositivos principales AWS IoT Greengrass. Esta función requiere la versión 2.7.0 o posterior del componente núcleo de Greengrass. • Versión actualizada para el lanzamiento de la versión 2.7.0 de Greengrass nucleus.

Componente	Detalles
Gestor Lambda	Está disponible la versión 2.2.5 del componente Lambda Manager. Nuevas características • Añade compatibilidad con los comodines de temas de MQTT en las fuentes de eventos en las que se suscribe a mensajes locales de publicación o suscripción. Esta función requiere la versión 2.6.0 o posterior del componente núcleo de Greengrass. • Versión actualizada para el lanzamiento de la versión 2.7.0 de Greengrass nucleus.

Lanzamiento: actualización del software AWS IoT Greengrass Core v2.6.0 el 27 de junio de 2022

Esta versión incluye la versión 2.6.0 del componente núcleo de Greengrass, los AWS nuevos componentes proporcionados y las actualizaciones AWS de los componentes proporcionados.

Fecha de lanzamiento: 27 de junio de 2022

Aspectos destacados del lanzamiento

- Comodín en los temas de publicación o suscripción locales: ahora puede usar los comodines de MQTT al suscribirse a temas de publicación o suscripción locales. Para obtener más información, consulte Publicar/suscribir mensajes locales y SubscribeToTopic.
- Compatibilidad con sombras de dispositivos cliente: ahora puede interactuar con las sombras de los dispositivos cliente en componentes personalizados y sincronizar las sombras de los dispositivos cliente con ellas. AWS IoT Core Para obtener más información, consulte Interactúa con las sombras de los dispositivos cliente y sincronízalas.
- Soporte local de MQTT 5 para dispositivos cliente: ahora puede implementar el broker MQTT 5 de EMQX para utilizar las funciones de MQTT 5 en la comunicación entre los dispositivos cliente y un dispositivo principal. Para obtener más información, consulte Bróker MQTT 5 (EMQX) y Connect los dispositivos cliente a los dispositivos principales.

- Variables de receta en las configuraciones de los componentes: ahora puede utilizar variables de receta específicas en las configuraciones de los componentes. Puede usar estas variables de receta al definir la configuración predeterminada de un componente en una receta o al configurar un componente en una implementación. Para obtener más información, consulte Variables de receta y Utilice variables de receta en las actualizaciones de fusión.
- Comodín en las políticas de autorización de IPC: ahora puede usar el * comodín para que coincida con cualquier combinación de caracteres en las políticas de autorización de comunicación entre procesos (IPC). Este comodín le permite permitir el acceso a varios recursos en una única política de autorización. Para obtener más información, consulte Comodín en las políticas de autorización.
- Operaciones de IPC que gestionan las implementaciones y los componentes locales: ahora puede desarrollar componentes personalizados que gestionen las implementaciones locales y ver los detalles de los componentes. Para obtener más información, consulte IPC: Administrar despliegues y componentes locales.
- Operaciones de IPC que autentican y autorizan los dispositivos cliente: ahora puede utilizar estas operaciones para crear un componente de agente local personalizado. Para obtener más información, consulte IPC: Autenticar y autorizar los dispositivos cliente.

Detalles de la versión

- Actualizaciones de componentes públicos

Actualizaciones de componentes públicos

En la siguiente tabla se enumeran los componentes AWS proporcionados que incluyen funciones nuevas y actualizadas.

> **Important**
>
> Al implementar un componente, AWS IoT Greengrass instala las últimas versiones compatibles de todas las dependencias de ese componente. Por este motivo, es posible que las nuevas versiones con parches de los componentes públicos AWS proporcionados se implementen automáticamente en sus dispositivos principales si agrega nuevos dispositivos a un grupo de cosas o si actualiza la implementación destinada a esos dispositivos. Algunas actualizaciones automáticas, como las actualizaciones de Nucleus, pueden provocar que los dispositivos se reinicien de forma inesperada.
> Para evitar actualizaciones no deseadas de un componente que se ejecuta en su dispositivo, le recomendamos que incluya directamente la versión que prefiera de ese componente al

crear una implementación. Para obtener más información sobre el comportamiento de las actualizaciones AWS IoT Greengrass del software principal, consulteActualice el software AWS IoT Greengrass principal (OTA).

Componente	Detalles
Núcleo de Greengrass	Está disponible la versión 2.6.0 del núcleo de Greengrass. Nuevas características • Añade compatibilidad con los caracteres comodín de MQTT al suscribirse a temas locales de publicación/suscripción. Para obtener más información, consulte Publicar/suscribir mensajes locales y Subscribe ToTopic. • Añade compatibilidad con variables de receta en las configuraciones de componentes, distintas de la variable de receta. *component_dependency_name* `:configuration:` *json_pointer* Puede usar estas variables de receta al definir un componente `DefaultConfiguration` en una receta o al configurar un componente en una implementación. Para habilitar esta función, defina la opción interpolateComponentConfigurationde configuración en`true`. Para obtener más información, consulte Variables de receta y Utilice variables de receta en las actualizaciones de fusión. • Añade una compatibilidad total con el * carácter comodín en las políticas de autorización de la comunicación entre procesos (IPC). Ahora puede especificar el * carácter de una cadena de recursos para que coincida con cualquier combinación de caracteres. Para obtener más información, consulte Comodín en las políticas de autorización. • Añade compatibilidad con componentes personalizados para llamar a las operaciones de IPC que utiliza la CLI de Greengrass. Puede usar estas operaciones de IPC para administrar las implementaciones locales, ver los detalles de los componentes y generar una contraseña que podrá usar para iniciar sesión en la consola de depuración local. Para obtener más información, consulte IPC: administrar las implementaciones y los componentes locales.

Componente	Detalles
	Mejoras y correcciones de errores • Soluciona un problema por el que los componentes dependientes no reaccionaban cuando sus dependencias principales se reiniciaban o cambiaban de estado en determinadas situaciones. • Mejora los mensajes de error que el dispositivo principal envía al servicio AWS IoT Greengrass en la nube cuando se produce un error en la implementación. • Soluciona un problema por el que el núcleo de Greengrass desplegab a una cosa dos veces en determinadas situaciones cuando el núcleo se reiniciaba. • Correcciones y mejoras menores adicionales. Para obtener más información, consulte las versiones en GitHub.
Broker MQTT 5 (EMQX)	Está disponible la versión 1.0.0 del nuevo componente de broker MQTT 5 de EMQX. Nuevas características • Añade soporte para el broker EMQX MQTT 5 local. Los dispositivos cliente se pueden conectar a este intermediario MQTT para comunicar se con un dispositivo principal mediante las funciones de MQTT 5.

Componente	Detalles
Gestor oculto	Está disponible la versión 2.2.0 del componente administrador de sombras. Nuevas características • Añade compatibilidad con el servicio paralelo local a través de la interfaz local de publicación/suscripción. Ahora puede comunicarse con el agente local de mensajes de publicación/suscripción sobre temas de MQTT ocultos para obtener, actualizar y eliminar las sombras en el dispositivo principal. Esta función le permite conectar los dispositivos cliente al servicio paralelo local mediante el puente MQTT para retransmitir mensajes sobre temas ocultos entre los dispositivos cliente y la interfaz local de publicación/suscripción. Esta función requiere la versión 2.6.0 o posterior del componente núcleo de Greengrass. Para conectar los dispositivos cliente al servicio paralelo local, también debe utilizar la versión 2.2.0 o posterior del componente MQTT bridge. • Añade la `direction` opción que puede configurar para personalizar la dirección de sincronización de las sombras entre el servicio oculto local y el. Nube de AWS Puede configurar esta opción para reducir el ancho de banda y las conexiones alNube de AWS.
Autenticación del dispositivo cliente	Está disponible la versión 2.2.0 del componente de autenticación del dispositivo cliente. Nuevas características • Añade compatibilidad con componentes personalizados para llamar a las operaciones de comunicación entre procesos (IPC) a fin de autenticar y autorizar los dispositivos cliente. Puede utilizar estas operaciones en un componente de intermediario MQTT personalizado, por ejemplo. Para obtener más información, consulte IPC: Autenticar y autorizar dispositivos cliente. • Agrega las `threadPoolSize` opciones `maxActiveAuthTokens` `cloudQueueSize` , y que puede configurar para ajustar el rendimiento de este componente.

Componente	Detalles
Puente MQTT	Está disponible la versión 2.2.0 del componente de puente MQTT. Nuevas características • Añade compatibilidad con los caracteres comodín (#y+) de los temas MQTT cuando se especifica la publicación o suscripción local como agente de mensajes de origen. Esta función requiere la versión 2.6.0 o posterior del componente núcleo de Greengrass. • Añade la `targetTopicPrefix` opción, que puede especificar, para configurar el puente MQTT de forma que añada un prefijo al tema de destino cuando retransmita un mensaje.
Greengrass CLI	Está disponible la versión 2.6.0 de la CLI de Greengrass. Nuevas características • Añade compatibilidad con componentes personalizados para llamar a las operaciones de comunicación entre procesos (IPC) que utiliza la CLI de Greengrass. Puede usar estas operaciones de IPC para administrar las implementaciones locales, ver los detalles de los componentes y generar una contraseña que pueda usar para iniciar sesión en la consola de depuración local. Para obtener más información, consulte IPC: administrar las implementaciones y los componentes locales. Mejoras y correcciones de errores • Correcciones y mejoras menores adicionales.

Lanzamiento: actualización del software AWS IoT Greengrass Core v2.5.6 el 31 de mayo de 2022

Esta versión incluye la versión 2.5.6 del componente núcleo de Greengrass y la versión 2.2.4 del componente administrador de registros.

Fecha de lanzamiento: 31 de mayo de 2022

Detalles del lanzamiento

- Actualizaciones de componentes públicos

Actualizaciones de componentes públicos

En la siguiente tabla se enumeran los componentes AWS proporcionados que incluyen funciones nuevas y actualizadas.

> **Important**
>
> Al implementar un componente, AWS IoT Greengrass instala las últimas versiones compatibles de todas las dependencias de ese componente. Por este motivo, es posible que las nuevas versiones con parches de los componentes públicos AWS proporcionados se implementen automáticamente en sus dispositivos principales si agrega nuevos dispositivos a un grupo de cosas o si actualiza la implementación destinada a esos dispositivos. Algunas actualizaciones automáticas, como las actualizaciones de Nucleus, pueden provocar que los dispositivos se reinicien de forma inesperada.
> Para evitar actualizaciones no deseadas de un componente que se ejecuta en su dispositivo, le recomendamos que incluya directamente la versión que prefiera de ese componente al crear una implementación. Para obtener más información sobre el comportamiento de las actualizaciones AWS IoT Greengrass del software principal, consulteActualice el software AWS IoT Greengrass principal (OTA).

Componente	Detalles
Núcleo de Greengrass	Está disponible la versión 2.5.6 del núcleo de Greengrass. Nuevas características • Añade compatibilidad con los módulos de seguridad de hardware que utilizan claves ECC. Puede utilizar un módulo de seguridad de hardware (HSM) para almacenar de forma segura la clave privada y el certificado del dispositivo. Para obtener más información, consulte Integración de la seguridad de hardware.

Componente	Detalles
	Mejoras y correcciones de errores • Soluciona un problema que provocaba que la implementación nunca se completara cuando se implementaba un componente con un script de instalación defectuoso en determinadas situaciones. • Mejora el rendimiento durante el inicio. • Mejoras y correcciones menores adicionales.
Gestor de registros	Está disponible la versión 2.2.4 del componente gestor de registros. Mejoras y correcciones de errores • Mejora la estabilidad al gestionar configuraciones no válidas. • Mejoras y correcciones menores adicionales.

Lanzamiento: actualización del software AWS IoT Greengrass Core v2.5.5 el 6 de abril de 2022

Esta versión incluye la versión 2.5.5 del componente núcleo de Greengrass.

Fecha de lanzamiento: 6 de abril de 2022

Detalles del lanzamiento

- Actualizaciones de componentes públicos

Actualizaciones de componentes públicos

En la siguiente tabla se enumeran los componentes AWS proporcionados que incluyen funciones nuevas y actualizadas.

> **Important**
>
> Al implementar un componente, AWS IoT Greengrass instala las últimas versiones compatibles de todas las dependencias de ese componente. Por este motivo, es posible que las nuevas versiones con parches de los componentes públicos AWS proporcionados se implementen automáticamente en sus dispositivos principales si agrega nuevos dispositivos

a un grupo de cosas o si actualiza la implementación destinada a esos dispositivos. Algunas actualizaciones automáticas, como las actualizaciones de Nucleus, pueden provocar que los dispositivos se reinicien de forma inesperada.
Para evitar actualizaciones no deseadas de un componente que se ejecuta en su dispositivo, le recomendamos que incluya directamente la versión que prefiera de ese componente al crear una implementación. Para obtener más información sobre el comportamiento de las actualizaciones AWS IoT Greengrass del software principal, consulteActualice el software AWS IoT Greengrass principal (OTA).

Componente	Detalles
Núcleo de Greengrass	Está disponible la versión 2.5.5 del núcleo de Greengrass. Nuevas características • Añade la variable de `GG_ROOT_CA_PATH` entorno para los componentes, de forma que pueda acceder al certificado raíz de la entidad emisora de certificados (CA) en los componentes personalizados. Mejoras y correcciones de errores • Añade compatibilidad con dispositivos Windows que utilizan un idioma de visualización distinto del inglés. • Actualiza la forma en que el núcleo de Greengrass analiza los argumentos booleanos del instalador, de modo que puede especificar un argumento booleano sin un valor booleano para especificar un valor. `true` Por ejemplo, ahora puede especificar en `--provision` lugar de instalar con el aprovisionamiento automático de recursos. `--provision true` • Soluciona un problema por el que el dispositivo principal no informaba de su estado al servicio en la AWS IoT Greengrass nube después del aprovisionamiento en determinadas situaciones. • Correcciones y mejoras menores adicionales.

Lanzamiento: actualización del software AWS IoT Greengrass Core v2.5.4 el 23 de marzo de 2022

Esta versión incluye la versión 2.5.4 del componente núcleo de Greengrass y la versión 2.0.10 del componente lanzador Lambda.

Fecha de lanzamiento: 23 de marzo de 2022

Detalles del lanzamiento

- Actualizaciones de componentes públicos

Actualizaciones de componentes públicos

En la siguiente tabla se enumeran los componentes AWS proporcionados que incluyen funciones nuevas y actualizadas.

> **Important**
>
> Al implementar un componente, AWS IoT Greengrass instala las últimas versiones compatibles de todas las dependencias de ese componente. Por este motivo, es posible que las nuevas versiones con parches de los componentes públicos AWS proporcionados se implementen automáticamente en sus dispositivos principales si agrega nuevos dispositivos a un grupo de cosas o si actualiza la implementación destinada a esos dispositivos. Algunas actualizaciones automáticas, como las actualizaciones de Nucleus, pueden provocar que los dispositivos se reinicien de forma inesperada.
> Para evitar actualizaciones no deseadas de un componente que se ejecuta en su dispositivo, le recomendamos que incluya directamente la versión que prefiera de ese componente al crear una implementación. Para obtener más información sobre el comportamiento de las actualizaciones AWS IoT Greengrass del software principal, consulteActualice el software AWS IoT Greengrass principal (OTA).

Componente	Detalles
Núcleo de Greengrass	Está disponible la versión 2.5.4 del núcleo de Greengrass.

Componente	Detalles
	Mejoras y correcciones de errores • Corrección de errores y mejoras generales.
Lanzador Lambda	Está disponible la versión 2.0.10 del componente Lanzador Lambda. Mejoras y correcciones de errores • Corrección de errores y mejoras generales.

Lanzamiento: actualización del software AWS IoT Greengrass Core v2.5.3 el 6 de enero de 2022

Esta versión incluye la versión 2.5.3 del componente núcleo de Greengrass y el nuevo componente proveedor PKCS #11.

Fecha de lanzamiento: 6 de enero de 2022

Aspectos destacados del lanzamiento

- Integración de la seguridad del hardware: ahora puede configurar el software AWS IoT Greengrass principal para que utilice una clave privada y un certificado que se almacenarán de forma segura en un módulo de seguridad de hardware (HSM). Para obtener más información, consulte Integración de la seguridad de hardware.

Detalles de la versión

- Actualizaciones de componentes públicos

Actualizaciones de componentes públicos

En la siguiente tabla se enumeran los componentes AWS proporcionados que incluyen funciones nuevas y actualizadas.

> **Important**
>
> Al implementar un componente, AWS IoT Greengrass instala las últimas versiones compatibles de todas las dependencias de ese componente. Por este motivo, es posible que

las nuevas versiones con parches de los componentes públicos AWS proporcionados se implementen automáticamente en sus dispositivos principales si agrega nuevos dispositivos a un grupo de cosas o si actualiza la implementación destinada a esos dispositivos. Algunas actualizaciones automáticas, como las actualizaciones de Nucleus, pueden provocar que los dispositivos se reinicien de forma inesperada.
Para evitar actualizaciones no deseadas de un componente que se ejecuta en su dispositivo, le recomendamos que incluya directamente la versión que prefiera de ese componente al crear una implementación. Para obtener más información sobre el comportamiento de las actualizaciones AWS IoT Greengrass del software principal, consulteActualice el software AWS IoT Greengrass principal (OTA).

Componente	Detalles
Núcleo de Greengrass	Está disponible la versión 2.5.3 del núcleo de Greengrass. Nuevas características • Añade soporte para la integración de la seguridad del hardware. Puede utilizar un módulo de seguridad de hardware (HSM) para almacenar de forma segura la clave privada y el certificado del dispositivo. Para obtener más información, consulte Integración de la seguridad de hardware. Mejoras y correcciones de errores • Corrige un problema con las excepciones de tiempo de ejecución cuando el núcleo establece conexiones con AWS IoT Core MQTT.
Proveedor PKCS #11	Está disponible la versión 2.0.0 del componente de proveedor PKCS #11. Nuevas características • Añade compatibilidad con la integración de la seguridad del hardware. Puede utilizar un módulo de seguridad de hardware (HSM) para almacenar de forma segura la clave privada y el certificado del dispositivo. Para obtener más información, consulte Integración de la seguridad de hardware.

Lanzamiento: actualización del software AWS IoT Greengrass Core v2.5.2 el 3 de diciembre de 2021

Esta versión incluye la versión 2.5.2 del componente núcleo de Greengrass.

Fecha de lanzamiento: 3 de diciembre de 2021

Detalles del lanzamiento

- Actualizaciones de componentes públicos

Actualizaciones de componentes públicos

En la siguiente tabla se enumeran los componentes AWS proporcionados que incluyen funciones nuevas y actualizadas.

> **Important**
>
> Al implementar un componente, AWS IoT Greengrass instala las últimas versiones compatibles de todas las dependencias de ese componente. Por este motivo, es posible que las nuevas versiones con parches de los componentes públicos AWS proporcionados se implementen automáticamente en sus dispositivos principales si agrega nuevos dispositivos a un grupo de cosas o si actualiza la implementación destinada a esos dispositivos. Algunas actualizaciones automáticas, como las actualizaciones de Nucleus, pueden provocar que los dispositivos se reinicien de forma inesperada.
> Para evitar actualizaciones no deseadas de un componente que se ejecuta en su dispositivo, le recomendamos que incluya directamente la versión que prefiera de ese componente al crear una implementación. Para obtener más información sobre el comportamiento de las actualizaciones AWS IoT Greengrass del software principal, consulteActualice el software AWS IoT Greengrass principal (OTA).

Componente	Detalles
Núcleo de Greengrass	Está disponible la versión 2.5.2 del núcleo de Greengrass.

Componente	Detalles
	Mejoras y correcciones de errores • Soluciona un problema por el que, una vez actualizado el núcleo de Greengrass, el servicio de Windows no se puede iniciar de nuevo después de detenerlo o reiniciar el dispositivo.
AWS IoT Device Defender	Está disponible la versión 3.0.1 del AWS IoT Device Defendercomponente. Esta versión del AWS IoT Device Defender componente espera parámetros de configuración diferentes a los de la versión 2.x. Si utiliza una configuración no predeterminada para la versión 2.x y desea actualizar de la v2.x a la v3.x, debe actualizar la configuración del componente. Para obtener más información, consulte la configuración de los componentes. AWS IoT Device Defender Nuevas características • Añade compatibilidad con los dispositivos principales que ejecutan Windows. • Cambia el tipo de componente de componente Lambda a componente genérico. Este componente ya no depende del componente antiguo del router de suscripciones para crear suscripciones. • Agrega el nuevo parámetro `UseInstaller` de configuración que permite deshabilitar opcionalmente el script de instalación que instala las dependencias de los componentes.

Lanzamiento: actualización del software AWS IoT Greengrass Core v2.5.1 el 23 de noviembre de 2021

Esta versión incluye la versión 2.5.1 del componente núcleo de Greengrass.

Fecha de lanzamiento: 23 de noviembre de 2021

Detalles de la versión

- Actualizaciones de componentes públicos

Actualizaciones de componentes públicos

En la siguiente tabla se enumeran los componentes AWS proporcionados que incluyen funciones nuevas y actualizadas.

> **Important**
>
> Al implementar un componente, AWS IoT Greengrass instala las últimas versiones compatibles de todas las dependencias de ese componente. Por este motivo, es posible que las nuevas versiones con parches de los componentes públicos AWS proporcionados se implementen automáticamente en sus dispositivos principales si agrega nuevos dispositivos a un grupo de cosas o si actualiza la implementación destinada a esos dispositivos. Algunas actualizaciones automáticas, como las actualizaciones de Nucleus, pueden provocar que los dispositivos se reinicien de forma inesperada.
> Para evitar actualizaciones no deseadas de un componente que se ejecuta en su dispositivo, le recomendamos que incluya directamente la versión que prefiera de ese componente al crear una implementación. Para obtener más información sobre el comportamiento de las actualizaciones AWS IoT Greengrass del software principal, consulteActualice el software AWS IoT Greengrass principal (OTA).

Componente	Detalles
Núcleo de Greengrass	Está disponible la versión 2.5.1 del núcleo de Greengrass. Mejoras y correcciones de errores • Añade compatibilidad con las versiones de 32 bits del entorno de ejecución de Java (JRE) en Windows. • Cambia el comportamiento de eliminación de grupos de cosas en los dispositivos principales cuya AWS IoT política no concede el `greengrass:ListThingGroupsForCoreDevice` permiso. Con esta versión, la implementación continúa, registra una advertencia y no elimina componentes al quitar el dispositivo principal de un grupo de cosas. Para obtener más información, consulte Implemente AWS IoT Greengrass componentes en los dispositivos. • Corrige un problema con las variables de entorno del sistema que el núcleo de Greengrass pone a disposición de los procesos de los

Componente	Detalles
	componentes de Greengrass. Ahora puede reiniciar un componente para que utilice las variables de entorno del sistema más recientes.

Lanzamiento: actualización del software AWS IoT Greengrass Core v2.5.0 el 12 de noviembre de 2021

Esta versión incluye la versión 2.5.0 del componente núcleo de Greengrass, los AWS nuevos componentes proporcionados y las actualizaciones AWS de los componentes proporcionados.

Fecha de lanzamiento: 12 de noviembre de 2021

Aspectos destacados del lanzamiento

- Compatibilidad con dispositivos Windows: ahora puede ejecutar el software AWS IoT Greengrass principal en dispositivos con sistemas operativos Windows. Para obtener más información, consulte Plataformas compatibles y requisitos y Compatibilidad de funciones de Greengrass por sistema operativo.
- Nuevo comportamiento de eliminación de grupos de cosas: ahora puede eliminar un dispositivo principal de un grupo de cosas para eliminar los componentes de ese grupo de cosas en la próxima implementación en ese dispositivo.

  > **Important**
  >
  > Como resultado de este cambio, la AWS IoT política de un dispositivo principal debe tener el `greengrass:ListThingGroupsForCoreDevice` permiso. Si usó el instalador de software AWS IoT Greengrass Core para aprovisionar recursos, la AWS IoT política predeterminada lo permite`greengrass:*`, e incluye este permiso. Para obtener más información, consulte Autenticación y autorización de dispositivos en AWS IoT Greengrass.

- Soporte de seguridad de hardware: ahora puede configurar el software AWS IoT Greengrass Core para que utilice un módulo de seguridad de hardware (HSM), de forma que pueda almacenar de forma segura la clave privada y el certificado del dispositivo. Para obtener más información, consulte Integración de la seguridad de hardware.

- Compatibilidad con proxy HTTPS: ahora puede configurar el software AWS IoT Greengrass Core para que se conecte a través de proxies HTTPS. Para obtener más información, consulte Realizar la conexión en el puerto 443 o a través de un proxy de red.

Detalles de la versión

- Actualizaciones de soporte de la plataforma
- Actualizaciones de componentes públicos

Actualizaciones de soporte de la plataforma

Plataforma	Detalles
Windows	AWS IoT Greengrassahora admite la ejecución del software AWS IoT Greengrass principal en las siguientes versiones de Windows: • Windows 10 • Windows Server 2019 Para obtener más información, consulte Plataformas compatibles y requisitos y Compatibilidad de funciones de Greengrass por sistema operativo.

Actualizaciones de componentes públicos

En la siguiente tabla se enumeran los componentes AWS proporcionados que incluyen funciones nuevas y actualizadas.

Important

Al implementar un componente, AWS IoT Greengrass instala las últimas versiones compatibles de todas las dependencias de ese componente. Por este motivo, es posible que las nuevas versiones con parches de los componentes públicos AWS proporcionados se implementen automáticamente en sus dispositivos principales si agrega nuevos dispositivos a un grupo de cosas o si actualiza la implementación destinada a esos dispositivos. Algunas actualizaciones automáticas, como las actualizaciones de Nucleus, pueden provocar que los dispositivos se reinicien de forma inesperada.

Para evitar actualizaciones no deseadas de un componente que se ejecuta en su dispositivo, le recomendamos que incluya directamente la versión que prefiera de ese componente al crear una implementación. Para obtener más información sobre el comportamiento de las actualizaciones AWS IoT Greengrass del software principal, consulteActualice el software AWS IoT Greengrass principal (OTA).

Componente	Detalles
Núcleo de Greengrass	Está disponible la versión 2.5.0 del núcleo de Greengrass. Nuevas características • Añade compatibilidad con los dispositivos principales que ejecutan Windows. • Cambia el comportamiento de la eliminación de grupos de cosas. Con esta versión, puede eliminar un dispositivo principal de un grupo de cosas para desinstalar los componentes de ese grupo de cosas en la siguiente implementación. Como resultado de este cambio, la AWS IoT política de un dispositivo principal debe tener el `greengrass:ListThingGroupsForCoreDevice` permiso. Si usó el instalador de software AWS IoT Greengrass Core para aprovisionar recursos, la AWS IoT política predeterminada lo permite `greengrass:*`, e incluye este permiso. Para obtener más información, consulte Autenticación y autorización de dispositivos en AWS IoT Greengrass. • Añade compatibilidad con las configuraciones de proxy HTTPS. Para obtener más información, consulte Realizar la conexión en el puerto 443 o a través de un proxy de red. • Añade el nuevo parámetro `windowsUser` de configuración. Puede usar este parámetro para especificar el usuario predeterminado que se utilizará para ejecutar los componentes en un dispositivo principal de Windows. Para obtener más información, consulte Configure el usuario que ejecuta los componentes. • Agrega las nuevas opciones de `httpClient` configuración que puede usar para personalizar los tiempos de espera de las solicitudes HTTP

Componente	Detalles
	a fin de mejorar el rendimiento en redes lentas. Para obtener más información, consulte el parámetro de configuración HttpClient. Mejoras y correcciones de errores • Corrige la opción del ciclo de vida de arranque para reiniciar el dispositivo principal desde un componente. • Añade compatibilidad con guiones en las variables de receta. • Corrige la autorización de IPC para los componentes de la función Lambda bajo demanda. • Mejora los mensajes de registro y cambia los registros no críticos de un `DEBUG` nivel `INFO` a otro, por lo que los registros son más útiles. • Elimina el `iot:DescribeCertificate` permiso de la función de intercambio de fichas predeterminada que el núcleo de Greengrass crea al instalar el software AWS IoT Greengrass Core con aprovisionamiento automático. El núcleo de Greengrass no utiliza este permiso. • Corrige un problema por el que el script de aprovisionamiento automático no requería el `iam:GetPolicy` permiso si estaba `iam:CreatePolicy` disponible para la misma política. • Correcciones y mejoras menores adicionales.
Greengrass CLI	Está disponible la versión 2.5.0 de la CLI de Greengrass. Nuevas características • Añade compatibilidad con los dispositivos principales que ejecutan Windows. • Añade el nuevo parámetro `AuthorizedWindowsGroups` de configuración que puede especificar para autorizar a los grupos del sistema a utilizar la CLI de Greengrass en dispositivos Windows. • Agrega el `windowsUser` parámetro para las implementaciones locales. Puede usar este parámetro para especificar el usuario que se utilizará para ejecutar los componentes en un dispositivo principal de Windows.

Componente	Detalles
CloudWatch métricas	Está disponible la versión 3.0.0 del componente de CloudWatchmétricas. Esta versión del componente de CloudWatch métricas espera parámetros de configuración diferentes a los de la versión 2.x. Si utiliza una configuración no predeterminada para la versión 2.x y desea actualizar de la v2.x a la v3.x, debe actualizar la configuración del componente. Para obtener más información, consulte la configuración de los componentes de métricas. CloudWatch Nuevas características • Añade compatibilidad con los dispositivos principales que ejecutan Windows. • Cambia el tipo de componente de componente Lambda a componente genérico. Este componente ya no depende del componente antiguo del router de suscripciones para crear suscripciones. • Agrega un nuevo parámetro de `InputTopic` configuración para especificar el tema al que se suscribe el componente para recibir mensajes. • Agrega un nuevo parámetro `OutputTopic` de configuración para especificar el tema en el que el componente publica las respuestas de estado. • Añade un nuevo parámetro `PubSubToIoTCore` de configuración para especificar si se deben publicar o suscribirse a los temas de AWS IoT Core MQTT. • Añade el nuevo parámetro `UseInstaller` de configuración que permite deshabilitar opcionalmente el script de instalación que instala las dependencias de los componentes. Mejoras y correcciones de errores Añade compatibilidad con marcas de tiempo duplicadas en los datos de entrada.

Componente	Detalles
Gestor Lambda	Está disponible la versión 2.2.0 del componente Lambda Manager. Mejoras y correcciones de errores • Soluciona un problema por el que las funciones de Lambda no podían escribir registros tras un reinicio. • Soluciona un problema que provocaba que el router de suscripciones antiguo enviara mensajes duplicados cuando había caracteres comodín en el tema. • Soluciona un problema por el que las funciones Lambda no ancladas no podían utilizar la biblioteca de comunicación entre procesos (IPC) de Greengrass en. SDK para dispositivos con AWS IoT

Lanzamiento: actualización del software AWS IoT Greengrass Core v2.4.0 el 3 de agosto de 2021

Esta versión incluye la versión 2.4.0 del componente núcleo de Greengrass, los AWS nuevos componentes proporcionados y las actualizaciones AWS de los componentes proporcionados.

Fecha de lanzamiento: 3 de agosto de 2021

Aspectos destacados del lanzamiento

- Límites de recursos del sistema: el componente núcleo de Greengrass ahora admite los límites de recursos del sistema. Puede configurar la cantidad máxima de uso de CPU y RAM que los procesos de cada componente pueden usar en el dispositivo principal. Para obtener más información, consulte Configure los límites de recursos del sistema para los componentes.
- Pausar y reanudar componentes: el núcleo de Greengrass ahora admite pausar y reanudar componentes. Puede utilizar la biblioteca de comunicación entre procesos (IPC) para desarrollar componentes personalizados que detengan y reanuden los procesos de otros componentes. Para obtener más información, consulte PauseComponent y ResumeComponent.
- Instale con el aprovisionamiento de AWS IoT flotas: utilice el nuevo complemento de aprovisionamiento de AWS IoT flotas para instalar el software AWS IoT Greengrass Core en los dispositivos que se conectan para aprovisionar los recursos necesarios. AWS IoT AWS Los dispositivos utilizan un certificado de reclamación para el aprovisionamiento. Puede incrustar el

certificado de reclamación en los dispositivos durante la fabricación, de modo que cada dispositivo pueda aprovisionarse en cuanto se conecte a Internet. Para obtener más información, consulte Instale el software AWS IoT Greengrass principal con aprovisionamiento AWS IoT de flota.

- Instale con aprovisionamiento personalizado: desarrolle un complemento de aprovisionamiento personalizado para aprovisionar AWS los recursos necesarios al instalar el software AWS IoT Greengrass principal en los dispositivos. Puede crear una aplicación Java que se ejecute durante la instalación para configurar los dispositivos principales de Greengrass para su caso de uso personalizado. Para obtener más información, consulte Instale el software AWS IoT Greengrass principal con aprovisionamiento de recursos personalizado.

Detalles de la versión

- Actualizaciones de componentes públicos

Actualizaciones de componentes públicos

En la siguiente tabla se enumeran los componentes AWS proporcionados que incluyen funciones nuevas y actualizadas.

> **Important**
>
> Al implementar un componente, AWS IoT Greengrass instala las últimas versiones compatibles de todas las dependencias de ese componente. Por este motivo, es posible que las nuevas versiones con parches de los componentes públicos AWS proporcionados se implementen automáticamente en sus dispositivos principales si agrega nuevos dispositivos a un grupo de cosas o si actualiza la implementación destinada a esos dispositivos. Algunas actualizaciones automáticas, como las actualizaciones de Nucleus, pueden provocar que los dispositivos se reinicien de forma inesperada.
>
> Para evitar actualizaciones no deseadas de un componente que se ejecuta en su dispositivo, le recomendamos que incluya directamente la versión que prefiera de ese componente al crear una implementación. Para obtener más información sobre el comportamiento de las actualizaciones AWS IoT Greengrass del software principal, consulteActualice el software AWS IoT Greengrass principal (OTA).

Componente	Detalles
Núcleo de Greengrass	Está disponible la versión 2.4.0 del núcleo de Greengrass. Nuevas características • Añade compatibilidad con los límites de recursos del sistema. Puede configurar la cantidad máxima de uso de CPU y RAM que los procesos de cada componente pueden usar en el dispositivo principal. Para obtener más información, consulte Configure los límites de recursos del sistema para los componentes. • Añade operaciones de IPC para pausar y reanudar los component es. Para obtener más información, consulte PauseComponent y ResumeComponent. • Añade compatibilidad con el aprovisionamiento de complementos. Puede especificar un archivo JAR para que se ejecute durante la instalación a fin de aprovisionar AWS los recursos necesarios para un dispositivo principal de Greengrass. El núcleo de Greengrass incluye una interfaz que puede implementar para desarrollar complementos de aprovisionamiento personalizados. Para obtener más información, consulte Instale el software AWS IoT Greengrass principal con aprovisio namiento de recursos personalizado. • Agrega el `thing-name-policy` argumento opcional al instalador del software AWS IoT Greengrass Core. Puede usar esta opción para especificar una AWS IoT política existente o personalizada al instalar el software AWS IoT Greengrass Core con aprovisionamiento automático de recursos. Mejoras y correcciones de errores • Actualiza la configuración de registro al inicio. Esto corrige un problema por el que la configuración de registro no se aplicaba al inicio. • Actualiza el enlace simbólico del cargador de núcleos para que apunte al almacén de componentes de la carpeta raíz de Greengrass durante la instalación. Esta actualización le permite eliminar el archivo JAR y otros artefactos del núcleo que se descargan al instalar el software AWS IoT Greengrass Core.

Componente	Detalles
	• Correcciones y mejoras menores adicionales. Para obtener más información, consulte las versiones en GitHub.
Greengrass CLI	Está disponible la versión 2.4.0 de la CLI de Greengrass. Nuevas características • Añade compatibilidad con los límites de recursos del sistema. Al crear una implementación local, puede configurar la cantidad máxima de uso de CPU y RAM que los procesos de cada componente pueden usar en el dispositivo principal. Para obtener más información, consulte Configure los límites de recursos del sistema para los componentes y el comando deploy create.
AWS IoTaprovisionamiento de flota por reclamación	El complemento AWS IoT de aprovisionamiento de flotas mediante reclamación ya está disponible. Para obtener más información, consulte Instale el software AWS IoT Greengrass principal con aprovisionamiento AWS IoT de flota. Nuevas características • Añade soporte para instalar el software AWS IoT Greengrass Core con el aprovisionamiento de AWS IoT flotas. Durante la instalación, los dispositivos se conectan AWS IoT para aprovisionar AWS los recursos necesarios y descargar los certificados de los dispositivos para utilizarlos en las operaciones habituales.

Lanzamiento: actualización del software AWS IoT Greengrass Core v2.3.0 el 29 de junio de 2021

Esta versión proporciona la versión 2.3.0 del componente núcleo de Greengrass.

Fecha de lanzamiento: 29 de junio de 2021

Aspectos destacados del lanzamiento

- Soporte de configuración de gran tamaño: el componente núcleo de Greengrass ahora admite documentos de despliegue de hasta 10 MB. Ahora puede implementar actualizaciones de configuración de mayor tamaño en los componentes de Greengrass.

> **Note**
>
> Para usar esta función, la AWS IoT política del dispositivo principal debe permitir el `greengrass:GetDeploymentConfiguration` permiso. Si utilizaste el instalador del software AWS IoT Greengrass principal para aprovisionar recursos, la AWS IoT política de tu dispositivo principal lo permite`greengrass:*`, e incluye este permiso. Para obtener más información, consulte Autenticación y autorización de dispositivos en AWS IoT Greengrass.

Detalles de la versión

- Actualizaciones de componentes públicos

Actualizaciones de componentes públicos

En la siguiente tabla se enumeran los componentes AWS proporcionados que incluyen funciones nuevas y actualizadas.

> **Important**
>
> Al implementar un componente, AWS IoT Greengrass instala las últimas versiones compatibles de todas las dependencias de ese componente. Por este motivo, es posible que las nuevas versiones con parches de los componentes públicos AWS proporcionados se implementen automáticamente en sus dispositivos principales si agrega nuevos dispositivos a un grupo de cosas o si actualiza la implementación destinada a esos dispositivos. Algunas actualizaciones automáticas, como las actualizaciones de Nucleus, pueden provocar que los dispositivos se reinicien de forma inesperada.
> Para evitar actualizaciones no deseadas de un componente que se ejecuta en su dispositivo, le recomendamos que incluya directamente la versión que prefiera de ese componente al crear una implementación. Para obtener más información sobre el comportamiento de las

actualizaciones AWS IoT Greengrass del software principal, consulteActualice el software AWS IoT Greengrass principal (OTA).

Componente	Detalles
Núcleo de Greengrass	Está disponible la versión 2.3.0 del núcleo de Greengrass. Nuevas características • Añade soporte para documentos de configuración de despliegues de hasta 10 MB, en lugar de 7 KB (para despliegues dirigidos a objetos) o 31 KB (para despliegues dirigidos a grupos de objetos). Para utilizar esta función, la AWS IoT política de un dispositivo principal debe permitir el `greengrass:GetDeploymentConfiguration` permiso. Si utilizaste el instalador del software AWS IoT Greengrass principal para aprovisionar recursos, la AWS IoT política de tu dispositivo principal lo permite `greengrass:*` , e incluye este permiso. Para obtener más información, consulte Autenticación y autorización de dispositivos en AWS IoT Greengrass. • Añade la variable de `iot:thingName` receta. Puedes usar esta variable de receta para obtener el nombre del dispositivo AWS IoT principal en una receta. Para obtener más información, consulte Variables de receta. Mejoras y correcciones de errores • Correcciones y mejoras menores adicionales. Para obtener más información, consulte las versiones en GitHub.

Lanzamiento: actualización del software AWS IoT Greengrass Core v2.2.0 el 18 de junio de 2021

Esta versión incluye la versión 2.2.0 del componente núcleo de Greengrass, los AWS nuevos componentes proporcionados y las actualizaciones AWS de los componentes proporcionados.

Fecha de lanzamiento: 18 de junio de 2021

Aspectos destacados del lanzamiento

- Compatibilidad con dispositivos cliente: los nuevos componentes AWS de dispositivos cliente proporcionados le permiten conectar los dispositivos cliente a sus dispositivos principales mediante la detección en la nube. Puede sincronizar los dispositivos cliente AWS IoT Core e interactuar con los dispositivos cliente en los componentes de Greengrass. Para obtener más información, consulte Interactúa con dispositivos IoT locales.
- Servicio paralelo local: el nuevo componente administrador de sombras habilita el servicio paralelo local en sus dispositivos principales. Puede utilizar este servicio de sombras para interactuar con las sombras locales sin conexión a Internet mediante las bibliotecas de comunicación entre procesos (IPC) de Greengrass del. SDK para dispositivos con AWS IoT También puede utilizar el componente administrador de sombras para sincronizar los estados de sombra locales con. AWS IoT Core Para obtener más información, consulte Interactúa con las sombras de los dispositivos.

Detalles de la versión

- Actualizaciones de componentes públicos

Actualizaciones de componentes públicos

En la siguiente tabla se enumeran los componentes AWS proporcionados que incluyen funciones nuevas y actualizadas.

> **Important**
>
> Al implementar un componente, AWS IoT Greengrass instala las últimas versiones compatibles de todas las dependencias de ese componente. Por este motivo, es posible que las nuevas versiones con parches de los componentes públicos AWS proporcionados se implementen automáticamente en sus dispositivos principales si agrega nuevos dispositivos a un grupo de cosas o si actualiza la implementación destinada a esos dispositivos. Algunas actualizaciones automáticas, como las actualizaciones de Nucleus, pueden provocar que los dispositivos se reinicien de forma inesperada.
>
> Para evitar actualizaciones no deseadas de un componente que se ejecuta en su dispositivo, le recomendamos que incluya directamente la versión que prefiera de ese componente al crear una implementación. Para obtener más información sobre el comportamiento de las actualizaciones AWS IoT Greengrass del software principal, consulteActualice el software AWS IoT Greengrass principal (OTA).

Componente	Detalles
Núcleo de Greengrass	Está disponible la versión 2.2.0 del núcleo de Greengrass. Nuevas características • Añade operaciones de IPC para la gestión local de las sombras. Mejoras y correcciones de errores • Reduce el tamaño del archivo JAR. • Reduce el uso de memoria. • Soluciona problemas por los que la configuración del registro no se actualizaba en determinados casos. • Correcciones y mejoras menores adicionales. Para obtener más información, consulte las versiones en GitHub.
Gestor en la sombra	Está disponible la versión 2.0.0 del nuevo componente de administrador de sombras. Nuevas características • Añade compatibilidad con sombras clásicas y con nombre. • Añade compatibilidad con la gestión local de sombras mediante IPC. • Añade compatibilidad con la sincronización oculta conAWS IoT Core.
Autenticación del dispositivo cliente	Está disponible la versión 2.0.0 del nuevo componente de autenticación del dispositivo cliente. Nuevas características • Añade compatibilidad con los dispositivos cliente de Greengrass, que son dispositivos IoT locales que se conectan a un dispositivo central a través de MQTT. • Añade compatibilidad con la autenticación y la autorización de los dispositivos cliente y sus acciones de MQTT.
Broker MQTT de Moquette	Está disponible la versión 2.0.0 del nuevo componente de broker MQTT de Moquette.

Componente	Detalles
	Nuevas características • Añade compatibilidad con un broker MQTT local de Moquette que gestiona la comunicación con los dispositivos cliente.
Puente MQTT	Está disponible la versión 2.0.0 del nuevo componente de puente MQTT. Nuevas características • Añade soporte para retransmitir mensajes entre el bróker MQTT local, el bróker local de publicación/suscripción de Greengrass y el bróker MQTT. AWS IoT Core
Detector de IP	Está disponible la versión 2.0.0 del nuevo componente detector de IP. Nuevas características • Añade soporte para enviar los puntos finales del intermediario MQTT local de un dispositivo principal al servicio en la AWS IoT Greengrass nube para que puedan conectarse los dispositivos cliente.
Gestor de registros	Está disponible la versión 2.1.1 del componente gestor de registros. Mejoras y correcciones de errores • Soluciona un problema por el que la configuración del registro del sistema no se actualizaba en algunos casos.
Detección de objetos DLR	Está disponible la versión 2.1.2 de la detección de objetos DLR. Mejoras y correcciones de errores • Corrige un problema de escalado de la imagen que provocaba que los recuadros delimitadores no fueran precisos en los resultados de la inferencia de detección de objetos del DLR de muestra.

Componente	Detalles
TensorFlow Detección de objetos de Lite	Está disponible la versión 2.1.1 de la detección de objetos TensorFlow Lite. Mejoras y correcciones de errores • Corrige un problema de escalado de la imagen que provocaba que los recuadros delimitadores no fueran precisos en los resultados de la inferencia de detección de objetos de TensorFlow Lite de muestra.

Lanzamiento: actualización del software AWS IoT Greengrass Core v2.1.0 el 26 de abril de 2021

Esta versión proporciona la versión 2.1.0 del componente núcleo de Greengrass y AWS actualiza los componentes proporcionados por él.

Fecha de lanzamiento: 26 de abril de 2021

Aspectos destacados del lanzamiento

- Integración de Docker Hub y Amazon Elastic Container Registry (Amazon ECR): el nuevo componente de administrador de aplicaciones de Docker le permite descargar imágenes públicas o privadas de Amazon ECR. También puede usar este componente para descargar imágenes públicas de Docker Hub y. AWS Marketplace Para obtener más información, consulte Ejecute un contenedor Docker.
- Dockerfile e imágenes de Docker para el software AWS IoT Greengrass Core: puede utilizar la imagen de Docker de Greengrass para ejecutarla en AWS IoT Greengrass un contenedor de Docker que utilice Amazon Linux 2 como sistema operativo base. También puedes usar el AWS IoT Greengrass Dockerfile para crear tu propia imagen de Greengrass. Para obtener más información, consulte Ejecute AWS IoT Greengrass el software principal en un contenedor de Docker.
- Soporte para marcos y plataformas de aprendizaje automático adicionales: puede implementar componentes de inferencia de aprendizaje automático de muestra que utilizan modelos previamente entrenados para realizar la clasificación de imágenes de muestra y la detección de objetos con TensorFlow Lite 2.5.0 y DLR 1.6.0. Esta versión también amplía los ejemplos de compatibilidad con el aprendizaje automático para los dispositivos Armv8 (AArch64). Para obtener más información, consulte Cómo realizar la inferencia de machine learning.

Detalles de la versión

Actualizaciones de soporte de la plataforma

Plataforma	Detalles
Docker	Ya AWS IoT Greengrass están disponibles un Dockerfile y una imagen de Docker. Dockerfile AWS IoT Greengrassproporciona un Dockerfile para crear una imagen de contenedor que tiene el software AWS IoT Greengrass principal y las dependencias instaladas en una imagen base de Amazon Linux 2 (x86_64). Puede modificar la imagen base del Dockerfile para que se ejecute en una arquitectura de plataforma diferente. AWS IoT Greengrass Imagen de Docker AWS IoT Greengrassproporciona una imagen de Docker prediseñada que tiene el software AWS IoT Greengrass principal y las dependencias instaladas en una imagen base de Amazon Linux 2 (x86_64). Para obtener más información, consulte Ejecute AWS IoT Greengrass el software principal en un contenedor de Docker.

Actualizaciones de componentes públicos

En la siguiente tabla se enumeran los componentes AWS proporcionados que incluyen funciones nuevas y actualizadas.

> **Important**
>
> Al implementar un componente, AWS IoT Greengrass instala las últimas versiones compatibles de todas las dependencias de ese componente. Por este motivo, es posible que

las nuevas versiones con parches de los componentes públicos AWS proporcionados se implementen automáticamente en sus dispositivos principales si agrega nuevos dispositivos a un grupo de cosas o si actualiza la implementación destinada a esos dispositivos. Algunas actualizaciones automáticas, como las actualizaciones de Nucleus, pueden provocar que los dispositivos se reinicien de forma inesperada.
Para evitar actualizaciones no deseadas de un componente que se ejecuta en su dispositivo, le recomendamos que incluya directamente la versión que prefiera de ese componente al crear una implementación. Para obtener más información sobre el comportamiento de las actualizaciones AWS IoT Greengrass del software principal, consulteActualice el software AWS IoT Greengrass principal (OTA).

Componente	Detalles

Núcleo de Greengrass	Está disponible la versión 2.1.0 del núcleo de Greengrass.
	
	Nuevas características
	• Admite la descarga de imágenes de Docker desde repositorios privados en Amazon ECR.
	• Añade los siguientes parámetros para personalizar la configuración de MQTT en los dispositivos principales:
	  • `maxInFlightPublishes` — El número máximo de mensajes QoS 1 de MQTT no confirmados que pueden estar en vuelo al mismo tiempo.
	  • `maxPublishRetry` — El número máximo de veces que se puede reintentar un mensaje que no se publica.
	• Añade el parámetro `fleetstatusservice` de configuración para configurar el intervalo en el que el dispositivo principal publica el estado del Nube de AWS dispositivo en.
	• Correcciones y mejoras menores adicionales. Para obtener más información, consulte las versiones en GitHub.
	
	Mejoras y correcciones de errores
	• Corrige un problema que provocaba que los despliegues ocultos se duplicaran al reiniciarse el núcleo.

Componente	Detalles
	• Corrige un problema que provocaba que el núcleo se bloqueara cuando detectaba una excepción de carga de servicio. • Mejora la resolución de dependencias de componentes para evitar que se produzca un error en una implementación que incluya una dependencia circular. • Corrige un problema que impedía volver a implementar un componente de un complemento si ese componente se había eliminado previamente del dispositivo principal. • Se ha corregido un problema que provocaba que la variable de `HOME` entorno se estableciera en el `/greengrass/v2 /work` directorio de los componentes de Lambda o de los componentes que se ejecutan como root. La `HOME` variable ahora está correctamente configurada en el directorio principal del usuario que ejecuta el componente. • Correcciones y mejoras menores adicionales. Para obtener más información, consulte las versiones en GitHub.
Gestor de aplicaciones Docker	Está disponible la versión 2.0.0 del nuevo componente del administrador de aplicaciones de Docker. Nuevas características • Administra las credenciales para descargar imágenes de repositorios privados en Amazon ECR. • Descarga imágenes públicas de Amazon ECR, Docker Hub y. AWS Marketplace
Lanzador Lambda	Está disponible la versión 2.0.4 del componente Lanzador Lambda. Mejoras y correcciones de errores • Soluciona un problema por el que el componente no pasa correctam ente `AddGroupOwner` al contenedor de funciones Lambda.

Componente	Detalles
Enrutador de suscripciones antiguo	Está disponible la versión 2.1.0 del componente de enrutador de suscripción antiguo. Mejoras y correcciones de errores • Añade soporte para especificar los nombres de los componentes en lugar de los ARN para `source` y `target`. Si especifica un nombre de componente para una suscripción, no necesita volver a configurar la suscripción cada vez que cambie la versión de la función Lambda.
Consola de depuración local	Está disponible la versión 2.1.0 del componente de la consola de depuración local. Nuevas características • Utiliza HTTPS para proteger la conexión a la consola de depuración local. HTTPS está activado de forma predeterminada. Mejoras y correcciones de errores • Puede descartar los mensajes de la barra flash en el editor de configura ción.
Gestor de registros	Está disponible la versión 2.1.0 del componente gestor de registros. Mejoras y correcciones de errores • Utilice valores predeterminados para `logFileDirectoryPath` y `logFileRegex` que funcionen para los componentes de Greengrass que imprimen con salida estándar (stdout) y error estándar (stderr). • Dirija correctamente el tráfico a través de un proxy de red configurado al cargar registros en Logs. CloudWatch • Maneje correctamente los dos puntos (`:`) en los nombres de los flujos de registro. CloudWatch Los nombres de los flujos de registro de registros no admiten signos de dos puntos. • Simplifique los nombres de los flujos de registro eliminando los nombres de los grupos de cosas del flujo de registro. • Elimine un mensaje de registro de errores que se imprime con un comportamiento normal.

Componente	Detalles
Clasificación de imágenes DLR	Está disponible la versión 2.1.1 del componente de clasificación de imágenes del DLR. Nuevas características • Utilice Deep Learning Runtime v1.6.0. • Añada compatibilidad con la clasificación de imágenes de muestra en las plataformas Armv8 (AArch64). Esto amplía el soporte de aprendizaje automático para los dispositivos principales de Greengrass que ejecutan NVIDIA Jetson, como el Jetson Nano. • Habilite la integración de la cámara para la inferencia de muestras. Utilice el nuevo parámetro de `UseCamera` configuración para permitir que el código de inferencia de muestra acceda a la cámara del dispositivo principal de Greengrass y ejecute la inferencia localmente en la imagen capturada. • Agregue soporte para publicar los resultados de la inferencia en. Nube de AWS Utilice el nuevo parámetro `PublishResultsOnTopic` de configuración para especificar el tema sobre el que desea publicar los resultados. • Añada el nuevo parámetro de `ImageDirectory` configuración que le permita especificar un directorio personalizado para la imagen en la que desee realizar la inferencia. Mejoras y correcciones de errores • Escriba los resultados de la inferencia en el archivo de registro del componente en lugar de en un archivo de inferencias independiente. • Utilice el módulo de registro del software AWS IoT Greengrass Core para registrar la salida de los componentes. • Utilice el SDK para dispositivos con AWS IoT para leer la configuración del componente y aplicar los cambios de configuración.

Componente	Detalles
Detección de objetos DLR	Está disponible la versión 2.1.1 del componente de detección de objetos del DLR. Nuevas características • Utilice Deep Learning Runtime v1.6.0. • Añada compatibilidad con la detección de objetos de muestra en las plataformas Armv8 (AArch64). Esto amplía el soporte de aprendizaje automático para los dispositivos principales de Greengrass que ejecutan NVIDIA Jetson, como el Jetson Nano. • Habilite la integración de la cámara para la inferencia de muestras. Utilice el nuevo parámetro de `UseCamera` configuración para permitir que el código de inferencia de muestra acceda a la cámara del dispositivo principal de Greengrass y ejecute la inferencia localmente en la imagen capturada. • Agregue soporte para publicar los resultados de la inferencia en. Nube de AWS Utilice el nuevo parámetro `PublishResultsOnTopic` de configuración para especificar el tema sobre el que desea publicar los resultados. • Añada el nuevo parámetro de `ImageDirectory` configuración que le permita especificar un directorio personalizado para la imagen en la que desee realizar la inferencia. Mejoras y correcciones de errores • Escriba los resultados de la inferencia en el archivo de registro del componente en lugar de en un archivo de inferencias independiente. • Utilice el módulo de registro del software AWS IoT Greengrass Core para registrar la salida de los componentes. • Utilice el SDK para dispositivos con AWS IoT para leer la configuración del componente y aplicar los cambios de configuración.

Componente	Detalles
Tienda de modelos de clasificación de imágenes DLR	Está disponible la versión 2.1.1 del componente de almacén de modelos de clasificación de imágenes del DLR. Nuevas características • Agregue un ejemplo de modelo de clasificación de imágenes de ResNet -50 para las plataformas Armv8 (AArch64). Esto amplía el soporte de aprendizaje automático para los dispositivos principales de Greengrass que ejecutan NVIDIA Jetson, como el Jetson Nano.
Tienda de modelos de detección de objetos DLR	Está disponible la versión 2.1.1 del componente de tienda de modelos de detección de objetos del DLR. Nuevas características • Añada un ejemplo del modelo de detección de objetos YoloV3 para las plataformas Armv8 (AArch64). Esto amplía el soporte de aprendizaje automático para los dispositivos principales de Greengrass que ejecutan NVIDIA Jetson, como el Jetson Nano.
Instalador de DLR	Está disponible la versión 1.6.1 del componente DLR. Nuevas características • Instale Deep Learning Runtime v1.6.0 y sus dependencias. • Añada soporte para la instalación de DLR en plataformas Armv8 (AArch64). Esto amplía el soporte de aprendizaje automático para los dispositivos principales de Greengrass que ejecutan NVIDIA Jetson, como el Jetson Nano. Mejoras y correcciones de errores • Instálelo SDK para dispositivos con AWS IoT en el entorno virtual para leer la configuración del componente y aplicar los cambios de configura ción. • Mejoras y correcciones de errores menores adicionales.

Componente	Detalles
TensorFlow Clasificación de imágenes ligera	Está disponible la versión 2.1.0 del nuevo componente de clasificación de imágenes TensorFlow Lite. Nuevas características • Añada soporte para la inferencia de clasificaciones de imágenes de muestra mediante TensorFlow Lite.
TensorFlow Detección de objetos Lite	Está disponible la versión 2.1.0 del nuevo componente de detección de objetos TensorFlow Lite. Nuevas características • Agregue soporte para la inferencia de detección de objetos de muestra mediante TensorFlow Lite.
TensorFlo w Tienda de modelos de clasificación de imágenes Lite	Está disponible la versión 2.1.0 del nuevo componente TensorFlow Lite de tienda de modelos de clasificación de imágenes. Nuevas características • Proporcione un modelo cuantificado MobileNet v1 previamente entrenado para inferir la clasificación de imágenes de muestra utilizando Lite. TensorFlow
TensorFlo w Tienda de modelos de detección de objetos Lite	Está disponible la versión 2.1.0 del nuevo componente TensorFlow Lite de tienda de modelos de detección de objetos. Nuevas características • Proporcione un MobileNet modelo de detección de disparo único (SSD) previamente entrenado en el conjunto de datos COCO para inferir la detección de objetos de muestra mediante Lite. TensorFlow
TensorFlow Lite	Está disponible la versión 2.5.0 del nuevo componente TensorFlow Lite. Nuevas características • Instale TensorFlow Lite v1.6.0 y sus dependencias en un entorno virtual en las plataformas Armv7, Armv8 (AArch64) y x86_64.

Lanzamiento: actualización del software AWS IoT Greengrass Core v2.0.5 el 9 de marzo de 2021

Esta versión proporciona la versión 2.0.5 del componente núcleo de Greengrass y AWS actualiza los componentes proporcionados por él. Soluciona un problema con la compatibilidad con el proxy de red y un problema con el punto final del plano de datos de Greengrass en las regiones de AWS China.

Fecha de lanzamiento: 9 de marzo de 2021

Actualizaciones de componentes públicos

En la siguiente tabla se enumeran los componentes AWS proporcionados que incluyen funciones nuevas y actualizadas.

> **Important**
>
> Al implementar un componente, AWS IoT Greengrass instala las últimas versiones compatibles de todas las dependencias de ese componente. Por este motivo, es posible que las nuevas versiones con parches de los componentes públicos AWS proporcionados se implementen automáticamente en sus dispositivos principales si agrega nuevos dispositivos a un grupo de cosas o si actualiza la implementación destinada a esos dispositivos. Algunas actualizaciones automáticas, como las actualizaciones de Nucleus, pueden provocar que los dispositivos se reinicien de forma inesperada.
> Para evitar actualizaciones no deseadas de un componente que se ejecuta en su dispositivo, le recomendamos que incluya directamente la versión que prefiera de ese componente al crear una implementación. Para obtener más información sobre el comportamiento de las actualizaciones AWS IoT Greengrass del software principal, consulteActualice el software AWS IoT Greengrass principal (OTA).

Componente	Detalles
Núcleo de Greengrass	Está disponible la versión 2.0.5 del núcleo de Greengrass.

Componente	Detalles
	Mejoras y correcciones de errores • Enruta correctamente el tráfico a través de un proxy de red configurado al descargar los componentes AWS proporcionados. • Utilice el punto final correcto del plano de datos de Greengrass en las regiones de AWS China.

Lanzamiento: actualización del software AWS IoT Greengrass Core v2.0.4 el 4 de febrero de 2021

Esta versión proporciona la versión 2.0.4 del componente núcleo de Greengrass. Incluye el nuevo `greengrassDataPlanePort` parámetro para configurar la comunicación HTTPS a través del puerto 443 y corrige errores. La política de IAM mínima ahora requiere que se ejecute el instalador de software AWS IoT Greengrass principal `iam:GetPolicy` y `sts:GetCallerIdentity` cuando se ejecute con `--provision true` él.

Fecha de lanzamiento: 4 de febrero de 2021

Actualizaciones de componentes públicos

En la siguiente tabla se enumeran los componentes AWS proporcionados que incluyen funciones nuevas y actualizadas.

> **Important**
>
> Al implementar un componente, AWS IoT Greengrass instala las últimas versiones compatibles de todas las dependencias de ese componente. Por este motivo, es posible que las nuevas versiones con parches de los componentes públicos AWS proporcionados se implementen automáticamente en sus dispositivos principales si agrega nuevos dispositivos a un grupo de cosas o si actualiza la implementación destinada a esos dispositivos. Algunas actualizaciones automáticas, como las actualizaciones de Nucleus, pueden provocar que los dispositivos se reinicien de forma inesperada.
> Para evitar actualizaciones no deseadas de un componente que se ejecuta en su dispositivo, le recomendamos que incluya directamente la versión que prefiera de ese componente al crear una implementación. Para obtener más información sobre el comportamiento de las

actualizaciones AWS IoT Greengrass del software principal, consulteActualice el software AWS IoT Greengrass principal (OTA).

Componente	Detalles
Núcleo de Greengrass	Está disponible la versión 2.0.4 del núcleo de Greengrass. Nuevas características • Habilita el tráfico HTTPS a través del puerto 443. Puede usar el nuevo parámetro de `greengrassDataPlanePort` configura ción de la versión 2.0.4 del componente núcleo para configurar la comunicación HTTPS para que viaje a través del puerto 443 en lugar del puerto predeterminado 8443. Para obtener más información, consulte Configure HTTPS a través del puerto 443. • Añade la variable de receta de ruta de trabajo. Puede utilizar esta variable de receta para obtener la ruta a las carpetas de trabajo de los componentes, que puede utilizar para compartir archivos entre los componentes y sus dependencias. Para obtener más información, consulte la variable de receta de ruta de trabajo. Mejoras y correcciones de errores • Impide la creación de la política de roles de intercambio de fichas AWS Identity and Access Management (IAM) si ya existe una política de roles. Como resultado de este cambio, el instalador ahora necesita la `iam:GetPolicy` y `sts:GetCallerIdentity` cuando se ejecuta con `--provision true` ella. Para obtener más información, consulte Política de IAM mínima para que el instalador aprovisione recursos. • Gestiona correctamente la cancelación de una implementación que aún no se ha registrado correctamente. • Actualiza la configuración para eliminar las entradas antiguas con marcas de tiempo más recientes al anular una implementación. • Correcciones y mejoras menores adicionales. Para obtener más información, consulte las versiones en GitHub.

Migrar desde AWS IoT Greengrass la versión 1

AWS IoT Greengrass Version 2es una versión principal del software, las API y la consola AWS IoT Greengrass principales. AWS IoT Greengrass V2introduce varias mejorasAWS IoT Greengrass V1, como las aplicaciones modulares, las implementaciones en grandes flotas de dispositivos y la compatibilidad con plataformas adicionales.

> **Note**
>
> A partir del 30 de junio de 2023, ya AWS IoT Greengrass Version 1 no recibirá actualizaciones de funciones, mejoras, correcciones de errores ni parches de seguridad. Para obtener más información, consulte la política de mantenimiento de AWS IoT Greengrass V1 Si lo usaAWS IoT Greengrass V1, le recomendamos encarecidamente que migre aAWS IoT Greengrass V2.

Siga las instrucciones de esta guía para migrar de AWS IoT Greengrass V1 aAWS IoT Greengrass V2.

¿Puedo ejecutar mis aplicaciones de la versión 1 en la versión 2?

La mayoría de las aplicaciones de la versión 1 pueden ejecutarse en los dispositivos principales de la versión 2 sin necesidad de cambiar el código de la aplicación. Si tus aplicaciones de la versión 1 utilizan la siguiente función, no podrás ejecutarlas en la versión 2.

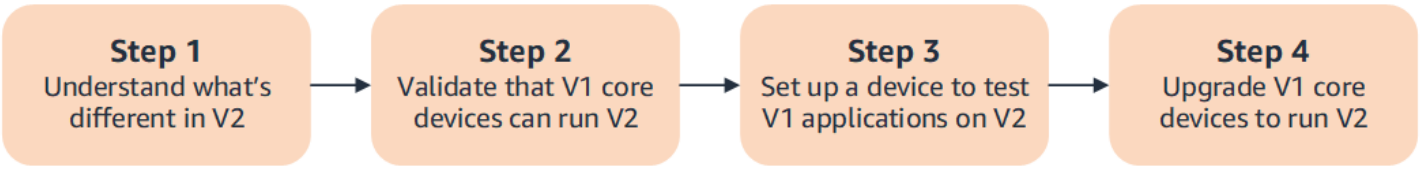
- Los tiempos de ejecución de las funciones Lambda en C y C++

Si las aplicaciones de la versión 1 utilizan alguna de las siguientes funciones, debe modificar el código de la aplicación para utilizar la SDK para dispositivos con AWS IoT versión 2 para ejecutar las aplicaciones. AWS IoT Greengrass V2

- Interactúe con el servicio paralelo local
- Publicar mensajes en dispositivos conectados localmente (dispositivos Greengrass)

Información general sobre la migración

En un nivel superior, puede utilizar el siguiente procedimiento para actualizar los dispositivos principales de AWS IoT Greengrass V1 aAWS IoT Greengrass V2. El procedimiento exacto que siga depende de los requisitos específicos de su entorno.



1. Comprenda las diferencias entre V1 y V2

   AWS IoT Greengrass V2presenta nuevos conceptos fundamentales para las flotas de dispositivos y el software desplegable, y la V2 simplifica varios conceptos de la V1.

   El servicio AWS IoT Greengrass V2 en la nube y el software AWS IoT Greengrass Core v2.x no son retrocompatibles con el servicio en la AWS IoT Greengrass V1 nube y el software Core v1.x. AWS IoT Greengrass Como resultado, las actualizaciones AWS IoT Greengrass V1 over-the-air (OTA) no pueden actualizar los dispositivos principales de la V1 a la V2.

2. Valide que los dispositivos principales de la V1 puedan ejecutar la V2

   Valide que un dispositivo central de la versión 1 pueda ejecutar el software AWS IoT Greengrass principal de la versión 2.x y AWS IoT Greengrass V2 sus funciones. AWS IoT Greengrass V2tiene requisitos de dispositivo diferentes a los de. AWS IoT Greengrass V1

3. Configure un nuevo dispositivo para probar las aplicaciones de la V1 en la V2

   Para minimizar el riesgo para sus dispositivos en producción, cree un nuevo dispositivo para probar las aplicaciones de la versión 1 en la versión 2. Tras instalar la versión 2.x del software AWS IoT Greengrass principal, puede crear e implementar AWS IoT Greengrass V2 componentes para migrar y probar sus AWS IoT Greengrass V1 aplicaciones.

4. Actualice los dispositivos principales de la V1 para ejecutar la V2

   Actualice un dispositivo principal V1 existente para ejecutar el software AWS IoT Greengrass Core v2.x y AWS IoT Greengrass V2 sus componentes. Para migrar una flota de dispositivos de la V1 a la V2, repita este paso para cada dispositivo de la flota.

Diferencias entre AWS IoT Greengrass V1 y AWS IoT Greengrass V2

AWS IoT Greengrass V2 presenta nuevos conceptos fundamentales para los dispositivos, las flotas y el software desplegable. En esta sección se describen los conceptos de la V1 que son diferentes en la V2.

Conceptos y terminología de Greengrass

Concepto	AWS IoT Greengrass V1	AWS IoT Greengrass V2
Código de la aplicación	En AWS IoT Greengrass V1, las funciones Lambda definen el software que se ejecuta en los dispositivos principales. En cada grupo de Greengrass, usted define las suscripciones y los recursos locales que utiliza la función. Para las funciones Lambda que el software AWS IoT Greengrass Core ejecuta en un entorno de ejecución Lambda en contenedores, debe definir los parámetros del contenedor, como los límites de memoria.	En AWS IoT Greengrass V2, los componentes son los módulos de software que se ejecutan en los dispositivos principales. • Cada componente tiene una receta que define los metadatos, los parámetros, las dependencias y los scripts del componente que se ejecutarán en cada paso del ciclo de vida del componente. • La receta también define los artefactos del componente, que son archivos binarios, como scripts, código compilado y recursos estáticos. • Al implementar un componente en un dispositivo principal, el dispositivo principal descarga la receta del componente y los artefactos para ejecutarlo.

Concepto	AWS IoT Greengrass V1	AWS IoT Greengrass V2
		Puede importar las funciones de Lambda de la versión 1 como componentes que se ejecutan en un entorno de ejecución de Lambda en. AWS IoT Greengrass V2 Al importar la función Lambda, se especifican las suscripciones, los recursos locales y los parámetros del contenedor de la función. Para obtener más informaci ón, consulte Paso 2: Crear e implementar AWS IoT Greengrass V2 componentes para migrar aplicaciones AWS IoT Greengrass V1. Para obtener más informaci ón sobre cómo crear componentes personalizados, consulteDesarrolle AWS IoT Greengrass componentes.

Concepto	AWS IoT Greengrass V1	AWS IoT Greengrass V2
AWS IoT Greengrass grupos e implementaciones	En AWS IoT Greengrass V1, un grupo define el dispositivo principal, la configuración y el software de ese dispositivo principal y la lista de AWS IoT elementos que se pueden conectar a ese dispositivo principal. Se crea una implementación para enviar la configuración de un grupo a un dispositivo principal.	En AWS IoT Greengrass V2, se utilizan las implementaciones para definir los componentes y las configuraciones de software que se ejecutan en los dispositivos principales. • Cada implementación se dirige a un dispositivo de un solo núcleo (que es una AWS IoT cosa) o a un grupo de AWS IoT cosas que puede contener varios dispositivos principales. • Las implementaciones en grupos de cosas son continuas, por lo que cuando se agrega un dispositivo principal a un grupo de cosas, recibe la configuración de software de ese grupo. Para obtener más información, consulte Implemente AWS IoT Greengrass componentes en los dispositivos. En AWS IoT Greengrass V2, también puede crear despliegues locales mediante la CLI de Greengrass para probar componentes de software personalizados

Concepto	AWS IoT Greengrass V1	AWS IoT Greengrass V2
		en el dispositivo en el que los desarrolla. Para obtener más información, consulte Crear AWS IoT Greengrass componentes.

Concepto	AWS IoT Greengrass V1	AWS IoT Greengrass V2
AWS IoT Greengrass Software básico	En este caso AWS IoT Greengrass V1, el software AWS IoT Greengrass principal es un paquete único que contiene el software y todas sus funciones. El dispositivo perimetral en el que se instala el software AWS IoT Greengrass Core se denomina núcleo Greengrass.	En este caso AWS IoT Greengrass V2, el software AWS IoT Greengrass Core es modular, por lo que puede elegir qué instalar para controlar el consumo de memoria. • El componente núcleo de Greengrass es la instalación mínima requerida del software AWS IoT Greengrass Core. El dispositivo perimetral en el que se instala el núcleo se denomina dispositivo central Greengrass. • El núcleo se encarga de las implementaciones, la organización y la gestión del ciclo de vida de otros componentes del dispositivo principal. • Las funciones como el administrador de transmisiones, el administrador de secretos y el administrador de registros son componentes que se implementan solo cuando se necesitan esas funciones. Para obtener más información, consulte AWS-componentes proporcionados.

Concepto	AWS IoT Greengrass V1	AWS IoT Greengrass V2
Connectors	En AWS IoT Greengras s V1, los conectores son módulos prediseñados que se implementan en los dispositi vos AWS IoT Greengrass V1 principales para interactuar con la infraestructura local AWS, los protocolos de los dispositivos y otros servicios en la nube.	En AWS IoT Greengras s V2, AWS proporciona componentes de Greengrass que implementan la funcional idad proporcionada por los conectores en la V1. Los siguientes AWS IoT Greengrass V2 componentes proporcionan la funcionalidad del conector Greengrass V1: • CloudWatch componente de métricas • AWS IoT Device Defender componente • Componente Firehose • Componente adaptador de protocolo Modbus-RTU • Componente Amazon SNS Para obtener más información, consulte AWS-componentes proporcionados.

Concepto	AWS IoT Greengrass V1	AWS IoT Greengrass V2
Dispositivos conectados (dispositivos Greengrass)	En AWS IoT Greengrass V1, los dispositivos conectados son AWS IoT elementos que se añaden a un grupo de Greengrass para conectarse al dispositivo principal de ese grupo y comunicarse a través de MQTT. Debe implementar ese grupo cada vez que añada o elimine un dispositivo conectado. Las suscripciones se utilizan para retransmitir mensajes entre los dispositivos conectados y las aplicaciones del dispositivo principal. AWS IoT Core En ambos AWS IoT Greengrass V1 y AWS IoT Greengrass V2, los dispositivos pueden ejecutar Freertos o usar la API de descubrimiento o SDK para dispositivos	En AWS IoT Greengrass V2, los dispositivos conectados se denominan dispositivos cliente de Greengrass. • Los dispositivos cliente se asocian a los dispositivos principales para conectarlos y comunicarse a través de MQTT. • Para autorizar la conexión de los dispositivos cliente, debe definir políticas de autorización que se puedan aplicar a grupos de dispositivos cliente, por lo que no es necesario crear una implementación para añadir o eliminar un dispositivo cliente. • Para retransmitir mensajes entre los dispositivos cliente y los componentes de Greengrass, puede configurar un componente de puente MQTT opcional. AWS IoT Core

Concepto	AWS IoT Greengrass V1	AWS IoT Greengrass V2
		con AWS IoTGreengrass para obtener información sobre los dispositivos principales a los que se pueden conectar. La API de descubrimiento de Greengrass es compatible con versiones anteriores, por lo que si tiene dispositivos cliente que se conectan a un dispositivo principal V1, puede conectarlos a un dispositivo principal V2 sin cambiar su código. Para obtener más informaci ón sobre los dispositivos cliente, consulteInteractúa con dispositivos IoT locales.

Concepto	AWS IoT Greengrass V1	AWS IoT Greengrass V2
Recursos locales	En AWS IoT Greengrass V1, las funciones de Lambda que se ejecutan en contenedores se pueden configurar para acceder a los volúmenes y dispositivos del sistema de archivos del dispositivo principal. Estos recursos del sistema de archivos se conocen como recursos locales.	En AWS IoT Greengrass V2, puede ejecutar componentes que sean funciones Lambda, contenedores de Docker o procesos nativos del sistema operativo o tiempos de ejecución personalizados. • Al importar una función Lambda contenerizada como componente, debe especificar los recursos locales que utiliza la función. • Las funciones Lambda no contenerizadas y los componentes que no son de Lambda pueden funcionar directamente con los recursos locales en los dispositivos principales, por lo que no es necesario especificar los recursos locales que utiliza el componente.

Concepto	AWS IoT Greengrass V1	AWS IoT Greengrass V2
Servicio paralelo local	En AWS IoT Greengrass V1, el servicio de sombras local está activado de forma predeterminada y solo admite sombras clásicas sin nombre. El SDK AWS IoT Greengras s principal se utiliza en las funciones de Lambda para interactuar con las sombras de los dispositivos.	En AWS IoT Greengrass V2, habilita el servicio de sombra local mediante la implement ación del componente de administrador de sombras. • Puede usar la SDK para dispositivos con AWS IoT V2 en funciones y componentes personali zados de Lambda para interactuar con las sombras de sus dispositivos. • El servicio de sombra local admite sombras con nombre. • El servicio de sombras local permite eliminar las sombras y sincronizarlas con AWS IoT Core ellas. Para obtener más informaci ón, consulte Interactúa con las sombras de los dispositivos.

Concepto	AWS IoT Greengrass V1	AWS IoT Greengrass V2

Suscripciones	En AWS IoT Greengrass V1, se definen las suscripciones para un grupo de Greengrass para especificar los canales de comunicación entre las funciones de Lambda, los conectores, los dispositivos conectados, el broker AWS IoT Core MQTT y el servicio paralelo local. Las suscripciones especifican dónde reciben las funciones de Lambda los mensajes de eventos para consumirlos como cargas útiles de funciones.	En AWS IoT Greengrass V2, se especifican los canales de comunicación sin utilizar suscripciones.
		• Los componentes administran sus propios canales de comunicación para interactuar con los mensajes locales de publicación/suscripción, los mensajes AWS IoT Core MQTT y el servicio alternativo local.
		  • Para desarrollar un componente que reaccione a los mensajes de otro componente o del intermediario AWS IoT Core MQTT, puede utilizar las interfaces de comunicación entre procesos (IPC) para la mensajería local de publicación/suscripción y la mensajería MQTT. AWS IoT Core
		  • Para desarrollar un componente que interactúe con el servicio paralelo local, puede utilizar la interfaz IPC para el servicio paralelo local.
		  • En la configuración del componente, se definen

Concepto	AWS IoT Greengrass V1	AWS IoT Greengrass V2
		políticas de autorización para especificar los temas y las sombras locales que el componente tiene permiso de uso. • Para configurar los canales de comunicación entre los dispositivos cliente, el agente local de publicación/suscripción y el agente AWS IoT Core MQTT, debe configurar e implementar el componente puente MQTT. El componente puente MQTT le permite interactuar con los dispositivos cliente en los componentes y retransmitir mensajes entre los dispositivos cliente y. AWS IoT Core

Concepto	AWS IoT Greengrass V1	AWS IoT Greengrass V2
Acceder a otros Servicios de AWS	En AWS IoT Greengrass V1, adjuntas un rol AWS Identity and Access Managemen t (IAM), denominado rol de grupo, a un grupo de Greengrass. El rol de grupo define los permisos a los que acceden las funciones y AWS IoT Greengrass características de Lambda en el dispositi vo principal de ese grupo. Servicios de AWS	En AWS IoT Greengrass V2, adjuntas un alias de AWS IoT rol a un dispositivo principal de Greengrass. El alias del rol apunta a un rol de IAM denominado rol de intercamb io de fichas. La función de intercambio de fichas define los permisos a los que acceden los componentes de Greengrass del dispositivo principal. Servicios de AWS Para obtener más informaci ón, consulte Autorizar a los dispositivos principales a interactuar con AWS los servicios.

Valide que los dispositivos principales V1 puedan ejecutar el software V2

El softwareAWS IoT Greengrass Core v2.x tiene requisitos diferentes a los del softwareAWS IoT Greengrass Core v1.x. Antes de actualizar los dispositivos principales de la V1 a la versión 2, revise los requisitos del dispositivo paraAWS IoT Greengrass V2. AWS IoT Greengrass V2actualmente no admite la migración de sistemas personalizados basados en Linux mediante el proyecto Yocto.

Puede utilizar AWS IoT Device Tester(IDT)AWS IoT Greengrass V2 para validar que los dispositivos cumplen con los requisitos para ejecutar el softwareAWS IoT Greengrass Core v2.x. IDT es un marco de pruebas descargable que se ejecuta en su computadora host y se conecta a los dispositivos para validarlos. Siga las instrucciones para usar IDT para ejecutar el conjunto deAWS IoT Greengrass calificaciones. Al configurar IDT, puede optar por validar si los dispositivos admiten funciones opcionales, como Docker, el aprendizaje automático (ML), la administración de flujos de datos y la integración de seguridad de hardware.

Si IDT informa de errores o errores en las pruebas de la versión 2 en un dispositivo principal de la V1, no podrá actualizar ese dispositivo de la V1 a la V2.

Configure un nuevo dispositivo central V2 para probar las aplicaciones V1

Configure un nuevo dispositivo AWS IoT Greengrass V2 principal para implementar y probar los componentes y AWS Lambda funciones AWS proporcionados para sus AWS IoT Greengrass V1 aplicaciones. También puede usar este dispositivo principal V2 para desarrollar y probar componentes Greengrass personalizados adicionales que ejecutan procesos nativos en los dispositivos principales. Después de probar sus aplicaciones en un dispositivo de núcleo V2, puede actualizar sus dispositivos principales V1 existentes a V2 e implementar los componentes de V2 que proporcionan la funcionalidad de la V1.

Paso 1: Instalar AWS IoT Greengrass V2 en un dispositivo nuevo

Instale el software AWS IoT Greengrass Core v2.x en un dispositivo nuevo. Puede seguir el tutorial de introducción para configurar un dispositivo y aprender a desarrollar e implementar componentes. En este tutorial, se utiliza el aprovisionamiento automático para configurar rápidamente un dispositivo. Cuando instale el software AWS IoT Greengrass Core v2.x, especifique el `--deploy-dev-tools` argumento para implementar la CLI de Greengrass, de modo que pueda desarrollar, probar y depurar componentes directamente en el dispositivo. Para obtener más información sobre otras opciones de instalación, incluida la forma de instalar el software AWS IoT Greengrass Core detrás de un proxy o mediante un módulo de seguridad de hardware (HSM), consulte. Instalación del software AWS IoT Greengrass Core

(Opcional) Habilita el registro en Amazon CloudWatch Logs

Para permitir que un dispositivo principal V2 cargue registros en Amazon CloudWatch Logs, puede implementar el componente AWS de administrador de registros proporcionado. Puede utilizar CloudWatch los registros para ver los registros de los componentes, de forma que pueda depurar y solucionar problemas sin tener acceso al sistema de archivos del dispositivo principal. Para obtener más información, consulte Supervisar AWS IoT Greengrass registros.

Paso 2: Crear e implementar AWS IoT Greengrass V2 componentes para migrar aplicaciones AWS IoT Greengrass V1

Puede ejecutar la mayoría de AWS IoT Greengrass V1 las aplicaciones en ellas AWS IoT Greengrass V2. Puede importar funciones Lambda como componentes en AWS IoT Greengrass V2 los que se ejecutan y puede usar los componentes AWS proporcionados que ofrecen la misma funcionalidad que los conectores. AWS IoT Greengrass

También puede desarrollar componentes personalizados para crear cualquier función o tiempo de ejecución que se ejecute en los dispositivos principales de Greengrass. Para obtener información sobre cómo desarrollar y probar componentes localmente, consulteCrear AWS IoT Greengrass componentes.

Temas

Importación de funciones Lambda V1

Puede importar funciones Lambda como AWS IoT Greengrass V2 componentes. Elija uno de los siguientes enfoques:

- Importe funciones Lambda V1 directamente como componentes de Greengrass.
- Actualice las funciones de Lambda para utilizar las bibliotecas de Greengrass en la SDK para dispositivos con AWS IoT versión 2 y, a continuación, importe las funciones de Lambda como componentes de Greengrass.
- Cree componentes personalizados que utilicen código que no sea de Lambda y la SDK para dispositivos con AWS IoT versión 2 para implementar la misma funcionalidad que sus funciones de Lambda.

Si su función Lambda usa funciones, como el administrador de transmisiones o secretos locales, debe definir las dependencias en los componentes AWS proporcionados que empaquetan estas funciones. Al implementar el componente de función Lambda, el despliegue también incluye el componente para cada función que defina como dependencia. En la implementación, puede configurar los parámetros, como los secretos que se van a implementar en el dispositivo principal. No todas las funciones de la V1 requieren una dependencia de componentes para la función Lambda en la V2. En la siguiente lista se describe cómo utilizar las funciones de la V1 en el componente de la función Lambda de la V2.

- Acceda a otros servicios AWS

  Si la función Lambda usa AWS credenciales para realizar solicitudes a otros AWS servicios, la función de intercambio de token del dispositivo principal debe permitir que el dispositivo principal realice las AWS operaciones que usa la función Lambda. Para obtener más información, consulte Autorizar a los dispositivos principales a interactuar con AWS los servicios.

- Gestor de transmisiones

  Si su función Lambda usa el administrador de flujos, especifíquelo `aws.greengrass.StreamManager` como una dependencia de componente al importar la función. Al implementar el componente del administrador de transmisiones, especifique los parámetros del administrador de transmisiones que desee configurar para los dispositivos principales de destino. La función de intercambio de tokens del dispositivo principal debe permitir que el dispositivo principal acceda a los Nube de AWS destinos que utilizas con Stream Manager. Para obtener más información, consulte Administrador de transmisiones.

- Secretos locales

  Si la función Lambda usa secretos locales, especifíquelos `aws.greengrass.SecretManager` como dependencia de componentes al importar la función. Al implementar el componente de administrador de secretos, especifique los recursos secretos que se van a implementar en los dispositivos principales de destino. La función de intercambio de fichas del dispositivo principal debe permitir que el dispositivo principal recupere los recursos secretos para desplegarlos. Para obtener más información, consulte Gestor secreto.

  Al implementar el componente de la función Lambda, configúrelo para que tenga una política de autorización de IPC que conceda permiso para usar la operación de GetSecretValue IPC en la V2. SDK para dispositivos con AWS IoT

- Sombras locales

Si la función Lambda interactúa con las sombras locales, debe actualizar el código de la función Lambda para usar la V2. SDK para dispositivos con AWS IoT También debe especificarlo `aws.greengrass.ShadowManager` como dependencia de un componente al importar la función. Para obtener más información, consulte Interactúa con las sombras de los dispositivos.

Al implementar el componente de la función Lambda, configúrelo para que tenga una política de autorización de IPC que conceda permiso para utilizar las operaciones de IPC ocultas en la V2. SDK para dispositivos con AWS IoT

- Suscripciones
  - Si su función Lambda se suscribe a mensajes de una fuente en la nube, especifique esas suscripciones como fuentes de eventos al importar la función.
  - Si la función de Lambda se suscribe a los mensajes de otra función de Lambda, o si la función de Lambda publica mensajes en u AWS IoT Core otras funciones de Lambda, configure e implemente el componente del router de suscripciones heredado al implementar la función de Lambda. Al implementar el componente del router de suscripciones heredado, especifique las suscripciones que utiliza la función Lambda.

    > **Note**
    >
    > El componente de router de suscripción antiguo solo es necesario si la función Lambda utiliza la `publish()` función del SDK AWS IoT Greengrass principal. Si actualiza el código de la función Lambda para utilizar la interfaz de comunicación entre procesos (IPC) de la SDK para dispositivos con AWS IoT V2, no necesitará implementar el componente de router de suscripción heredado. Para obtener más información, consulte los siguientes servicios de comunicación entre procesos:
    >
    > - Publicar/suscribir mensajes locales
    > - Publicar/suscribir mensajes MQTT AWS IoT Core

  - Si la función Lambda se suscribe a los mensajes de los dispositivos locales conectados, especifique esas suscripciones como fuentes de eventos al importar la función. También debe configurar e implementar el componente de puente MQTT para retransmitir los mensajes desde los dispositivos conectados a los temas locales de publicación o suscripción que especifique como fuentes de eventos.
  - Si su función Lambda publica mensajes en dispositivos conectados localmente, debe actualizar el código de la función Lambda para usar la SDK para dispositivos con AWS IoT V2 para

publicar mensajes locales de publicación/suscripción. También debe configurar e implementar el componente de puente MQTT para retransmitir los mensajes desde el intermediario local de mensajes de publicación/suscripción a los dispositivos conectados.

- Volúmenes y dispositivos locales

  Si la función Lambda en contenedores accede a volúmenes o dispositivos locales, especifique esos volúmenes y dispositivos al importar la función Lambda. Esta función no requiere una dependencia de componentes.

Para obtener más información, consulte AWS LambdaFunciones de ejecución.

Utilice conectores V1

Puede implementar los componentes AWS proporcionados que ofrecen la misma funcionalidad que algunos AWS IoT Greengrass conectores. Al crear la implementación, puede configurar los parámetros de los conectores.

Los siguientes AWS IoT Greengrass V2 componentes proporcionan la funcionalidad del conector Greengrass V1:

- CloudWatch componente de métricas
- AWS IoT Device Defender componente
- Componente Firehose
- Componente adaptador de protocolo Modbus-RTU
- Componente Amazon SNS

Ejecute contenedores de Docker

AWS IoT Greengrass V2 no proporciona un componente que sustituya directamente al conector de implementación de aplicaciones Docker de la versión 1. Sin embargo, puede usar el componente del administrador de aplicaciones de Docker para descargar imágenes de Docker y, a continuación, crear componentes personalizados que ejecuten contenedores de Docker a partir de las imágenes descargadas. Para obtener más información, consulte Ejecute un contenedor Docker y Gestor de aplicaciones Docker.

Ejecute una inferencia de aprendizaje automático

AWS IoT Greengrass V2 proporciona un componente de Amazon SageMaker Edge Manager que instala el agente de Amazon SageMaker Edge Manager y le permite utilizar modelos SageMaker compilados en NEO como componentes de modelos en los dispositivos principales de Greengrass. AWS IoT Greengrass V2 también proporciona componentes que instalan Deep Learning Runtime y TensorFlow Lite en su dispositivo. Puede utilizar el modelo DLR y TensorFlow Lite correspondientes y los componentes de inferencia para realizar la clasificación de imágenes de muestra y la inferencia de detección de objetos. Para utilizar otros marcos de aprendizaje automático, como MXNet TensorFlow, puede desarrollar sus propios componentes personalizados que utilicen estos marcos.

Conecta dispositivos Greengrass V1

Los dispositivos conectados en AWS IoT Greengrass V1 se denominan dispositivos cliente en AWS IoT Greengrass V2. AWS IoT Greengrass V2 la compatibilidad con dispositivos cliente es compatible con versiones anteriores AWS IoT Greengrass V1, por lo que puede conectar los dispositivos cliente V1 a los dispositivos principales V2 sin cambiar su código de aplicación. Para permitir que los dispositivos cliente se conecten a un dispositivo central V2, implemente los componentes de Greengrass que permitan la compatibilidad con los dispositivos cliente y asocie los dispositivos cliente al dispositivo principal. Para retransmitir mensajes entre los dispositivos cliente, el servicio AWS IoT Core en la nube y los componentes de Greengrass (incluidas las funciones Lambda), implemente y configure el componente puente MQTT. Puede implementar el componente detector de IP para detectar automáticamente la información de conectividad, o puede administrar manualmente los puntos finales. Para obtener más información, consulte Interactúa con dispositivos IoT locales.

Habilite el servicio paralelo local

En AWS IoT Greengrass V2, el servicio de sombra local se implementa mediante el componente AWS de administrador de sombras proporcionado. AWS IoT Greengrass V2 también incluye soporte para sombras con nombre asignado. Para permitir que sus componentes interactúen con las sombras locales y sincronicen los estados de las sombras AWS IoT Core, configure e implemente el componente administrador de sombras y utilice las operaciones de IPC ocultas en el código del componente. Para obtener más información, consulte Interactúa con las sombras de los dispositivos.

Intégrelo con AWS IoT SiteWise

Si utilizas tu dispositivo principal V1 como AWS IoT SiteWise puerta de enlace, sigue las instrucciones para configurar tu nuevo dispositivo central V2 como AWS IoT SiteWise puerta de

enlace. AWS IoT SiteWise proporciona un script de instalación que despliega los AWS IoT SiteWise componentes automáticamente.

Paso 3: Pruebe sus aplicaciones AWS IoT Greengrass V2

Tras crear e implementar los componentes de la V2 en el nuevo dispositivo principal de la V2, compruebe que las aplicaciones cumplen sus expectativas. Puede consultar los registros del dispositivo para ver los mensajes de salida estándar (stdout) y de error estándar (stderr) de sus componentes. Para obtener más información, consulte Supervisar AWS IoT Greengrass registros.

Si implementó la CLI de Greengrass en el dispositivo principal, puede usarla para depurar los componentes y sus configuraciones. Para obtener más información, consulte Comandos CLI de Greengrass.

Tras comprobar que las aplicaciones funcionan en un dispositivo principal V2, puede implementar los componentes Greengrass de la aplicación en otros dispositivos principales. Si ha desarrollado componentes personalizados que ejecutan procesos nativos o contenedores de Docker, primero debe publicar esos componentes en el AWS IoT Greengrass servicio para implementarlos en otros dispositivos principales.

Actualice los dispositivos principales de Greengrass V1 a Greengrass V2

Tras comprobar que las aplicaciones y los componentes funcionan en un dispositivo AWS IoT Greengrass V2 principal, puede instalar el software AWS IoT Greengrass principal v2.x en los dispositivos que actualmente ejecutan la versión 1.x, como los dispositivos de producción. A continuación, despliegue los componentes de Greengrass V2 para ejecutar las aplicaciones de Greengrass en los dispositivos.

Para actualizar una flota de dispositivos de la V1 a la V2, complete estos pasos para cada dispositivo que desee actualizar. Puede usar grupos de cosas para implementar componentes de la V2 en una flota de dispositivos principales.

> **Tip**
>
> Le recomendamos que cree un script para automatizar el proceso de actualización de una flota de dispositivos. Si AWS Systems Managersolía administrar su flota, puede usar Systems Manager para ejecutar ese script en cada dispositivo para actualizar su flota de la V1 a la V2.

Puede ponerse en contacto con su representante de AWS Enterprise Support si tiene preguntas sobre la mejor manera de automatizar el proceso de actualización.

Paso 1: Instale la versión AWS IoT Greengrass 2.x del software principal

Elija una de las siguientes opciones para instalar el software AWS IoT Greengrass Core v2.x en un dispositivo básico V1:

- Actualice en menos pasos

  Para realizar la actualización en menos pasos, puede desinstalar el software v1.x antes de instalar el software v2.x.
- Actualice con un tiempo de inactividad mínimo

  Para realizar la actualización con un tiempo de inactividad mínimo, puede instalar ambas versiones del software AWS IoT Greengrass Core al mismo tiempo. Tras instalar el software AWS IoT Greengrass Core v2.x y comprobar que las aplicaciones de la V2 funcionan correctamente, debe desinstalar el software AWS IoT Greengrass Core v1.x. Antes de elegir esta opción, tenga en cuenta la RAM adicional necesaria para ejecutar ambas versiones del software AWS IoT Greengrass Core al mismo tiempo.

Desinstale AWS IoT Greengrass Core v1.x antes de instalar la v2.x

Si desea realizar la actualización de forma secuencial, desinstale la versión 1.x del software AWS IoT Greengrass principal antes de instalar la versión 2.x en el dispositivo.

Para desinstalar la versión 1.x del software principal AWS IoT Greengrass

1. Si la AWS IoT Greengrass versión 1.x del software principal se ejecuta como un servicio, debe detener, deshabilitar y eliminar el servicio.

   a. Detenga el servicio AWS IoT Greengrass Core software v1.x en ejecución.

   ```
   sudo systemctl stop greengrass
   ```

   b. Espere hasta que se detenga el servicio. Puede utilizar el `list` comando para comprobar el estado del servicio.

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

c. Deshabilite el servicio.

```
sudo systemctl disable greengrass
```

d. Elimine el servicio.

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

2. Si la AWS IoT Greengrass versión 1.x del software principal no se ejecuta como un servicio, utilice el siguiente comando para detener el daemon. Sustituya *greengrass-root* por el nombre de la carpeta raíz de Greengrass. La ubicación predeterminada es `/greengrass`.

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

3. (Opcional) Haga una copia de seguridad de su carpeta raíz de Greengrass y, si corresponde, de su carpeta de escritura personalizada, en una carpeta diferente de su dispositivo.

a. Use el siguiente comando para copiar la carpeta raíz actual de Greengrass en una carpeta diferente y, a continuación, elimine la carpeta raíz.

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

b. Use el siguiente comando para mover la carpeta de escritura a otra carpeta y, a continuación, elimine la carpeta de escritura.

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

A continuación, puede utilizar las instrucciones de instalación AWS IoT Greengrass V2 para instalar el software en el dispositivo.

> **Tip**
>
> Para reutilizar la identidad de un dispositivo principal al migrarlo de la V1 a la V2, sigue las instrucciones para instalar el software AWS IoT Greengrass principal con aprovisionamiento

manual. En primer lugar, elimine el software principal de la versión 1 del dispositivo y, a continuación, reutilice el componente y el certificado del AWS IoT dispositivo principal de la versión 1 y actualice las AWS IoT políticas del certificado para conceder los permisos que requiere el software de la versión 2.x.

Instale el software AWS IoT Greengrass Core v2.x en un dispositivo que ya ejecute la v1.x

Si instala el software AWS IoT Greengrass Core v2.x en un dispositivo que ya ejecuta el software AWS IoT Greengrass Core v1.x, tenga en cuenta lo siguiente:

- El nombre del dispositivo principal V2 AWS IoT debe ser único. No utilices el mismo nombre que el de tu dispositivo principal V1.
- Los puertos que utilice para la versión 2.x del software AWS IoT Greengrass Core deben ser diferentes de los puertos que utilice para la versión 1.x.
  - Configure el administrador de transmisiones V1 para que utilice un puerto distinto del 8088. Para obtener más información, consulte Configurar el administrador de transmisiones.
  - Configure el broker MQTT de la versión 1 para que utilice un puerto distinto del 8883. Para obtener más información, consulte Configurar el puerto MQTT para la mensajería local.
- AWS IoT Greengrass V2no ofrece la opción de cambiar el nombre del servicio del sistema Greengrass. Si ejecuta Greengrass como un servicio del sistema, debe realizar una de las siguientes acciones para evitar conflictos en los nombres de los servicios del sistema:
  - Cambie el nombre del servicio Greengrass a la versión 1.x antes de instalar la versión 2.x.
  - Instale el software AWS IoT Greengrass Core v2.x sin un servicio del sistema y, a continuación, configure manualmente el software como un servicio del sistema con un nombre distinto de. `greengrass`

  Para cambiar el nombre del servicio Greengrass a la versión 1.x

  1. Detenga el servicio AWS IoT Greengrass Core software v1.x.

     ```
     sudo systemctl stop greengrass
     ```

  2. Espere a que se detenga el servicio. El servicio puede tardar unos minutos en detenerse. Puede utilizar el `list-units` comando para comprobar si el servicio se ha detenido.

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

3. Deshabilite el servicio.

```
sudo systemctl disable greengrass
```

4. Cambie el nombre del servicio.

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

5. Vuelva a cargar el servicio e inícielo.

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

A continuación, puede utilizar las instrucciones de instalación AWS IoT Greengrass V2 para instalar el software en su dispositivo.

> **Tip**
>
> Para reutilizar la identidad de un dispositivo principal al migrarlo de la V1 a la V2, sigue las instrucciones para instalar el software AWS IoT Greengrass principal con aprovisionamiento manual. En primer lugar, elimine el software principal de la versión 1 del dispositivo y, a continuación, reutilice el componente y el certificado del AWS IoT dispositivo principal de la versión 1 y actualice las AWS IoT políticas del certificado para conceder los permisos que requiere el software de la versión 2.x.

Paso 2: Implemente AWS IoT Greengrass V2 los componentes en los dispositivos principales

Tras instalar el software AWS IoT Greengrass Core v2.x en su dispositivo, cree una implementación que incluya los siguientes recursos. Para implementar componentes en una flota de dispositivos similares, cree una implementación para un grupo de cosas que contenga esos dispositivos.

- Componentes de la función Lambda que creó a partir de las funciones de Lambda de la V1. Para obtener más información, consulte AWS LambdaFunciones de ejecución.
- Si usa suscripciones V1, el componente del router de suscripciones antiguo.
- Si usa el administrador de transmisiones, el componente del administrador de transmisiones. Para obtener más información, consulte Gestione los flujos de datos en los dispositivos principales de Greengrass.
- Si usa secretos locales, el componente administrador de secretos.
- Si utiliza conectores V1, los componentes del AWS conector proporcionados.
- Si usa contenedores Docker, el componente del administrador de aplicaciones de Docker. Para obtener más información, consulte Ejecute un contenedor Docker.
- Si utilizas la inferencia de aprendizaje automático, los componentes para el aprendizaje automático son compatibles. Para obtener más información, consulte Cómo realizar la inferencia de machine learning.
- Si utiliza dispositivos conectados, los componentes del dispositivo cliente son compatibles. También debe habilitar la compatibilidad con los dispositivos cliente y asociar los dispositivos cliente a su dispositivo principal. Para obtener más información, consulte Interactúa con dispositivos IoT locales.
- Si usa sombras de dispositivos, el componente administrador de sombras. Para obtener más información, consulte Interactúa con las sombras de los dispositivos.
- Si carga registros de los dispositivos principales de Greengrass a Amazon CloudWatch Logs, el componente del administrador de registros. Para obtener más información, consulte Supervisar AWS IoT Greengrass registros.
- Si realiza la integración conAWS IoT SiteWise, siga las instrucciones para configurar el dispositivo principal V2 como AWS IoT SiteWise puerta de enlace. AWS IoT SiteWiseproporciona un script de instalación que despliega los AWS IoT SiteWise componentes automáticamente.
- Componentes definidos por el usuario que usted desarrolló para implementar una funcionalidad personalizada.

Para obtener información sobre la creación y la revisión de despliegues, consulte. Implemente AWS IoT Greengrass componentes en los dispositivos

Tutorial: Introducción a AWS IoT Greengrass V2

Puede completar este tutorial de introducción para conocer las características básicas deAWS IoT Greengrass V2. En este tutorial, aprenderá a hacer lo siguiente:

1. Instale y configure el software AWS IoT Greengrass Core en un dispositivo Linux, como una Raspberry Pi o un dispositivo Windows. Este dispositivo es un dispositivo central de Greengrass.
2. Desarrolle un componente Hello World en su dispositivo principal de Greengrass. Los componentes son módulos de software que se ejecutan en los dispositivos principales de Greengrass.
3. Cargue ese componente AWS IoT Greengrass V2 en. Nube de AWS
4. Implemente ese componente desde su dispositivo principal de Greengrass. Nube de AWS

> **Note**
>
> En este tutorial se describe cómo configurar un entorno de desarrollo y se exploran las características deAWS IoT Greengrass. Para obtener más información sobre cómo instalar y configurar los dispositivos de producción, consulte lo siguiente:
>
> - Configuración de los dispositivos AWS IoT Greengrass principales
> - Instalación del software AWS IoT Greengrass Core

Puedes dedicar de 20 a 30 minutos a este tutorial.

Temas

Requisitos previos

Para completar este tutorial de introducción, necesita lo siguiente:

- Una Cuenta de AWS. Si no dispone de una, consulte Paso 1: configurar una AWS cuenta.
- El uso de un Región de AWSque apoyeAWS IoT Greengrass V2. Para ver una lista completa de las regiones admitidas, consulte AWS IoT Greengrass V2Puntos de conexión y cuotas en la Referencia general de AWS.
- Un usuario AWS Identity and Access Management (IAM) con permisos de administrador.
- Un dispositivo para configurar como un dispositivo principal de Greengrass, como una Raspberry Pi con sistema operativo Raspberry Pi (anteriormente llamada Raspbian) o un dispositivo Windows 10. Debe tener permisos de administrador en este dispositivo o poder adquirir privilegios de administrador, por ejemplo, a través de. `sudo` Este dispositivo debe tener una conexión a Internet.

  También puede optar por utilizar un dispositivo diferente que cumpla con los requisitos para instalar y ejecutar el software AWS IoT Greengrass Core. Para obtener más información, consulte Plataformas compatibles y requisitos.

  Si su ordenador de desarrollo cumple estos requisitos, puede configurarlo como su dispositivo principal de Greengrass en este tutorial.
- Python 3.5 o posterior instalado para todos los usuarios del dispositivo y agregado a la variable de `PATH` entorno. En Windows, también debe tener instalado el Lanzador de Python para Windows para todos los usuarios.

  > **Important**
  >
  > En Windows, Python no se instala para todos los usuarios de forma predeterminada. Al instalar Python, debe personalizar la instalación para configurarla para que el software AWS IoT Greengrass principal ejecute scripts de Python. Por ejemplo, si utiliza el instalador gráfico de Python, haga lo siguiente:
  >
  > 1. Seleccione Instalar el lanzador para todos los usuarios (recomendado).
  > 2. Elija Customize installation.
  > 3. Elija Next.
  > 4. Seleccione Install for all users.
  > 5. Seleccione Add Python to environment variables.

6. Elija Instalar.

Para obtener más información, consulte Uso de Python en Windows en la documentación de Python 3.

- AWS Command Line Interface(AWS CLI) instalado y configurado con credenciales en su ordenador de desarrollo y en su dispositivo. Asegúrese de utilizar las mismas Región de AWS para configurarlo AWS CLI en su ordenador de desarrollo y en su dispositivo. Para usarlo AWS IoT Greengrass V2 conAWS CLI, debe tener una de las siguientes versiones o una posterior:
  - Versión AWS CLI V1 mínima: v1.18.197
  - Versión AWS CLI V2 mínima: v2.1.11

  Tip

  Puede ejecutar el siguiente comando para comprobar la versión de la AWS CLI que dispone.

  ```
  aws --version
  ```

  Para obtener más información, consulte Instalación, actualización y desinstalación AWS CLI y configuración de AWS CLI en la Guía del AWS Command Line Interface usuario.

  Note

  Si utiliza un dispositivo ARM de 32 bits, como una Raspberry Pi con un sistema operativo de 32 bits, instale la V1. AWS CLI AWS CLI La versión 2 no está disponible para dispositivos ARM de 32 bits. Para obtener más información, consulte Instalación, actualización y desinstalación de la AWS CLI versión 1.

Paso 1: configurar una AWS cuenta

Inscríbase en una Cuenta de AWS

Si no tiene uno Cuenta de AWS, complete los siguientes pasos para crearlo.

Para suscribirte a una Cuenta de AWS

1. Abra https://portal.aws.amazon.com/billing/signup.
2. Siga las instrucciones que se le indiquen.

   Parte del procedimiento de registro consiste en recibir una llamada telefónica e indicar un código de verificación en el teclado del teléfono.

   Cuando te registras en un Cuenta de AWS, Usuario raíz de la cuenta de AWSse crea un. El usuario raíz tendrá acceso a todos los Servicios de AWS y recursos de esa cuenta. Como práctica recomendada de seguridad, asigne acceso administrativo a un usuario y utilice únicamente el usuario raíz para realizar tareas que requieren acceso de usuario raíz.

AWS te envía un correo electrónico de confirmación una vez finalizado el proceso de registro. Puede ver la actividad de la cuenta y administrar la cuenta en cualquier momento entrando en https://aws.amazon.com/ y seleccionando Mi cuenta.

Creación de un usuario con acceso administrativo

Después de crear un usuario administrativo Cuenta de AWS, asegúrelo Usuario raíz de la cuenta de AWS AWS IAM Identity Center, habilite y cree un usuario administrativo para no usar el usuario root en las tareas diarias.

Proteja su Usuario raíz de la cuenta de AWS

1. Inicie sesión AWS Management Consolecomo propietario de la cuenta seleccionando el usuario root e introduciendo su dirección de Cuenta de AWS correo electrónico. En la siguiente página, escriba su contraseña.

   Para obtener ayuda para iniciar sesión con el usuario raíz, consulte Signing in as the root user en la Guía del usuario de AWS Sign-In .
2. Active la autenticación multifactor (MFA) para el usuario raíz.

   Para obtener instrucciones, consulte Habilitar un dispositivo MFA virtual para el usuario Cuenta de AWS raíz (consola) en la Guía del usuario de IAM.

Creación de un usuario con acceso administrativo

1. Activar IAM Identity Center.

Consulte las instrucciones en Activar AWS IAM Identity Center en la Guía del usuario de AWS IAM Identity Center .

2. En IAM Identity Center, conceda acceso administrativo a un usuario.

   Para ver un tutorial sobre su uso Directorio de IAM Identity Center como fuente de identidad, consulte Configurar el acceso de los usuarios con la configuración predeterminada Directorio de IAM Identity Center en la Guía del AWS IAM Identity Center usuario.

Iniciar sesión como usuario con acceso de administrador

- Para iniciar sesión con el usuario de IAM Identity Center, utilice la URL de inicio de sesión que se envió a la dirección de correo electrónico cuando creó el usuario de IAM Identity Center.

  Para obtener ayuda para iniciar sesión con un usuario del Centro de identidades de IAM, consulte Iniciar sesión en el portal de AWS acceso en la Guía del AWS Sign-In usuario.

Concesión de acceso a usuarios adicionales

1. En IAM Identity Center, cree un conjunto de permisos que siga la práctica recomendada de aplicar permisos de privilegios mínimos.

   Para conocer las instrucciones, consulte Create a permission set en la Guía del usuario de AWS IAM Identity Center .

2. Asigne usuarios a un grupo y, a continuación, asigne el acceso de inicio de sesión único al grupo.

   Para conocer las instrucciones, consulte Add groups en la Guía del usuario de AWS IAM Identity Center .

Paso 2: Configure su entorno

Siga los pasos de esta sección para configurar un dispositivo Linux o Windows para usarlo como dispositivo AWS IoT Greengrass principal.

Configura un dispositivo Linux (Raspberry Pi)

En estos pasos se supone que utilizas una Raspberry Pi con el sistema operativo Raspberry Pi. Si utilizas un dispositivo o sistema operativo diferente, consulta la documentación correspondiente a tu dispositivo.

Para configurar una Raspberry Pi para AWS IoT Greengrass V2

1. Activa SSH en tu Raspberry Pi para conectarte a ella de forma remota. Para obtener más información, consulte SSH (Secure shell) en la documentación de Raspberry Pi.
2. Busca la dirección IP de tu Raspberry Pi para conectarte a ella mediante SSH. Para ello, puedes ejecutar el siguiente comando en tu Raspberry Pi.

   ```
   hostname -I
   ```

3. Conéctate a tu Raspberry Pi con SSH.

   En tu ordenador de desarrollo, ejecuta el siguiente comando. Sustituya el nombre de *usuario* por el nombre del usuario para iniciar sesión y *pi-ip-address*sustitúyalo por la dirección IP que encontró en el paso anterior.

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

   > **Important**
   >
   > Si tu equipo de desarrollo usa una versión anterior de Windows, es posible que no tengas el `ssh` comando o que lo tengas `ssh` pero no puedas conectarte a tu Raspberry Pi. Para conectarte a tu Raspberry Pi, puedes instalar y configurar PuTTY, que es un cliente SSH de código abierto y gratuito. Consulta la documentación de PuTTY para conectarte a tu Raspberry Pi.

4. Instala el motor de ejecución de Java, que el software AWS IoT Greengrass Core necesita para ejecutarse. En tu Raspberry Pi, usa los siguientes comandos para instalar Java 11.

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

   Cuando se complete la instalación, ejecuta el siguiente comando para comprobar que Java se ejecuta en tu Raspberry Pi.

```
java -version
```

El comando imprime la versión de Java que se ejecuta en el dispositivo. El resultado puede tener un aspecto similar al del siguiente ejemplo.

```
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)
```

Consejo: Defina los parámetros del núcleo en una Raspberry Pi

Si su dispositivo es una Raspberry Pi, puede completar los siguientes pasos para ver y actualizar los parámetros del núcleo de Linux:

1. Abra el archivo `/boot/cmdline.txt`. Este archivo especifica los parámetros del núcleo de Linux que se aplicarán al arrancar la Raspberry Pi.

   Por ejemplo, en un sistema basado en Linux, puede ejecutar el siguiente comando para usar GNU nano y abrir el archivo.

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

2. Compruebe que el `/boot/cmdline.txt` archivo contiene los siguientes parámetros del núcleo. El `systemd.unified_cgroup_hierarchy=0` parámetro especifica el uso de cgroups v1 en lugar de cgroups v2.

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

   Si el `/boot/cmdline.txt` archivo no contiene estos parámetros o los contiene con valores diferentes, actualice el archivo para que contenga estos parámetros y valores.

3. Si ha actualizado el `/boot/cmdline.txt` archivo, reinicie la Raspberry Pi para aplicar los cambios.

   ```
   sudo reboot
   ```

Configure un dispositivo Linux (otro)

Para configurar un dispositivo Linux para AWS IoT Greengrass V2

1. Instale el motor de ejecución de Java, que el software AWS IoT Greengrass principal necesita para ejecutarse. Le recomendamos que utilice las versiones de soporte a largo plazo de Amazon Corretto u OpenJDK. Se requiere la versión 8 o superior. Los siguientes comandos muestran cómo instalar OpenJDK en su dispositivo.

   - Para distribuciones basadas en Debian o en Ubuntu:

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

   - Para distribuciones basadas en Red Hat:

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

   - En Amazon Linux 2:

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

   - Para Amazon Linux 2023:

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

   Cuando se complete la instalación, ejecute el siguiente comando para comprobar que Java se ejecuta en su dispositivo Linux.

   ```
   java -version
   ```

   El comando imprime la versión de Java que se ejecuta en el dispositivo. Por ejemplo, en una distribución basada en Debian, el resultado podría tener un aspecto similar al del siguiente ejemplo.

   ```
   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. (Opcional) Cree el usuario y el grupo predeterminados del sistema que ejecutan los componentes del dispositivo. También puede optar por permitir que el instalador del software AWS IoT Greengrass principal cree este usuario y grupo durante la instalación con el argumento del `--component-default-user` instalador. Para obtener más información, consulte Argumentos de instalación.

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

3. Compruebe que el usuario que ejecuta el software AWS IoT Greengrass principal (normalmente`root`) tiene permiso para ejecutar `sudo` con cualquier usuario y grupo.

   a. Ejecute el siguiente comando para abrir el `/etc/sudoers` archivo.

      ```
      sudo visudo
      ```

   b. Compruebe que el permiso del usuario es similar al del ejemplo siguiente.

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

4. (Opcional) Para ejecutar funciones Lambda en contenedores, debe habilitar cgroups v1 y debe habilitar y montar los cgroups de memoria y dispositivos. Si no planea ejecutar funciones Lambda en contenedores, puede omitir este paso.

   Para habilitar estas opciones de cgroups, arranque el dispositivo con los siguientes parámetros del núcleo de Linux.

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

   Para obtener información sobre cómo ver y configurar los parámetros del núcleo de su dispositivo, consulte la documentación del sistema operativo y del gestor de arranque. Siga las instrucciones para configurar permanentemente los parámetros del núcleo.

5. Instale todas las demás dependencias necesarias en su dispositivo tal y como se indica en Requisitos de los dispositivos la lista de requisitos de.

Configura un dispositivo Windows

Para configurar un dispositivo Windows para AWS IoT Greengrass V2

1. Instale el motor de ejecución de Java, que el software AWS IoT Greengrass principal necesita para ejecutarse. Le recomendamos que utilice las versiones de soporte a largo plazo de Amazon Corretto u OpenJDK. Se requiere la versión 8 o superior.
2. Compruebe si Java está disponible en la variable de sistema PATH y agréguelo si no lo está. La LocalSystem cuenta ejecuta el software AWS IoT Greengrass principal, por lo que debe agregar Java a la variable de sistema PATH en lugar de a la variable de usuario PATH de su usuario. Haga lo siguiente:
   a. Pulse la tecla Windows para abrir el menú de inicio.
   b. Escriba **`environment variables`** para buscar las opciones del sistema en el menú de inicio.
   c. En los resultados de la búsqueda del menú de inicio, seleccione Editar las variables de entorno del sistema para abrir la ventana de propiedades del sistema.
   d. Seleccione Variables de entorno... para abrir la ventana Variables de entorno.
   e. En Variables de sistema, seleccione Ruta y, a continuación, elija Editar. En la ventana Editar variables de entorno, puede ver cada ruta en una línea independiente.
   f. Compruebe si la ruta a la `bin` carpeta de la instalación de Java está presente. La ruta puede tener un aspecto similar al del siguiente ejemplo.

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

   g. Si la `bin` carpeta de la instalación de Java no aparece en Path, seleccione Nueva para añadirla y, a continuación, pulse Aceptar.
3. Abre la línea de comandos de Windows (`cmd.exe`) como administrador.
4. Cree el usuario predeterminado en la LocalSystem cuenta del dispositivo Windows. Sustituya la *`contraseña`* por una contraseña segura.

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

> **Tip**
>
> Según la configuración de Windows, es posible que la contraseña del usuario caduque en una fecha futura. Para garantizar que sus aplicaciones de Greengrass sigan funcionando, controle cuándo caduque la contraseña y actualícela antes de que caduque. También puede configurar la contraseña del usuario para que nunca caduque.
>
> - Para comprobar cuándo caducan un usuario y su contraseña, ejecuta el siguiente comando.
>
>   ```
>   net user ggc_user | findstr /C:expires
>   ```
>
> - Para configurar la contraseña de un usuario para que no caduque nunca, ejecute el siguiente comando.
>
>   ```
>   wmic UserAccount where "Name='ggc_user'" set PasswordExpires=False
>   ```
>
> - Si utilizas Windows 10 o una versión posterior, donde el `wmic` comando está obsoleto, ejecuta el siguiente PowerShell comando.
>
>   ```
>   Get-CimInstance -Query "SELECT * from Win32_UserAccount WHERE name =
>    'ggc_user'" | Set-CimInstance -Property @{PasswordExpires="False"}
>   ```

5. Descargue e instale la PsExecutilidad de Microsoft en el dispositivo.
6. Utilice la PsExec utilidad para almacenar el nombre de usuario y la contraseña del usuario predeterminado en la instancia de Credential Manager de la LocalSystem cuenta. Sustituya la *contraseña* por la contraseña del usuario que configuró anteriormente.

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

   Si se PsExec License Agreementabre, Acceptacepte la licencia y ejecute el comando.

> **Note**
>
> En los dispositivos Windows, la LocalSystem cuenta ejecuta el núcleo de Greengrass y debe usar la PsExec utilidad para almacenar la información de usuario predeterminada en la LocalSystem cuenta. El uso de la aplicación Credential Manager almacena esta

información en la cuenta de Windows del usuario que ha iniciado sesión actualmente, en lugar de en la LocalSystem cuenta.

Paso 3: instale el softwareAWS IoT Greengrass principal

Siga los pasos de esta sección para configurar su Raspberry Pi de como dispositivoAWS IoT Greengrass principal que puede usar para el desarrollo local. En esta sección, descarga y ejecuta un instalador que hace lo siguiente para configurar el softwareAWS IoT Greengrass principal para su dispositivo:

- Instala el componente Nucleus de Greengrass. El núcleo es un componente obligatorio y es el requisito mínimo para ejecutar el softwareAWS IoT Greengrass Core en un dispositivo. Para obtener más información, consulte Componente Greengrass de Información sobre rendimiento.
- Registra tu dispositivo como unaAWS IoT cosa y descarga un certificado digital que permite que tu dispositivo se conecte aAWS. Para obtener más información, consulte Autenticación y autorización de dispositivos en AWS IoT Greengrass.
- Añade laAWS IoT cosa del dispositivo a un grupo de cosas, que es un grupo o una flota deAWS IoT cosas. Los grupos de cosas le permiten gestionar flotas de dispositivos principales de Greengrass. Al implementar componentes de software en sus dispositivos, puede optar por implementarlos en dispositivos individuales o en grupos de dispositivos. Para obtener más información, consulte Administración de dispositivosAWS IoT en la Guía paraAWS IoT Core desarrolladores.
- Crea el rol de IAM que permite que el dispositivo principal de Greengrass interactúe conAWS los servicios. De forma predeterminada, esta función permite que el dispositivo interactúe con Amazon LogsAWS IoT y envíe registros a AmazonCloudWatch Logs. Para obtener más información, consulte Autorizar a los dispositivos principales a interactuar con AWS los servicios.
- Instala la interfaz de línea deAWS IoT Greengrass comandos (`greengrass-cli`), que puede utilizar para probar los componentes personalizados que desarrolle en el dispositivo principal. Para obtener más información, consulte Interfaz de línea de comandos Greengrass.

Instalación del software AWS IoT Greengrass principal (consola)

1. Inicie sesión en la consola de AWS IoT Greengrass.
2. En Comenzar a usar Greengrass, selecciona Configurar un dispositivo principal.

3. En el paso 1: Registrar un dispositivo principal de Greengrass, en Nombre del dispositivo principal, introduzca el nombre del dispositivo AWS IoT principal de Greengrass. Si el elemento no existe, el instalador lo crea.
4. En el paso 2: Añadir a un grupo de cosas para aplicar una implementación continua, en Grupo de AWS IoT cosas, elija el grupo de cosas al que quiere añadir su dispositivo principal.
   - Si selecciona Introducir un nombre de grupo nuevo y, a continuación, en Nombre de grupo de cosas, introduzca el nombre del nuevo grupo que desee crear. El instalador crea el nuevo grupo automáticamente.
   - Si selecciona Seleccionar un grupo existente y, en Nombre del grupo Thing, elija el grupo existente que desee usar.
   - Si selecciona Sin grupo, el instalador no añadirá el dispositivo principal a un grupo de cosas.
5. En el paso 3: Instalar el software Greengrass Core, complete los siguientes pasos.
   a. Elija el sistema operativo de su dispositivo principal: Linux o Windows.
   b. Proporcione sus AWS credenciales al dispositivo para que el instalador pueda aprovisionar los recursos de IAM AWS IoT y los de su dispositivo principal. Para aumentar la seguridad, le recomendamos que obtenga credenciales temporales para un rol de IAM que permita únicamente los permisos mínimos necesarios para el aprovisionamiento. Para obtener más información, consulte Política de IAM mínima para que el instalador aprovisione recursos.

      > **Note**
      >
      > El instalador no guarda ni almacena tus credenciales.

      En el dispositivo, realice una de las siguientes acciones para recuperar las credenciales y ponerlas a disposición del instalador del software AWS IoT Greengrass principal:

      - (Recomendado) Utilice credenciales temporales de AWS IAM Identity Center
        i. Proporcione el ID de la clave de acceso, la clave de acceso secreta y el token de sesión del Centro de identidades de IAM. Para obtener más información, consulte Actualización manual de credenciales en Obtener y actualizar credenciales temporales en la guía del usuario del IAM Identity Center.
        ii. Ejecute los siguientes comandos para proporcionar las credenciales al software AWS IoT Greengrass principal.

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="
```

- Utilice credenciales de seguridad temporales de un rol de IAM:

  i. Proporcione el ID de la clave de acceso, la clave de acceso secreta y el token de sesión de la función de IAM que asuma. Para obtener más información sobre cómo recuperar estas credenciales, consulte Solicitud de credenciales de seguridad temporales en la Guía del usuario de IAM.

  ii. Ejecute los siguientes comandos para proporcionar las credenciales al software AWS IoT Greengrass principal.

  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="
```

- Utilice credenciales de larga duración de un usuario de IAM:

  i. Proporcione el ID de la clave de acceso y la clave de acceso secreta de su usuario de IAM. Puede crear un usuario de IAM para el aprovisionamiento y eliminarlo más adelante. Para obtener información sobre la política de IAM que debe proporcionarse al usuario, consulte. Política de IAM mínima para que el instalador aprovisione recursos Para obtener más información sobre cómo recuperar credenciales de larga duración, consulte Administrar las claves de acceso para los usuarios de IAM en la Guía del usuario de IAM.

  ii. Ejecute los siguientes comandos para proporcionar las credenciales al software AWS IoT Greengrass principal.

  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. (Opcional) Si creó un usuario de IAM para aprovisionar su dispositivo Greengrass, elimínelo.

iv. (Opcional) Si utilizó el ID de clave de acceso y la clave de acceso secreta de un usuario de IAM existente, actualice las claves del usuario para que dejen de ser válidas. Para obtener más información, consulte Actualización de las claves de acceso en la guía del AWS Identity and Access Management usuario.

c. En Ejecutar el instalador, complete los pasos siguientes.

i. En Descargar el instalador, selecciona Copiar y ejecuta el comando copiado en tu dispositivo principal. Este comando descarga la última versión del software AWS IoT Greengrass principal y la descomprime en el dispositivo.

ii. En Ejecutar el instalador, selecciona Copiar y ejecuta el comando copiado en tu dispositivo principal. Este comando usa los nombres de AWS IoT cosas y grupos de cosas que especificó anteriormente para ejecutar el instalador del software AWS IoT Greengrass principal y configurar AWS los recursos del dispositivo principal.

Este comando también hace lo siguiente:

- Configure el software AWS IoT Greengrass principal como un servicio del sistema que se ejecute durante el arranque. En los dispositivos Linux, esto requiere el sistema de inicio Systemd.

> **Important**
>
> En los dispositivos principales de Windows, debe configurar el software AWS IoT Greengrass Core como un servicio del sistema.

- Implemente el componente AWS IoT Greengrass CLI, que es una herramienta de línea de comandos que le permite desarrollar componentes personalizados de Greengrass en el dispositivo principal.
- Especifique si desea utilizar el usuario del `ggc_user` sistema para ejecutar los componentes de software en el dispositivo principal. En los dispositivos Linux, este comando también especifica el uso del grupo `ggc_group` del sistema, y el instalador crea el usuario y el grupo del sistema automáticamente.

Al ejecutar este comando, debería ver los siguientes mensajes para indicar que el instalador se ha realizado correctamente.

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

> **Note**
>
> Si tiene un dispositivo Linux y este no tiene systemd, el instalador no configurará el software como un servicio del sistema y no verá el mensaje de éxito al configurar el núcleo como un servicio del sistema.

Instalación del software AWS IoT Greengrass principal (CLI)

Para instalar y configurar el software AWS IoT Greengrass principal

1. En su dispositivo principal de Greengrass, ejecute el siguiente comando para cambiar al directorio principal.

   Linux or Unix

   ```
   cd ~
   ```

   Windows Command Prompt (CMD)

   ```
   cd %USERPROFILE%
   ```

   PowerShell

   ```
   cd ~
   ```

2. En su dispositivo principal, descargue el software AWS IoT Greengrass Core a un archivo denominado`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
```

Al descargar este software, acepta el acuerdo de licencia del software de Greengrass Core.

3. Descomprime el software AWS IoT Greengrass Core en una carpeta de tu dispositivo. *GreengrassInstaller*Sustitúyalo por la carpeta que desee usar.

   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. Ejecute el siguiente comando para iniciar el instalador del software AWS IoT Greengrass Core. Este comando hace lo siguiente:

   - Cree los AWS recursos que el dispositivo principal necesita para funcionar.
   - Configure el software AWS IoT Greengrass Core como un servicio del sistema que se ejecute durante el arranque. En los dispositivos Linux, esto requiere el sistema de inicio Systemd.

> **Important**
>
> En los dispositivos principales de Windows, debe configurar el software AWS IoT Greengrass Core como un servicio del sistema.

- Implemente el componente AWS IoT Greengrass CLI, que es una herramienta de línea de comandos que le permite desarrollar componentes personalizados de Greengrass en el dispositivo principal.
- Especifique si desea utilizar el usuario del `ggc_user` sistema para ejecutar los componentes de software en el dispositivo principal. En los dispositivos Linux, este comando también especifica el uso del grupo `ggc_group` del sistema, y el instalador crea el usuario y el grupo del sistema automáticamente.

Sustituya los valores de los argumentos en el comando de la siguiente manera.

a. *`/greengrass/v2`*o *`C:\greengrass\v2`*: la ruta a la carpeta raíz que se utilizará para instalar el software AWS IoT Greengrass principal.

b. *`GreengrassInstaller`*. La ruta a la carpeta en la que desempaquetó el instalador del software AWS IoT Greengrass Core.

c. *`region`*. Región de AWSEn la que se buscan o crean los recursos.

d. *`MyGreengrassCore`*. El nombre del AWS IoT dispositivo principal de Greengrass. Si la cosa no existe, el instalador la crea. El instalador descarga los certificados para autenticarse como talAWS IoT. Para obtener más información, consulte Autenticación y autorización de dispositivos en AWS IoT Greengrass.

   > **Note**
   >
   > El nombre de la cosa no puede contener dos puntos (`:`).

e. *`MyGreengrassCoreGroup`*. El nombre del grupo de AWS IoT cosas de su dispositivo principal de Greengrass. Si el grupo de cosas no existe, el instalador lo crea y lo añade. Si el grupo de cosas existe y tiene una implementación activa, el dispositivo principal descarga y ejecuta el software que especifique la implementación.

> **Note**
>
> El nombre del grupo de cosas no puede contener dos puntos (`:`).

f. *`Greengrass V2 IoT ThingPolicy`*. El nombre de la AWS IoT política que permite a los dispositivos principales de Greengrass comunicarse con AWS IoT y. AWS IoT Greengrass Si la AWS IoT política no existe, el instalador crea una AWS IoT política permisiva con este nombre. Puede restringir los permisos de esta política según su caso de uso. Para obtener más información, consulte AWS IoTPolítica mínima para los dispositivos AWS IoT Greengrass V2 principales.

g. *`Greengrass V2 TokenExchangeRole`*. El nombre de la función de IAM que permite al dispositivo principal de Greengrass obtener AWS credenciales temporales. Si la función no existe, el instalador la crea y adjunta una política denominada. *`GreengrassV2TokenExchangeRole`*`Access` Para obtener más información, consulte Autorizar a los dispositivos principales a interactuar con AWS los servicios.

h. *`GreengrassCoreTokenExchangeRoleAlias`*. El alias de la función de IAM que permite al dispositivo principal de Greengrass obtener credenciales temporales más adelante. Si el alias del rol no existe, el instalador lo crea y lo dirige al rol de IAM que especifique. Para obtener más información, consulte Autorizar a los dispositivos principales a interactuar con AWS los servicios.

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

Si utiliza un AWS IoT Greengrass dispositivo con memoria limitada, puede controlar la cantidad de memoria que utiliza el software AWS IoT Greengrass Core. Para controlar la asignación de memoria, puede configurar las opciones de tamaño de pila de la JVM en el parámetro de `jvmOptions` configuración del componente core. Para obtener más información, consulte Controle la asignación de memoria con las opciones de JVM.

Al ejecutar este comando, deberían aparecer los siguientes mensajes para indicar que el instalador se ha realizado correctamente.

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

> **Note**
>
> Si tiene un dispositivo Linux y este no tiene systemd, el instalador no configurará el software como un servicio del sistema y no verá el mensaje de éxito al configurar el núcleo como un servicio del sistema.

(Opcional) Ejecute el software Greengrass (Linux)

Si instaló el software como un servicio del sistema, el instalador ejecutará el software automáticamente. De lo contrario, debe ejecutar el software. Para comprobar si el instalador configuró el software como un servicio del sistema, busque la siguiente línea en el resultado del instalador.

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

Si no ve este mensaje, haga lo siguiente para ejecutar el software:

1. Ejecute el siguiente comando para ejecutar el software.

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

   El software imprime el siguiente mensaje si se inicia correctamente.

   ```
   Launched Nucleus successfully.
   ```

2. Debe dejar abierta la consola de comandos actual para que el software AWS IoT Greengrass principal siga funcionando. Si utilizas SSH para conectarte al dispositivo principal, ejecuta el siguiente comando en tu ordenador de desarrollo para abrir una segunda sesión de SSH que podrás utilizar para ejecutar comandos adicionales en el dispositivo principal. Sustituya el

nombre de *usuario* por el nombre del usuario para iniciar sesión y *pi-ip-address*sustitúyalo por la dirección IP del dispositivo.

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

Para obtener más información sobre cómo interactuar con el servicio del sistema Greengrass, consulte. Configurar el núcleo de Greengrass como un servicio del sistema

Verifique la instalación de la CLI de Greengrass en el dispositivo

La implementación de la CLI de Greengrass puede tardar hasta un minuto. Ejecute el siguiente comando para comprobar el estado de la implementación. Sustituya *MyGreengrassCore* por el nombre de su dispositivo principal.

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

`coreDeviceExecutionStatus`Indica el estado de la implementación en el dispositivo principal. Cuando el estado sea`SUCCEEDED`, ejecute el siguiente comando para comprobar que la CLI de Greengrass está instalada y se ejecuta. */greengrass/v2*Sustitúyala por la ruta a la carpeta raíz.

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
```

El comando genera información de ayuda para la CLI de Greengrass. Si `greengrass-cli` no lo encuentra, es posible que la implementación no haya podido instalar la CLI de Greengrass. Para obtener más información, consulte Solución de problemas AWS IoT Greengrass V2.

También puede ejecutar el siguiente comando para implementar manualmente la AWS IoT Greengrass CLI en su dispositivo.

- Sustituya la *región* por la Región de AWS que utilice. Asegúrese de utilizar la misma Región de AWS que utilizó para configurar AWS CLI el dispositivo.
- Sustituye el *identificador de la cuenta* por tu Cuenta de AWS identificador.
- Sustituya *MyGreengrassCore* por el nombre de su dispositivo principal.

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**
>
> Puede añadir */greengrass/v2*/bin (Linux) o *C:\greengrass\v2*\bin (Windows) a la variable de PATH entorno para que se ejecute `greengrass-cli` sin su ruta absoluta.

El softwareAWS IoT Greengrass principal y las herramientas de desarrollo local se ejecutan en su dispositivo. A continuación, puede desarrollar unAWS IoT Greengrass componente Hello World en su dispositivo.

Paso 4: Desarrolle y pruebe un componente en su dispositivo

Un componente es un módulo de software que se ejecuta en los dispositivos AWS IoT Greengrass principales. Los componentes le permiten crear y administrar aplicaciones complejas como bloques de construcción discretos que puede reutilizar de un dispositivo principal de Greengrass a otro. Cada componente se compone de una receta y artefactos.

- Recetas

  Cada componente contiene un archivo de recetas, que define sus metadatos. La receta también especifica los parámetros de configuración del componente, las dependencias de los componentes, el ciclo de vida y la compatibilidad de la plataforma. El ciclo de vida del componente define los comandos que instalan, ejecutan y apagan el componente. Para obtener más información, consulte AWS IoT Greengrass referencia de recetas de componentes.

  Puede definir recetas en formato JSON o YAML.

- Artefactos

  Los componentes pueden tener cualquier número de artefactos, que son componentes binarios. Los artefactos pueden incluir scripts, código compilado, recursos estáticos y cualquier otro archivo que consuma un componente. Los componentes también pueden consumir artefactos de las dependencias de los componentes.

Con AWS IoT Greengrass, puede usar la CLI de Greengrass para desarrollar y probar componentes localmente en un dispositivo central de Greengrass sin interactuar con la nube. AWS Cuando complete su componente local, podrá usar la receta y los artefactos del componente para crear ese componente en el AWS IoT Greengrass servicio en la AWS nube y, a continuación, implementarlo en todos sus dispositivos principales de Greengrass. Para obtener más información sobre los componentes, consulte Desarrolle AWS IoT Greengrass componentes.

En esta sección, aprenderá a crear y ejecutar un componente básico de Hello World de forma local en su dispositivo principal.

Para desarrollar un componente de Hello World en tu dispositivo

1. Cree una carpeta para sus componentes con subcarpetas para recetas y artefactos. Ejecute los siguientes comandos en su dispositivo principal de Greengrass para crear estas carpetas

y cambiarlas a la carpeta de componentes. Sustituya *~/greengrassv2* o *%USERPROFILE%\greengrassv2* por la ruta a la carpeta que se utilizará para el desarrollo local.

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. Utilice un editor de texto para crear un archivo de recetas que defina los metadatos, los parámetros, las dependencias, el ciclo de vida y la capacidad de la plataforma de su componente. Incluya la versión del componente en el nombre del archivo de recetas para poder identificar qué receta refleja qué versión del componente. Puedes elegir el formato YAML o JSON para tu receta.

   Por ejemplo, en un sistema basado en Linux, puede ejecutar el siguiente comando para usar GNU nano para crear el archivo.

   JSON

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

   YAML

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

> **Note**
>
> AWS IoT Greengrass usa versiones semánticas para los componentes. Las versiones semánticas siguen un sistema de números de principal.secundario.parche. Por ejemplo, la versión `1.0.0` representa la primera versión principal de un componente. Para obtener más información, consulte la especificación de la versión semántica.

3. Pegue la siguiente receta en el archivo.

   JSON

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

```
  ]
}
```

YAML

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

La `ComponentConfiguration` sección de esta receta define un parámetro,`Message`, que por defecto es. `world` La `Manifests` sección define un manifiesto, que es un conjunto de instrucciones y artefactos del ciclo de vida de una plataforma. Puede definir varios manifiestos para especificar diferentes instrucciones de instalación para distintas plataformas, por ejemplo. En el manifiesto, la `Lifecycle` sección indica al dispositivo principal de Greengrass que ejecute el script Hello World con `Message` el valor del parámetro como argumento.

4. Ejecute el siguiente comando para crear una carpeta para los artefactos componentes.

   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**
>
> Debe utilizar el siguiente formato para la ruta de la carpeta de artefactos. Incluya el nombre y la versión del componente que especifique en la receta.
>
> ```
> artifacts/componentName/componentVersion/
> ```

5. Utilice un editor de texto para crear un archivo de artefactos de script de Python para su componente Hello World.

   Por ejemplo, en un sistema basado en Linux, puede ejecutar el siguiente comando para usar GNU nano para crear el archivo.

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

   Copie y pegue el siguiente script de Python en el archivo.

   ```
   import sys

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

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

6. Utilice la AWS IoT Greengrass CLI local para gestionar los componentes de su dispositivo principal de Greengrass.

   Ejecute el siguiente comando para implementar el componente en el AWS IoT Greengrass núcleo. Sustituya */greengrass/v2* o *C:\greengrass\v2* por la carpeta AWS IoT

Greengrass V2 raíz y sustituya *~/greengrassv2 o %USERPROFILE%\ greengrassv2* por la carpeta de desarrollo de componentes.

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"
```

Este comando añade el componente que utiliza la receta en `recipes` y el script de Python en`artifacts`. La `--merge` opción añade o actualiza el componente y la versión que especifique.

7. El software AWS IoT Greengrass Core guarda la salida estándar del proceso del componente en los archivos de registro de la `logs` carpeta. Ejecute el siguiente comando para comprobar que el componente Hello World se ejecuta e imprime los mensajes.

   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
   ```

El `type` comando escribe el contenido del archivo en la terminal. Ejecute este comando varias veces para observar los cambios en el archivo.

PowerShell

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

Debería ver mensajes similares a los del siguiente ejemplo.

```
Hello, world!
```

> **Note**
>
> Si el archivo no existe, es posible que la implementación local aún no esté completa. Si el archivo no existe en 15 segundos, es probable que se haya producido un error en la implementación. Esto puede ocurrir si la receta no es válida, por ejemplo. Ejecute el siguiente comando para ver el archivo de registro AWS IoT Greengrass principal. Este archivo incluye registros del servicio de despliegue del dispositivo principal de Greengrass.
>
> Linux or Unix
>
> ```
> sudo tail -f /greengrass/v2/logs/greengrass.log
> ```
>
> Windows Command Prompt (CMD)
>
> ```
> type C:\greengrass\v2\logs\greengrass.log
> ```
>
> El `type` comando escribe el contenido del archivo en la terminal. Ejecute este comando varias veces para observar los cambios en el archivo.
>
> PowerShell
>
> ```
> gc C:\greengrass\v2\logs\greengrass.log -Tail 10 -Wait
> ```

8. Modifique el componente local para iterar y probar el código. Ábrelo `hello_world.py` en un editor de texto y añade el siguiente código en la línea 4 para editar el mensaje que registra el AWS IoT Greengrass núcleo.

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

   El `hello_world.py` script debería tener ahora el siguiente contenido.

```
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. Ejecute el siguiente comando para actualizar el componente con los cambios.

   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"
```

Este comando actualiza el `com.example.HelloWorld` componente con el último artefacto de Hello World.

10. Ejecute el siguiente comando para reiniciar el componente. Al reiniciar un componente, el dispositivo principal utiliza los cambios más recientes.

    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. Vuelva a comprobar el registro para comprobar que el componente Hello World imprime el nuevo mensaje.

    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
    ```

    El `type` comando escribe el contenido del archivo en la terminal. Ejecute este comando varias veces para observar los cambios en el archivo.

    PowerShell

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

Debería ver mensajes similares a los del siguiente ejemplo.

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

12. Puede actualizar los parámetros de configuración del componente para probar diferentes configuraciones. Al implementar un componente, puede especificar una actualización de la configuración, que defina cómo modificar la configuración del componente en el dispositivo principal. Puede especificar los valores de configuración que desea restablecer a los valores predeterminados y los nuevos valores de configuración que se van a combinar en el dispositivo principal. Para obtener más información, consulte Actualizar las configuraciones de los componentes.

    Haga lo siguiente:

    a. Utilice un editor de texto para crear un archivo llamado `hello-world-config-update.json` a contener la actualización de la configuración

       Por ejemplo, en un sistema basado en Linux, puede ejecutar el siguiente comando para usar GNU nano para crear el archivo.

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

    b. Copia y pega el siguiente objeto JSON en el archivo. Este objeto JSON define una actualización de configuración que fusiona el valor `friend` con el `Message` parámetro para actualizar su valor. Esta actualización de configuración no especifica ningún valor que se vaya a restablecer. No es necesario restablecer el `Message` parámetro porque la actualización de combinación reemplaza el valor existente.

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

    c. Ejecute el siguiente comando para implementar la actualización de configuración en el componente Hello World.

Linux or Unix

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

Windows Command Prompt (CMD)

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

PowerShell

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

d. Vuelva a comprobar el registro para comprobar que el componente Hello World genera el nuevo mensaje.

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
```

El `type` comando escribe el contenido del archivo en la terminal. Ejecute este comando varias veces para observar los cambios en el archivo.

PowerShell

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

Debería ver mensajes similares a los del siguiente ejemplo.

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

13. Cuando termines de probar el componente, retíralo del dispositivo principal. Ejecute el siguiente comando de la .

    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**
    >
    > Este paso es necesario para volver a implementar el componente en el dispositivo principal después de cargarlo en él AWS IoT Greengrass. De lo contrario, se produce un error en la implementación y se produce un error de compatibilidad de versiones porque la implementación local especifica una versión diferente del componente.

    Ejecute el siguiente comando y compruebe que el `com.example.HelloWorld` componente no aparece en la lista de componentes del dispositivo.

    Linux or Unix

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

Windows Command Prompt (CMD)

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

PowerShell

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

Su componente Hello World está completo y ahora puede subirlo al servicio AWS IoT Greengrass en la nube. A continuación, puede implementar el componente en los dispositivos principales de Greengrass.

Paso 5: Cree su componente en el AWS IoT Greengrass servicio

Cuando termine de desarrollar un componente en su dispositivo principal, podrá cargarlo en el AWS IoT Greengrass servicio del Nube de AWS. También puede crear el componente directamente en la AWS IoT Greengrass consola. AWS IoT Greengrass proporciona un servicio de administración de componentes que aloja sus componentes para que pueda desplegarlos en dispositivos individuales o en flotas de dispositivos. Para cargar un componente al AWS IoT Greengrass servicio, siga estos pasos:

- Cargue los artefactos de los componentes a un depósito de S3.
- Añada el URI de Amazon Simple Storage Service (Amazon S3) de cada artefacto a la receta del componente.
- Cree un componente a AWS IoT Greengrass partir de la receta del componente.

En esta sección, debe completar estos pasos en su dispositivo principal de Greengrass para cargar su componente Hello World en el AWS IoT Greengrass servicio.

Cree su componente en AWS IoT Greengrass (consola)

1. Utilice un depósito de S3 en su AWS cuenta para alojar los artefactos de los AWS IoT Greengrass componentes. Al implementar el componente en un dispositivo principal, el dispositivo descarga los artefactos del componente del depósito.

   Puede usar un depósito de S3 existente o puede crear uno nuevo.

a. En la consola Amazon S3, en Buckets, selecciona Create bucket.

b. En el nombre del bucket, introduzca un nombre de bucket único. Por ejemplo, puede utilizar **`greengrass-component-artifacts-`*`region`*`-`*`123456789012`***. Sustituya *`123456789012`* por el ID de su AWS cuenta y *`la región`* por el Región de AWS que utilizó en este tutorial.

c. Para AWS la región, selecciona la AWS región que utilizas para este tutorial.

d. Elija Crear bucket.

e. En Buckets, elige el bucket que has creado y carga el `hello_world.py` script en la `artifacts/com.example.HelloWorld/1.0.0` carpeta del bucket. Para obtener información sobre cómo cargar objetos en buckets de S3, consulte Carga de objetos en la Guía del usuario de Amazon Simple Storage Service.

f. Copia el URI de S3 del `hello_world.py` objeto en el bucket de S3. Este URI debería tener un aspecto similar al del siguiente ejemplo. Sustituya DOC-EXAMPLE-BUCKET por el nombre del depósito S3.

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

2. Permita que el dispositivo principal acceda a los artefactos de los componentes del depósito de S3.

   Cada dispositivo principal tiene una función de IAM del dispositivo principal que le permite interactuar con los registros AWS IoT y enviarlos a la AWS nube. Esta función de dispositivo no permite el acceso a los depósitos de S3 de forma predeterminada, por lo que debe crear y adjuntar una política que permita al dispositivo principal recuperar los artefactos de los componentes del depósito de S3.

   Si la función de tu dispositivo ya te permite acceder al bucket de S3, puedes saltarte este paso. De lo contrario, cree una política de IAM que permita el acceso y asójela al rol, de la siguiente manera:

   a. En el menú de navegación de la consola de IAM, seleccione Políticas y, a continuación, elija Crear política.

   b. En la pestaña JSON, reemplace el contenido del marcador de posición por la política siguiente. Sustituya DOC-EXAMPLE-BUCKET por el nombre del depósito de S3 que contiene los artefactos componentes para su descarga en el dispositivo principal.

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

c. Elija Siguiente.

d. En la sección de detalles de la política, escriba Nombre. **`MyGreengrassV2ComponentArtifactPolicy`**

e. Elija Crear política.

f. En el menú de navegación de la consola de IAM, seleccione Función y, a continuación, elija el nombre de la función para el dispositivo principal. Especificó este nombre de función al instalar el software AWS IoT Greengrass principal. Si no especificó ningún nombre, el nombre predeterminado es`GreengrassV2TokenExchangeRole`.

g. En Permisos, selecciona Añadir permisos y, a continuación, selecciona Adjuntar políticas.

h. En la página Añadir permisos, selecciona la casilla de verificación situada junto a la `MyGreengrassV2ComponentArtifactPolicy` política que has creado y, a continuación, selecciona Añadir permisos.

3. Utilice la receta del componente para crear un componente en la AWS IoT Greengrass consola.

a. En el menú de navegación de la AWS IoT Greengrass consola, elija Componentes y, a continuación, elija Crear componente.

b. En Información del componente, selecciona Introducir receta como JSON. La receta del marcador de posición debería tener un aspecto similar al del siguiente ejemplo.

```
{
  "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. Sustituya el URI del marcador de posición de cada `Artifacts` sección por el URI S3 de su `hello_world.py` objeto.

d. Seleccione Crear componente.

e. En com.example. HelloWorlden la página del componente, compruebe que el estado del componente es desplegable.

Cree su componente en AWS IoT Greengrass ()AWS CLI

Para cargar su componente Hello World

1. Utilice un depósito de S3 en el suyo Cuenta de AWS para alojar los artefactos de los AWS IoT Greengrass componentes. Al implementar el componente en un dispositivo principal, el dispositivo descarga los artefactos del componente del depósito.

   Puede usar un bucket de S3 existente o ejecutar el siguiente comando para crear un bucket. Este comando crea un depósito con su Cuenta de AWS ID y Región de AWS forma un nombre de depósito único. Sustituya *123456789012* por su Cuenta de AWS ID y *la región* por la Región de AWS que utilizó en este tutorial.

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

   El comando genera la siguiente información si la solicitud se realiza correctamente.

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

2. Permita que el dispositivo principal acceda a los artefactos de los componentes del depósito S3.

   Cada dispositivo principal tiene una función de IAM del dispositivo principal que le permite interactuar con los registros AWS IoT y enviarlos. Nube de AWS Esta función de dispositivo no permite el acceso a los depósitos de S3 de forma predeterminada, por lo que debe crear y adjuntar una política que permita al dispositivo principal recuperar los artefactos de los componentes del depósito de S3.

   Si la función principal del dispositivo ya permite el acceso al bucket de S3, puedes saltarte este paso. De lo contrario, cree una política de IAM que permita el acceso y asójela al rol, de la siguiente manera:

   a. Cree un archivo llamado `component-artifact-policy.json` y copie el siguiente JSON en el archivo. Esta política permite el acceso a todos los archivos de un bucket de S3. Sustituya DOC-EXAMPLE-BUCKET por el nombre del depósito de S3.

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

b. Ejecute el siguiente comando para crear la política a partir del documento de política incluido en. `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
```

Copie la política Amazon Resource Name (ARN) de los metadatos de la política en la salida. Utilice este ARN para adjuntar esta política a la función de dispositivo principal en el siguiente paso.

c. Ejecute el siguiente comando para adjuntar la política a la función de dispositivo principal. Sustituya *GreengrassV2 TokenExchangeRole* por el nombre de la función del

dispositivo principal. Especificó este nombre de función al instalar el software AWS IoT Greengrass Core. Sustituya el ARN de la política por el ARN del paso anterior.

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
```

Si el comando no tiene salida, se ha realizado correctamente. El dispositivo principal ahora puede acceder a los artefactos que cargue en este depósito de S3.

3. Cargue el artefacto del script de Python Hello World en el bucket de S3.

   Ejecute el siguiente comando para cargar el script en la misma ruta del depósito en el que se encuentra el script en su AWS IoT Greengrass núcleo. Sustituya DOC-EXAMPLE-BUCKET por el nombre del depósito de 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
```

El comando genera una línea que comienza con `upload:` si la solicitud se realiza correctamente.

4. Añada el URI de Amazon S3 del artefacto a la receta del componente.

   El URI de Amazon S3 está compuesto por el nombre del bucket y la ruta al objeto artefacto del bucket. El URI de Amazon S3 de su artefacto de script es el URI en el que cargó el artefacto en el paso anterior. Este URI debería tener un aspecto similar al del siguiente ejemplo. Sustituya DOC-EXAMPLE-BUCKET por el nombre del depósito S3.

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

   Para añadir el artefacto a la receta, añada una lista `Artifacts` que contenga una estructura con la URI de Amazon S3.

   JSON

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

   Abra el archivo de la receta en un editor de texto.

Por ejemplo, en un sistema basado en Linux, puede ejecutar el siguiente comando para usar GNU nano para crear el archivo.

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

Agrega el artefacto a la receta. El archivo de la receta debe tener un aspecto similar al del siguiente ejemplo.

```
{
  "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
```

Abre el archivo de recetas en un editor de texto.

Por ejemplo, en un sistema basado en Linux, puede ejecutar el siguiente comando para usar GNU nano para crear el archivo.

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

Agrega el artefacto a la receta. El archivo de la receta debe tener un aspecto similar al del siguiente ejemplo.

```
---
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. Cree un recurso componente a AWS IoT Greengrass partir de la receta. Ejecute el siguiente comando para crear el componente a partir de la receta, que se proporciona como un archivo binario.

   JSON

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

   YAML

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

   La respuesta es similar a la del siguiente ejemplo si la solicitud se realiza correctamente.

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

```
  }
}
```

`arn`Copie el elemento del resultado para comprobar el estado del componente en el siguiente paso.

> **Note**
>
> También puedes ver tu componente Hello World en la AWS IoT Greengrass consola, en la página de componentes.

6. Compruebe que el componente se crea y está listo para su implementación. Al crear un componente, su estado es`REQUESTED`. A continuación, AWS IoT Greengrass valida que el componente sea desplegable. Puede ejecutar el siguiente comando para consultar el estado del componente y comprobar que el componente se puede implementar. Sustituya el `arn` por el ARN del paso anterior.

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

Si el componente se valida, la respuesta indica que el estado del componente es. `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"
    }
```

```
  ]
}
```

Su componente Hello World ya está disponible en AWS IoT Greengrass. Puede volver a implementarlo en este dispositivo principal de Greengrass o en otros dispositivos principales.

Paso 6: Implemente su componente

ConAWS IoT Greengrass, puede implementar componentes en dispositivos individuales o grupos de dispositivos. Al implementar un componente, AWS IoT Greengrass instala y ejecuta el software de ese componente en cada dispositivo de destino. Usted especifica los componentes que se van a implementar y la actualización de configuración que se va a implementar para cada componente. También puede controlar cómo se despliega la implementación en los dispositivos a los que se dirige la implementación. Para obtener más información, consulte Implemente AWS IoT Greengrass componentes en los dispositivos.

En esta sección, vuelve a implementar el componente Hello World en su dispositivo principal de Greengrass.

Implemente su componente (consola)

1. En el menú de navegación de la AWS IoT Greengrassconsola, elija Componentes.
2. En la página Componentes, en la pestaña Mis componentes, elija com.example.HelloWorld.
3. En la página com.example.HelloWorld, elija Implementar.
4. En Añadir a la implementación, elija Crear nueva implementación y, a continuación, elija Siguiente.
5. En la página Especificar detalles, haga lo siguiente:
   a. En el cuadro Name (Nombre), introduzca **`Deployment for MyGreengrassCore`**.
   b. En Destino de despliegue, elige Dispositivo principal y el AWS IoT nombre del dispositivo principal. El valor predeterminado de este tutorial es *`MyGreengrassCore`*.
   c. Elija Siguiente.
6. En la página Seleccionar componentes, en Mis componentes, compruebe que el com.example.HelloWorldcomponente esté seleccionado y elija Siguiente.
7. En la página Configurar componentes com.example.HelloWorld, elija y haga lo siguiente:

a. Seleccione Configurar componente.

b. En Actualización de la configuración, en Configuración para fusionar, introduzca la siguiente configuración.

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

Esta actualización de configuración establece el `Message` parámetro Hello World en el dispositivo de esta implementación. `universe`

c. Seleccione Confirmar.

d. Elija Siguiente.

8. En la página Configurar ajustes avanzados, mantenga los ajustes de configuración predeterminados y seleccione Siguiente.

9. En la página Revisar, elija Implementar.

10. Compruebe que la implementación se complete correctamente. La implementación puede tardar varios minutos en completarse. Consulte el registro de Hello World para comprobar el cambio. Ejecute el siguiente comando en su dispositivo principal de Greengrass.

Linux or Unix

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

Windows Command Prompt (CMD)

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

PowerShell

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

Debería ver mensajes similares a los del siguiente ejemplo.

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

> **Note**
>
> Si los mensajes de registro no cambian, la implementación ha fallado o no ha llegado al dispositivo principal. Esto puede ocurrir si el dispositivo principal no está conectado a Internet o no tiene permisos para recuperar artefactos del bucket de S3. Ejecuta el siguiente comando en tu dispositivo principal para ver el archivo de registro del software AWS IoT Greengrass principal. Este archivo incluye registros del servicio de despliegue del dispositivo principal de Greengrass.
>
> Linux or Unix
>
> ```
> sudo tail -f /greengrass/v2/logs/greengrass.log
> ```
>
> Windows Command Prompt (CMD)
>
> ```
> type C:\greengrass\v2\\logs\\greengrass.log
> ```
>
> El `type` comando escribe el contenido del archivo en la terminal. Ejecute este comando varias veces para observar los cambios en el archivo.
>
> PowerShell
>
> ```
> gc C:\greengrass\v2\\logs\\greengrass.log -Tail 10 -Wait
> ```
>
> Para obtener más información, consulte Solución de problemas AWS IoT Greengrass V2.

Implemente su componente (AWS CLI)

Para implementar su componente Hello World

1. En su computadora de desarrollo, cree un archivo llamado `hello-world-deployment.json` y copie el siguiente JSON en el archivo. Este archivo define los componentes y las configuraciones que se van a implementar.

```
{
```

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

Este archivo de configuración especifica que se debe implementar la versión `1.0.0` del componente Hello World que desarrolló y publicó en el procedimiento anterior. `configurationUpdate`Especifica combinar la configuración del componente en una cadena codificada en JSON. Esta actualización de configuración establece el `Message` parámetro Hello World en `universe` el dispositivo de esta implementación.

2. Ejecute el siguiente comando para implementar el componente en su dispositivo principal de Greengrass. Puede desplegarlo en objetos, que son dispositivos individuales, o en grupos de objetos, que son grupos de dispositivos. *`MyGreengrassCore`*Sustitúyalo por el AWS IoT nombre del dispositivo principal.

   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
   ```

   El comando genera una respuesta similar a la del siguiente ejemplo.

```
{
  "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. Compruebe que la implementación se complete correctamente. La implementación puede tardar varios minutos en completarse. Consulte el registro de Hello World para comprobar el cambio. Ejecute el siguiente comando en su dispositivo principal de Greengrass.

   Linux or Unix

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

   Windows Command Prompt (CMD)

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

   PowerShell

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

   Debería ver mensajes similares a los del siguiente ejemplo.

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

   > **Note**
   >
   > Si los mensajes de registro no cambian, la implementación ha fallado o no ha llegado al dispositivo principal. Esto puede ocurrir si el dispositivo principal no está conectado a Internet o no tiene permisos para recuperar artefactos del bucket de S3. Ejecuta el siguiente comando en tu dispositivo principal para ver el archivo de registro del software AWS IoT Greengrass principal. Este archivo incluye registros del servicio de despliegue del dispositivo principal de Greengrass.

Linux or Unix

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

Windows Command Prompt (CMD)

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

El `type` comando escribe el contenido del archivo en la terminal. Ejecute este comando varias veces para observar los cambios en el archivo.

PowerShell

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

Para obtener más información, consulte Solución de problemas AWS IoT Greengrass V2.

Pasos siguientes

Has completado este tutorial. El softwareAWS IoT Greengrass principal y el componente Hello World se ejecutan en el dispositivo. Además, su componente Hello World está disponible en el servicio en laAWS IoT Greengrass nube para implementarlo en otros dispositivos. Para obtener más información acerca de los temas que se utilizan en este tutorial, visite lo siguiente:

- Crear AWS IoT Greengrass componentes
- Publique componentes para desplegarlos en sus dispositivos principales
- Implemente AWS IoT Greengrass componentes en los dispositivos

Configuración de los dispositivos AWS IoT Greengrass principales

Complete las tareas de esta sección para instalar, configurar y ejecutar el software AWS IoT Greengrass principal.

> **Note**
>
> En esta sección se describe la instalación y configuración avanzadas del software AWS IoT Greengrass principal. Si es la primera vez que lo AWS IoT Greengrass V2 utiliza, le recomendamos que complete primero el tutorial de introducción para configurar un dispositivo principal y explorar sus funciones. AWS IoT Greengrass

Plataformas compatibles y requisitos

Antes de empezar, asegúrate de cumplir los siguientes requisitos para instalar y ejecutar el software AWS IoT Greengrass Core.

> **Tip**
>
> Puede buscar los dispositivos para los que cumplan los requisitos AWS IoT Greengrass V2 en el catálogo de dispositivos AWS asociados.

Temas

Plataformas admitidas

AWS IoT Greengrass es compatible oficialmente con dispositivos que ejecutan las siguientes plataformas. Los dispositivos con plataformas no incluidas en esta lista pueden funcionar, pero AWS IoT Greengrass las pruebas solo se realizan en estas plataformas especificadas.

Linux

Arquitecturas:

- Armv7l
- Armv8 (AArch64)
- x86_64

Windows

Arquitecturas:

- x86_64

Versiones:

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

> **Note**
>
> Algunas AWS IoT Greengrass funciones no son compatibles actualmente con los dispositivos Windows. Para obtener más información, consulte Compatibilidad de funciones de Greengrass por sistema operativo y Consideraciones sobre las características de los dispositivos Windows.

Las plataformas Linux también se pueden ejecutar AWS IoT Greengrass V2 en un contenedor Docker. Para obtener más información, consulte Ejecute AWS IoT Greengrass el software principal en un contenedor de Docker.

Para crear un sistema operativo personalizado basado en Linux, puede usar la BitBake receta del proyecto. AWS IoT Greengrass V2`meta-aws` El `meta-aws` proyecto proporciona recetas que puede utilizar para desarrollar capacidades de software de AWS vanguardia en sistemas Linux integrados que se crean con OpenEmbeddedlos marcos de compilación del Proyecto Yocto. El Proyecto Yocto

es un proyecto de colaboración de código abierto que le ayuda a crear sistemas personalizados basados en Linux para aplicaciones integradas, independientemente de la arquitectura de hardware. La BitBake receta para AWS IoT Greengrass V2 instalar, configurar y ejecutar automáticamente el software Core en su dispositivo. AWS IoT Greengrass

Requisitos de los dispositivos

Los dispositivos deben cumplir los siguientes requisitos para instalar y ejecutar la versión 2.x del software AWS IoT Greengrass principal.

> **Note**
>
> Puede utilizarlo AWS IoT Device Tester AWS IoT Greengrass para comprobar que su dispositivo puede ejecutar el software AWS IoT Greengrass principal y comunicarse con el. Nube de AWS Para obtener más información, consulte Uso AWS IoT Device Tester para AWS IoT Greengrass V2.

Linux

- El uso de un Región de AWSque admita AWS IoT Greengrass V2. Para ver una lista completa de las regiones admitidas, consulte Puntos de conexión y cuotas de AWS IoT Greengrass V2 en la Referencia general de AWS.
- Un mínimo de 256 MB de espacio en disco disponible para el software AWS IoT Greengrass Core. Este requisito no incluye los componentes implementados en el dispositivo principal.
- Se asignó un mínimo de 96 MB de RAM al software AWS IoT Greengrass principal. Este requisito no incluye los componentes que se ejecutan en el dispositivo principal. Para obtener más información, consulte Controle la asignación de memoria con las opciones de JVM.
- Java Runtime Environment (JRE) versión 8 o superior. Java debe estar disponible en la variable de entorno PATH del dispositivo. Para utilizar Java para desarrollar componentes personalizados, debe instalar un kit de desarrollo de Java (JDK). Le recomendamos que utilice las versiones de soporte a largo plazo de Amazon Corretto u OpenJDK. Se requiere la versión 8 o superior.
- Biblioteca C de GNU (glibc), versión 2.25 o superior.
- Debe ejecutar el software AWS IoT Greengrass Core como usuario root. `sudo`Utilícelo, por ejemplo.

- El usuario root que ejecuta el software AWS IoT Greengrass principal, por ejemplo`root`, debe tener permiso para ejecutar `sudo` con cualquier usuario y grupo. El `/etc/sudoers` archivo debe conceder permiso a este usuario para que se ejecute `sudo` como otros grupos. El permiso de entrada del usuario `/etc/sudoers` debería tener el aspecto que se muestra en el siguiente ejemplo.

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

- El dispositivo principal debe poder realizar solicitudes salientes a un conjunto de puntos finales y puertos. Para obtener más información, consulte Permitir el tráfico del dispositivo a través de un proxy o firewall.
- El `/tmp` directorio debe montarse con `exec` permisos.
- Todos los siguientes comandos de shell:
  - `ps -ax -o pid,ppid`
  - `sudo`
  - `sh`
  - `kill`
  - `cp`
  - `chmod`
  - `rm`
  - `ln`
  - `echo`
  - `exit`
  - `id`
  - `uname`
  - `grep`
- Es posible que su dispositivo también requiera los siguientes comandos de shell opcionales:
  - `systemctl` (opcional). Este comando se utiliza para configurar el software AWS IoT Greengrass Core como un servicio del sistema.
  - (Opcional) `useraddgroupadd`, `yusermod`. Estos comandos se utilizan para configurar el `ggc_user` usuario y el grupo `ggc_group` del sistema.
  - `mkfifo` (opcional). Este comando se utiliza para ejecutar funciones Lambda como componentes.

- Para configurar los límites de recursos del sistema para los procesos de los componentes, el dispositivo debe ejecutar la versión 2.6.24 o posterior del kernel de Linux.
- Para ejecutar las funciones de Lambda, el dispositivo debe cumplir requisitos adicionales. Para obtener más información, consulte Requisitos de la función de Lambda.

Windows

- El uso de un dispositivo Región de AWSque admita AWS IoT Greengrass V2. Para ver una lista completa de las regiones admitidas, consulte Puntos de conexión y cuotas de AWS IoT Greengrass V2 en la Referencia general de AWS.
- Un mínimo de 256 MB de espacio en disco disponible para el software AWS IoT Greengrass Core. Este requisito no incluye los componentes implementados en el dispositivo principal.
- Se asignó un mínimo de 160 MB de RAM al software AWS IoT Greengrass principal. Este requisito no incluye los componentes que se ejecutan en el dispositivo principal. Para obtener más información, consulte Controle la asignación de memoria con las opciones de JVM.
- Java Runtime Environment (JRE) versión 8 o superior. Java debe estar disponible en la variable de sistema PATH del dispositivo. Para utilizar Java para desarrollar componentes personalizados, debe instalar un kit de desarrollo de Java (JDK). Le recomendamos que utilice las versiones de soporte a largo plazo de Amazon Corretto u OpenJDK. Se requiere la versión 8 o superior.

  > **Note**
  >
  > Para usar la versión 2.5.0 del núcleo de Greengrass, debe usar una versión de 64 bits del Java Runtime Environment (JRE). La versión 2.5.1 del núcleo de Greengrass admite JRE de 32 y 64 bits.

- El usuario que instala el software AWS IoT Greengrass Core debe ser administrador.
- Debe instalar el software AWS IoT Greengrass Core como un servicio del sistema. Especifique `--setup-system-service true` cuándo va a instalar el software.
- Cada usuario que ejecute los procesos componentes debe existir en la LocalSystem cuenta y el nombre y la contraseña del usuario deben estar en la instancia de Credential Manager de la LocalSystem cuenta. Puede configurar este usuario siguiendo las instrucciones para instalar el software AWS IoT Greengrass principal.

- El dispositivo principal debe poder realizar solicitudes salientes a un conjunto de puntos finales y puertos. Para obtener más información, consulte Permitir el tráfico del dispositivo a través de un proxy o firewall.

Requisitos de la función de Lambda

El dispositivo debe cumplir los siguientes requisitos para ejecutar las funciones Lambda:

- Un sistema operativo basado en Linux.
- El dispositivo debe tener el `mkfifo` comando shell.
- El dispositivo debe ejecutar las bibliotecas de lenguajes de programación que requiere una función Lambda. Debe instalar las bibliotecas necesarias en el dispositivo y añadirlas a la variable de `PATH` entorno. Greengrass admite todas las versiones compatibles con Lambda de los tiempos de ejecución de Python, Node.js y Java. Greengrass no aplica ninguna restricción adicional a las versiones de tiempo de ejecución de Lambda obsoletas. Para obtener más información sobre la AWS IoT Greengrass compatibilidad con los tiempos de ejecución de Lambda, consulte. AWS LambdaFunciones de ejecución
- Para ejecutar funciones Lambda en contenedores, el dispositivo debe cumplir los siguientes requisitos:
  - Kernel de Linux versión 4.4 o posterior.
  - El núcleo debe ser compatible con cgroups v1 y usted debe habilitar y montar los siguientes cgroups:
    - El grupo de memoria AWS IoT Greengrass para establecer el límite de memoria para las funciones Lambda en contenedores.
    - El grupo de dispositivos de Lambda en contenedores funciona para acceder a los dispositivos o volúmenes del sistema.

    El software AWS IoT Greengrass Core no es compatible con cgroups v2.

    Para cumplir con este requisito, arranque el dispositivo con los siguientes parámetros del núcleo de Linux.

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

> **Tip**
>
> En una Raspberry Pi, edite el `/boot/cmdline.txt` archivo para configurar los parámetros del núcleo del dispositivo.

- Debe habilitar las siguientes configuraciones del núcleo de Linux en el dispositivo:
  - Espacio de nombres:
    - `CONFIG_IPC_NS`
    - `CONFIG_UTS_NS`
    - `CONFIG_USER_NS`
    - `CONFIG_PID_NS`
  - Grupos de control:
    - `CONFIG_CGROUP_DEVICE`
    - `CONFIG_CGROUPS`
    - `CONFIG_MEMCG`
  - Otros:
    - `CONFIG_POSIX_MQUEUE`
    - `CONFIG_OVERLAY_FS`
    - `CONFIG_HAVE_ARCH_SECCOMP_FILTER`
    - `CONFIG_SECCOMP_FILTER`
    - `CONFIG_KEYS`
    - `CONFIG_SECCOMP`
    - `CONFIG_SHMEM`

> **Tip**
>
> Consulte la documentación de su distribución de Linux para obtener información sobre cómo verificar y configurar los parámetros del núcleo de Linux. También puede utilizar AWS IoT Device Tester for AWS IoT Greengrass para comprobar que el dispositivo cumple estos requisitos. Para obtener más información, consulte Uso AWS IoT Device Tester para AWS IoT Greengrass V2.

Consideraciones sobre las características de los dispositivos Windows

Algunas AWS IoT Greengrass funciones no son compatibles actualmente con los dispositivos Windows. Revisa las diferencias de funciones para confirmar si un dispositivo Windows cumple tus requisitos. Para obtener más información, consulte Compatibilidad de funciones de Greengrass por sistema operativo.

Configure un Cuenta de AWS

Si no tiene uno Cuenta de AWS, complete los siguientes pasos para crearlo.

Para suscribirte a una Cuenta de AWS

1. Abra https://portal.aws.amazon.com/billing/signup.
2. Siga las instrucciones que se le indiquen.

   Parte del procedimiento de registro consiste en recibir una llamada telefónica e indicar un código de verificación en el teclado del teléfono.

   Cuando te registras en un Cuenta de AWS, Usuario raíz de la cuenta de AWSse crea un. El usuario raíz tendrá acceso a todos los Servicios de AWS y recursos de esa cuenta. Como práctica recomendada de seguridad, asigne acceso administrativo a un usuario y utilice únicamente el usuario raíz para realizar tareas que requieren acceso de usuario raíz.

Para crear un usuario administrador, elija una de las siguientes opciones.

Elegir una forma de administrar el administrador	Para	Haga esto	También puede
En IAM Identity Center (recomendado)	Usar credenciales a corto plazo para acceder a AWS. Esto se ajusta a las prácticas recomendadas de seguridad. Para obtener información sobre las prácticas recomendadas, consulte Prácticas recomendadas de seguridad en IAM en la Guía del usuario de IAM.	Siga las instrucciones en Introducción en la Guía del usuario de AWS IAM Identity Center .	Configure el acceso programático configurando el AWS CLI que se utilizará AWS IAM Identity Center en la Guía del AWS Command Line Interface usuario.
En IAM (no recomendado)	Usar credenciales a largo plazo para acceder a AWS.	Siga las instrucciones en Creación del primer grupo de usuarios y usuario de administrador de IAM en la Guía del usuario de IAM.	Configurar el acceso programático mediante Administración de las claves de acceso de los usuarios de IAM en la Guía del usuario de IAM.

Instalación del software AWS IoT Greengrass Core

AWS IoT Greengrassse extiende AWS a los dispositivos periféricos para que puedan actuar en función de los datos que generan y, al mismo tiempo, los utilizan Nube de AWS para la administración, el análisis y el almacenamiento duradero. Instale el software AWS IoT Greengrass Core en los dispositivos periféricos para integrarlo con AWS IoT Greengrass y elNube de AWS.

> **Important**
>
> Antes de descargar e instalar el software AWS IoT Greengrass Core, compruebe que su dispositivo principal cumpla los requisitos para instalar y ejecutar el software AWS IoT Greengrass Core v2.0.

El software AWS IoT Greengrass Core incluye un instalador que configura su dispositivo como un dispositivo principal de Greengrass. Al ejecutar el instalador, puede configurar las opciones, como la carpeta raíz y la carpeta Región de AWS a utilizar. Puede elegir que el instalador cree los recursos necesarios AWS IoT y de IAM por usted. También puede optar por implementar herramientas de desarrollo local para configurar un dispositivo que utilice para el desarrollo de componentes personalizados.

El software AWS IoT Greengrass principal requiere lo siguiente AWS IoT y los recursos de IAM para conectarse Nube de AWS y funcionar:

- Un objeto de AWS IoT. Al registrar un dispositivo como una AWS IoT cosa, ese dispositivo puede utilizar un certificado digital para autenticarse. AWS Este certificado permite que el dispositivo se comunique con AWS IoT yAWS IoT Greengrass. Para obtener más información, consulte Autenticación y autorización de dispositivos en AWS IoT Greengrass.
- (Opcional) Un grupo de AWS IoT cosas. Los grupos de cosas se utilizan para gestionar las flotas de los dispositivos principales de Greengrass. Al implementar componentes de software en sus dispositivos, puede optar por implementarlos en dispositivos individuales o en grupos de dispositivos. Puede añadir un dispositivo a un grupo de cosas para implementar los componentes de software de ese grupo de cosas en el dispositivo. Para obtener más información, consulte Implemente AWS IoT Greengrass componentes en los dispositivos.
- Un rol de IAM. Los dispositivos principales de Greengrass utilizan el proveedor de AWS IoT Core credenciales para autorizar las llamadas a AWS los servicios con una función de IAM. Esta función permite que el dispositivo interactúe con Amazon LogsAWS IoT, envíe registros a Amazon

CloudWatch Logs y descargue artefactos de componentes personalizados de Amazon Simple Storage Service (Amazon S3). Para obtener más información, consulte Autorizar a los dispositivos principales a interactuar con AWS los servicios.

- Un alias de AWS IoT rol. Los dispositivos principales de Greengrass utilizan el alias del rol para identificar el rol de IAM que se va a utilizar. El alias del rol le permite cambiar el rol de IAM pero mantener la misma configuración del dispositivo. Para obtener más información, consulte Autorizar llamadas directas a AWS los servicios en la Guía para AWS IoT Coredesarrolladores.

Elija una de las siguientes opciones para instalar el software AWS IoT Greengrass principal en su dispositivo principal.

- Instalación rápida

  Elija esta opción para configurar un dispositivo principal de Greengrass en el menor número de pasos posible. El instalador crea los recursos AWS IoT de IAM necesarios para usted. Esta opción requiere que proporcione AWS credenciales al instalador para crear recursos en suCuenta de AWS.

  No puede usar esta opción para realizar la instalación detrás de un firewall o un proxy de red. Si tus dispositivos están protegidos por un firewall o un proxy de red, considera la posibilidad de instalarlos manualmente.

  Para obtener más información, consulte Instale el software AWS IoT Greengrass principal con aprovisionamiento automático de recursos.

- Instalación manual

  Elija esta opción para crear los AWS recursos necesarios manualmente o para instalarlos detrás de un firewall o un proxy de red. Al realizar una instalación manual, no necesita conceder permiso al instalador para crear recursos en la suyaCuenta de AWS, ya que crea los recursos necesarios AWS IoT y de IAM. También puedes configurar el dispositivo para que se conecte al puerto 443 o a través de un proxy de red. También puede configurar el software AWS IoT Greengrass Core para que utilice una clave privada y un certificado que se almacenen en un módulo de seguridad de hardware (HSM), un módulo de plataforma segura (TPM) u otro elemento criptográfico.

  Para obtener más información, consulte Instale el software AWS IoT Greengrass principal con aprovisionamiento manual de recursos.

- Instalación con aprovisionamiento de flota AWS IoT

Elija esta opción para crear los AWS recursos necesarios a partir de una plantilla de aprovisionamiento de AWS IoT flotas. Puede elegir esta opción para crear dispositivos similares en una flota o si fabrica dispositivos que sus clientes activen más adelante, como vehículos o dispositivos domésticos inteligentes. Los dispositivos utilizan certificados identificativos para autenticar y aprovisionar AWS recursos, incluido un certificado de cliente X.509 que el dispositivo utiliza Nube de AWS para conectarse al sistema operativo normal. Puede incrustar o mostrar los certificados de reclamación en el hardware del dispositivo durante la fabricación, y puede utilizar el mismo certificado de reclamación y la misma clave para aprovisionar varios dispositivos. También puede configurar los dispositivos para que se conecten en el puerto 443 o mediante un proxy de red.

Para obtener más información, consulte Instale el software AWS IoT Greengrass principal con aprovisionamiento AWS IoT de flota.

- Instalación con aprovisionamiento personalizado

  Elija esta opción para desarrollar una aplicación Java personalizada que aprovisione los AWS recursos necesarios. Puede elegir esta opción si crea sus propios certificados de cliente X.509 o si desea tener más control sobre el proceso de aprovisionamiento. AWS IoT Greengrassproporciona una interfaz que puede implementar para intercambiar información entre su aplicación de aprovisionamiento personalizada y el instalador del software AWS IoT Greengrass principal.

  Para obtener más información, consulte Instale el software AWS IoT Greengrass principal con aprovisionamiento de recursos personalizado.

AWS IoT Greengrass también proporciona entornos en contenedores que ejecutan el software de AWS IoT Greengrass Core. Puede usar un Dockerfile para ejecutarlo AWS IoT Greengrass en un contenedor Docker.

Temas

Instale el software AWS IoT Greengrass principal con aprovisionamiento automático de recursos

El software AWS IoT Greengrass Core incluye un instalador que configura su dispositivo como un dispositivo principal de Greengrass. Para configurar un dispositivo rápidamente, el instalador puede proporcionar la AWS IoT AWS IoT cosa, el grupo de cosas, la función de IAM y el alias de la AWS IoT función que el dispositivo principal necesita para funcionar. El instalador también puede implementar las herramientas de desarrollo locales en el dispositivo principal, de modo que usted pueda usar el dispositivo para desarrollar y probar componentes de software personalizados. El instalador necesita AWS credenciales para aprovisionar estos recursos y crear la implementación.

Si no puede proporcionar AWS las credenciales al dispositivo, puede aprovisionar los AWS recursos que el dispositivo principal necesita para funcionar. También puede implementar las herramientas de desarrollo en un dispositivo principal para usarlas como dispositivo de desarrollo. Esto le permite conceder menos permisos al dispositivo al ejecutar el instalador. Para obtener más información, consulte Instale el software AWS IoT Greengrass principal con aprovisionamiento manual de recursos.

> **Important**
>
> Antes de descargar el software AWS IoT Greengrass Core, compruebe que su dispositivo principal cumpla los requisitos para instalar y ejecutar el software AWS IoT Greengrass Core v2.0.

Temas

Configura el entorno del dispositivo

Siga los pasos de esta sección para configurar un dispositivo Linux o Windows para usarlo como dispositivo AWS IoT Greengrass principal.

Configura un dispositivo Linux

Para configurar un dispositivo Linux para AWS IoT Greengrass V2

1. Instale el motor de ejecución de Java, que el software AWS IoT Greengrass principal necesita para ejecutarse. Le recomendamos que utilice las versiones de soporte a largo plazo de Amazon Corretto u OpenJDK. Se requiere la versión 8 o superior. Los siguientes comandos le muestran cómo instalar OpenJDK en su dispositivo.

   - Para distribuciones basadas en Debian o en Ubuntu:

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

   - Para distribuciones basadas en Red Hat:

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

   - En Amazon Linux 2:

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

   - Para Amazon Linux 2023:

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

   Cuando se complete la instalación, ejecute el siguiente comando para comprobar que Java se ejecuta en su dispositivo Linux.

   ```
   java -version
   ```

   El comando imprime la versión de Java que se ejecuta en el dispositivo. Por ejemplo, en una distribución basada en Debian, el resultado podría tener un aspecto similar al del siguiente ejemplo.

   ```
   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. (Opcional) Cree el usuario y el grupo predeterminados del sistema que ejecutan los componentes del dispositivo. También puede optar por permitir que el instalador del software AWS IoT Greengrass principal cree este usuario y grupo durante la instalación con el argumento del `--component-default-user` instalador. Para obtener más información, consulte Argumentos de instalación.

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

3. Compruebe que el usuario que ejecuta el software AWS IoT Greengrass principal (normalmente`root`) tiene permiso para ejecutar `sudo` con cualquier usuario y grupo.

   a. Ejecute el siguiente comando para abrir el `/etc/sudoers` archivo.

      ```
      sudo visudo
      ```

   b. Compruebe que el permiso del usuario es similar al del ejemplo siguiente.

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

4. (Opcional) Para ejecutar funciones Lambda en contenedores, debe habilitar cgroups v1 y debe habilitar y montar los cgroups de memoria y dispositivos. Si no planea ejecutar funciones Lambda en contenedores, puede omitir este paso.

   Para habilitar estas opciones de cgroups, arranque el dispositivo con los siguientes parámetros del kernel de Linux.

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

   Para obtener información sobre cómo ver y configurar los parámetros del núcleo de su dispositivo, consulte la documentación del sistema operativo y del gestor de arranque. Siga las instrucciones para configurar permanentemente los parámetros del núcleo.

5. Instale todas las demás dependencias necesarias en su dispositivo tal y como se indica en Requisitos de los dispositivos la lista de requisitos de.

Configura un dispositivo Windows

> **Note**
>
> Esta función está disponible para la versión 2.5.0 y versiones posteriores del componente núcleo de Greengrass.

Para configurar un dispositivo Windows para AWS IoT Greengrass V2

1. Instale el motor de ejecución de Java, que el software AWS IoT Greengrass principal necesita para ejecutarse. Le recomendamos que utilice las versiones de soporte a largo plazo de Amazon Corretto u OpenJDK. Se requiere la versión 8 o superior.
2. Compruebe si Java está disponible en la variable de sistema PATH y agréguelo si no lo está. La LocalSystem cuenta ejecuta el software AWS IoT Greengrass principal, por lo que debe agregar Java a la variable de sistema PATH en lugar de a la variable de usuario PATH de su usuario. Haga lo siguiente:
   a. Pulse la tecla Windows para abrir el menú de inicio.
   b. Escriba **`environment variables`** para buscar las opciones del sistema en el menú de inicio.
   c. En los resultados de la búsqueda del menú de inicio, seleccione Editar las variables de entorno del sistema para abrir la ventana de propiedades del sistema.
   d. Seleccione Variables de entorno... para abrir la ventana Variables de entorno.
   e. En Variables de sistema, seleccione Ruta y, a continuación, elija Editar. En la ventana Editar variables de entorno, puede ver cada ruta en una línea independiente.
   f. Compruebe si la ruta a la `bin` carpeta de la instalación de Java está presente. La ruta puede tener un aspecto similar al del siguiente ejemplo.

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

   g. Si la `bin` carpeta de la instalación de Java no aparece en Path, seleccione Nueva para añadirla y, a continuación, pulse Aceptar.
3. Abre la línea de comandos de Windows (`cmd.exe`) como administrador.
4. Cree el usuario predeterminado en la LocalSystem cuenta del dispositivo Windows. Sustituya la *`contraseña`* por una contraseña segura.

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

> **Tip**
>
> Según la configuración de Windows, es posible que la contraseña del usuario caduque en una fecha futura. Para garantizar que sus aplicaciones de Greengrass sigan funcionando, controle cuándo caduque la contraseña y actualícela antes de que caduque. También puede configurar la contraseña del usuario para que nunca caduque.
>
> - Para comprobar cuándo caducan un usuario y su contraseña, ejecuta el siguiente comando.
>
>   ```
>   net user ggc_user | findstr /C:expires
>   ```
>
> - Para configurar la contraseña de un usuario para que no caduque nunca, ejecute el siguiente comando.
>
>   ```
>   wmic UserAccount where "Name='ggc_user'" set PasswordExpires=False
>   ```
>
> - Si utilizas Windows 10 o una versión posterior, donde el wmiccomando está obsoleto, ejecuta el siguiente PowerShell comando.
>
>   ```
>   Get-CimInstance -Query "SELECT * from Win32_UserAccount WHERE name =
>    'ggc_user'" | Set-CimInstance -Property @{PasswordExpires="False"}
>   ```

5. Descargue e instale la PsExecutilidad de Microsoft en el dispositivo.
6. Utilice la PsExec utilidad para almacenar el nombre de usuario y la contraseña del usuario predeterminado en la instancia de Credential Manager de la LocalSystem cuenta. Sustituya la *contraseña* por la contraseña del usuario que configuró anteriormente.

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

   Si se PsExec License Agreementabre, Acceptacepte la licencia y ejecute el comando.

> **Note**
>
> En los dispositivos Windows, la LocalSystem cuenta ejecuta el núcleo de Greengrass y debe usar la PsExec utilidad para almacenar la información de usuario predeterminada en la LocalSystem cuenta. El uso de la aplicación Credential Manager almacena esta información en la cuenta de Windows del usuario que ha iniciado sesión actualmente, en lugar de en la LocalSystem cuenta.

Proporcione AWS las credenciales al dispositivo

Proporcione AWS las credenciales del dispositivo para que el instalador pueda aprovisionar los AWS recursos necesarios. Para obtener más información sobre los permisos necesarios, consulte Política de IAM mínima para que el instalador aprovisione recursos.

Para proporcionar AWS credenciales al dispositivo

- Proporcione sus AWS credenciales al dispositivo para que el instalador pueda aprovisionar los recursos de IAM AWS IoT y los de su dispositivo principal. Para aumentar la seguridad, le recomendamos que obtenga credenciales temporales para un rol de IAM que permita únicamente los permisos mínimos necesarios para el aprovisionamiento. Para obtener más información, consulte Política de IAM mínima para que el instalador aprovisione recursos.

  > **Note**
  >
  > El instalador no guarda ni almacena tus credenciales.

  En el dispositivo, realice una de las siguientes acciones para recuperar las credenciales y ponerlas a disposición del instalador del software AWS IoT Greengrass principal:

  - (Recomendado) Utilice credenciales temporales de AWS IAM Identity Center

    a. Proporcione el ID de la clave de acceso, la clave de acceso secreta y el token de sesión del Centro de identidades de IAM. Para obtener más información, consulte Actualización manual de credenciales en Obtener y actualizar credenciales temporales en la guía del usuario del IAM Identity Center.

b. Ejecute los siguientes comandos para proporcionar las credenciales al software AWS IoT Greengrass principal.

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="
```

- Utilice credenciales de seguridad temporales de un rol de IAM:

a. Proporcione el ID de la clave de acceso, la clave de acceso secreta y el token de sesión de la función de IAM que asuma. Para obtener más información sobre cómo recuperar estas credenciales, consulte Solicitud de credenciales de seguridad temporales en la Guía del usuario de IAM.

b. Ejecute los siguientes comandos para proporcionar las credenciales al software AWS IoT Greengrass principal.

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="
```

- Utilice credenciales de larga duración de un usuario de IAM:

  a. Proporcione el ID de la clave de acceso y la clave de acceso secreta de su usuario de IAM. Puede crear un usuario de IAM para el aprovisionamiento y eliminarlo más adelante. Para obtener información sobre la política de IAM que debe proporcionarse al usuario, consulte. Política de IAM mínima para que el instalador aprovisione recursos Para obtener más información sobre cómo recuperar credenciales de larga duración, consulte Administrar las claves de acceso para los usuarios de IAM en la Guía del usuario de IAM.

  b. Ejecute los siguientes comandos para proporcionar las credenciales al software AWS IoT Greengrass principal.

  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. (Opcional) Si creó un usuario de IAM para aprovisionar su dispositivo Greengrass, elimínelo.

  d. (Opcional) Si utilizó el ID de clave de acceso y la clave de acceso secreta de un usuario de IAM existente, actualice las claves del usuario para que dejen de ser válidas. Para

obtener más información, consulte Actualización de las claves de acceso en la guía del AWS Identity and Access Management usuario.

Descargue el software AWS IoT Greengrass principal

Puede descargar la última versión del software AWS IoT Greengrass Core desde la siguiente ubicación:

- https://d2s8p88vqu9w66.cloudfront.net/releases/ greengrass-nucleus-latest .zip

> **Note**
>
> Puede descargar una versión específica del software AWS IoT Greengrass Core desde la siguiente ubicación. Sustituya la *versión* por la versión que desea descargar.
>
> ```
> https://d2s8p88vqu9w66.cloudfront.net/releases/greengrass-version.zip
> ```

Para descargar el software AWS IoT Greengrass principal

1. En su dispositivo principal, descargue el software AWS IoT Greengrass Core en un archivo denominado`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
   ```

Al descargar este software, acepta el acuerdo de licencia del software de Greengrass Core.

2. (Opcional) Para verificar la firma del software Greengrass Nucleus

> **Note**
>
> Esta función está disponible con la versión 2.9.5 y posteriores del núcleo de Greengrass.

a. Usa el siguiente comando para verificar la firma del artefacto núcleo de Greengrass:

Linux or Unix

```
jarsigner -verify -certs -verbose greengrass-nucleus-latest.zip
```

Windows Command Prompt (CMD)

El nombre del archivo puede tener un aspecto diferente según la versión de JDK que instale. *jdk17.0.6_10*Sustitúyalo por la versión de JDK que instaló.

```
"C:\\Program Files\\Amazon Corretto\\jdk17.0.6_10\\bin\\jarsigner.exe" -
verify -certs -verbose greengrass-nucleus-latest.zip
```

PowerShell

El nombre del archivo puede tener un aspecto diferente en función de la versión de JDK que instale. *jdk17.0.6_10*Sustitúyalo por la versión de JDK que instaló.

```
'C:\\Program Files\\Amazon Corretto\\jdk17.0.6_10\\bin\\jarsigner.exe' -
verify -certs -verbose greengrass-nucleus-latest.zip
```

b. La `jarsigner` invocación produce un resultado que indica los resultados de la verificación.

i. Si el archivo zip del núcleo de Greengrass está firmado, el resultado contiene la siguiente declaración:

```
jar verified.
```

ii. Si el archivo zip del núcleo de Greengrass no está firmado, el resultado contiene la siguiente declaración:

```
jar is unsigned.
```

c. Si ha proporcionado la `-certs` opción Jarsigner junto con `-verbose` las opciones `-verify` y, el resultado también incluye información detallada del certificado de firmante.

3. Descomprime el software AWS IoT Greengrass principal en una carpeta de tu dispositivo. *`GreengrassInstaller`*Sustitúyalo por la carpeta que desee usar.

   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. (Opcional) Ejecute el siguiente comando para ver la versión del software AWS IoT Greengrass principal.

   ```
   java -jar ./GreengrassInstaller/lib/Greengrass.jar --version
   ```

> **Important**
>
> Si instala una versión del núcleo de Greengrass anterior a la v2.4.0, no elimine esta carpeta después de instalar el software Core. AWS IoT Greengrass El software AWS IoT Greengrass Core utiliza los archivos de esta carpeta para ejecutarse.

Si descargó la última versión del software, instale la versión 2.4.0 o posterior y podrá eliminar esta carpeta después de instalar el software AWS IoT Greengrass principal.

Instalación del software AWS IoT Greengrass Core

Ejecute el instalador con argumentos que especifiquen lo siguiente:

- Cree los AWS recursos que el dispositivo principal necesita para funcionar.
- Especifique si desea utilizar el usuario `ggc_user` del sistema para ejecutar los componentes de software en el dispositivo principal. En los dispositivos Linux, este comando también especifica el uso del grupo `ggc_group` del sistema, y el instalador crea el usuario y el grupo del sistema automáticamente.
- Configure el software AWS IoT Greengrass principal como un servicio del sistema que se ejecute durante el arranque. En los dispositivos Linux, esto requiere el sistema de inicio Systemd.

> **Important**
>
> En los dispositivos principales de Windows, debe configurar el software AWS IoT Greengrass Core como un servicio del sistema.

Para configurar un dispositivo de desarrollo con herramientas de desarrollo local, especifique el `--deploy-dev-tools true` argumento. Una vez finalizada la instalación, el despliegue de las herramientas de desarrollo local puede tardar hasta un minuto.

Para obtener más información sobre los argumentos que puede especificar, consulteArgumentos de instalación.

> **Note**
>
> Si utiliza un AWS IoT Greengrass dispositivo con memoria limitada, puede controlar la cantidad de memoria que utiliza el software AWS IoT Greengrass Core. Para controlar la asignación de memoria, puede configurar las opciones de tamaño de pila de la JVM en el parámetro de `jvmOptions` configuración del componente core. Para obtener más información, consulte Controle la asignación de memoria con las opciones de JVM.

Para instalar el software de AWS IoT Greengrass Core

1. Ejecute el instalador AWS IoT Greengrass Core. Reemplace los valores de los argumentos en su comando de la siguiente manera.

   a. */greengrass/v2*o *C:\greengrass\v2*: la ruta a la carpeta raíz que se utilizará para instalar el software AWS IoT Greengrass principal.

   b. *GreengrassInstaller*. La ruta a la carpeta en la que desempaquetó el instalador del software AWS IoT Greengrass Core.

   c. *region*. Región de AWSEn la que se buscan o crean los recursos.

   d. *MyGreengrassCore*. El nombre del AWS IoT dispositivo principal de Greengrass. Si la cosa no existe, el instalador la crea. El instalador descarga los certificados para autenticarse como talAWS IoT. Para obtener más información, consulte Autenticación y autorización de dispositivos en AWS IoT Greengrass.

   > **Note**
   >
   > El nombre de la cosa no puede contener dos puntos (`:`).

   e. *MyGreengrassCoreGroup*. El nombre del grupo de AWS IoT cosas de su dispositivo principal de Greengrass. Si el grupo de cosas no existe, el instalador lo crea y lo añade. Si el grupo de cosas existe y tiene una implementación activa, el dispositivo principal descarga y ejecuta el software que especifique la implementación.

   > **Note**
   >
   > El nombre del grupo de cosas no puede contener dos puntos (`:`).

   f. *Greengrass V2 IoT ThingPolicy*. El nombre de la AWS IoT política que permite a los dispositivos principales de Greengrass comunicarse con AWS IoT y. AWS IoT Greengrass Si la AWS IoT política no existe, el instalador crea una AWS IoT política permisiva con este nombre. Puede restringir los permisos de esta política según su caso de uso. Para obtener más información, consulte AWS IoTPolítica mínima para los dispositivos AWS IoT Greengrass V2 principales.

   g. *Greengrass V2 TokenExchangeRole*. El nombre de la función de IAM que permite al dispositivo principal de Greengrass obtener AWS credenciales temporales. Si la función no existe, el instalador la crea y adjunta una política denominada.

*GreengrassV2TokenExchangeRole*Access Para obtener más información, consulte Autorizar a los dispositivos principales a interactuar con AWS los servicios.

h. *GreengrassCoreTokenExchangeRoleAlias*. El alias de la función de IAM que permite al dispositivo principal de Greengrass obtener credenciales temporales más adelante. Si el alias del rol no existe, el instalador lo crea y lo dirige al rol de IAM que especifique. Para obtener más información, consulte Autorizar a los dispositivos principales a interactuar con AWS los servicios.

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**
>
> En los dispositivos principales de Windows, debe especificar si `--setup-system-service true` desea configurar el software AWS IoT Greengrass principal como un servicio del sistema.

Si lo consigue, el instalador imprime los siguientes mensajes:

- Si lo especifica`--provision`, el instalador `Successfully configured Nucleus with provisioned resource details` imprimirá si configuró los recursos correctamente.
- Si lo especifica`--deploy-dev-tools`, el instalador imprimirá `Configured Nucleus to deploy aws.greengrass.Cli component` si creó la implementación correctamente.
- Si lo especifica`--setup-system-service true`, el instalador imprimirá `Successfully set up Nucleus as a system service` si configuró y ejecutó el software como un servicio.
- Si no lo especifica`--setup-system-service true`, el instalador imprimirá `Launched Nucleus successfully` si se ejecutó correctamente y ejecutó el software.

2. Omita este paso si instaló la Núcleo de Greengrass versión 2.0.4 o una versión posterior. Si descargó la última versión del software, instaló la versión 2.0.4 o posterior.

   Ejecute el siguiente comando para configurar los permisos de archivo necesarios para la carpeta raíz AWS IoT Greengrass del software principal. */greengrass/v2*Sustitúyala por la carpeta raíz que especificó en el comando de instalación y sustituya */greengrass* por la carpeta principal de la carpeta raíz.

   ```
   sudo chmod 755 /greengrass/v2 && sudo chmod 755 /greengrass
   ```

Si instaló el software AWS IoT Greengrass principal como un servicio del sistema, el instalador ejecutará el software automáticamente. De lo contrario, debe ejecutar el software manualmente. Para obtener más información, consulte Ejecute el software AWS IoT Greengrass principal.

> **Note**
>
> De forma predeterminada, la función de IAM que crea el instalador no permite el acceso a los artefactos de los componentes de los depósitos de S3. Para implementar componentes personalizados que definan artefactos en Amazon S3, debe añadir permisos al rol para permitir que su dispositivo principal recupere artefactos de componentes. Para obtener más información, consulte Permita el acceso a los depósitos de S3 para los artefactos de los componentes.
>
> Si aún no tiene un depósito de S3 para los artefactos de los componentes, puede añadir estos permisos más adelante, después de crear un depósito.

Para obtener más información sobre cómo configurar y usar el softwareAWS IoT Greengrass, consulte lo siguiente:

- Configurar el software AWS IoT Greengrass principal
- Desarrolle AWS IoT Greengrass componentes
- Implemente AWS IoT Greengrass componentes en los dispositivos
- Interfaz de línea de comandos Greengrass

Instale el software AWS IoT Greengrass principal con aprovisionamiento manual de recursos

El software AWS IoT Greengrass Core incluye un instalador que configura su dispositivo como un dispositivo principal de Greengrass. Para configurar un dispositivo manualmente, puede crear los recursos necesarios AWS IoT y de IAM para que los utilice el dispositivo. Si crea estos recursos manualmente, no necesita proporcionar AWS las credenciales al instalador.

Al instalar manualmente el software AWS IoT Greengrass Core, también puede configurar el dispositivo para que utilice un proxy de red o se conecte AWS al puerto 443. Es posible que tengas que especificar estas opciones de configuración si tu dispositivo funciona con un firewall o un proxy de red, por ejemplo. Para obtener más información, consulte Realizar la conexión en el puerto 443 o a través de un proxy de red.

También puede configurar el software AWS IoT Greengrass Core para que utilice un módulo de seguridad de hardware (HSM) a través de la interfaz PKCS #11. Esta función le permite almacenar de forma segura los archivos de certificados y claves privadas para que no queden expuestos ni duplicados en el software. Puede almacenar claves privadas y certificados en un módulo de hardware, como un HSM, un módulo de plataforma segura (TPM) u otro elemento criptográfico. Esta función solo está disponible en dispositivos Linux. Para obtener más información sobre la seguridad del hardware y los requisitos para su uso, consulteIntegración de la seguridad de hardware.

> **Important**
>
> Antes de descargar el software AWS IoT Greengrass Core, compruebe que su dispositivo principal cumpla los requisitos para instalar y ejecutar el software AWS IoT Greengrass Core v2.0.

Temas

Recupere los puntos finales AWS IoT

Obtenga los AWS IoT puntos finales que desee y guárdelos para usarlos más adelante. Cuenta de AWS El dispositivo utiliza estos puntos finales para conectarse a ellos. AWS IoT Haga lo siguiente:

1. Obtenga el punto final AWS IoT de datos para su. Cuenta de AWS

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

   La respuesta es similar a la del siguiente ejemplo, si la solicitud se realiza correctamente.

```
{
  "endpointAddress": "device-data-prefix-ats.iot.us-west-2.amazonaws.com"
}
```

2. Obtenga el punto final de AWS IoT credenciales para su. Cuenta de AWS

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

La respuesta es similar a la del siguiente ejemplo, si la solicitud se realiza correctamente.

```
{
  "endpointAddress": "device-credentials-prefix.credentials.iot.us-
west-2.amazonaws.com"
}
```

Crea cualquier cosa AWS IoT

AWS IoT las cosas representan dispositivos y entidades lógicas a las que se conectan AWS IoT. Los dispositivos principales de Greengrass son AWS IoT cosas. Cuando registras un dispositivo como una AWS IoT cosa, ese dispositivo puede usar un certificado digital para autenticarse. AWS

En esta sección, crearás AWS IoT algo que represente tu dispositivo.

Para crear cualquier AWS IoT cosa

1. Crea cualquier AWS IoT cosa para tu dispositivo. En tu ordenador de desarrollo, ejecuta el siguiente comando.

   - Sustituya *MyGreengrassCore* por el nombre de la cosa que vaya a utilizar. Este nombre también es el nombre de su dispositivo principal de Greengrass.

     > **Note**
     >
     > El nombre del objeto no puede contener dos puntos (`:`).

```
aws iot create-thing --thing-name MyGreengrassCore
```

Si la solicitud se realiza correctamente, la respuesta es similar a la del siguiente ejemplo.

```
{
  "thingName": "MyGreengrassCore",
  "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/MyGreengrassCore",
  "thingId": "8cb4b6cd-268e-495d-b5b9-1713d71dbf42"
}
```

2. (Opcional) Añada el AWS IoT elemento a un grupo de elementos nuevo o existente. Los grupos de cosas se utilizan para gestionar las flotas de los dispositivos principales de Greengrass. Al implementar componentes de software en sus dispositivos, puede dirigirlos a dispositivos individuales o a grupos de dispositivos. Puede añadir un dispositivo a un grupo de cosas con una implementación activa de Greengrass para implementar los componentes de software de ese grupo de cosas en el dispositivo. Haga lo siguiente:

   a. (Opcional) Cree un grupo de AWS IoT cosas.

      - *MyGreengrassCoreGroup*Sustitúyalo por el nombre del grupo de cosas que desee crear.

      > **Note**
      >
      > El nombre del grupo de cosas no puede contener dos puntos (`:`).

      ```
      aws iot create-thing-group --thing-group-name MyGreengrassCoreGroup
      ```

      La respuesta es similar a la del siguiente ejemplo, si la solicitud se realiza correctamente.

      ```
      {
        "thingGroupName": "MyGreengrassCoreGroup",
        "thingGroupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/
      MyGreengrassCoreGroup",
        "thingGroupId": "4df721e1-ff9f-4f97-92dd-02db4e3f03aa"
      }
      ```

   b. Añada la AWS IoT cosa a un grupo de cosas.

      - Sustituye *MyGreengrassCore* por el nombre de tu AWS IoT objeto.

- *MyGreengrassCoreGroup*Sustitúyalo por el nombre del grupo de cosas.

```
aws iot add-thing-to-thing-group --thing-name MyGreengrassCore --thing-group-
name MyGreengrassCoreGroup
```

El comando no tiene ningún resultado si la solicitud se realiza correctamente.

Crea el certificado de la cosa

Cuando registras un dispositivo como una AWS IoT cosa, ese dispositivo puede usar un certificado digital para autenticarse AWS. Este certificado permite que el dispositivo se comunique con AWS IoT y AWS IoT Greengrass.

En esta sección, puede crear y descargar certificados que el dispositivo puede usar para conectarse AWS.

Si desea configurar el software AWS IoT Greengrass principal para que utilice un módulo de seguridad de hardware (HSM) para almacenar de forma segura la clave privada y el certificado, siga los pasos para crear el certificado a partir de una clave privada de un HSM. De lo contrario, siga los pasos para crear el certificado y la clave privada en el AWS IoT servicio. La función de seguridad de hardware solo está disponible en dispositivos Linux. Para obtener más información sobre la seguridad del hardware y los requisitos para su uso, consulteIntegración de la seguridad de hardware.

Cree el certificado y la clave privada en el AWS IoT servicio

Para crear el certificado de cosa

1. Crea una carpeta en la que descargues los certificados de la AWS IoT cosa.

   ```
   mkdir greengrass-v2-certs
   ```

2. Crea y descarga los certificados de la AWS IoT cosa.

   ```
   aws iot create-keys-and-certificate --set-as-active --certificate-pem-outfile
    greengrass-v2-certs/device.pem.crt --public-key-outfile greengrass-v2-certs/
   public.pem.key --private-key-outfile greengrass-v2-certs/private.pem.key
   ```

   La respuesta es similar a la del siguiente ejemplo, si la solicitud se realiza correctamente.

```
{
  "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-----\
"
  }
}
```

Guarde el nombre de recurso de Amazon (ARN) del certificado para configurarlo más adelante.

Cree el certificado a partir de una clave privada en un HSM

> **Note**
>
> Esta función está disponible para la versión 2.5.3 y versiones posteriores del componente núcleo de Greengrass. AWS IoT Greengrass actualmente no admite esta función en los dispositivos principales de Windows.

Para crear el certificado de la cosa

1. En el dispositivo principal, inicialice un token PKCS #11 en el HSM y genere una clave privada. La clave privada debe ser una clave RSA con un tamaño de clave RSA-2048 (o mayor) o una clave ECC.

   > **Note**
   >
   > Para utilizar un módulo de seguridad de hardware con claves ECC, debe utilizar Greengrass nucleus v2.5.6 o posterior.
   > Para usar un módulo de seguridad de hardware y un administrador de secretos, debe usar un módulo de seguridad de hardware con claves RSA.

   Consulte la documentación de su HSM para obtener información sobre cómo inicializar el token y generar la clave privada. Si su HSM admite ID de objeto, especifique un ID de objeto al generar la clave privada. Guarde el ID de ranura, el PIN de usuario, la etiqueta del objeto y el ID del objeto (si su HSM utiliza alguno) que especifique al inicializar el token y generar la clave privada. Estos valores se utilizan más adelante cuando se importa el certificado de la cosa al HSM y se configura el AWS IoT Greengrass software Core.
2. Cree una solicitud de firma de certificado (CSR) a partir de la clave privada. AWS IoT usa esta CSR para crear un certificado de cosas para la clave privada que generaste en el HSM. Para obtener información sobre cómo crear una CSR a partir de la clave privada, consulte la documentación de su HSM. La CSR es un archivo, como. `iotdevicekey.csr`
3. Copie la CSR del dispositivo a su ordenador de desarrollo. Si SSH y SCP están habilitados en el ordenador de desarrollo y en el dispositivo, puede utilizar el `scp` comando del ordenador de desarrollo para transferir la CSR. Sustituya la *`dirección IP del dispositivo por la`*

dirección IP de su dispositivo y sustituya *~/iotdevicekey.csr* por la ruta al archivo CSR del dispositivo.

```
scp device-ip-address:~/iotdevicekey.csr iotdevicekey.csr
```

4. En tu ordenador de desarrollo, crea una carpeta en la que descargues el certificado del dispositivo. AWS IoT

```
mkdir greengrass-v2-certs
```

5. Utilice el archivo CSR para crear y descargar el certificado del dispositivo en AWS IoT su ordenador de desarrollo.

```
aws iot create-certificate-from-csr --set-as-active --certificate-signing-
request=file://iotdevicekey.csr --certificate-pem-outfile greengrass-v2-certs/
device.pem.crt
```

La respuesta es similar a la del siguiente ejemplo, si la solicitud se realiza correctamente.

```
{
  "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-----"
```

```
}
```

Guarde el ARN del certificado para usarlo más adelante para configurarlo.

Configure el certificado del objeto

Adjunte el certificado del AWS IoT objeto al elemento que creó anteriormente y añada una AWS IoT política al certificado para definir los AWS IoT permisos del dispositivo principal.

Para configurar el certificado de la cosa

1. Adjunte el certificado a la AWS IoT cosa.

   - Reemplace *MyGreengrassCore* con el nombre de su AWS IoT cosa.
   - Sustituya el certificado Amazon Resource Name (ARN) por el ARN del certificado que creó en el paso anterior.

   ```
   aws iot attach-thing-principal --thing-name MyGreengrassCore
    --principal arn:aws:iot:us-west-2:123456789012:cert/
   aa0b7958770878eabe251d8a7ddd547f4889c524c9b574ab9fbf65f32248b1d4
   ```

   El comando no tiene ningún resultado si la solicitud se realiza correctamente.

2. Cree y adjunte una AWS IoT política que defina los AWS IoT permisos de su dispositivo principal de Greengrass. La siguiente política permite el acceso a todos los temas de MQTT y a las operaciones de Greengrass, de modo que su dispositivo funcione con aplicaciones personalizadas y con cambios futuros que requieran nuevas operaciones de Greengrass. Puede restringir esta política en función de su caso de uso. Para obtener más información, consulte AWS IoTPolítica mínima para los dispositivos AWS IoT Greengrass V2 principales.

   Si ya ha configurado un dispositivo principal de Greengrass, puede adjuntar su AWS IoT política en lugar de crear una nueva.

   Haga lo siguiente:

   a. Cree un archivo que contenga el documento AWS IoT de política que requieren los dispositivos principales de Greengrass.

Por ejemplo, en un sistema basado en Linux, puede ejecutar el siguiente comando para usar GNU nano para crear el archivo.

```
nano greengrass-v2-iot-policy.json
```

Copia el siguiente JSON en el archivo.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish",
        "iot:Subscribe",
        "iot:Receive",
        "iot:Connect",
        "greengrass:*"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

b. Cree una AWS IoT política a partir del documento de política.

- Sustituya *GreengrassV2IoT* por el nombre ThingPolicy de la política que se va a crear.

```
aws iot create-policy --policy-name GreengrassV2IoTThingPolicy --policy-
document file://greengrass-v2-iot-policy.json
```

Si la solicitud se realiza correctamente, la respuesta es similar a la del siguiente ejemplo.

```
{
  "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. Adjunta la AWS IoT política al certificado de la AWS IoT cosa.

- Sustituya *GreengrassV2IoT* por el nombre ThingPolicy de la política que desee adjuntar.
- Sustituya el ARN de destino por el ARN del certificado de su dispositivo. AWS IoT

```
aws iot attach-policy --policy-name GreengrassV2IoTThingPolicy
 --target arn:aws:iot:us-west-2:123456789012:cert/
aa0b7958770878eabe251d8a7ddd547f4889c524c9b574ab9fbf65f32248b1d4
```

El comando no tiene ningún resultado si la solicitud se realiza correctamente.

Crea un rol de intercambio de fichas

Los dispositivos principales de Greengrass utilizan una función de servicio de IAM, denominada función de intercambio de fichas, para autorizar las llamadas a los servicios. AWS El dispositivo utiliza el proveedor de AWS IoT credenciales para obtener AWS credenciales temporales para esta función, lo que permite al dispositivo interactuar con Amazon Logs AWS IoT, enviar registros a Amazon CloudWatch Logs y descargar artefactos de componentes personalizados de Amazon S3.

Para obtener más información, consulte Autorizar a los dispositivos principales a interactuar con AWS los servicios.

Se utiliza un alias de AWS IoT rol para configurar el rol de intercambio de fichas para los dispositivos principales de Greengrass. Los alias de rol le permiten cambiar el rol de intercambio de fichas de un dispositivo, pero mantener la configuración del dispositivo igual. Para obtener más información, consulte Autorizar llamadas directas a AWS los servicios en la Guía para AWS IoT Core desarrolladores.

En esta sección, se crea una función de IAM de intercambio de fichas y un alias de AWS IoT función que apunte a esa función. Si ya ha configurado un dispositivo principal de Greengrass, puede utilizar su función de intercambio de fichas y su alias de función en lugar de crear otros nuevos. A continuación, configura el dispositivo para que utilice ese rol y ese alias. AWS IoT

Para crear un rol de IAM para el intercambio de fichas

1. Cree una función de IAM que su dispositivo pueda utilizar como función de intercambio de fichas. Haga lo siguiente:

   a. Cree un archivo que contenga el documento de política de confianza que requiere la función de intercambio de tokens.

      Por ejemplo, en un sistema basado en Linux, puede ejecutar el siguiente comando para usar GNU nano para crear el archivo.

      ```
      nano device-role-trust-policy.json
      ```

      Copia el siguiente JSON en el archivo.

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

```
}
```

b. Cree la función de intercambio de fichas con el documento de política de confianza.

- Sustituya el *TokenExchangerol de GreengrassV2* por el nombre del rol de IAM que desee crear.

```
aws iam create-role --role-name GreengrassV2TokenExchangeRole --assume-role-
policy-document file://device-role-trust-policy.json
```

Si la solicitud se realiza correctamente, la respuesta es similar a la del siguiente ejemplo.

```
{
  "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. Cree un archivo que contenga el documento de política de acceso que requiere la función de intercambio de tokens.

Por ejemplo, en un sistema basado en Linux, puede ejecutar el siguiente comando para usar GNU nano para crear el archivo.

```
nano device-role-access-policy.json
```

Copia el siguiente JSON en el archivo.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:DescribeLogStreams",
        "s3:GetBucketLocation"
      ],
      "Resource": "*"
    }
  ]
}
```

> **Note**
>
> Esta política de acceso no permite el acceso a los artefactos de los componentes de los depósitos de S3. Para implementar componentes personalizados que definan artefactos en Amazon S3, debe añadir permisos al rol para permitir que su dispositivo principal recupere artefactos de componentes. Para obtener más información, consulte Permita el acceso a los depósitos de S3 para los artefactos de los componentes.
>
> Si aún no tiene un depósito de S3 para los artefactos de los componentes, puede añadir estos permisos más adelante, después de crear un depósito.

d. Cree la política de IAM a partir del documento de política.

- Sustituya *GreengrassV2 TokenExchange RoleAccess* por el nombre de la política de IAM que desee crear.

```
aws iam create-policy --policy-name GreengrassV2TokenExchangeRoleAccess --
policy-document file://device-role-access-policy.json
```

Si la solicitud se realiza correctamente, la respuesta es similar a la del siguiente ejemplo.

```
{
  "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. Adjunte la política de IAM a la función de intercambio de fichas.

- Sustituya la *TokenExchangefunción GreengrassV2* por el nombre de la función de IAM.
- Sustituya el ARN de la política por el ARN de la política de IAM que creó en el paso anterior.

```
aws iam attach-role-policy --role-name GreengrassV2TokenExchangeRole --policy-
arn arn:aws:iam::123456789012:policy/GreengrassV2TokenExchangeRoleAccess
```

El comando no tiene ningún resultado si la solicitud se realiza correctamente.

2. Cree un alias de AWS IoT rol que apunte al rol de intercambio de fichas.

- *GreengrassCoreTokenExchangeRoleAlias*Sustitúyalo por el nombre del alias del rol que se va a crear.
- Sustituya el ARN del rol por el ARN del rol de IAM que creó en el paso anterior.

```
aws iot create-role-alias --role-alias GreengrassCoreTokenExchangeRoleAlias --role-
arn arn:aws:iam::123456789012:role/GreengrassV2TokenExchangeRole
```

Si la solicitud se realiza correctamente, la respuesta es similar a la del ejemplo siguiente.

```
{
  "roleAlias": "GreengrassCoreTokenExchangeRoleAlias",
  "roleAliasArn": "arn:aws:iot:us-west-2:123456789012:rolealias/
GreengrassCoreTokenExchangeRoleAlias"
}
```

> **Note**
>
> Para crear un alias de rol, debe tener permiso para transferir el rol de IAM de intercambio de fichas. AWS IoT Si recibe un mensaje de error al intentar crear un alias de rol, compruebe que su AWS usuario tiene este permiso. Para obtener más información, consulte Conceder permisos a un usuario para transferir un rol a un AWS servicio en la Guía del AWS Identity and Access Management usuario.

3. Cree y adjunte una AWS IoT política que permita a su dispositivo principal de Greengrass utilizar el alias del rol para asumir el rol de intercambio de fichas. Si ya ha configurado un dispositivo principal de Greengrass, puede adjuntar su AWS IoT política de alias de rol en lugar de crear una nueva. Haga lo siguiente:

   a. (Opcional) Cree un archivo que contenga el documento AWS IoT de política que requiere el alias del rol.

      Por ejemplo, en un sistema basado en Linux, puede ejecutar el siguiente comando para usar GNU nano para crear el archivo.

      ```
      nano greengrass-v2-iot-role-alias-policy.json
      ```

      Copia el siguiente JSON en el archivo.

      - Sustituya el ARN del recurso por el ARN del alias de su rol.

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

```
      "Action": "iot:AssumeRoleWithCertificate",
      "Resource": "arn:aws:iot:us-west-2:123456789012:rolealias/
GreengrassCoreTokenExchangeRoleAlias"
    }
  ]
}
```

b. Cree una AWS IoT política a partir del documento de política.

- Sustituya la *GreengrassCoreTokenExchangeRoleAliaspolítica* por el nombre de la AWS IoT política que se va a crear.

```
aws iot create-policy --policy-name GreengrassCoreTokenExchangeRoleAliasPolicy
 --policy-document file://greengrass-v2-iot-role-alias-policy.json
```

Si la solicitud se realiza correctamente, la respuesta es similar a la del ejemplo siguiente.

```
{
  "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. Adjunta la AWS IoT política al certificado de la AWS IoT cosa.

- Sustituya la *GreengrassCoreTokenExchangeRoleAliaspolítica* por el nombre de la AWS IoT política de alias del rol.
- Sustituya el ARN de destino por el ARN del certificado de su dispositivo. AWS IoT

```
aws iot attach-policy --policy-name GreengrassCoreTokenExchangeRoleAliasPolicy
 --target arn:aws:iot:us-west-2:123456789012:cert/
aa0b7958770878eabe251d8a7ddd547f4889c524c9b574ab9fbf65f32248b1d4
```

El comando no tiene ningún resultado si la solicitud se realiza correctamente.

Descargue los certificados al dispositivo

Anteriormente, descargaste el certificado de tu dispositivo en tu ordenador de desarrollo. En esta sección, copia el certificado en el dispositivo principal para configurar el dispositivo con los certificados a los que se conecta AWS IoT. También descargas el certificado de la autoridad de certificación raíz (CA) de Amazon. Si utiliza un HSM, también importará el archivo de certificado al HSM en esta sección.

- Si creó anteriormente el certificado y la clave privada en el AWS IoT servicio, siga los pasos para descargar los certificados con la clave privada y los archivos de certificado.
- Si anteriormente creó el certificado Thing a partir de una clave privada en un módulo de seguridad de hardware (HSM), siga los pasos para descargar los certificados con la clave privada y el certificado en un HSM.

Descargue los certificados con la clave privada y los archivos de certificado

Para descargar certificados al dispositivo

1. Copia AWS IoT el certificado de la máquina de desarrollo al dispositivo. Si SSH y SCP están habilitados en el ordenador de desarrollo y en el dispositivo, puede utilizar el `scp` comando del ordenador de desarrollo para transferir el certificado. Sustituya la *dirección IP del dispositivo por la dirección* IP de su dispositivo.

   ```
   scp -r greengrass-v2-certs/ device-ip-address:~
   ```

2. Cree la carpeta raíz de Greengrass en el dispositivo. Más adelante instalará el software AWS IoT Greengrass principal en esta carpeta.

   Linux or Unix

   - */greengrass/v2*Sustitúyalo por la carpeta que desee utilizar.

```
sudo mkdir -p /greengrass/v2
```

Windows Command Prompt

- Sustituya *C:\greengrass\v2* por la carpeta que desee utilizar.

```
mkdir C:\greengrass\v2
```

PowerShell

- Sustituya *C:\greengrass\v2* por la carpeta que desee utilizar.

```
mkdir C:\greengrass\v2
```

3. (Solo para Linux) Establezca los permisos de la carpeta principal de la raíz de Greengrass.

   - Sustituya */greengrass* por el archivo principal de la carpeta raíz.

   ```
   sudo chmod 755 /greengrass
   ```

4. Copia los certificados de la AWS IoT cosa a la carpeta raíz de Greengrass.

   Linux or Unix

   - */greengrass/v2*Sustitúyala por la carpeta raíz de Greengrass.

   ```
   sudo cp -R ~/greengrass-v2-certs/* /greengrass/v2
   ```

   Windows Command Prompt

   - Sustituya *C:\greengrass\v2* por la carpeta que vaya a utilizar.

   ```
   robocopy %USERPROFILE%\greengrass-v2-certs C:\greengrass\v2 /E
   ```

PowerShell

- Sustituya *C:\greengrass\v2* por la carpeta que desee utilizar.

```
cp -Path ~\greengrass-v2-certs\* -Destination C:\greengrass\v2
```

5. Descarga el certificado de la autoridad de certificación raíz (CA) de Amazon. AWS IoT Los certificados están asociados al certificado de CA raíz de Amazon de forma predeterminada.

   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
   ```

Descargue los certificados con la clave privada y el certificado en un HSM

> **Note**
>
> Esta función está disponible para la versión 2.5.3 y versiones posteriores del componente núcleo de Greengrass. AWS IoT Greengrass actualmente no admite esta función en los dispositivos principales de Windows.

Para descargar certificados al dispositivo

1. Copia AWS IoT el certificado de la máquina de desarrollo al dispositivo. Si SSH y SCP están habilitados en el ordenador de desarrollo y en el dispositivo, puede utilizar el `scp` comando

del ordenador de desarrollo para transferir el certificado. Sustituya la *dirección IP del dispositivo por la dirección* IP de su dispositivo.

```
scp -r greengrass-v2-certs/ device-ip-address:~
```

2. Cree la carpeta raíz de Greengrass en el dispositivo. Más adelante instalará el software AWS IoT Greengrass principal en esta carpeta.

   Linux or Unix

   - */greengrass/v2*Sustitúyalo por la carpeta que desee utilizar.

   ```
   sudo mkdir -p /greengrass/v2
   ```

   Windows Command Prompt

   - Sustituya *C:\greengrass\v2* por la carpeta que desee utilizar.

   ```
   mkdir C:\greengrass\v2
   ```

   PowerShell

   - Sustituya *C:\greengrass\v2* por la carpeta que desee utilizar.

   ```
   mkdir C:\greengrass\v2
   ```

3. (Solo para Linux) Establezca los permisos de la carpeta principal de la raíz de Greengrass.

   - Sustituya */greengrass* por el archivo principal de la carpeta raíz.

   ```
   sudo chmod 755 /greengrass
   ```

4. Importe el archivo de certificado de la cosa,`~/greengrass-v2-certs/device.pem.crt`, al HSM. Consulte la documentación de su HSM para saber cómo importar certificados al mismo. Importe el certificado con el mismo token, ID de ranura, PIN de usuario, etiqueta de objeto e ID de objeto (si su HSM utiliza alguno) con los que generó la clave privada en el HSM anteriormente.

> **Note**
>
> Si generaste la clave privada anteriormente sin un ID de objeto y el certificado tiene un ID de objeto, establece el ID de objeto de la clave privada con el mismo valor que el certificado. Consulte la documentación de su HSM para obtener información sobre cómo configurar el ID de objeto para el objeto de clave privada.

5. (Opcional) Elimine el archivo de certificado de la cosa para que solo exista en el HSM.

```
rm ~/greengrass-v2-certs/device.pem.crt
```

6. Descarga el certificado de la autoridad de certificación raíz (CA) de Amazon. AWS IoT Los certificados están asociados al certificado de CA raíz de Amazon de forma predeterminada.

   Linux or Unix

   ```
   sudo curl -o /greengrass/v2/AmazonRootCA1.pem https://www.amazontrust.com/
   repository/AmazonRootCA1.pem
   ```

   Windows Command Prompt (CMD)

   ```
   curl -o C:\greengrass\v2\\AmazonRootCA1.pem https://www.amazontrust.com/
   repository/AmazonRootCA1.pem
   ```

   PowerShell

   ```
   iwr -Uri https://www.amazontrust.com/repository/AmazonRootCA1.pem -OutFile C:
   \greengrass\v2\\AmazonRootCA1.pem
   ```

Configura el entorno del dispositivo

Siga los pasos de esta sección para configurar un dispositivo Linux o Windows para usarlo como dispositivo AWS IoT Greengrass principal.

Configura un dispositivo Linux

Para configurar un dispositivo Linux para AWS IoT Greengrass V2

1. Instale el motor de ejecución de Java, que el software AWS IoT Greengrass principal necesita para ejecutarse. Le recomendamos que utilice las versiones de soporte a largo plazo de Amazon Corretto u OpenJDK. Se requiere la versión 8 o superior. Los siguientes comandos le muestran cómo instalar OpenJDK en su dispositivo.

   - Para distribuciones basadas en Debian o en Ubuntu:

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

   - Para distribuciones basadas en Red Hat:

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

   - En Amazon Linux 2:

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

   - En Amazon Linux 2023:

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

   Cuando se complete la instalación, ejecute el siguiente comando para comprobar que Java se ejecuta en su dispositivo Linux.

   ```
   java -version
   ```

   El comando imprime la versión de Java que se ejecuta en el dispositivo. Por ejemplo, en una distribución basada en Debian, el resultado podría tener un aspecto similar al del siguiente ejemplo.

   ```
   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. (Opcional) Cree el usuario y el grupo predeterminados del sistema que ejecutan los componentes del dispositivo. También puede optar por permitir que el instalador del software AWS IoT Greengrass principal cree este usuario y grupo durante la instalación con el argumento del `--component-default-user` instalador. Para obtener más información, consulte Argumentos de instalación.

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

3. Compruebe que el usuario que ejecuta el software AWS IoT Greengrass principal (normalmente`root`) tiene permiso para ejecutar `sudo` con cualquier usuario y grupo.

   a. Ejecute el siguiente comando para abrir el `/etc/sudoers` archivo.

      ```
      sudo visudo
      ```

   b. Compruebe que el permiso del usuario es similar al del ejemplo siguiente.

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

4. (Opcional) Para ejecutar funciones Lambda en contenedores, debe habilitar cgroups v1 y debe habilitar y montar los cgroups de memoria y dispositivos. Si no planea ejecutar funciones Lambda en contenedores, puede omitir este paso.

   Para habilitar estas opciones de cgroups, arranque el dispositivo con los siguientes parámetros del kernel de Linux.

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

   Para obtener información sobre cómo ver y configurar los parámetros del núcleo de su dispositivo, consulte la documentación del sistema operativo y del gestor de arranque. Siga las instrucciones para configurar permanentemente los parámetros del núcleo.

5. Instale todas las demás dependencias necesarias en su dispositivo tal y como se indica en Requisitos de los dispositivos la lista de requisitos de.

Configura un dispositivo Windows

> **Note**
>
> Esta función está disponible para la versión 2.5.0 y versiones posteriores del componente núcleo de Greengrass.

Para configurar un dispositivo Windows para AWS IoT Greengrass V2

1. Instale el motor de ejecución de Java, que el software AWS IoT Greengrass principal necesita para ejecutarse. Le recomendamos que utilice las versiones de soporte a largo plazo de Amazon Corretto u OpenJDK. Se requiere la versión 8 o superior.
2. Compruebe si Java está disponible en la variable de sistema PATH y agréguelo si no lo está. La LocalSystem cuenta ejecuta el software AWS IoT Greengrass principal, por lo que debe añadir Java a la variable de sistema PATH en lugar de a la variable de usuario PATH de su usuario. Haga lo siguiente:
   a. Pulse la tecla Windows para abrir el menú de inicio.
   b. Escriba **`environment variables`** para buscar las opciones del sistema en el menú de inicio.
   c. En los resultados de la búsqueda del menú de inicio, seleccione Editar las variables de entorno del sistema para abrir la ventana de propiedades del sistema.
   d. Seleccione Variables de entorno... para abrir la ventana Variables de entorno.
   e. En Variables de sistema, seleccione Ruta y, a continuación, elija Editar. En la ventana Editar variables de entorno, puede ver cada ruta en una línea independiente.
   f. Compruebe si la ruta a la `bin` carpeta de la instalación de Java está presente. La ruta puede tener un aspecto similar al del siguiente ejemplo.

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

   g. Si la `bin` carpeta de la instalación de Java no aparece en Path, seleccione Nueva para añadirla y, a continuación, pulse Aceptar.
3. Abre la línea de comandos de Windows (`cmd.exe`) como administrador.
4. Cree el usuario predeterminado en la LocalSystem cuenta del dispositivo Windows. Sustituya la *`contraseña`* por una contraseña segura.

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

> **Tip**
>
> Según la configuración de Windows, es posible que la contraseña del usuario caduque en una fecha futura. Para garantizar que sus aplicaciones de Greengrass sigan funcionando, controle cuándo caduque la contraseña y actualícela antes de que caduque. También puede configurar la contraseña del usuario para que nunca caduque.
>
> - Para comprobar cuándo caducan un usuario y su contraseña, ejecuta el siguiente comando.
>
>   ```
>   net user ggc_user | findstr /C:expires
>   ```
>
> - Para configurar la contraseña de un usuario para que no caduque nunca, ejecute el siguiente comando.
>
>   ```
>   wmic UserAccount where "Name='ggc_user'" set PasswordExpires=False
>   ```
>
> - Si utilizas Windows 10 o una versión posterior, donde el wmiccomando está obsoleto, ejecuta el siguiente PowerShell comando.
>
>   ```
>   Get-CimInstance -Query "SELECT * from Win32_UserAccount WHERE name =
>    'ggc_user'" | Set-CimInstance -Property @{PasswordExpires="False"}
>   ```

5. Descargue e instale la PsExecutilidad de Microsoft en el dispositivo.
6. Utilice la PsExec utilidad para almacenar el nombre de usuario y la contraseña del usuario predeterminado en la instancia de Credential Manager de la LocalSystem cuenta. Sustituya la *contraseña* por la contraseña del usuario que configuró anteriormente.

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

   Si se PsExec License Agreementabre, Acceptacepte la licencia y ejecute el comando.

> **Note**
>
> En los dispositivos Windows, la LocalSystem cuenta ejecuta el núcleo de Greengrass y debe usar la PsExec utilidad para almacenar la información de usuario predeterminada en la LocalSystem cuenta. El uso de la aplicación Credential Manager almacena esta información en la cuenta de Windows del usuario que ha iniciado sesión actualmente, en lugar de en la LocalSystem cuenta.

Descargue el software AWS IoT Greengrass principal

Puede descargar la última versión del software AWS IoT Greengrass Core desde la siguiente ubicación:

- https://d2s8p88vqu9w66.cloudfront.net/releases/greengrass-nucleus-latest.zip

> **Note**
>
> Puede descargar una versión específica del software AWS IoT Greengrass Core desde la siguiente ubicación. Sustituya la *versión* por la versión que desea descargar.
>
> ```
> https://d2s8p88vqu9w66.cloudfront.net/releases/greengrass-version.zip
> ```

Para descargar el software AWS IoT Greengrass principal

1. En su dispositivo principal, descargue el software AWS IoT Greengrass Core en un archivo denominado`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
```

Al descargar este software, acepta el acuerdo de licencia del software de Greengrass Core.

2. (Opcional) Para verificar la firma del software Greengrass Nucleus

> **Note**
>
> Esta función está disponible con la versión 2.9.5 y posteriores del núcleo de Greengrass.

a. Usa el siguiente comando para verificar la firma del artefacto núcleo de Greengrass:

Linux or Unix

```
jarsigner -verify -certs -verbose greengrass-nucleus-latest.zip
```

Windows Command Prompt (CMD)

El nombre del archivo puede tener un aspecto diferente según la versión de JDK que instale. *jdk17.0.6_10*Sustitúyalo por la versión de JDK que instaló.

```
"C:\\Program Files\\Amazon Corretto\\jdk17.0.6_10\\bin\\jarsigner.exe" -
verify -certs -verbose greengrass-nucleus-latest.zip
```

PowerShell

El nombre del archivo puede tener un aspecto diferente en función de la versión de JDK que instale. *jdk17.0.6_10*Sustitúyalo por la versión de JDK que instaló.

```
'C:\\Program Files\\Amazon Corretto\\jdk17.0.6_10\\bin\\jarsigner.exe' -
verify -certs -verbose greengrass-nucleus-latest.zip
```

b. La `jarsigner` invocación produce un resultado que indica los resultados de la verificación.

   i. Si el archivo zip del núcleo de Greengrass está firmado, el resultado contiene la siguiente declaración:

   ```
   jar verified.
   ```

   ii. Si el archivo zip del núcleo de Greengrass no está firmado, el resultado contiene la siguiente declaración:

   ```
   jar is unsigned.
   ```

c. Si ha proporcionado la `-certs` opción Jarsigner junto con `-verbose` las opciones `-verify` y, el resultado también incluye información detallada del certificado de firmante.

3. Descomprime el software AWS IoT Greengrass principal en una carpeta de tu dispositivo. *`GreengrassInstaller`*Sustitúyalo por la carpeta que desee usar.

   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. (Opcional) Ejecute el siguiente comando para ver la versión del software AWS IoT Greengrass principal.

```
java -jar ./GreengrassInstaller/lib/Greengrass.jar --version
```

> **Important**
>
> Si instala una versión del núcleo de Greengrass anterior a la v2.4.0, no elimine esta carpeta después de instalar el software Core. AWS IoT Greengrass El software AWS IoT Greengrass Core utiliza los archivos de esta carpeta para ejecutarse.
> Si descargó la última versión del software, instale la versión 2.4.0 o posterior y podrá eliminar esta carpeta después de instalar el software AWS IoT Greengrass principal.

Instale el software principal AWS IoT Greengrass

Ejecute el instalador con argumentos que especifiquen las siguientes acciones:

- Realice la instalación desde un archivo de configuración parcial que especifique el uso de AWS los recursos y certificados que creó anteriormente. El software AWS IoT Greengrass Core utiliza un archivo de configuración que especifica la configuración de todos los componentes de Greengrass del dispositivo. El instalador crea un archivo de configuración completo a partir del archivo de configuración parcial que usted proporciona.
- Especifique si desea utilizar el usuario `ggc_user` del sistema para ejecutar los componentes de software en el dispositivo principal. En los dispositivos Linux, este comando también especifica el uso del grupo `ggc_group` del sistema, y el instalador crea el usuario y el grupo del sistema automáticamente.
- Configure el software AWS IoT Greengrass principal como un servicio del sistema que se ejecute durante el arranque. En los dispositivos Linux, esto requiere el sistema de inicio Systemd.

> **Important**
>
> En los dispositivos principales de Windows, debe configurar el software AWS IoT Greengrass Core como un servicio del sistema.

Para obtener más información sobre los argumentos que puede especificar, consulteArgumentos de instalación.

> **Note**
>
> Si utiliza un AWS IoT Greengrass dispositivo con memoria limitada, puede controlar la cantidad de memoria que utiliza el software AWS IoT Greengrass Core. Para controlar la asignación de memoria, puede configurar las opciones de tamaño de pila de la JVM en el parámetro de `jvmOptions` configuración del componente core. Para obtener más información, consulte Controle la asignación de memoria con las opciones de JVM.

- Si creó anteriormente el certificado y la clave privada en el AWS IoT servicio, siga los pasos para instalar el software AWS IoT Greengrass Core con los archivos de clave privada y certificado.
- Si anteriormente creó el certificado Thing a partir de una clave privada en un módulo de seguridad de hardware (HSM), siga los pasos para instalar el software AWS IoT Greengrass Core con la clave privada y el certificado en un HSM.

Instale el software AWS IoT Greengrass principal con la clave privada y los archivos de certificado

Para instalar el software AWS IoT Greengrass Core

1. Compruebe la versión del software AWS IoT Greengrass principal.

   - *`GreengrassInstaller`*Sustitúyala por la ruta a la carpeta que contiene el software.

   ```
   java -jar ./GreengrassInstaller/lib/Greengrass.jar --version
   ```

2. Utilice un editor de texto para crear un archivo de configuración con el nombre `config.yaml` para proporcionárselo al instalador.

   Por ejemplo, en un sistema basado en Linux, puede ejecutar el siguiente comando para usar GNU nano para crear el archivo.

   ```
   nano GreengrassInstaller/config.yaml
   ```

   Copia el siguiente contenido de YAML en el archivo. Este archivo de configuración parcial especifica los parámetros del sistema y los parámetros del núcleo de 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"
```

A continuación, proceda del modo siguiente:

- Sustituya cada instancia de */greengrass/v2* por la carpeta raíz de Greengrass.
- Sustituya *MyGreengrassCore* por el nombre de la AWS IoT cosa.
- Sustituya la versión *2.12.6* por la versión del software AWS IoT Greengrass Core.
- Sustituya *us-west-2* por Región de AWS el lugar donde creó los recursos.
- *GreengrassCoreTokenExchangeRoleAlias*Sustitúyalo por el nombre del alias de la función de intercambio de fichas.
- `iotDataEndpoint`Sustitúyalo por el punto final de AWS IoT datos.
- Sustituya el `iotCredEndpoint` punto final por el de sus AWS IoT credenciales.

> **Note**
>
> En este archivo de configuración, puede personalizar otras opciones de configuración de Nucleus, como los puertos y el proxy de red que se van a utilizar, como se muestra en el siguiente ejemplo. Para obtener más información, consulte Configuración del núcleo de 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. Ejecute el instalador y especifique si desea `--init-config` proporcionar el archivo de configuración.

   - Sustituya */greengrass/v2* o *C:\greengrass\v2* por la carpeta raíz de Greengrass.
   - Sustituya cada instancia de por *GreengrassInstaller*la carpeta en la que desempaquetó el instalador.

   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**
>
> En los dispositivos principales de Windows, debe especificar si `--setup-system-service true` desea configurar el software AWS IoT Greengrass principal como un servicio del sistema.

Si lo especifica`--setup-system-service true`, el instalador `Successfully set up Nucleus as a system service` imprimirá si configuró y ejecutó el software como un servicio del sistema. De lo contrario, el instalador no mostrará ningún mensaje si instala el software correctamente.

> **Note**
>
> No puede usar el `deploy-dev-tools` argumento para implementar herramientas de desarrollo local cuando ejecuta el instalador sin el `--provision true` argumento. Para obtener información sobre cómo implementar la CLI de Greengrass directamente en su dispositivo, consulte. Interfaz de línea de comandos Greengrass

4. Verifique la instalación consultando los archivos de la carpeta raíz.

Linux or Unix

```
ls /greengrass/v2
```

Windows Command Prompt (CMD)

```
dir C:\greengrass\v2
```

PowerShell

```
ls C:\greengrass\v2
```

Si la instalación se realizó correctamente, la carpeta raíz contiene varias carpetas`config`, como`packages`, `ylogs`.

Instale el software AWS IoT Greengrass principal con la clave privada y el certificado en un HSM

> **Note**
>
> Esta función está disponible para la versión 2.5.3 y versiones posteriores del componente núcleo de Greengrass. AWS IoT Greengrass actualmente no admite esta función en los dispositivos principales de Windows.

Para instalar el software AWS IoT Greengrass principal

1. Compruebe la versión del software AWS IoT Greengrass principal.

   - *`GreengrassInstaller`*Sustitúyala por la ruta a la carpeta que contiene el software.

   ```
   java -jar ./GreengrassInstaller/lib/Greengrass.jar --version
   ```

2. Para permitir que el software AWS IoT Greengrass principal utilice la clave privada y el certificado del HSM, instale el componente de proveedor PKCS #11 al instalar el software AWS IoT Greengrass principal. El componente de proveedor PKCS #11 es un complemento que

puede configurar durante la instalación. Puede descargar la última versión del componente de proveedor PKCS #11 desde la siguiente ubicación:

- https://d2s8p88vqu9w66.cloudfront.net/releases/Pkcs11Provider/aws.greengrass.crypto.Pkcs11Provider-latest.jar

Descargue el complemento del proveedor PKCS #11 en un archivo denominado`aws.greengrass.crypto.Pkcs11Provider.jar`. *`GreengrassInstaller`*Sustitúyalo por la carpeta que desee usar.

```
curl -s https://d2s8p88vqu9w66.cloudfront.net/releases/Pkcs11Provider/
aws.greengrass.crypto.Pkcs11Provider-latest.jar > GreengrassInstaller/
aws.greengrass.crypto.Pkcs11Provider.jar
```

Al descargar este software, acepta el acuerdo de licencia del software de Greengrass Core.

3. Utilice un editor de texto para crear un archivo de configuración con el nombre `config.yaml` para proporcionárselo al instalador.

   Por ejemplo, en un sistema basado en Linux, puede ejecutar el siguiente comando para usar GNU nano para crear el archivo.

   ```
   nano GreengrassInstaller/config.yaml
   ```

   Copia el siguiente contenido de YAML en el archivo. Este archivo de configuración parcial especifica los parámetros del sistema, los parámetros del núcleo de Greengrass y los parámetros del proveedor 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"
```

A continuación, proceda del modo siguiente:

- Sustituya cada instancia de *iotdevicekey* de los URI del PKCS #11 por la etiqueta de objeto en la que creó la clave privada e importó el certificado.
- Sustituya cada instancia de */greengrass/v2* por la carpeta raíz de Greengrass.
- Sustituya *MyGreengrassCore* por el nombre de la AWS IoT cosa.
- Sustituya la versión *2.12.6* por la versión del software AWS IoT Greengrass Core.
- Sustituya *us-west-2* por Región de AWS el lugar donde creó los recursos.
- *GreengrassCoreTokenExchangeRoleAlias*Sustitúyalo por el nombre del alias de la función de intercambio de fichas.
- `iotDataEndpoint`Sustitúyalo por el punto final de AWS IoT datos.
- Sustituya el `iotCredEndpoint` punto final por el de sus AWS IoT credenciales.
- Sustituya los parámetros de configuración del `aws.greengrass.crypto.Pkcs11Provider` componente por los valores de la configuración del HSM en el dispositivo principal.

> **Note**
>
> En este archivo de configuración, puede personalizar otras opciones de configuración del núcleo, como los puertos y el proxy de red que se van a utilizar, como se muestra en el siguiente ejemplo. Para obtener más información, consulte Configuración del núcleo de 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. Ejecute el instalador y especifique si desea `--init-config` proporcionar el archivo de configuración.

   - */greengrass/v2*Sustitúyala por la carpeta raíz de Greengrass.
   - Sustituya cada instancia de por *GreengrassInstaller*la carpeta en la que desempaquetó el instalador.

```
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**
>
> En los dispositivos principales de Windows, debe especificar si `--setup-system-service true` desea configurar el software AWS IoT Greengrass principal como un servicio del sistema.

Si lo especifica`--setup-system-service true`, el instalador `Successfully set up Nucleus as a system service` imprimirá si configuró y ejecutó el software como un servicio del sistema. De lo contrario, el instalador no mostrará ningún mensaje si instala el software correctamente.

> **Note**
>
> No puede usar el `deploy-dev-tools` argumento para implementar herramientas de desarrollo local cuando ejecuta el instalador sin el `--provision true` argumento. Para obtener información sobre cómo implementar la CLI de Greengrass directamente en su dispositivo, consulte. Interfaz de línea de comandos Greengrass

5. Verifique la instalación consultando los archivos de la carpeta raíz.

   Linux or Unix

   ```
   ls /greengrass/v2
   ```

   Windows Command Prompt (CMD)

   ```
   dir C:\greengrass\v2
   ```

   PowerShell

   ```
   ls C:\greengrass\v2
   ```

Si la instalación se realizó correctamente, la carpeta raíz contiene varias carpetas`config`, como`packages`, `ylogs`.

Si instaló el software AWS IoT Greengrass principal como un servicio del sistema, el instalador ejecutará el software automáticamente. De lo contrario, debe ejecutar el software manualmente. Para obtener más información, consulte Ejecute el software AWS IoT Greengrass principal.

Para obtener más información acerca de cómo configurar y utilizar el software AWS IoT Greengrass, consulte lo siguiente:

- Configurar el software AWS IoT Greengrass principal
- Desarrolle AWS IoT Greengrass componentes
- Implemente AWS IoT Greengrass componentes en los dispositivos
- Interfaz de línea de comandos Greengrass

Instale el software AWS IoT Greengrass principal con aprovisionamiento AWS IoT de flota

Esta función está disponible para la versión 2.4.0 y versiones posteriores del componente núcleo de Greengrass.

Con el aprovisionamiento de AWS IoT flotas, puede configurarlo AWS IoT para generar y entregar de forma segura los certificados de dispositivo X.509 y las claves privadas a sus dispositivos cuando se conectan a AWS IoT ellos por primera vez. AWS IoT proporciona certificados de cliente firmados por la autoridad de certificación (CA) raíz de Amazon. También puede configurarlo AWS IoT para especificar grupos de cosas, tipos de cosas y permisos para los dispositivos principales de Greengrass que aprovisiona con el aprovisionamiento de flotas. Defina una plantilla de aprovisionamiento para definir cómo AWS IoT aprovisiona cada dispositivo. La plantilla de aprovisionamiento especifica el objeto, la política y los recursos de certificado que se van a crear para un dispositivo durante el aprovisionamiento. Para obtener más información, consulte las plantillas de aprovisionamiento en la AWS IoT Core Guía para desarrolladores.

AWS IoT Greengrass proporciona un complemento de aprovisionamiento de AWS IoT flotas que puede usar para instalar el software AWS IoT Greengrass principal utilizando los AWS recursos creados por el aprovisionamiento de AWS IoT flotas. El complemento de aprovisionamiento de

flotas utiliza el aprovisionamiento por reclamación. Los dispositivos utilizan un certificado de aprovisionamiento y una clave privada para obtener un certificado de dispositivo X.509 y una clave privada únicos que pueden utilizar para sus operaciones habituales. Puede incrustar el certificado de reclamación y la clave privada en cada dispositivo durante la fabricación, de modo que sus clientes puedan activar los dispositivos más adelante, cuando cada dispositivo entre en funcionamiento. Puede utilizar el mismo certificado de reclamación y la misma clave privada para varios dispositivos. Para obtener más información, consulte Aprovisionamiento por reclamación en la Guía para AWS IoT Core desarrolladores.

> **Note**
>
> Actualmente, el complemento de aprovisionamiento de flotas no admite el almacenamiento de archivos de certificados y claves privadas en un módulo de seguridad de hardware (HSM). Para usar un HSM, instala el software AWS IoT Greengrass principal con aprovisionamiento manual.

Para instalar el software AWS IoT Greengrass Core con el aprovisionamiento de AWS IoT flota, debe configurar los recursos Cuenta de AWS que AWS IoT utiliza para aprovisionar los dispositivos principales de Greengrass. Estos recursos incluyen una plantilla de aprovisionamiento, certificados de reclamación y una función de IAM para el intercambio de fichas. Después de crear estos recursos, puede reutilizarlos para aprovisionar varios dispositivos principales de una flota. Para obtener más información, consulte Configurar el aprovisionamiento de AWS IoT flota para los dispositivos principales de Greengrass.

> **Important**
>
> Antes de descargar el software AWS IoT Greengrass Core, compruebe que su dispositivo principal cumpla los requisitos para instalar y ejecutar el software AWS IoT Greengrass Core v2.0.

Temas

- Descargue el software AWS IoT Greengrass principal
- Descargue el complemento de aprovisionamiento AWS IoT de flotas
- Instale el software AWS IoT Greengrass principal
- Configurar el aprovisionamiento de AWS IoT flota para los dispositivos principales de Greengrass
- Configurar el complemento de aprovisionamiento de AWS IoT flotas
- AWS IoT registro de cambios del complemento de aprovisionamiento de flotas

Requisitos previos

Para instalar el software AWS IoT Greengrass Core con el aprovisionamiento de AWS IoT flota, primero debe configurar el aprovisionamiento de AWS IoT flota para los dispositivos principales de Greengrass. Tras completar estos pasos una vez, puede utilizar el aprovisionamiento de flotas para instalar el software AWS IoT Greengrass Core en cualquier número de dispositivos.

Recupere los puntos finales AWS IoT

Obtenga sus AWS IoT Cuenta de AWS puntos finales y guárdelos para usarlos más tarde. El dispositivo utiliza estos puntos finales para conectarse a ellos. AWS IoT Haga lo siguiente:

1. Obtenga el punto final AWS IoT de datos para su. Cuenta de AWS

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

   La respuesta es similar a la del siguiente ejemplo, si la solicitud se realiza correctamente.

   ```
   {
     "endpointAddress": "device-data-prefix-ats.iot.us-west-2.amazonaws.com"
   }
   ```

2. Obtenga el punto final de AWS IoT credenciales para su. Cuenta de AWS

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

   La respuesta es similar a la del siguiente ejemplo, si la solicitud se realiza correctamente.

   ```
   {
     "endpointAddress": "device-credentials-prefix.credentials.iot.us-
   west-2.amazonaws.com"
   ```

```
}
```

Descargue los certificados al dispositivo

El dispositivo utiliza un certificado de reclamación y una clave privada para autenticar su solicitud de aprovisionamiento de AWS recursos y adquirir un certificado de dispositivo X.509. Puede incrustar el certificado de reclamación y la clave privada en el dispositivo durante la fabricación o copiar el certificado y la clave en el dispositivo durante la instalación. En esta sección, debe copiar el certificado de reclamación y la clave privada en el dispositivo. También descargas el certificado de la autoridad de certificación (CA) raíz de Amazon en el dispositivo.

> **Important**
>
> El aprovisionamiento afirma que las claves privadas deben estar protegidas en todo momento, incluso en los dispositivos principales de Greengrass. Te recomendamos que utilices CloudWatch las estadísticas y los registros de Amazon para detectar indicios de uso indebido, como el uso no autorizado del certificado de reclamación para aprovisionar dispositivos. Si detectas un uso indebido, desactiva el certificado de notificación de aprovisionamiento para que no se pueda utilizar para el aprovisionamiento de dispositivos. Para obtener más información, consulte Monitorear AWS IoT en la Guía para desarrolladores de AWS IoT Core .
>
> Para ayudarte a gestionar mejor el número de dispositivos y los dispositivos que se registran automáticamente en el tuyo Cuenta de AWS, puedes especificar un enlace previo al aprovisionamiento al crear una plantilla de aprovisionamiento de flotas. Un enlace de preaprovisionamiento es una AWS Lambda función que valida los parámetros de plantilla que proporcionan los dispositivos durante el registro. Por ejemplo, puede crear un enlace de preaprovisionamiento que compare el ID de un dispositivo con una base de datos para comprobar que el dispositivo tiene permiso de aprovisionamiento. Para obtener más información, consulta los ganchos de aprovisionamiento previo en la Guía para desarrolladores.AWS IoT Core

Para descargar los certificados de reclamación al dispositivo

1. Copie el certificado de reclamación y la clave privada en el dispositivo. Si SSH y SCP están habilitados en el ordenador de desarrollo y en el dispositivo, puede utilizar el `scp` comando del ordenador de desarrollo para transferir el certificado de reclamación y la clave privada.

El siguiente comando de ejemplo transfiere estos archivos al dispositivo a una carpeta denominada `claim-certs` en el ordenador de desarrollo. Sustituya *la dirección IP del dispositivo* por la dirección IP de su dispositivo.

```
scp -r claim-certs/ device-ip-address:~
```

2. Cree la carpeta raíz de Greengrass en el dispositivo. Más adelante instalará el software AWS IoT Greengrass principal en esta carpeta.

   Linux or Unix

   - */greengrass/v2*Sustitúyalo por la carpeta que desee utilizar.

   ```
   sudo mkdir -p /greengrass/v2
   ```

   Windows Command Prompt

   - Sustituya *C:\greengrass\v2* por la carpeta que desee utilizar.

   ```
   mkdir C:\greengrass\v2
   ```

   PowerShell

   - Sustituya *C:\greengrass\v2* por la carpeta que desee utilizar.

   ```
   mkdir C:\greengrass\v2
   ```

3. (Solo para Linux) Establezca los permisos de la carpeta principal de la raíz de Greengrass.

   - Sustituya */greengrass* por el archivo principal de la carpeta raíz.

   ```
   sudo chmod 755 /greengrass
   ```

4. Mueva los certificados de reclamación a la carpeta raíz de Greengrass.

   - Sustituya */greengrass/v2* o *C:\greengrass\v2* por la carpeta raíz de 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. Descarga el certificado de la autoridad de certificación raíz (CA) de Amazon. AWS IoT Los certificados están asociados al certificado de CA raíz de Amazon de forma predeterminada.

   Linux or Unix

   ```
   sudo curl -o /greengrass/v2/AmazonRootCA1.pem https://www.amazontrust.com/
   repository/AmazonRootCA1.pem
   ```

   Windows Command Prompt (CMD)

   ```
   curl -o C:\greengrass\v2\\AmazonRootCA1.pem https://www.amazontrust.com/
   repository/AmazonRootCA1.pem
   ```

   PowerShell

   ```
   iwr -Uri https://www.amazontrust.com/repository/AmazonRootCA1.pem -OutFile C:
   \greengrass\v2\\AmazonRootCA1.pem
   ```

Configura el entorno del dispositivo

Siga los pasos de esta sección para configurar un dispositivo Linux o Windows para usarlo como dispositivo AWS IoT Greengrass principal.

Configura un dispositivo Linux

Para configurar un dispositivo Linux para AWS IoT Greengrass V2

1. Instale el motor de ejecución de Java, que el software AWS IoT Greengrass principal necesita para ejecutarse. Le recomendamos que utilice las versiones de soporte a largo plazo de Amazon Corretto u OpenJDK. Se requiere la versión 8 o superior. Los siguientes comandos muestran cómo instalar OpenJDK en su dispositivo.

   - Para distribuciones basadas en Debian o en Ubuntu:

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

   - Para distribuciones basadas en Red Hat:

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

   - En Amazon Linux 2:

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

   - En Amazon Linux 2023:

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

   Cuando se complete la instalación, ejecute el siguiente comando para comprobar que Java se ejecuta en su dispositivo Linux.

   ```
   java -version
   ```

   El comando imprime la versión de Java que se ejecuta en el dispositivo. Por ejemplo, en una distribución basada en Debian, el resultado podría tener un aspecto similar al del siguiente ejemplo.

   ```
   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. (Opcional) Cree el usuario y el grupo predeterminados del sistema que ejecutan los componentes del dispositivo. También puede optar por permitir que el instalador del software AWS IoT Greengrass principal cree este usuario y grupo durante la instalación con el argumento del `--component-default-user` instalador. Para obtener más información, consulte Argumentos de instalación.

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

3. Compruebe que el usuario que ejecuta el software AWS IoT Greengrass principal (normalmente`root`) tiene permiso para ejecutar `sudo` con cualquier usuario y grupo.

   a. Ejecute el siguiente comando para abrir el `/etc/sudoers` archivo.

      ```
      sudo visudo
      ```

   b. Compruebe que el permiso del usuario es similar al del ejemplo siguiente.

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

4. (Opcional) Para ejecutar funciones Lambda en contenedores, debe habilitar cgroups v1 y debe habilitar y montar los cgroups de memoria y dispositivos. Si no planea ejecutar funciones Lambda en contenedores, puede omitir este paso.

   Para habilitar estas opciones de cgroups, arranque el dispositivo con los siguientes parámetros del núcleo de Linux.

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

   Para obtener información sobre cómo ver y configurar los parámetros del núcleo de su dispositivo, consulte la documentación del sistema operativo y del gestor de arranque. Siga las instrucciones para configurar permanentemente los parámetros del núcleo.

5. Instale todas las demás dependencias necesarias en su dispositivo tal y como se indica en Requisitos de los dispositivos la lista de requisitos de.

Configura un dispositivo Windows

> **Note**
>
> Esta función está disponible para la versión 2.5.0 y versiones posteriores del componente núcleo de Greengrass.

Para configurar un dispositivo Windows para AWS IoT Greengrass V2

1. Instale el motor de ejecución de Java, que el software AWS IoT Greengrass principal necesita para ejecutarse. Le recomendamos que utilice las versiones de soporte a largo plazo de Amazon Corretto u OpenJDK. Se requiere la versión 8 o superior.
2. Compruebe si Java está disponible en la variable de sistema PATH y agréguelo si no lo está. La LocalSystem cuenta ejecuta el software AWS IoT Greengrass principal, por lo que debe añadir Java a la variable de sistema PATH en lugar de a la variable de usuario PATH de su usuario. Haga lo siguiente:
   a. Pulse la tecla Windows para abrir el menú de inicio.
   b. Escriba **`environment variables`** para buscar las opciones del sistema en el menú de inicio.
   c. En los resultados de la búsqueda del menú de inicio, seleccione Editar las variables de entorno del sistema para abrir la ventana de propiedades del sistema.
   d. Seleccione Variables de entorno... para abrir la ventana Variables de entorno.
   e. En Variables de sistema, seleccione Ruta y, a continuación, elija Editar. En la ventana Editar variables de entorno, puede ver cada ruta en una línea independiente.
   f. Compruebe si la ruta a la `bin` carpeta de la instalación de Java está presente. La ruta puede tener un aspecto similar al del siguiente ejemplo.

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

   g. Si la `bin` carpeta de la instalación de Java no aparece en Path, seleccione Nueva para añadirla y, a continuación, pulse Aceptar.
3. Abre la línea de comandos de Windows (`cmd.exe`) como administrador.
4. Cree el usuario predeterminado en la LocalSystem cuenta del dispositivo Windows. Sustituya la *`contraseña`* por una contraseña segura.

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

> **Tip**
>
> Según la configuración de Windows, es posible que la contraseña del usuario caduque en una fecha futura. Para garantizar que sus aplicaciones de Greengrass sigan funcionando, controle cuándo caduque la contraseña y actualícela antes de que caduque. También puede configurar la contraseña del usuario para que nunca caduque.
>
> - Para comprobar cuándo caducan un usuario y su contraseña, ejecuta el siguiente comando.
>
>   ```
>   net user ggc_user | findstr /C:expires
>   ```
>
> - Para configurar la contraseña de un usuario para que no caduque nunca, ejecute el siguiente comando.
>
>   ```
>   wmic UserAccount where "Name='ggc_user'" set PasswordExpires=False
>   ```
>
> - Si utilizas Windows 10 o una versión posterior, donde el wmiccomando está obsoleto, ejecuta el siguiente PowerShell comando.
>
>   ```
>   Get-CimInstance -Query "SELECT * from Win32_UserAccount WHERE name =
>    'ggc_user'" | Set-CimInstance -Property @{PasswordExpires="False"}
>   ```

5. Descargue e instale la PsExecutilidad de Microsoft en el dispositivo.
6. Utilice la PsExec utilidad para almacenar el nombre de usuario y la contraseña del usuario predeterminado en la instancia de Credential Manager de la LocalSystem cuenta. Sustituya la *contraseña* por la contraseña del usuario que configuró anteriormente.

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

   Si se PsExec License Agreementabre, Acceptacepte la licencia y ejecute el comando.

> **Note**
>
> En los dispositivos Windows, la LocalSystem cuenta ejecuta el núcleo de Greengrass y debe usar la PsExec utilidad para almacenar la información de usuario predeterminada en la LocalSystem cuenta. El uso de la aplicación Credential Manager almacena esta información en la cuenta de Windows del usuario que ha iniciado sesión actualmente, en lugar de en la LocalSystem cuenta.

Descargue el software AWS IoT Greengrass principal

Puede descargar la última versión del software AWS IoT Greengrass Core desde la siguiente ubicación:

- https://d2s8p88vqu9w66.cloudfront.net/releases/greengrass-nucleus-latest.zip

> **Note**
>
> Puede descargar una versión específica del software AWS IoT Greengrass Core desde la siguiente ubicación. Sustituya la *versión* por la versión que desea descargar.
>
> ```
> https://d2s8p88vqu9w66.cloudfront.net/releases/greengrass-version.zip
> ```

Para descargar el software AWS IoT Greengrass principal

1. En su dispositivo principal, descargue el software AWS IoT Greengrass Core en un archivo denominado`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
```

Al descargar este software, acepta el acuerdo de licencia del software de Greengrass Core.

2. (Opcional) Para verificar la firma del software Greengrass Nucleus

> **Note**
>
> Esta función está disponible con la versión 2.9.5 y posteriores del núcleo de Greengrass.

a. Usa el siguiente comando para verificar la firma del artefacto núcleo de Greengrass:

Linux or Unix

```
jarsigner -verify -certs -verbose greengrass-nucleus-latest.zip
```

Windows Command Prompt (CMD)

El nombre del archivo puede tener un aspecto diferente según la versión de JDK que instale. *jdk17.0.6_10*Sustitúyalo por la versión de JDK que instaló.

```
"C:\\Program Files\\Amazon Corretto\\jdk17.0.6_10\\bin\\jarsigner.exe" -
verify -certs -verbose greengrass-nucleus-latest.zip
```

PowerShell

El nombre del archivo puede tener un aspecto diferente en función de la versión de JDK que instale. *jdk17.0.6_10*Sustitúyalo por la versión de JDK que instaló.

```
'C:\\Program Files\\Amazon Corretto\\jdk17.0.6_10\\bin\\jarsigner.exe' -
verify -certs -verbose greengrass-nucleus-latest.zip
```

b. La `jarsigner` invocación produce un resultado que indica los resultados de la verificación.

   i. Si el archivo zip del núcleo de Greengrass está firmado, el resultado contiene la siguiente declaración:

   ```
   jar verified.
   ```

   ii. Si el archivo zip del núcleo de Greengrass no está firmado, el resultado contiene la siguiente declaración:

   ```
   jar is unsigned.
   ```

c. Si ha proporcionado la `-certs` opción Jarsigner junto con `-verbose` las opciones `-verify` y, el resultado también incluye información detallada del certificado de firmante.

3. Descomprime el software AWS IoT Greengrass principal en una carpeta de tu dispositivo. *`GreengrassInstaller`*Sustitúyalo por la carpeta que desee usar.

   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. (Opcional) Ejecute el siguiente comando para ver la versión del software AWS IoT Greengrass principal.

```
java -jar ./GreengrassInstaller/lib/Greengrass.jar --version
```

Important

Si instala una versión del núcleo de Greengrass anterior a la v2.4.0, no elimine esta carpeta después de instalar el software Core. AWS IoT Greengrass El software AWS IoT Greengrass Core utiliza los archivos de esta carpeta para ejecutarse.
Si descargó la última versión del software, instale la versión 2.4.0 o posterior y podrá eliminar esta carpeta después de instalar el software AWS IoT Greengrass principal.

Descargue el complemento de aprovisionamiento AWS IoT de flotas

Puedes descargar la última versión del complemento de aprovisionamiento de AWS IoT flotas desde la siguiente ubicación:

- https://d2s8p88vqu9w66.cloudfront.net/releases/aws-greengrass - /fleetprovisioningbyclaim-latest.jar FleetProvisioning ByClaim

Note

Puede descargar una versión específica del complemento de aprovisionamiento de AWS IoT flotas desde la siguiente ubicación. Sustituya la *versión* por la versión que desea descargar. Para obtener más información sobre cada versión del complemento de aprovisionamiento de flotas, consulteAWS IoT registro de cambios del complemento de aprovisionamiento de flotas.

```
https://d2s8p88vqu9w66.cloudfront.net/releases/aws-greengrass-
FleetProvisioningByClaim/fleetprovisioningbyclaim-version.jar
```

El complemento de aprovisionamiento de flotas es de código abierto. Para ver su código fuente, consulta el complemento de aprovisionamiento de AWS IoT flotas en. GitHub

Para descargar el complemento de aprovisionamiento de AWS IoT flotas

- En su dispositivo, descargue el complemento de aprovisionamiento de AWS IoT flotas en un archivo denominado. `aws.greengrass.FleetProvisioningByClaim.jar` *`GreengrassInstaller`*Sustitúyalo por la carpeta que desee usar.

  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
  ```

  Al descargar este software, acepta el acuerdo de licencia del software de Greengrass Core.

Instale el software AWS IoT Greengrass principal

Ejecute el instalador con argumentos que especifiquen las siguientes acciones:

- Realice la instalación desde un archivo de configuración parcial que especifique el uso del complemento de aprovisionamiento de flotas para aprovisionar AWS recursos. El software AWS IoT Greengrass Core utiliza un archivo de configuración que especifica la configuración de todos los componentes de Greengrass del dispositivo. El instalador crea un archivo de configuración completo a partir del archivo de configuración parcial que usted proporciona y de AWS los recursos que crea el complemento de aprovisionamiento de flotas.
- Especifique si desea utilizar el usuario `ggc_user` del sistema para ejecutar los componentes de software en el dispositivo principal. En los dispositivos Linux, este comando también especifica

el uso del grupo `ggc_group` del sistema, y el instalador crea el usuario y el grupo del sistema automáticamente.

- Configure el software AWS IoT Greengrass principal como un servicio del sistema que se ejecute durante el arranque. En los dispositivos Linux, esto requiere el sistema de inicio Systemd.

> **Important**
>
> En los dispositivos principales de Windows, debe configurar el software AWS IoT Greengrass Core como un servicio del sistema.

Para obtener más información sobre los argumentos que puede especificar, consulteArgumentos de instalación.

> **Note**
>
> Si utiliza un AWS IoT Greengrass dispositivo con memoria limitada, puede controlar la cantidad de memoria que utiliza el software AWS IoT Greengrass Core. Para controlar la asignación de memoria, puede configurar las opciones de tamaño de pila de la JVM en el parámetro de `jvmOptions` configuración del componente core. Para obtener más información, consulte Controle la asignación de memoria con las opciones de JVM.

Para instalar el software Core AWS IoT Greengrass

1. Compruebe la versión del software AWS IoT Greengrass principal.

   - *`GreengrassInstaller`*Sustitúyala por la ruta a la carpeta que contiene el software.

   ```
   java -jar ./GreengrassInstaller/lib/Greengrass.jar --version
   ```

2. Utilice un editor de texto para crear un archivo de configuración con el nombre `config.yaml` para proporcionárselo al instalador.

   Por ejemplo, en un sistema basado en Linux, puede ejecutar el siguiente comando para usar GNU nano para crear el archivo.

   ```
   nano GreengrassInstaller/config.yaml
   ```

Copia el siguiente contenido de YAML en el archivo. Este archivo de configuración parcial especifica los parámetros del complemento de aprovisionamiento de flotas. Para obtener más información sobre las opciones que puede especificar, consulteConfigurar el complemento de aprovisionamiento de AWS IoT flotas.

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"
```

A continuación, proceda del modo siguiente:

- Sustituya la versión *2.12.6* por la versión del software AWS IoT Greengrass Core.
- Sustituya cada instancia de */greengrass/v2* o *C:\greengrass\v2* por la carpeta raíz de Greengrass.

> **Note**
>
> En los dispositivos Windows, debe especificar los separadores de rutas como barras invertidas dobles (\\), como. `C:\\greengrass\\v2`

- Sustituya *us-west-2* por AWS la región en la que creó la plantilla de aprovisionamiento y otros recursos.
- Sustitúyalo por su `iotDataEndpoint` punto final de datos. AWS IoT
- Sustituya el `iotCredentialEndpoint` punto final por el de sus AWS IoT credenciales.
- *GreengrassCoreTokenExchangeRoleAlias*Sustitúyalo por el nombre del alias de la función de intercambio de fichas.
- *GreengrassFleetProvisioningTemplate*Sustitúyalo por el nombre de la plantilla de aprovisionamiento de flota.
- Sustitúyala por la ruta al certificado de reclamación del dispositivo. `claimCertificatePath`
- Sustitúyala por la ruta a la clave privada del certificado de reclamación del dispositivo. `claimCertificatePrivateKeyPath`
- Sustituya los parámetros de la plantilla (`templateParameters`) por los valores que se utilizarán para aprovisionar el dispositivo. Este ejemplo hace referencia a la plantilla de ejemplo que define `ThingName` los `ThingGroupName` parámetros.

Note

En este archivo de configuración, puede personalizar otras opciones de configuración, como los puertos y el proxy de red que se van a utilizar, como se muestra en el siguiente ejemplo. Para obtener más información, consulte Configuración del núcleo de 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"
```

Para usar un proxy HTTPS, debe usar la versión 1.1.0 o posterior del complemento de aprovisionamiento de flotas. Además, debe especificar lo siguiente `rootCaPathsystem`, como se muestra en el siguiente ejemplo.

Linux or Unix

```
---
system:
  rootCaPath: "/greengrass/v2/AmazonRootCA1.pem"
services:
  ...
```

Windows

```
---
system:
  rootCaPath: "C:\\greengrass\\v2\\AmazonRootCA1.pem"
services:
  ...
```

3. Ejecute el instalador. Especifique si `--trusted-plugin` desea proporcionar el complemento de aprovisionamiento de flotas y especifique si `--init-config` desea proporcionar el archivo de configuración.

   - */greengrass/v2*Sustitúyala por la carpeta raíz de Greengrass.
   - Sustituya cada instancia de por *GreengrassInstaller*la carpeta en la que desempaquetó el instalador.

   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

En los dispositivos principales de Windows, debe especificar si `--setup-system-service true` desea configurar el software AWS IoT Greengrass principal como un servicio del sistema.

Si lo especifica`--setup-system-service true`, el instalador `Successfully set up Nucleus as a system service` imprimirá si configuró y ejecutó el software como un servicio del sistema. De lo contrario, el instalador no mostrará ningún mensaje si instala el software correctamente.

Note

No puede usar el `deploy-dev-tools` argumento para implementar herramientas de desarrollo local cuando ejecuta el instalador sin el `--provision true` argumento.

Para obtener información sobre cómo implementar la CLI de Greengrass directamente en su dispositivo, consulte. Interfaz de línea de comandos Greengrass

4. Verifique la instalación consultando los archivos de la carpeta raíz.

   Linux or Unix

   ```
   ls /greengrass/v2
   ```

   Windows Command Prompt (CMD)

   ```
   dir C:\greengrass\v2
   ```

   PowerShell

   ```
   ls C:\greengrass\v2
   ```

   Si la instalación se realizó correctamente, la carpeta raíz contiene varias carpetas`config`, como`packages`, `ylogs`.

Si instaló el software AWS IoT Greengrass principal como un servicio del sistema, el instalador ejecutará el software automáticamente. De lo contrario, debe ejecutar el software manualmente. Para obtener más información, consulte Ejecute el software AWS IoT Greengrass principal.

Para obtener más información acerca de cómo configurar y utilizar el software AWS IoT Greengrass, consulte lo siguiente:

- Configurar el software AWS IoT Greengrass principal
- Desarrolle AWS IoT Greengrass componentes
- Implemente AWS IoT Greengrass componentes en los dispositivos
- Interfaz de línea de comandos Greengrass

Configurar el aprovisionamiento de AWS IoT flota para los dispositivos principales de Greengrass

Para instalar el software AWS IoT Greengrass Core con el aprovisionamiento de flotas, primero debe configurar los siguientes recursos en su. Cuenta de AWS Estos recursos permiten que los dispositivos se registren AWS IoT y funcionen como dispositivos principales de Greengrass. Siga los pasos de esta sección una vez para crear y configurar estos recursos en suCuenta de AWS.

- Una función de IAM de intercambio de fichas, que los dispositivos principales utilizan para autorizar las llamadas a AWS los servicios.
- Un alias de AWS IoT función que apunta a la función de intercambio de fichas.
- (Opcional) Una AWS IoT política que los dispositivos principales utilizan para autorizar las llamadas a los AWS IoT Greengrass servicios AWS IoT y. Esta AWS IoT política debe permitir el `iot:AssumeRoleWithCertificate` permiso para el alias de la AWS IoT función que apunta a la función de intercambio de fichas.

  Puede usar una AWS IoT política única para todos los dispositivos principales de su flota, o puede configurar la plantilla de aprovisionamiento de la flota para crear una AWS IoT política para cada dispositivo principal.
- Una plantilla de aprovisionamiento de AWS IoT flota. Esta plantilla debe especificar lo siguiente:
  - Cualquier AWS IoT cosa, un recurso. Puede especificar una lista de grupos de cosas existentes para implementar componentes en cada dispositivo cuando esté en línea.
  - Un recurso AWS IoT de políticas. Este recurso puede definir una de las siguientes propiedades:
    - El nombre de una AWS IoT política existente. Si elige esta opción, los dispositivos principales que cree a partir de esta plantilla utilizarán la misma AWS IoT política y podrá gestionar sus permisos como una flota.
    - Un documento AWS IoT de política. Si elige esta opción, cada dispositivo principal que cree a partir de esta plantilla utilizará una AWS IoT política única y podrá administrar los permisos de cada dispositivo principal individual.
  - Un recurso AWS IoT de certificados. Este recurso de certificado debe usar el `AWS::IoT::Certificate::Id` parámetro para adjuntar el certificado al dispositivo principal. Para obtener más información, consulte el ust-in-time aprovisionamiento de J en la Guía para AWS IoT desarrolladores.
- Un certificado de AWS IoT aprovisionamiento y una clave privada para la plantilla de aprovisionamiento de la flota. Puede incrustar este certificado y esta clave privada en los

dispositivos durante la fabricación, de modo que los dispositivos puedan registrarse y aprovisionarse por sí mismos cuando se conecten a Internet.

> **Important**
>
> El aprovisionamiento afirma que las claves privadas deben estar protegidas en todo momento, incluso en los dispositivos principales de Greengrass. Te recomendamos que utilices CloudWatch las estadísticas y los registros de Amazon para detectar indicios de uso indebido, como el uso no autorizado del certificado de reclamación para aprovisionar dispositivos. Si detectas un uso indebido, desactiva el certificado de notificación de aprovisionamiento para que no se pueda utilizar para el aprovisionamiento de dispositivos. Para obtener más información, consulte Monitorear AWS IoT en la Guía para desarrolladores de AWS IoT Core.
> Para ayudarte a gestionar mejor la cantidad de dispositivos y los dispositivos que se registran automáticamente en el tuyoCuenta de AWS, puedes especificar un enlace previo al aprovisionamiento al crear una plantilla de aprovisionamiento de flotas. Un enlace de preaprovisionamiento es una AWS Lambda función que valida los parámetros de plantilla que proporcionan los dispositivos durante el registro. Por ejemplo, puede crear un enlace de preaprovisionamiento que compare el ID de un dispositivo con una base de datos para comprobar que el dispositivo tiene permiso de aprovisionamiento. Para obtener más información, consulta los ganchos de aprovisionamiento previo en la Guía para desarrolladores. AWS IoT Core

- Una AWS IoT política que se adjunta al certificado de notificación de aprovisionamiento para permitir que los dispositivos se registren y utilicen la plantilla de aprovisionamiento de flotas.

Temas

- Crea una función de intercambio de fichas
- Creación de una política de AWS IoT
- Cree una plantilla de aprovisionamiento de flota
- Cree un certificado de notificación de aprovisionamiento y una clave privada

Crea una función de intercambio de fichas

Los dispositivos principales de Greengrass utilizan una función de servicio de IAM, denominada función de intercambio de fichas, para autorizar las llamadas a los servicios. AWS El dispositivo

utiliza el proveedor de AWS IoT credenciales para obtener AWS credenciales temporales para esta función, lo que permite al dispositivo interactuar con Amazon LogsAWS IoT, enviar registros a Amazon CloudWatch Logs y descargar artefactos de componentes personalizados de Amazon S3. Para obtener más información, consulte Autorizar a los dispositivos principales a interactuar con AWS los servicios.

Se utiliza un alias de AWS IoT rol para configurar el rol de intercambio de fichas para los dispositivos principales de Greengrass. Los alias de rol le permiten cambiar el rol de intercambio de fichas de un dispositivo, pero mantener la configuración del dispositivo igual. Para obtener más información, consulte Autorizar llamadas directas a AWS los servicios en la Guía para AWS IoT Coredesarrolladores.

En esta sección, se crea una función de IAM de intercambio de fichas y un alias de AWS IoT función que apunte a esa función. Si ya ha configurado un dispositivo principal de Greengrass, puede utilizar su función de intercambio de fichas y su alias de función en lugar de crear otros nuevos.

Para crear un rol de IAM de intercambio de fichas

1. Cree una función de IAM que su dispositivo pueda utilizar como función de intercambio de fichas. Haga lo siguiente:

   a. Cree un archivo que contenga el documento de política de confianza que requiere la función de intercambio de tokens.

      Por ejemplo, en un sistema basado en Linux, puede ejecutar el siguiente comando para usar GNU nano para crear el archivo.

      ```
      nano device-role-trust-policy.json
      ```

      Copia el siguiente JSON en el archivo.

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

```
    }
  ]
}
```

b. Cree la función de intercambio de fichas con el documento de política de confianza.

- Sustituya *GreengrassV2 TokenExchangeRole* por el nombre del rol de IAM que desee crear.

```
aws iam create-role --role-name GreengrassV2TokenExchangeRole --assume-role-
policy-document file://device-role-trust-policy.json
```

Si la solicitud se realiza correctamente, la respuesta es similar a la del siguiente ejemplo.

```
{
  "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. Cree un archivo que contenga el documento de política de acceso que requiere la función de intercambio de tokens.

Por ejemplo, en un sistema basado en Linux, puede ejecutar el siguiente comando para usar GNU nano para crear el archivo.

```
nano device-role-access-policy.json
```

Copia el siguiente JSON en el archivo.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:DescribeLogStreams",
        "s3:GetBucketLocation"
      ],
      "Resource": "*"
    }
  ]
}
```

> **Note**
>
> Esta política de acceso no permite el acceso a los artefactos de los componentes de los depósitos de S3. Para implementar componentes personalizados que definan artefactos en Amazon S3, debe añadir permisos al rol para permitir que su dispositivo principal recupere artefactos de componentes. Para obtener más información, consulte Permita el acceso a los depósitos de S3 para los artefactos de los componentes.
>
> Si aún no tiene un depósito de S3 para los artefactos de los componentes, puede añadir estos permisos más adelante, después de crear un depósito.

d. Cree la política de IAM a partir del documento de política.

- Sustituya *GreengrassV2TokenExchangeRoleAccess* por el nombre de la política de IAM que desee crear.

```
aws iam create-policy --policy-name GreengrassV2TokenExchangeRoleAccess --
policy-document file://device-role-access-policy.json
```

Si la solicitud se realiza correctamente, la respuesta es similar a la del siguiente ejemplo.

```
{
  "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. Adjunte la política de IAM a la función de intercambio de fichas.

- Sustituya *GreengrassV2 TokenExchangeRole* por el nombre de la función de IAM.
- Sustituya el ARN de la política por el ARN de la política de IAM que creó en el paso anterior.

```
aws iam attach-role-policy --role-name GreengrassV2TokenExchangeRole --policy-
arn arn:aws:iam::123456789012:policy/GreengrassV2TokenExchangeRoleAccess
```

El comando no tiene ningún resultado si la solicitud se realiza correctamente.

2. Cree un alias de AWS IoT rol que apunte al rol de intercambio de fichas.

- *GreengrassCoreTokenExchangeRoleAlias*Sustitúyalo por el nombre del alias del rol que se va a crear.
- Sustituya el ARN del rol por el ARN del rol de IAM que creó en el paso anterior.

```
aws iot create-role-alias --role-alias GreengrassCoreTokenExchangeRoleAlias --role-
arn arn:aws:iam::123456789012:role/GreengrassV2TokenExchangeRole
```

Si la solicitud se realiza correctamente, la respuesta es similar a la del ejemplo siguiente.

```
{
  "roleAlias": "GreengrassCoreTokenExchangeRoleAlias",
  "roleAliasArn": "arn:aws:iot:us-west-2:123456789012:rolealias/
GreengrassCoreTokenExchangeRoleAlias"
}
```

> **Note**
>
> Para crear un alias de rol, debe tener permiso para transferir el rol de IAM de intercambio de fichas. AWS IoT Si recibe un mensaje de error al intentar crear un alias de rol, compruebe que su AWS usuario tiene este permiso. Para obtener más información, consulte Conceder permisos a un usuario para transferir un rol a un AWS servicio en la Guía del AWS Identity and Access Management usuario.

Creación de una política de AWS IoT

Después de registrar un dispositivo como una AWS IoT cosa, ese dispositivo puede usar un certificado digital para autenticarseAWS. Este certificado incluye una o más AWS IoT políticas que definen los permisos que un dispositivo puede usar con el certificado. Estas políticas permiten que el dispositivo se comunique con AWS IoT yAWS IoT Greengrass.

Con el aprovisionamiento de AWS IoT flotas, los dispositivos se conectan AWS IoT para crear y descargar un certificado de dispositivo. En la plantilla de aprovisionamiento de flotas que cree en la siguiente sección, puede especificar si desea AWS IoT adjuntar la misma AWS IoT política a todos los certificados de los dispositivos o crear una nueva política para cada dispositivo.

En esta sección, creará una AWS IoT política que se AWS IoT adjunte a los certificados de todos los dispositivos. Con este enfoque, puede administrar los permisos de todos los dispositivos como una flota. Si prefieres crear una nueva AWS IoT política para cada dispositivo, puedes saltarte esta sección y consultar la política que contiene cuando definas la plantilla de tu flota.

Para crear una política de AWS IoT

- Cree una AWS IoT política que defina los AWS IoT permisos para su flota de dispositivos principales de Greengrass. La siguiente política permite el acceso a todos los temas de MQTT y a las operaciones de Greengrass, de modo que su dispositivo funcione con aplicaciones personalizadas y con cambios futuros que requieran nuevas operaciones de Greengrass. Esta política también permite el `iot:AssumeRoleWithCertificate` permiso, que permite a sus dispositivos utilizar la función de intercambio de fichas que creó en la sección anterior. Puedes restringir esta política en función de tu caso de uso. Para obtener más información, consulte AWS IoTPolítica mínima para los dispositivos AWS IoT Greengrass V2 principales.

  Haga lo siguiente:

  a. Cree un archivo que contenga el documento AWS IoT de política que requieren los dispositivos principales de Greengrass.

     Por ejemplo, en un sistema basado en Linux, puede ejecutar el siguiente comando para usar GNU nano para crear el archivo.

     ```
     nano greengrass-v2-iot-policy.json
     ```

     Copia el siguiente JSON en el archivo.

     - Sustituya el `iot:AssumeRoleWithCertificate` recurso por el ARN del alias de AWS IoT rol que creó en la sección anterior.

     ```
     {
       "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. Cree una AWS IoT política a partir del documento de política.

- Sustituya *GreengrassV2IoT* por el nombre ThingPolicy de la política que se va a crear.

```
aws iot create-policy --policy-name GreengrassV2IoTThingPolicy --policy-
document file://greengrass-v2-iot-policy.json
```

Si la solicitud se realiza correctamente, la respuesta es similar a la del siguiente ejemplo.

```
{
  "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"
}
```

Cree una plantilla de aprovisionamiento de flota

AWS IoTLas plantillas de aprovisionamiento de flotas definen cómo aprovisionar los AWS IoT elementos, las políticas y los certificados. Para aprovisionar los dispositivos principales de Greengrass con el complemento de aprovisionamiento de flotas, debe crear una plantilla que especifique lo siguiente:

- Cualquier AWS IoT cosa, un recurso. Puede especificar una lista de grupos de cosas existentes para implementar componentes en cada dispositivo cuando esté en línea.
- Un recurso AWS IoT de políticas. Este recurso puede definir una de las siguientes propiedades:
  - El nombre de una AWS IoT política existente. Si elige esta opción, los dispositivos principales que cree a partir de esta plantilla utilizarán la misma AWS IoT política y podrá gestionar sus permisos como una flota.
  - Un documento AWS IoT de política. Si elige esta opción, cada dispositivo principal que cree a partir de esta plantilla utilizará una AWS IoT política única y podrá administrar los permisos de cada dispositivo principal individual.
- Un recurso AWS IoT de certificados. Este recurso de certificado debe usar el `AWS::IoT::Certificate::Id` parámetro para adjuntar el certificado al dispositivo principal. Para obtener más información, consulte el ust-in-time aprovisionamiento de J en la Guía para AWS IoT desarrolladores.

En la plantilla, puede especificar si desea añadirlo a una lista de grupos de cosas existentes. AWS IoT Cuando el dispositivo principal se conecta AWS IoT Greengrass por primera vez, recibe despliegues de Greengrass para cada grupo de elementos del que es miembro. Puede usar grupos de cosas para implementar el software más reciente en cada dispositivo tan pronto como se conecte a Internet. Para obtener más información, consulte Implemente AWS IoT Greengrass componentes en los dispositivos.

El AWS IoT servicio requiere permisos para crear y actualizar AWS IoT recursos en sus dispositivos Cuenta de AWS al aprovisionar sus dispositivos. Para dar acceso al AWS IoT servicio, debe crear un rol de IAM y proporcionarlo al crear la plantilla. AWS IoTproporciona una política gestionada que permite el acceso a todos los permisos que se AWS IoT puedan utilizar al aprovisionar dispositivos. AWSIoTThingsRegistration Puede usar esta política administrada o crear una política personalizada que limite los permisos de la política administrada para su caso de uso.

En esta sección, crea una función de IAM que permite AWS IoT aprovisionar recursos para los dispositivos y crea una plantilla de aprovisionamiento de flotas que utiliza esa función de IAM.

Para crear una plantilla de aprovisionamiento de flotas

1. Cree una función de IAM que AWS IoT pueda asumir el aprovisionamiento de recursos en su. Cuenta de AWS Haga lo siguiente:

   a. Cree un archivo que contenga el documento de política de confianza que le AWS IoT permita asumir el rol.

      Por ejemplo, en un sistema basado en Linux, puede ejecutar el siguiente comando para usar GNU nano para crear el archivo.

      ```
      nano aws-iot-trust-policy.json
      ```

      Copia el siguiente JSON en el archivo.

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

   b. Cree un rol de IAM con el documento de política de confianza.

- *GreengrassFleetProvisioningRole*Sustitúyalo por el nombre del rol de IAM que desee crear.

```
aws iam create-role --role-name GreengrassFleetProvisioningRole --assume-role-
policy-document file://aws-iot-trust-policy.json
```

Si la solicitud se realiza correctamente, la respuesta es similar a la del ejemplo siguiente.

```
{
  "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. Revise la AWSIoTThingsRegistrationpolítica, que permite el acceso a todos los permisos que se AWS IoT puedan utilizar al aprovisionar dispositivos. Puedes usar esta política administrada o crear una política personalizada que defina los permisos restringidos para tu caso de uso. Si decide crear una política personalizada, hágalo ahora.

d. Adjunte la política de IAM a la función de aprovisionamiento de flota.

- Reemplace *GreengrassFleetProvisioningRole* por el nombre del rol de IAM.
- Si creó una política personalizada en el paso anterior, sustituya el ARN de la política de IAM por el ARN de la política de IAM que vaya a utilizar.

```
aws iam attach-role-policy --role-name GreengrassFleetProvisioningRole --
policy-arn arn:aws:iam::aws:policy/service-role/AWSIoTThingsRegistration
```

El comando no tiene ningún resultado si la solicitud se realiza correctamente.

2. (Opcional) Cree un enlace de preaprovisionamiento, que es una AWS Lambda función que valida los parámetros de plantilla que los dispositivos proporcionan durante el registro. Puedes usar un enlace de preaprovisionamiento para tener más control sobre qué dispositivos y cuántos están integrados en tu dispositivo. Cuenta de AWS Para obtener más información, consulta los ganchos de aprovisionamiento previo en la Guía para desarrolladores. AWS IoT Core
3. Cree una plantilla de aprovisionamiento de flotas. Haga lo siguiente:

   a. Cree un archivo que contenga el documento de plantilla de aprovisionamiento.

      Por ejemplo, en un sistema basado en Linux, puede ejecutar el siguiente comando para usar GNU nano para crear el archivo.

      ```
      nano greengrass-fleet-provisioning-template.json
      ```

      Escriba el documento de plantilla de aprovisionamiento. Puede empezar con el siguiente ejemplo de plantilla de aprovisionamiento, que especifica la creación de AWS IoT algo con las siguientes propiedades:

      - El nombre de la cosa es el valor que se especifica en el parámetro de la `ThingName` plantilla.
      - La cosa es un miembro del grupo de cosas que se especifica en el parámetro `ThingGroupName` de plantilla. El grupo de cosas debe existir en suCuenta de AWS.
      - El certificado de la cosa lleva `GreengrassV2IoTThingPolicy` adjunto el nombre de la AWS IoT política.

      Para obtener más información, consulte las plantillas de aprovisionamiento en la Guía para AWS IoT Core desarrolladores.

      ```
      {
        "Parameters": {
          "ThingName": {
      ```

```
      "Type": "String"
    },
    "ThingGroupName": {
      "Type": "String"
    },
    "AWS::IoT::Certificate::Id": {
      "Type": "String"
    }
  },
  "Resources": {
    "MyThing": {
      "OverrideSettings": {
        "AttributePayload": "REPLACE",
        "ThingGroups": "REPLACE",
        "ThingTypeName": "REPLACE"
      },
      "Properties": {
        "AttributePayload": {},
        "ThingGroups": [
          {
            "Ref": "ThingGroupName"
          }
        ],
        "ThingName": {
          "Ref": "ThingName"
        }
      },
      "Type": "AWS::IoT::Thing"
    },
    "MyPolicy": {
      "Properties": {
        "PolicyName": "GreengrassV2IoTThingPolicy"
      },
      "Type": "AWS::IoT::Policy"
    },
    "MyCertificate": {
      "Properties": {
        "CertificateId": {
          "Ref": "AWS::IoT::Certificate::Id"
        },
        "Status": "Active"
      },
      "Type": "AWS::IoT::Certificate"
    }
```

```
  }
}
```

> **Note**
>
> *MyThingMyPolicy*, y *MyCertificate*son nombres arbitrarios que identifican cada especificación de recursos de la plantilla de aprovisionamiento de flotas. AWS IoTno utiliza estos nombres en los recursos que crea a partir de la plantilla. Puede utilizar estos nombres o sustituirlos por valores que le ayuden a identificar cada recurso de la plantilla.

b. Cree la plantilla de aprovisionamiento de flota a partir del documento de plantilla de aprovisionamiento.

- *GreengrassFleetProvisioningTemplate*Sustitúyala por el nombre de la plantilla que desee crear.
- Sustituya la descripción de la plantilla por una descripción de la plantilla.
- Sustituya el ARN del rol de aprovisionamiento por el ARN del rol que creó anteriormente.

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**
>
> Si ha creado un enlace de preaprovisionamiento, especifique el ARN de la función Lambda del enlace de preaprovisionamiento con el argumento. `--pre-provisioning-hook`
>
> ```
> --pre-provisioning-hook targetArn=arn:aws:lambda:us-
> west-2:123456789012:function:GreengrassPreProvisioningHook
> ```

La respuesta es similar a la del siguiente ejemplo, si la solicitud se realiza correctamente.

```
{
    "templateArn": "arn:aws:iot:us-west-2:123456789012:provisioningtemplate/
GreengrassFleetProvisioningTemplate",
    "templateName": "GreengrassFleetProvisioningTemplate",
    "defaultVersionId": 1
}
```

Cree un certificado de notificación de aprovisionamiento y una clave privada

Los certificados de reclamación son certificados X.509 que permiten que los dispositivos se registren como AWS IoT cosas y recuperen un certificado de dispositivo X.509 exclusivo para utilizarlo en las operaciones habituales. Tras crear un certificado de reclamación, debe adjuntar una AWS IoT política que permita a los dispositivos utilizarlo para crear certificados de dispositivos únicos y aprovisionarlos con una plantilla de aprovisionamiento de flota. Los dispositivos con el certificado de reclamación se

pueden aprovisionar únicamente con la plantilla de aprovisionamiento que usted permita en la AWS IoT política.

En esta sección, crea el certificado de reclamación y lo configura para que los dispositivos lo utilicen con la plantilla de aprovisionamiento de flota que creó en la sección anterior.

> **Important**
>
> El aprovisionamiento afirma que las claves privadas deben estar protegidas en todo momento, incluso en los dispositivos principales de Greengrass. Te recomendamos que utilices CloudWatch las estadísticas y los registros de Amazon para detectar indicios de uso indebido, como el uso no autorizado del certificado de reclamación para aprovisionar dispositivos. Si detectas un uso indebido, desactiva el certificado de notificación de aprovisionamiento para que no se pueda utilizar para el aprovisionamiento de dispositivos. Para obtener más información, consulte Monitorear AWS IoT en la Guía para desarrolladores de AWS IoT Core.
>
> Para ayudarte a gestionar mejor la cantidad de dispositivos y los dispositivos que se registran automáticamente en el tuyoCuenta de AWS, puedes especificar un enlace previo al aprovisionamiento al crear una plantilla de aprovisionamiento de flotas. Un enlace de preaprovisionamiento es una AWS Lambda función que valida los parámetros de plantilla que proporcionan los dispositivos durante el registro. Por ejemplo, puede crear un enlace de preaprovisionamiento que compare el ID de un dispositivo con una base de datos para comprobar que el dispositivo tiene permiso de aprovisionamiento. Para obtener más información, consulta los ganchos de aprovisionamiento previo en la Guía para desarrolladores. AWS IoT Core

Para crear un certificado de notificación de aprovisionamiento y una clave privada

1. Cree una carpeta en la que pueda descargar el certificado de reclamación y la clave privada.

   ```
   mkdir claim-certs
   ```

2. Cree y guarde un certificado y una clave privada para usarlos en el aprovisionamiento. AWS IoTproporciona certificados de cliente firmados por la autoridad de certificación (CA) raíz de Amazon.

Linux or Unix

```
aws iot create-keys-and-certificate \
  --certificate-pem-outfile "claim-certs/claim.pem.crt" \
  --public-key-outfile "claim-certs/claim.public.pem.key" \
  --private-key-outfile "claim-certs/claim.private.pem.key" \
  --set-as-active
```

Windows Command Prompt (CMD)

```
aws iot create-keys-and-certificate ^
  --certificate-pem-outfile "claim-certs/claim.pem.crt" ^
  --public-key-outfile "claim-certs/claim.public.pem.key" ^
  --private-key-outfile "claim-certs/claim.private.pem.key" ^
  --set-as-active
```

PowerShell

```
aws iot create-keys-and-certificate `
  --certificate-pem-outfile "claim-certs/claim.pem.crt" `
  --public-key-outfile "claim-certs/claim.public.pem.key" `
  --private-key-outfile "claim-certs/claim.private.pem.key" `
  --set-as-active
```

La respuesta contiene información sobre el certificado, si la solicitud se realiza correctamente. Guarde el ARN del certificado para usarlo más adelante.

3. Cree y adjunte una AWS IoT política que permita a los dispositivos usar el certificado para crear certificados de dispositivo únicos y aprovisionarlos con la plantilla de aprovisionamiento de flota. La siguiente política permite el acceso a la API MQTT de aprovisionamiento de dispositivos. Para obtener más información, consulte la API MQTT de aprovisionamiento de dispositivos en la Guía para desarrolladores. AWS IoT Core

   Haga lo siguiente:

   a. Cree un archivo que contenga el documento AWS IoT de política que requieren los dispositivos principales de Greengrass.

Por ejemplo, en un sistema basado en Linux, puede ejecutar el siguiente comando para usar GNU nano para crear el archivo.

```
nano greengrass-provisioning-claim-iot-policy.json
```

Copia el siguiente JSON en el archivo.

- Sustituya cada instancia de la *región* por aquella en la Región de AWS que configuró el aprovisionamiento de la flota.
- Sustituya cada instancia de *account-id por su ID*. Cuenta de AWS
- Sustituya cada instancia de *GreengrassFleetProvisioningTemplate*por el nombre de la plantilla de aprovisionamiento de flota que creó en la sección anterior.

```
{
  "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. Cree una AWS IoT política a partir del documento de política.

- *GreengrassProvisioningClaimPolicy*Sustitúyala por el nombre de la política que se va a crear.

```
aws iot create-policy --policy-name GreengrassProvisioningClaimPolicy --policy-
document file://greengrass-provisioning-claim-iot-policy.json
```

Si la solicitud se realiza correctamente, la respuesta es similar a la del ejemplo siguiente.

```
{
  "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. Adjunte la AWS IoT política al certificado de notificación de aprovisionamiento.

   - *GreengrassProvisioningClaimPolicy*Sustitúyala por el nombre de la política que se va a adjuntar.
   - Sustituya el ARN de destino por el ARN del certificado de notificación de aprovisionamiento.

   ```
   aws iot attach-policy --policy-name GreengrassProvisioningClaimPolicy --
   target arn:aws:iot:us-west-2:123456789012:cert/
   aa0b7958770878eabe251d8a7ddd547f4889c524c9b574ab9fbf65f32248b1d4
   ```

   El comando no tiene ningún resultado si la solicitud se realiza correctamente.

Ahora tiene un certificado de notificación de aprovisionamiento y una clave privada que los dispositivos pueden usar para registrarse AWS IoT y aprovisionarse como dispositivos principales de Greengrass. Puede incrustar el certificado de reclamación y la clave privada en los dispositivos durante la fabricación, o bien copiar el certificado y la clave en los dispositivos antes de instalar el software AWS IoT Greengrass Core. Para obtener más información, consulte Instale el software AWS IoT Greengrass principal con aprovisionamiento AWS IoT de flota.

Configurar el complemento de aprovisionamiento de AWS IoT flotas

El complemento de aprovisionamiento de AWS IoT flotas proporciona los siguientes parámetros de configuración que puede personalizar al instalar el software AWS IoT Greengrass Core con el aprovisionamiento de flotas.

`rootPath`

La ruta a la carpeta que se va a utilizar como raíz del software AWS IoT Greengrass principal.

`awsRegion`

La Región de AWS que utiliza el complemento de aprovisionamiento de flotas para aprovisionar AWS recursos.

`iotDataEndpoint`

El punto final AWS IoT de datos para su. Cuenta de AWS

`iotCredentialEndpoint`

El punto final de AWS IoT credenciales para suCuenta de AWS.

`iotRoleAlias`

El alias del AWS IoT rol que apunta a un rol de IAM de intercambio de fichas. El proveedor de AWS IoT credenciales asume esta función para permitir que el dispositivo principal de Greengrass interactúe con AWS los servicios. Para obtener más información, consulte Autorizar a los dispositivos principales a interactuar con AWS los servicios.

`provisioningTemplate`

La plantilla de aprovisionamiento de AWS IoT flota que se utilizará para AWS aprovisionar recursos. Esta plantilla debe especificar lo siguiente:

- Cualquier AWS IoT cosa, un recurso. Puede especificar una lista de grupos de cosas existentes para implementar componentes en cada dispositivo cuando esté en línea.
- Un recurso AWS IoT de políticas. Este recurso puede definir una de las siguientes propiedades:
  - El nombre de una AWS IoT política existente. Si elige esta opción, los dispositivos principales que cree a partir de esta plantilla utilizarán la misma AWS IoT política y podrá gestionar sus permisos como una flota.
  - Un documento AWS IoT de política. Si elige esta opción, cada dispositivo principal que cree a partir de esta plantilla utilizará una AWS IoT política única y podrá administrar los permisos de cada dispositivo principal individual.
- Un recurso AWS IoT de certificados. Este recurso de certificado debe usar el `AWS::IoT::Certificate::Id` parámetro para adjuntar el certificado al dispositivo principal. Para obtener más información, consulte el ust-in-time aprovisionamiento de J en la Guía para AWS IoT desarrolladores.

Para obtener más información, consulte las plantillas de aprovisionamiento en la Guía para AWS IoT Core desarrolladores.

`claimCertificatePath`

La ruta al certificado de solicitud de aprovisionamiento para la plantilla de aprovisionamiento que especifique en. `provisioningTemplate` Para obtener más información, consulte CreateProvisioningClaim en la Referencia de la API de AWS IoT Core.

`claimCertificatePrivateKeyPath`

La ruta a la clave privada del certificado de notificación de aprovisionamiento para la plantilla de aprovisionamiento que especifique en. `provisioningTemplate` Para obtener más información, consulte CreateProvisioningClaim en la Referencia de la API de AWS IoT Core.

> **Important**
>
> El aprovisionamiento afirma que las claves privadas deben estar protegidas en todo momento, incluso en los dispositivos principales de Greengrass. Te recomendamos que utilices CloudWatch las estadísticas y los registros de Amazon para detectar indicios de uso indebido, como el uso no autorizado del certificado de reclamación para aprovisionar dispositivos. Si detectas un uso indebido, desactiva el certificado de notificación de aprovisionamiento para que no se pueda utilizar para el aprovisionamiento de dispositivos. Para obtener más información, consulte Monitorear AWS IoT en la Guía para desarrolladores de AWS IoT Core.
>
> Para ayudarte a gestionar mejor el número de dispositivos y los dispositivos que se registran automáticamente en el tuyoCuenta de AWS, puedes especificar un enlace previo al aprovisionamiento al crear una plantilla de aprovisionamiento de flotas. Un enlace de preaprovisionamiento es una AWS Lambda función que valida los parámetros de plantilla que proporcionan los dispositivos durante el registro. Por ejemplo, puede crear un enlace de preaprovisionamiento que compare el ID de un dispositivo con una base de datos para comprobar que el dispositivo tiene permiso de aprovisionamiento. Para obtener más información, consulta los ganchos de aprovisionamiento previo en la Guía para desarrolladores. AWS IoT Core

`rootCaPath`

La ruta al certificado de la autoridad de certificación (CA) raíz de Amazon.

`templateParameters`

(Opcional) El mapa de parámetros que se debe proporcionar a la plantilla de aprovisionamiento de la flota. Para obtener más información, consulte la sección de parámetros de las plantillas de aprovisionamiento en la Guía para desarrolladores. AWS IoT Core

`deviceId`

(Opcional) El identificador del dispositivo que se utilizará como ID de cliente cuando el complemento de aprovisionamiento de flotas cree una conexión MQTT con la que. AWS IoT

Predeterminado: un UUID aleatorio.

`mqttPort`

(Opcional) El puerto que se utilizará para las conexiones MQTT.

Valor predeterminado: `8883`

`proxyUrl`

(Opcional) La URL del servidor proxy en el formato`scheme://userinfo@host:port`. Para usar un proxy HTTPS, debe usar la versión 1.1.0 o posterior del complemento de aprovisionamiento de flotas.

- `scheme`— El esquema, que debe ser `http` o. `https`

  > **Important**
  >
  > Los dispositivos principales de Greengrass deben ejecutar Greengrass nucleus v2.5.0 o posterior para usar proxies HTTPS.
  > Si configura un proxy HTTPS, debe añadir el certificado de CA del servidor proxy al certificado de CA raíz de Amazon del dispositivo principal. Para obtener más información, consulte Habilite el dispositivo principal para que confíe en un proxy HTTPS.

- `userinfo`— (Opcional) La información del nombre de usuario y la contraseña. Si especifica esta información en`url`, el dispositivo principal de Greengrass ignora los `username` campos y. `password`
- `host`— El nombre de host o la dirección IP del servidor proxy.
- `port`— (Opcional) El número de puerto. Si no especificas el puerto, el dispositivo principal de Greengrass utilizará los siguientes valores predeterminados:

- `http`— 80
- `https`— 443

`proxyUserName`

(Opcional) El nombre de usuario que autentica el servidor proxy.

`proxyPassword`

(Opcional) El nombre de usuario que autentica el servidor proxy.

CSRPath

(Opcional) La ruta al archivo de solicitud de firma de certificado (CSR) que se utilizará para crear el certificado del dispositivo a partir de una CSR. Para obtener más información, consulte Aprovisionamiento por reclamación en la AWS IoT Core guía para desarrolladores.

csrPrivateKeyRuta

(Opcional, obligatorio si `csrPath` se declara) La ruta a la clave privada utilizada para generar la CSR. La clave privada debe haberse utilizado para generar la CSR. Para obtener más información, consulte Aprovisionamiento por reclamación en la guía para AWS IoT Coredesarrolladores.

AWS IoT registro de cambios del complemento de aprovisionamiento de flotas

En la siguiente tabla se describen los cambios en cada versión del complemento AWS IoT Fleet Provisioning by Claim (`aws.greengrass.FleetProvisioningByClaim`).

Versión	Cambios
1.2.1	Mejoras y correcciones de errores • Soluciona un problema por el que el complemento de aprovisionamiento de flotas estaba fuera de línea durante el inicio de Greengrass Nucleus. El complemento de aprovisionamiento de flotas ahora reintenta indefinid amente las llamadas de conexión MQTT.
1.2.0	Mejoras y correcciones de errores • Añade compatibilidad con el aprovisionamiento de dispositivos mediante una solicitud de firma de certificado con una ruta de clave privada configurable.

Versión	Cambios
	• Correcciones y mejoras menores.
1.1.0	Mejoras y correcciones de errores • Añade compatibilidad con otros formatos de rutas de archivos al configurar el complemento en dispositivos Windows. • Añade compatibilidad con las configuraciones de proxy de red HTTPS. Para obtener más información, consulte Realizar la conexión en el puerto 443 o a través de un proxy de red y Habilite el dispositivo principal para que confíe en un proxy HTTPS.
1.0.0	Versión inicial.

Instale el software AWS IoT Greengrass principal con aprovisionamiento de recursos personalizado

Esta función está disponible para la versión 2.4.0 y versiones posteriores del componente núcleo de Greengrass.

El instalador de software AWS IoT Greengrass Core proporciona una interfaz Java que se puede implementar en un complemento personalizado que aprovisione los recursos necesarios. AWS Puede desarrollar un complemento de aprovisionamiento para utilizar certificados de cliente X.509 personalizados o ejecutar pasos de aprovisionamiento complejos que otros procesos de instalación no admiten. Para obtener más información, consulte Cómo crear sus propios certificados de cliente en la AWS IoT Core Guía para desarrolladores.

Para ejecutar un complemento de aprovisionamiento personalizado al instalar el software AWS IoT Greengrass principal, debe crear un archivo JAR que se proporciona al instalador. El instalador ejecuta el complemento y el complemento devuelve una configuración de aprovisionamiento que define los AWS recursos del dispositivo principal de Greengrass. El instalador usa esta información para configurar el software AWS IoT Greengrass principal en el dispositivo. Para obtener más información, consulte Desarrollar complementos de aprovisionamiento personalizados.

> **Important**
>
> Antes de descargar el software AWS IoT Greengrass Core, compruebe que su dispositivo principal cumpla los requisitos para instalar y ejecutar el software AWS IoT Greengrass Core v2.0.

Temas

Requisitos previos

Para instalar el software AWS IoT Greengrass Core con aprovisionamiento personalizado, debe disponer de lo siguiente:

- Un archivo JAR para un complemento de aprovisionamiento personalizado que implementa el. `DeviceIdentityInterface` El complemento de aprovisionamiento personalizado debe devolver valores para cada parámetro de configuración del núcleo y del sistema. De lo contrario, debe proporcionar esos valores en el archivo de configuración durante la instalación. Para obtener más información, consulte Desarrollar complementos de aprovisionamiento personalizados.

Configure el entorno del dispositivo

Siga los pasos de esta sección para configurar un dispositivo Linux o Windows para usarlo como dispositivo AWS IoT Greengrass principal.

Configura un dispositivo Linux

Para configurar un dispositivo Linux para AWS IoT Greengrass V2

1. Instale el motor de ejecución de Java, que el software AWS IoT Greengrass principal necesita para ejecutarse. Le recomendamos que utilice las versiones de soporte a largo plazo de Amazon

Corretto u OpenJDK. Se requiere la versión 8 o superior. Los siguientes comandos muestran cómo instalar OpenJDK en su dispositivo.

- Para distribuciones basadas en Debian o en Ubuntu:

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

- Para distribuciones basadas en Red Hat:

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

- En Amazon Linux 2:

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

- En Amazon Linux 2023:

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

Cuando se complete la instalación, ejecute el siguiente comando para comprobar que Java se ejecuta en su dispositivo Linux.

```
java -version
```

El comando imprime la versión de Java que se ejecuta en el dispositivo. Por ejemplo, en una distribución basada en Debian, el resultado podría tener un aspecto similar al del siguiente ejemplo.

```
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. (Opcional) Cree el usuario y el grupo predeterminados del sistema que ejecutan los componentes del dispositivo. También puede optar por permitir que el instalador del software AWS IoT Greengrass principal cree este usuario y grupo durante la instalación con el argumento del `--component-default-user` instalador. Para obtener más información, consulte Argumentos de instalación.

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

```
sudo groupadd --system ggc_group
```

3. Compruebe que el usuario que ejecuta el software AWS IoT Greengrass principal (normalmente`root`) tiene permiso para ejecutar `sudo` con cualquier usuario y grupo.

   a. Ejecute el siguiente comando para abrir el `/etc/sudoers` archivo.

   ```
   sudo visudo
   ```

   b. Compruebe que el permiso del usuario es similar al del ejemplo siguiente.

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

4. (Opcional) Para ejecutar funciones Lambda en contenedores, debe habilitar cgroups v1 y debe habilitar y montar los cgroups de memoria y dispositivos. Si no planea ejecutar funciones Lambda en contenedores, puede omitir este paso.

   Para habilitar estas opciones de cgroups, arranque el dispositivo con los siguientes parámetros del kernel de Linux.

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

   Para obtener información sobre cómo ver y configurar los parámetros del núcleo de su dispositivo, consulte la documentación del sistema operativo y del gestor de arranque. Siga las instrucciones para configurar permanentemente los parámetros del núcleo.

5. Instale todas las demás dependencias necesarias en su dispositivo tal y como se indica en Requisitos de los dispositivos la lista de requisitos de.

Configura un dispositivo Windows

> **Note**
>
> Esta función está disponible para la versión 2.5.0 y versiones posteriores del componente núcleo de Greengrass.

Para configurar un dispositivo Windows para AWS IoT Greengrass V2

1. Instale el motor de ejecución de Java, que el software AWS IoT Greengrass principal necesita para ejecutarse. Le recomendamos que utilice las versiones de soporte a largo plazo de Amazon Corretto u OpenJDK. Se requiere la versión 8 o superior.
2. Compruebe si Java está disponible en la variable de sistema PATH y agréguelo si no lo está. La LocalSystem cuenta ejecuta el software AWS IoT Greengrass principal, por lo que debe añadir Java a la variable de sistema PATH en lugar de a la variable de usuario PATH de su usuario. Haga lo siguiente:

   a. Pulse la tecla Windows para abrir el menú de inicio.

   b. Escriba **`environment variables`** para buscar las opciones del sistema en el menú de inicio.

   c. En los resultados de la búsqueda del menú de inicio, seleccione Editar las variables de entorno del sistema para abrir la ventana de propiedades del sistema.

   d. Seleccione Variables de entorno... para abrir la ventana Variables de entorno.

   e. En Variables de sistema, seleccione Ruta y, a continuación, elija Editar. En la ventana Editar variables de entorno, puede ver cada ruta en una línea independiente.

   f. Compruebe si la ruta a la `bin` carpeta de la instalación de Java está presente. La ruta puede tener un aspecto similar al del siguiente ejemplo.

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

   g. Si la `bin` carpeta de la instalación de Java no aparece en Path, seleccione Nueva para añadirla y, a continuación, pulse Aceptar.
3. Abre la línea de comandos de Windows (`cmd.exe`) como administrador.
4. Cree el usuario predeterminado en la LocalSystem cuenta del dispositivo Windows. Sustituya la *`contraseña`* por una contraseña segura.

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

   > **Tip**
   >
   > Según la configuración de Windows, es posible que la contraseña del usuario caduque en una fecha futura. Para garantizar que sus aplicaciones de Greengrass sigan

funcionando, controle cuándo caduque la contraseña y actualícela antes de que caduque. También puede configurar la contraseña del usuario para que nunca caduque.

- Para comprobar cuándo caducan un usuario y su contraseña, ejecuta el siguiente comando.

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

- Para configurar la contraseña de un usuario para que no caduque nunca, ejecute el siguiente comando.

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

- Si utilizas Windows 10 o una versión posterior, donde el wmiccomando está obsoleto, ejecuta el siguiente PowerShell comando.

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

5. Descargue e instale la PsExecutilidad de Microsoft en el dispositivo.
6. Utilice la PsExec utilidad para almacenar el nombre de usuario y la contraseña del usuario predeterminado en la instancia de Credential Manager de la LocalSystem cuenta. Sustituya la *contraseña* por la contraseña del usuario que configuró anteriormente.

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

Si se PsExec License Agreementabre, Acceptacepte la licencia y ejecute el comando.

> **Note**
>
> En los dispositivos Windows, la LocalSystem cuenta ejecuta el núcleo de Greengrass y debe usar la PsExec utilidad para almacenar la información de usuario predeterminada en la LocalSystem cuenta. El uso de la aplicación Credential Manager almacena esta información en la cuenta de Windows del usuario que ha iniciado sesión actualmente, en lugar de en la LocalSystem cuenta.

Descargue el software AWS IoT Greengrass principal

Puede descargar la última versión del software AWS IoT Greengrass Core desde la siguiente ubicación:

- https://d2s8p88vqu9w66.cloudfront.net/releases/greengrass-nucleus-latest.zip

> **Note**
>
> Puede descargar una versión específica del software AWS IoT Greengrass Core desde la siguiente ubicación. Sustituya la *versión* por la versión que desea descargar.
>
> ```
> https://d2s8p88vqu9w66.cloudfront.net/releases/greengrass-version.zip
> ```

Para descargar el software AWS IoT Greengrass principal

1. En su dispositivo principal, descargue el software AWS IoT Greengrass Core en un archivo denominado`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
   ```

   Al descargar este software, acepta el acuerdo de licencia del software de Greengrass Core.

2. (Opcional) Para verificar la firma del software Greengrass Nucleus

> **Note**
>
> Esta función está disponible con la versión 2.9.5 y posteriores del núcleo de Greengrass.

a. Usa el siguiente comando para verificar la firma del artefacto núcleo de Greengrass:

   Linux or Unix

   ```
   jarsigner -verify -certs -verbose greengrass-nucleus-latest.zip
   ```

   Windows Command Prompt (CMD)

   El nombre del archivo puede tener un aspecto diferente según la versión de JDK que instale. *jdk17.0.6_10*Sustitúyalo por la versión de JDK que instaló.

   ```
   "C:\\Program Files\\Amazon Corretto\\jdk17.0.6_10\\bin\\jarsigner.exe" -
   verify -certs -verbose greengrass-nucleus-latest.zip
   ```

   PowerShell

   El nombre del archivo puede tener un aspecto diferente en función de la versión de JDK que instale. *jdk17.0.6_10*Sustitúyalo por la versión de JDK que instaló.

   ```
   'C:\\Program Files\\Amazon Corretto\\jdk17.0.6_10\\bin\\jarsigner.exe' -
   verify -certs -verbose greengrass-nucleus-latest.zip
   ```

b. La `jarsigner` invocación produce un resultado que indica los resultados de la verificación.

   i. Si el archivo zip del núcleo de Greengrass está firmado, el resultado contiene la siguiente declaración:

      ```
      jar verified.
      ```

   ii. Si el archivo zip del núcleo de Greengrass no está firmado, el resultado contiene la siguiente declaración:

      ```
      jar is unsigned.
      ```

c. Si ha proporcionado la `-certs` opción Jarsigner junto con `-verbose` las opciones `-verify` y, el resultado también incluye información detallada del certificado de firmante.

3. Descomprime el software AWS IoT Greengrass principal en una carpeta de tu dispositivo. *`GreengrassInstaller`*Sustitúyalo por la carpeta que desee usar.

   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. (Opcional) Ejecute el siguiente comando para ver la versión del software AWS IoT Greengrass principal.

   ```
   java -jar ./GreengrassInstaller/lib/Greengrass.jar --version
   ```

> **Important**
>
> Si instala una versión del núcleo de Greengrass anterior a la v2.4.0, no elimine esta carpeta después de instalar el software Core. AWS IoT Greengrass El software AWS IoT Greengrass Core utiliza los archivos de esta carpeta para ejecutarse.
> Si descargó la última versión del software, instale la versión 2.4.0 o posterior y podrá eliminar esta carpeta después de instalar el software AWS IoT Greengrass principal.

Instale el software principal AWS IoT Greengrass

Ejecute el instalador con argumentos que especifiquen las siguientes acciones:

- Realice la instalación desde un archivo de configuración parcial que especifique el uso de su complemento de aprovisionamiento personalizado para aprovisionar AWS recursos. El software AWS IoT Greengrass Core utiliza un archivo de configuración que especifica la configuración de todos los componentes de Greengrass del dispositivo. El instalador crea un archivo de configuración completo a partir del archivo de configuración parcial que usted proporciona y de los AWS recursos que crea el complemento de aprovisionamiento personalizado.
- Especifique si desea utilizar el usuario `ggc_user` del sistema para ejecutar los componentes de software en el dispositivo principal. En los dispositivos Linux, este comando también especifica el uso del grupo `ggc_group` del sistema, y el instalador crea el usuario y el grupo del sistema automáticamente.
- Configure el software AWS IoT Greengrass principal como un servicio del sistema que se ejecute durante el arranque. En los dispositivos Linux, esto requiere el sistema de inicio Systemd.

> **Important**
>
> En los dispositivos principales de Windows, debe configurar el software AWS IoT Greengrass Core como un servicio del sistema.

Para obtener más información sobre los argumentos que puede especificar, consulteArgumentos de instalación.

> **Note**
>
> Si utiliza un AWS IoT Greengrass dispositivo con memoria limitada, puede controlar la cantidad de memoria que utiliza el software AWS IoT Greengrass Core. Para controlar la asignación de memoria, puede configurar las opciones de tamaño de pila de la JVM en el parámetro de `jvmOptions` configuración del componente core. Para obtener más información, consulte Controle la asignación de memoria con las opciones de JVM.

Para instalar el software AWS IoT Greengrass Core (Linux)

1. Compruebe la versión del software AWS IoT Greengrass principal.

   - *`GreengrassInstaller`*Sustitúyala por la ruta a la carpeta que contiene el software.

```
java -jar ./GreengrassInstaller/lib/Greengrass.jar --version
```

2. Utilice un editor de texto para crear un archivo de configuración con el nombre `config.yaml` para proporcionárselo al instalador.

   Por ejemplo, en un sistema basado en Linux, puede ejecutar el siguiente comando para usar GNU nano para crear el archivo.

   ```
   nano GreengrassInstaller/config.yaml
   ```

   Copia el siguiente contenido de YAML en el archivo.

   ```
   ---
   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: ""
   ```

   A continuación, proceda del modo siguiente:

   - Sustituya la versión *2.12.6* por la versión del AWS IoT Greengrass software Core.

- Sustituya cada instancia de */greengrass/v2* por la carpeta raíz de Greengrass.
- (Opcional) Especifique los valores de configuración del sistema y del núcleo. Debe establecer estos valores si su complemento de aprovisionamiento no los proporciona.
- (Opcional) Especifique los parámetros de configuración para proporcionarlos a su complemento de aprovisionamiento.

> **Note**
>
> En este archivo de configuración, puede personalizar otras opciones de configuración, como los puertos y el proxy de red que se van a utilizar, como se muestra en el siguiente ejemplo. Para obtener más información, consulte Configuración del núcleo de 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. Ejecute el instalador. Especifique si `--trusted-plugin` desea proporcionar su complemento de aprovisionamiento personalizado y especifique si desea `--init-config` proporcionar el archivo de configuración.

   - Sustituya */greengrass/v2* o *C:\greengrass\v2* por la carpeta raíz de Greengrass.
   - Sustituya cada instancia de por *GreengrassInstaller*la carpeta en la que desempaquetó el instalador.
   - Sustituya la ruta al archivo JAR del complemento de aprovisionamiento personalizado por la ruta al archivo JAR del complemento.

   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**
>
> En los dispositivos principales de Windows, debe `--setup-system-service true` especificar la configuración del software AWS IoT Greengrass principal como un servicio del sistema.

Si lo especifica`--setup-system-service true`, el instalador `Successfully set up Nucleus as a system service` imprimirá si configuró y ejecutó el software como un servicio del sistema. De lo contrario, el instalador no mostrará ningún mensaje si instala el software correctamente.

> **Note**
>
> No puede usar el `deploy-dev-tools` argumento para implementar herramientas de desarrollo local cuando ejecuta el instalador sin el `--provision true` argumento. Para obtener información sobre cómo implementar la CLI de Greengrass directamente en su dispositivo, consulte. Interfaz de línea de comandos Greengrass

4. Verifique la instalación consultando los archivos de la carpeta raíz.

   Linux or Unix

   ```
   ls /greengrass/v2
   ```

   Windows Command Prompt (CMD)

   ```
   dir C:\greengrass\v2
   ```

   PowerShell

   ```
   ls C:\greengrass\v2
   ```

Si la instalación se realizó correctamente, la carpeta raíz contiene varias carpetas`config`, como`packages`, `ylogs`.

Si ha instalado el software AWS IoT Greengrass principal como un servicio del sistema, el instalador ejecutará el software automáticamente. De lo contrario, debe ejecutar el software manualmente. Para obtener más información, consulte Ejecute el software AWS IoT Greengrass principal.

Para obtener más información acerca de cómo configurar y utilizar el software AWS IoT Greengrass, consulte lo siguiente:

- Configurar el software AWS IoT Greengrass principal
- Desarrolle AWS IoT Greengrass componentes
- Implemente AWS IoT Greengrass componentes en los dispositivos
- Interfaz de línea de comandos Greengrass

Desarrollar complementos de aprovisionamiento personalizados

Para desarrollar un complemento de aprovisionamiento personalizado, cree una clase Java que implemente el`com.aws.greengrass.provisioning.DeviceIdentityInterface`. Puede incluir el archivo JAR del núcleo de Greengrass en su proyecto para acceder a esta interfaz y a sus clases. Esta interfaz define un método que introduce una configuración de complemento y genera una configuración de aprovisionamiento. La configuración de aprovisionamiento define las configuraciones para el sistema y elComponente de núcleo de Greengrass. LaAWS IoT GreengrassEl instalador de software principal utiliza esta configuración de aprovisionamiento para configurar elAWS IoT GreengrassSoftware principal de un dispositivo.

Después de desarrollar un complemento de aprovisionamiento personalizado, créelo como un archivo JAR que puede proporcionar alAWS IoT GreengrassInstalador de software principal para ejecutar el complemento durante la instalación. El instalador ejecuta su complemento de aprovisionamiento personalizado en la misma JVM que utiliza el instalador, por lo que puede crear un JAR que contenga solo el código del complemento.

> **Note**
>
> LaAWS IoTPlugin de aprovisionamiento deimplementa el`DeviceIdentityInterface`para utilizar el aprovisionamiento de flotas durante la instalación. El complemento de

aprovisionamiento de flotas es de código abierto, por lo que puede explorar su código fuente para ver un ejemplo de cómo utilizar la interfaz del complemento de aprovisionamiento. Para obtener más información, consulte laAWS IoTPlugin de aprovisionamiento deen GitHub.

Temas

Requisitos

Para desarrollar un complemento de aprovisionamiento personalizado, debe crear una clase Java que cumpla los siguientes requisitos:

- Usa el`com.aws.greengrass`paquete o un paquete dentro del`com.aws.greengrass`paquete.
- Tiene un constructor sin argumentos.
- Implemente el`DeviceIdentityInterface`. Para obtener más información, consulte Implemente el DeviceIdentityInterface interfaz.

Implemente el DeviceIdentityInterface interfaz

Para utilizar el`com.aws.greengrass.provisioning.DeviceIdentityInterface`interfaz en su complemento personalizado, agregue el núcleo de Greengrass como dependencia a su proyecto.

Para utilizar el DeviceIdentityInterface en un proyecto de complemento de aprovisionamiento personalizado

- Puede agregar el archivo JAR del núcleo de Greengrass como biblioteca o agregar el núcleo de Greengrass como dependencia de Maven. Realice alguna de las siguientes acciones:
  - Para agregar el archivo JAR del núcleo de Greengrass como biblioteca, descargue laAWS IoT GreengrassSoftware principal, que contiene el núcleo de Greengrass JAR. Puede descargar la versión más reciente deAWS IoT GreengrassSoftware Core (Software Core) desde la ubicación siguiente:
    - https://d2s8p88vqu9w66.cloudfront.net/releases/greengrass-nucleus-latest.zip

Encontrará el archivo JAR del núcleo de Greengrass (`Greengrass.jar`) en el`lib`carpeta del archivo ZIP. Añade este archivo JAR a tu proyecto.

- Para consumir el núcleo de Greengrass en un proyecto de Maven, añada una dependencia del`nucleus`artefacto en el`com.aws.greengrass`grupo. También debe agregar el`greengrass-common`, porque el núcleo de Greengrass no está disponible en el repositorio central de Maven.

```
<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>
```

Interfaz de interfaz de DeviceIdentityInterface

La`com.aws.greengrass.provisioning.DeviceIdentityInterface`tiene la forma siguiente.

> **Note**
>
> También puede explorar estas clases en elpaquete com.aws.greengrass.provisioningdelCódigo fuente del núcleo de Greengrassen 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 {}
```

Cada valor de configuración del`SystemConfiguration`y`NucleusConfiguration`es necesario para instalar elAWS IoT GreengrassSoftware principal, pero puedes volver`null`. Si el complemento de aprovisionamiento personalizado regresa`null`para cualquier valor de configuración, debe proporcionar ese valor en la configuración del sistema o núcleo al crear el`config.yaml`archivo que se debe proporcionar a laAWS IoT GreengrassInstalador de software de Core. Si su complemento de aprovisionamiento personalizado devuelve un valor no nulo para una opción que también define en`config.yaml`y, a continuación, el instalador sustituye el valor de`config.yaml`con el valor devuelto por el complemento.

Argumentos de instalación

El software AWS IoT Greengrass principal incluye un instalador que configura el software y proporciona los AWS recursos necesarios para que funcione el dispositivo principal de Greengrass. El instalador incluye los siguientes argumentos que puede especificar para configurar la instalación:

`-h`, `--help`

(Opcional) Muestra la información de ayuda del instalador.

`--version`

(Opcional) Muestra la versión del software AWS IoT Greengrass principal.

`-Droot`

(Opcional) La ruta a la carpeta que se va a utilizar como raíz del software AWS IoT Greengrass principal.

> **Note**
>
> Este argumento establece una propiedad de JVM, por lo que debe especificarla antes de `-jar` ejecutar el instalador. Por ejemplo, especifique `java -Droot="/greengrass/v2" -jar /path/to/Greengrass.jar`.

Valor predeterminado:

- Linux: `~/.greengrass`
- Windows: `%USERPROFILE%/.greengrass`

`-ar`, `--aws-region`

El Región de AWS que el software AWS IoT Greengrass Core utiliza para recuperar o crear AWS los recursos necesarios.

`-p`, `--provision`

(Opcional) Puede registrar este dispositivo como una AWS IoT cosa y aprovisionar los AWS recursos que necesite el dispositivo principal. Si lo especifica`true`, el software AWS IoT Greengrass principal aprovisiona una AWS IoT cosa, (opcional) un grupo de AWS IoT cosas, una función de IAM y un alias de AWS IoT función.

Valor predeterminado: `false`

`-tn`, `--thing-name`

(Opcional) El nombre del elemento AWS IoT que se registra como este dispositivo principal. Si el elemento con ese nombre no existe en el tuyoCuenta de AWS, será creado por el software AWS IoT Greengrass Core.

> **Note**
>
> El nombre de la cosa no puede contener dos puntos (`:`).

Debe especificar si `--provision true` desea aplicar este argumento.

Predeterminado: `GreengrassV2IotThing_` más un UUID aleatorio.

`-tgn`, `--thing-group-name`

(Opcional) El nombre del grupo de AWS IoT cosas al que se añade el objeto de AWS IoT este dispositivo principal. Si una implementación se dirige a este grupo de elementos, este dispositivo principal recibe esa implementación cuando se conecta a élAWS IoT Greengrass. Si el grupo de cosas con este nombre no existe en el suyoCuenta de AWS, el software AWS IoT Greengrass principal lo crea.

> **Note**
>
> El nombre del grupo de cosas no puede contener dos puntos (`:`).

Debe especificar si `--provision true` desea aplicar este argumento.

`-tpn`, `--thing-policy-name`

Esta función está disponible para la versión 2.4.0 y versiones posteriores del componente núcleo de Greengrass.

(Opcional) El nombre de la AWS IoT política que se debe adjuntar al certificado Thing de AWS IoT este dispositivo principal. Si la AWS IoT política con este nombre no existe en el suyoCuenta de AWS, el software AWS IoT Greengrass principal la crea.

El software AWS IoT Greengrass Core crea una AWS IoT política permisiva de forma predeterminada. Puede limitar el alcance de esta política o crear una política personalizada en la

que restrinja los permisos según su caso de uso. Para obtener más información, consulte AWS IoTPolítica mínima para los dispositivos AWS IoT Greengrass V2 principales.

Debe especificar si `--provision true` desea aplicar este argumento.

Valor predeterminado: `GreengrassV2IoTThingPolicy`

`-trn`, `--tes-role-name`

(Opcional) El nombre de la función de IAM que se va a utilizar para adquirir AWS las credenciales que permiten al dispositivo principal interactuar con AWS los servicios. Si el rol con este nombre no existe en su cuentaCuenta de AWS, el software AWS IoT Greengrass principal lo crea con la `GreengrassV2TokenExchangeRoleAccess` política. Este rol no tiene acceso a los depósitos de S3 donde alojas los artefactos de los componentes. Por lo tanto, debes añadir permisos a los cubos y objetos de S3 de tus artefactos al crear un componente. Para obtener más información, consulte Autorizar a los dispositivos principales a interactuar con AWS los servicios.

Debe especificar si desea aplicar este argumento`--provision true`.

Valor predeterminado: `GreengrassV2TokenExchangeRole`

`-tra`, `--tes-role-alias-name`

(Opcional) El nombre del alias del AWS IoT rol que apunta al rol de IAM que proporciona AWS las credenciales para este dispositivo principal. Si el alias del rol con este nombre no existe en tu cuentaCuenta de AWS, el software AWS IoT Greengrass principal lo crea y lo dirige al rol de IAM que especifiques.

Debe especificar si desea `--provision true` aplicar este argumento.

Valor predeterminado: `GreengrassV2TokenExchangeRoleAlias`

`-ss`, `--setup-system-service`

(Opcional) Puede configurar el software AWS IoT Greengrass principal como un servicio del sistema que se ejecute cuando se inicie el dispositivo. El nombre del servicio del sistema es`greengrass`. Para obtener más información, consulte Configurar el núcleo de Greengrass como un servicio del sistema.

En los sistemas operativos Linux, este argumento requiere que el sistema de inicio systemd esté disponible en el dispositivo.

> **Important**
>
> En los dispositivos principales de Windows, debe configurar el software AWS IoT Greengrass Core como un servicio del sistema.

Valor predeterminado: `false`

`-u, --component-default-user`

El nombre o ID del usuario que el software AWS IoT Greengrass principal utiliza para ejecutar los componentes. Por ejemplo, puede especificar **`ggc_user`**. Este valor es obligatorio cuando se ejecuta el instalador en sistemas operativos Windows.

En los sistemas operativos Linux, también puede especificar el grupo de forma opcional. Especifique el usuario y el grupo separados por dos puntos. Por ejemplo, **`ggc_user:ggc_group`**.

Las siguientes consideraciones adicionales se aplican a los sistemas operativos Linux:

- Si se ejecuta como root, el usuario del componente predeterminado es el usuario definido en el archivo de configuración. Si el archivo de configuración no define un usuario, el valor predeterminado es. `ggc_user:ggc_group` Si existen `ggc_user` o `ggc_group` no, el software los crea.
- Si se ejecuta como un usuario que no es root, el software AWS IoT Greengrass Core utiliza ese usuario para ejecutar los componentes.
- Si no especifica un grupo, el software AWS IoT Greengrass Core utiliza el grupo principal del usuario del sistema.

Para obtener más información, consulte Configure el usuario que ejecuta los componentes.

`-d, --deploy-dev-tools`

(Opcional) Puede descargar e implementar el componente CLI de Greengrass en este dispositivo principal. Puede usar esta herramienta para desarrollar y depurar componentes en este dispositivo principal.

> **Important**
>
> Le recomendamos que utilice este componente únicamente en entornos de desarrollo, no en entornos de producción. Este componente proporciona acceso a información

y operaciones que normalmente no necesitará en un entorno de producción. Siga el principio de privilegios mínimos implementando este componente solo en los dispositivos principales donde lo necesite.

Debe especificar si `--provision true` desea aplicar este argumento.

Valor predeterminado: `false`

`-init`, `--init-config`

(Opcional) La ruta al archivo de configuración que se utilizará para instalar el software AWS IoT Greengrass principal. Puede usar esta opción para configurar nuevos dispositivos principales con una configuración de núcleo específica, por ejemplo.

Important

El archivo de configuración que especifique se fusiona con el archivo de configuración existente en el dispositivo principal. Esto incluye los componentes y las configuraciones de los componentes del dispositivo principal. Se recomienda que el archivo de configuración solo muestre las configuraciones que está intentando cambiar.

`-tp`, `--trusted-plugin`

(Opcional) La ruta a un archivo JAR para cargarlo como un complemento de confianza. Utilice esta opción para proporcionar los archivos JAR del complemento de aprovisionamiento, por ejemplo, para instalarlo con el aprovisionamiento de flotas o el aprovisionamiento personalizado, o para instalarlo con la clave privada y el certificado en un módulo de seguridad de hardware.

`-s`, `--start`

(Opcional) Puede iniciar el software AWS IoT Greengrass principal después de que se haya instalado y, de forma opcional, aprovisionar recursos.

Valor predeterminado: `true`

Ejecute el software AWS IoT Greengrass principal

Después de instalar el software AWS IoT Greengrass Core, ejecútelo para conectar el dispositivo aAWS IoT Greengrass.

Al instalar el software AWS IoT Greengrass Core, puede especificar si desea instalarlo como un servicio del sistema con systemd. Si elige esta opción, el instalador ejecutará el software automáticamente y lo configurará para que se ejecute al arrancar el dispositivo.

> **Important**
>
> En los dispositivos principales de Windows, debe configurar el software AWS IoT Greengrass Core como un servicio del sistema.

Temas

Compruebe si el software AWS IoT Greengrass Core se ejecuta como un servicio del sistema

Al instalar el software AWS IoT Greengrass Core, puede especificar el `--setup-system-service true` argumento para instalar el software AWS IoT Greengrass Core como un servicio del sistema. Los dispositivos Linux requieren que el sistema systemd init configure el software AWS IoT Greengrass Core como un servicio del sistema. Si utiliza esta opción, el instalador ejecuta el software automáticamente y lo configura para que se ejecute al arrancar el dispositivo. El instalador muestra el siguiente mensaje si instala correctamente el software AWS IoT Greengrass principal como un servicio del sistema.

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

Si ya instaló el software AWS IoT Greengrass Core y no dispone de la salida del instalador, puede comprobar si el software está instalado como un servicio del sistema.

Para comprobar si el software AWS IoT Greengrass principal está instalado como un servicio del sistema

- Ejecute el siguiente comando para comprobar el estado del servicio del sistema Greengrass.

Linux or Unix (systemd)

```
sudo systemctl status greengrass.service
```

La respuesta es similar a la del siguiente ejemplo si el software AWS IoT Greengrass principal está instalado como un servicio del sistema y está activo.

```
# 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...
```

Si `systemctl greengrass.service` no lo encuentra, significa que el software AWS IoT Greengrass principal no está instalado como un servicio del sistema. Para ejecutar el software, consulteEjecute el software AWS IoT Greengrass principal sin un servicio de sistema.

Windows Command Prompt (CMD)

```
sc query greengrass
```

La respuesta es similar a la del siguiente ejemplo si el software AWS IoT Greengrass principal está instalado como un servicio de Windows y está activo.

```
SERVICE_NAME: greengrass
        TYPE               : 10  WIN32_OWN_PROCESS
        STATE              : 4  RUNNING
                                (STOPPABLE, NOT_PAUSABLE, ACCEPTS_SHUTDOWN)
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0x0
```

PowerShell

```
Get-Service greengrass
```

La respuesta es similar a la del siguiente ejemplo si el software AWS IoT Greengrass principal está instalado como un servicio de Windows y está activo.

```
Status   Name               DisplayName
------   ----               -----------
Running  greengrass         greengrass
```

Ejecute el software AWS IoT Greengrass Core como un servicio del sistema

Si el software AWS IoT Greengrass principal está instalado como un servicio del sistema, puede usar el administrador de servicios del sistema para iniciar, detener y administrar el software. Para obtener más información, consulte Configurar el núcleo de Greengrass como un servicio del sistema.

Para ejecutar el software AWS IoT Greengrass principal

- Ejecute el siguiente comando para iniciar el software de AWS IoT Greengrass Core.

  Linux or Unix (systemd)

  ```
  sudo systemctl start greengrass.service
  ```

  Windows Command Prompt (CMD)

  ```
  sc start greengrass
  ```

  PowerShell

  ```
  Start-Service greengrass
  ```

Ejecute el software AWS IoT Greengrass principal sin un servicio de sistema

En los dispositivos principales de Linux, si el software AWS IoT Greengrass principal no está instalado como un servicio del sistema, puede ejecutar el script de carga del software para ejecutarlo.

Para ejecutar el software AWS IoT Greengrass principal sin un servicio del sistema

- Ejecute el siguiente comando para iniciar el software de AWS IoT Greengrass Core. Si ejecuta este comando en una terminal, debe mantener abierta la sesión de la terminal para que el software AWS IoT Greengrass principal siga funcionando.
  - Sustituya */greengrass/v2* o *C:\greengrass\v2* por la carpeta raíz de Greengrass que utilice.

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

El software imprime el siguiente mensaje si se inicia correctamente.

```
Launched Nucleus successfully.
```

Ejecute AWS IoT Greengrass el software principal en un contenedor de Docker

AWS IoT Greengrass se puede configurar para que se ejecute en un contenedor de Docker. Docker es una plataforma que proporciona las herramientas para crear, ejecutar, probar e implementar aplicaciones basadas en contenedores de Linux. Al ejecutar una imagen de AWS IoT Greengrass Docker, puede elegir si desea proporcionar sus AWS credenciales al contenedor de Docker y permitir que el instalador del software AWS IoT Greengrass Core aprovisione automáticamente los AWS recursos que un dispositivo principal de Greengrass necesita para funcionar. Si no desea proporcionar AWS credenciales, puede aprovisionar AWS recursos manualmente y ejecutar el software AWS IoT Greengrass Core en el contenedor de Docker.

Temas

- Plataformas compatibles y requisitos

- AWS IoT Greengrass Descargas del software Docker
- Elija cómo aprovisionar los recursos AWS
- Cree la imagen del AWS IoT Greengrass contenedor a partir de un Dockerfile
- Se ejecuta AWS IoT Greengrass en un contenedor Docker con aprovisionamiento automático de recursos
- Se ejecuta AWS IoT Greengrass en un contenedor Docker con aprovisionamiento manual de recursos
- Solución de problemas de AWS IoT Greengrass en un contenedor Docker

Plataformas compatibles y requisitos

Los ordenadores host deben cumplir los siguientes requisitos mínimos para instalar y ejecutar el software AWS IoT Greengrass principal en un contenedor de Docker:

- Un sistema operativo basado en Linux con conexión a Internet.
- Docker Engine versión 18.09 o posterior.
- (Opcional) Docker Compose versión 1.22 o posterior. Docker Compose solo es necesario si quieres usar la CLI de Docker Compose para ejecutar tus imágenes de Docker.

Para ejecutar los componentes de la función Lambda dentro del contenedor de Docker, debe configurar el contenedor para que cumpla con los requisitos adicionales. Para obtener más información, consulte Requisitos de la función de Lambda.

Ejecute los componentes en modo de proceso

AWS IoT Greengrass no admite la ejecución de funciones de Lambda o componentes AWS proporcionados en un entorno de ejecución aislado dentro del AWS IoT Greengrass contenedor de Docker. Debe ejecutar estos componentes en modo de proceso sin ningún tipo de aislamiento.

Al configurar un componente de la función Lambda, defina el modo de aislamiento en Sin contenedor. Para obtener más información, consulte AWS LambdaFunciones de ejecución.

Al implementar cualquiera de los siguientes componentes AWS proporcionados, actualice la configuración de cada componente para establecer el `containerMode` parámetro en. `NoContainer` Para obtener más información sobre las actualizaciones de configuración, consulteActualizar las configuraciones de los componentes.

- CloudWatch métricas
- Device Defender
- Firehose
- adaptador de protocolo Modbus-RTU
- Amazon SNS

AWS IoT Greengrass Descargas del software Docker

AWS IoT Greengrass proporciona un Dockerfile para crear una imagen de contenedor que tiene el software AWS IoT Greengrass principal y las dependencias instaladas en una imagen base de Amazon Linux 2 (x86_64). Puede modificar la imagen base del Dockerfile para que se ejecute en una arquitectura de plataforma diferente. AWS IoT Greengrass

Descargue el paquete Dockerfile desde. GitHub

El Dockerfile usa una versión anterior de Greengrass. Debe actualizar el archivo para usar la versión de Greengrass que desee. Para obtener información sobre cómo crear la imagen del AWS IoT Greengrass contenedor a partir del Dockerfile, consulte. Cree la imagen del AWS IoT Greengrass contenedor a partir de un Dockerfile

Elija cómo aprovisionar los recursos AWS

Al instalar el software AWS IoT Greengrass Core en un contenedor de Docker, puede elegir si desea aprovisionar automáticamente los AWS recursos que un dispositivo principal de Greengrass necesita para funcionar o usar los recursos que aprovisiona manualmente.

- Aprovisionamiento automático de recursos: el instalador proporciona la AWS IoT AWS IoT cosa, el grupo de cosas, la función de IAM y el alias de la AWS IoT función al ejecutar la imagen del AWS IoT Greengrass contenedor por primera vez. El instalador también puede implementar las herramientas de desarrollo locales en el dispositivo principal, de modo que usted pueda usar el dispositivo para desarrollar y probar componentes de software personalizados. Para aprovisionar estos recursos automáticamente, debe proporcionar AWS credenciales como variables de entorno a la imagen de Docker.

  Para utilizar el aprovisionamiento automático, debe configurar la variable de entorno Docker `PROVISION=true` y montar un archivo de credenciales para proporcionar sus AWS credenciales al contenedor.

- Aprovisionamiento manual de recursos: si no quieres proporcionar AWS credenciales al contenedor, puedes aprovisionar los AWS recursos manualmente antes de ejecutar la imagen del contenedor. AWS IoT Greengrass Debe crear un archivo de configuración para proporcionar información sobre estos recursos al instalador del software AWS IoT Greengrass principal dentro del contenedor de Docker.

  Para utilizar el aprovisionamiento manual, debe configurar la variable de entorno Docker. `PROVISION=false` El aprovisionamiento manual es la opción predeterminada.

Para obtener más información, consulte Cree la imagen del AWS IoT Greengrass contenedor a partir de un Dockerfile.

Cree la imagen del AWS IoT Greengrass contenedor a partir de un Dockerfile

AWSproporciona un Dockerfile que puede descargar y usar para ejecutar AWS IoT Greengrass el software principal en un contenedor de Docker. Los archivos Docker contienen código fuente para crear imágenes de contenedores. AWS IoT Greengrass

Antes de crear una imagen de AWS IoT Greengrass contenedor, debe configurar su Dockerfile para seleccionar la versión del software AWS IoT Greengrass principal que desea instalar. También puedes configurar las variables de entorno para elegir cómo aprovisionar los recursos durante la instalación y personalizar otras opciones de instalación. En esta sección se describe cómo configurar y crear una imagen de AWS IoT Greengrass Docker a partir de un Dockerfile.

Descargue el paquete Dockerfile

Puede descargar el paquete AWS IoT Greengrass Dockerfile desde: GitHub

Repositorio Docker de AWS Greengrass

Tras descargar el paquete, extraiga el contenido a la *`download-directory/`*`aws-greengrass-docker-`*`nucleus-version`* carpeta de su ordenador. El Dockerfile usa una versión anterior de Greengrass. Debe actualizar el archivo para usar la versión de Greengrass que desee.

Especifique la versión del software AWS IoT Greengrass principal

Usa el siguiente argumento de compilación en el Dockerfile para especificar la versión del software AWS IoT Greengrass principal que deseas usar en la imagen de AWS IoT Greengrass Docker. De forma predeterminada, el Dockerfile usa la última versión del software Core. AWS IoT Greengrass

`GREENGRASS_RELEASE_VERSION`

La versión del software AWS IoT Greengrass principal. De forma predeterminada, el Dockerfile descarga la última versión disponible del núcleo de Greengrass. Establezca el valor en la versión del núcleo que desee descargar.

Configuración de las variables de entorno

Las variables de entorno le permiten personalizar la forma en que se instala el software AWS IoT Greengrass Core en el contenedor de Docker. Puede configurar las variables de entorno para la imagen de AWS IoT Greengrass Docker de varias maneras.

- Para usar las mismas variables de entorno para crear varias imágenes, configura las variables de entorno directamente en el Dockerfile.
- Si lo utiliza `docker run` para iniciar el contenedor, pasa variables de entorno como argumentos en el comando o establece las variables de entorno en un archivo de variables de entorno y, a continuación, pasa el archivo como argumento. Para obtener más información sobre cómo configurar las variables de entorno en Docker, consulta las variables de entorno en la documentación de Docker.
- Si solías `docker-compose up` iniciar tu contenedor, configura las variables de entorno en un archivo de variables de entorno y, a continuación, pasa el archivo como argumento. Para obtener más información sobre cómo configurar las variables de entorno en Compose, consulta la documentación de Docker.

Puedes configurar las siguientes variables de entorno para la imagen de AWS IoT Greengrass Docker.

> **Note**
>
> No modifique la `TINI_KILL_PROCESS_GROUP` variable en el Dockerfile. Esta variable permite reenviarlos `SIGTERM` a todos los PID del grupo de PID para que el software AWS IoT Greengrass principal se cierre correctamente cuando se detiene el contenedor de Docker.

`GGC_ROOT_PATH`

(Opcional) La ruta a la carpeta dentro del contenedor que se utilizará como raíz del software Core. AWS IoT Greengrass

Valor predeterminado: `/greengrass/v2`

`PROVISION`

(Opcional) Determina si el AWS IoT Greengrass núcleo AWS aprovisiona recursos.

- Si lo especifica`true`, el software AWS IoT Greengrass Core registra la imagen del contenedor como una AWS IoT cosa y aprovisiona los AWS recursos que requiere el dispositivo principal de Greengrass. El software AWS IoT Greengrass Core aprovisiona AWS IoT cualquier cosa, (opcional) un grupo de AWS IoT cosas, un rol de IAM y un alias de AWS IoT rol. Para obtener más información, consulte Se ejecuta AWS IoT Greengrass en un contenedor Docker con aprovisionamiento automático de recursos.
- Si lo especifica`false`, debe crear un archivo de configuración para proporcionárselo al instalador AWS IoT Greengrass principal que especifique el uso de los AWS recursos y certificados que creó manualmente. Para obtener más información, consulte Se ejecuta AWS IoT Greengrass en un contenedor Docker con aprovisionamiento manual de recursos.

Valor predeterminado: `false`

`AWS_REGION`

(Opcional) El Región de AWS que el software AWS IoT Greengrass principal utiliza para recuperar o crear AWS los recursos necesarios.

Valor predeterminado: `us-east-1`.

`THING_NAME`

(Opcional) El nombre de AWS IoT lo que se registra como este dispositivo principal. Si el elemento con este nombre no existe en el tuyoCuenta de AWS, será creado por el software AWS IoT Greengrass Core.

Debe especificar si `PROVISION=true` desea aplicar este argumento.

Predeterminado: `GreengrassV2IotThing_` más un UUID aleatorio.

`THING_GROUP_NAME`

(Opcional) El nombre del grupo de elementos al AWS IoT que se agrega el dispositivo principal. AWS IoT Si una implementación se dirige a este grupo, este y otros dispositivos principales de ese grupo reciben ese despliegue cuando se conectan. AWS IoT Greengrass Si el grupo de cosas con este nombre no existe en su empresaCuenta de AWS, el software AWS IoT Greengrass Core lo crea.

Debe especificar si `PROVISION=true` desea aplicar este argumento.

`TES_ROLE_NAME`

(Opcional) El nombre de la función de IAM que se utilizará para adquirir AWS las credenciales que permiten al dispositivo principal de Greengrass interactuar con AWS los servicios. Si el rol con este nombre no existe en su cuentaCuenta de AWS, el software AWS IoT Greengrass principal lo crea con la `GreengrassV2TokenExchangeRoleAccess` política. Este rol no tiene acceso a los depósitos de S3 donde alojas los artefactos de los componentes. Por lo tanto, debes añadir permisos a los cubos y objetos de S3 de tus artefactos al crear un componente. Para obtener más información, consulte Autorizar a los dispositivos principales a interactuar con AWS los servicios.

Valor predeterminado: `GreengrassV2TokenExchangeRole`

`TES_ROLE_ALIAS_NAME`

(Opcional) El nombre del alias del AWS IoT rol que apunta al rol de IAM que proporciona AWS las credenciales para el dispositivo principal de Greengrass. Si el alias del rol con este nombre no existe en su cuentaCuenta de AWS, el software AWS IoT Greengrass principal lo crea y lo dirige al rol de IAM que especifique.

Valor predeterminado: `GreengrassV2TokenExchangeRoleAlias`

`COMPONENT_DEFAULT_USER`

(Opcional) El nombre o ID del usuario y grupo del sistema que el software AWS IoT Greengrass principal utiliza para ejecutar los componentes. Especifique el usuario y el grupo, separados por dos puntos. El grupo es opcional. Por ejemplo, puede especificar **`ggc_user:ggc_group`** o **`ggc_user`**.

- Si se ejecuta como root, el usuario y el grupo que defina el archivo de configuración son de forma predeterminada. Si el archivo de configuración no define un usuario ni un grupo, el valor predeterminado es. `ggc_user:ggc_group` Si existen `ggc_user` o `ggc_group` no existen, el software los crea.
- Si se ejecuta como un usuario que no es root, el software AWS IoT Greengrass Core utiliza ese usuario para ejecutar los componentes.
- Si no especifica un grupo, el software AWS IoT Greengrass Core utiliza el grupo principal del usuario del sistema.

Para obtener más información, consulte Configure el usuario que ejecuta los componentes.

`DEPLOY_DEV_TOOLS`

Define si se debe descargar e implementar el componente CLI de Greengrass en la imagen del contenedor. Puede usar la CLI de Greengrass para desarrollar y depurar componentes localmente.

> **Important**
>
> Le recomendamos que utilice este componente únicamente en entornos de desarrollo, no en entornos de producción. Este componente proporciona acceso a información y operaciones que normalmente no necesitará en un entorno de producción. Siga el principio de privilegios mínimos implementando este componente solo en los dispositivos principales donde lo necesite.

Valor predeterminado: `false`

`INIT_CONFIG`

(Opcional) La ruta al archivo de configuración que se utilizará para instalar el software AWS IoT Greengrass principal. Puede usar esta opción para configurar nuevos dispositivos principales de Greengrass con una configuración de núcleo específica o para especificar recursos aprovisionados manualmente, por ejemplo. Debe montar el archivo de configuración en la ruta que especifique en este argumento.

`TRUSTED_PLUGIN`

Esta función está disponible para la versión 2.4.0 y versiones posteriores del componente núcleo de Greengrass.

(Opcional) La ruta a un archivo JAR para cargarlo como un complemento de confianza. Utilice esta opción para proporcionar archivos JAR a los complementos de aprovisionamiento, por ejemplo, para instalarlos con el aprovisionamiento de flota o el aprovisionamiento personalizado.

`THING_POLICY_NAME`

Esta función está disponible para la versión 2.4.0 y versiones posteriores del componente núcleo de Greengrass.

(Opcional) El nombre de la AWS IoT política que se debe adjuntar al certificado Thing de AWS IoT este dispositivo principal. Si la AWS IoT política con este nombre no existe en su software AWS IoT Greengrass Core Cuenta de AWS la crea.

Debe especificar si `PROVISION=true` desea aplicar este argumento.

> **Note**
>
> El software AWS IoT Greengrass Core crea una AWS IoT política permisiva de forma predeterminada. Puede limitar el alcance de esta política o crear una política personalizada en la que restrinja los permisos según su caso de uso. Para obtener más información, consulte AWS IoTPolítica mínima para los dispositivos AWS IoT Greengrass V2 principales.

Especifica las dependencias que deseas instalar

La instrucción RUN del AWS IoT Greengrass Dockerfile prepara el entorno del contenedor para ejecutar el instalador del software AWS IoT Greengrass principal. Puede personalizar las dependencias que se instalan antes de que el instalador del software AWS IoT Greengrass principal se ejecute en el contenedor de Docker.

Cree la imagen AWS IoT Greengrass

Usa el AWS IoT Greengrass Dockerfile para crear una imagen de AWS IoT Greengrass contenedor. Puede usar la CLI de Docker o la CLI de Docker Compose para crear la imagen e iniciar el contenedor. También puede usar la CLI de Docker para crear la imagen y, a continuación, usar Docker Compose para iniciar el contenedor a partir de esa imagen.

Docker

1. En la máquina host, ejecuta el siguiente comando para cambiar al directorio que contiene el Dockerfile configurado.

   ```
   cd download-directory/aws-greengrass-docker-nucleus-version
   ```

2. Ejecute el siguiente comando para crear la imagen del AWS IoT Greengrass contenedor a partir del Dockerfile.

   ```
   sudo docker build -t "platform/aws-iot-greengrass:nucleus-version" ./
   ```

Docker Compose

1. En la máquina host, ejecuta el siguiente comando para cambiar al directorio que contiene el Dockerfile y el archivo Compose.

   ```
   cd download-directory/aws-greengrass-docker-nucleus-version
   ```

2. Ejecuta el siguiente comando para usar el archivo Compose para crear la imagen del AWS IoT Greengrass contenedor.

   ```
   docker-compose -f docker-compose.yml build
   ```

Ha creado correctamente la imagen del AWS IoT Greengrass contenedor. La imagen de Docker tiene instalado el software AWS IoT Greengrass principal. Ahora puede ejecutar el software AWS IoT Greengrass principal en un contenedor de Docker.

Se ejecuta AWS IoT Greengrass en un contenedor Docker con aprovisionamiento automático de recursos

En este tutorial, se muestra cómo instalar y ejecutar el software AWS IoT Greengrass principal en un contenedor Docker con AWS recursos aprovisionados automáticamente y herramientas de desarrollo local. Puedes usar este entorno de desarrollo para explorar las AWS IoT Greengrass funciones de un contenedor Docker. El software requiere AWS credenciales para aprovisionar estos recursos e implementar las herramientas de desarrollo local.

Si no puede proporcionar AWS las credenciales al contenedor, puede aprovisionar los AWS recursos que el dispositivo principal necesita para funcionar. También puede implementar las herramientas de desarrollo en un dispositivo principal para usarlas como dispositivo de desarrollo. Esto le permite conceder menos permisos al dispositivo al ejecutar el contenedor. Para obtener más información, consulte Se ejecuta AWS IoT Greengrass en un contenedor Docker con aprovisionamiento manual de recursos.

Requisitos previos

Para completar este tutorial, necesitará lo siguiente.

- Una Cuenta de AWS. Si no dispone de una, consulte Configure un Cuenta de AWS.
- Un usuario de AWS IAM con permisos para aprovisionar los recursos AWS IoT de IAM para un dispositivo principal de Greengrass. El instalador del software AWS IoT Greengrass principal utiliza sus AWS credenciales para aprovisionar automáticamente estos recursos. Para obtener información sobre la política mínima de IAM para aprovisionar recursos automáticamente, consultePolítica de IAM mínima para que el instalador aprovisione recursos.
- Una imagen de AWS IoT Greengrass Docker. Puede crear una imagen a partir del AWS IoT Greengrass Dockerfile.
- El ordenador host en el que se ejecuta el contenedor de Docker debe cumplir los siguientes requisitos:
  - Un sistema operativo basado en Linux con conexión a Internet.
  - Docker Engine versión 18.09 o posterior.
  - (Opcional) Docker Compose versión 1.22 o posterior. Docker Compose solo es necesario si quieres usar la CLI de Docker Compose para ejecutar tus imágenes de Docker.

Configuración de sus credenciales de AWS

En este paso, crea un archivo de credenciales en la computadora host que contiene sus credenciales de seguridad. AWS Al ejecutar la imagen de AWS IoT Greengrass Docker, debe montar la carpeta que contiene este archivo de credenciales `/root/.aws/` en el contenedor de Docker. El AWS IoT Greengrass instalador usa estas credenciales para aprovisionar recursos en su. Cuenta de AWS Para obtener información sobre la política de IAM mínima que el instalador requiere para aprovisionar automáticamente los recursos, consultePolítica de IAM mínima para que el instalador aprovisione recursos.

1. Recupere una de las siguientes opciones.

   - Credenciales a largo plazo para un usuario de IAM. Para obtener información sobre cómo recuperar credenciales de larga duración, consulte Administrar las claves de acceso para los usuarios de IAM en la Guía del usuario de IAM.
   - (Recomendado) Credenciales temporales para un rol de IAM. Para obtener información sobre cómo recuperar credenciales temporales, consulte Uso de credenciales de seguridad temporales con la Guía del AWS CLI usuario de IAM.

2. Cree una carpeta en la que coloque el archivo de credenciales.

   ```
   mkdir ./greengrass-v2-credentials
   ```

3. Utilice un editor de texto para crear un archivo de configuración con el nombre `credentials` de la `./greengrass-v2-credentials` carpeta.

   Por ejemplo, puede ejecutar el siguiente comando para usar GNU nano para crear el `credentials` archivo.

   ```
   nano ./greengrass-v2-credentials/credentials
   ```

4. Añada sus AWS credenciales al `credentials` archivo en el siguiente formato.

   ```
   [default]
   aws_access_key_id = AKIAIOSFODNN7EXAMPLE
   aws_secret_access_key = wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
   aws_session_token
    = AQoEXAMPLEH4aoAH0gNCAPy...truncated...zrkuWJOgQs8IZZaIv2BXIa2R4Olgk
   ```

   `aws_session_token`Inclúyalo solo para credenciales temporales.

> **Important**
>
> Elimine el archivo de credenciales del equipo host después de iniciar el AWS IoT Greengrass contenedor. Si no elimina el archivo de credenciales, sus AWS credenciales permanecerán montadas dentro del contenedor. Para obtener más información, consulte Ejecute el software AWS IoT Greengrass principal en un contenedor.

Cree un archivo de entorno

Este tutorial utiliza un archivo de entorno para configurar las variables de entorno que se pasarán al instalador del software AWS IoT Greengrass Core dentro del contenedor de Docker. También puede usar el `--env` argumento `-e` o en su `docker run` comando para establecer las variables de entorno en el contenedor de Docker o puede establecer las variables en un `environment` bloque del `docker-compose.yml` archivo.

1. Usa un editor de texto para crear un archivo de entorno llamado `.env`.

   Por ejemplo, en un sistema basado en Linux, puede ejecutar el siguiente comando para usar GNU nano y crear el directorio actual `.env`.

   ```
   nano .env
   ```

2. Copie el siguiente contenido en el archivo.

   ```
   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
   ```

   A continuación, sustituya los valores siguientes.

   - *`/greengrass/v2`*. La carpeta raíz de Greengrass que desea usar para la instalación. Utilice la variable de `GGC_ROOT` entorno para establecer este valor.
   - *`region`*. El Región de AWS lugar donde creó los recursos.
   - *`MyGreengrassCore`*. Es el nombre del objeto de AWS IoT. Si la cosa no existe, el instalador la crea. El instalador descarga los certificados para autenticarse como talAWS IoT.
   - *`MyGreengrassCoreGroup`*. El nombre del grupo de AWS IoT cosas. Si el grupo de cosas no existe, el instalador lo crea y lo añade. Si el grupo de cosas existe y tiene una implementación activa, el dispositivo principal descarga y ejecuta el software que especifique la implementación.
   - *`Greengrass V2 TokenExchangeRole`*. Sustitúyalo por el nombre de la función de intercambio de fichas de IAM que permite al dispositivo principal de Greengrass obtener AWS

credenciales temporales. *Si el rol no existe, el instalador lo crea y crea y adjunta una política denominada GreengrassV2 Access. TokenExchangeRole* Para obtener más información, consulte Autorizar a los dispositivos principales a interactuar con AWS los servicios.

- *GreengrassCoreTokenExchangeRoleAlias*. El alias del rol de intercambio de fichas. Si el alias del rol no existe, el instalador lo crea y lo dirige al rol de intercambio de tokens de IAM que especifique. Para obtener más información, consulte

> **Note**
>
> Puede configurar la variable de `DEPLOY_DEV_TOOLS` entorno `true` para implementar el componente CLI de Greengrass, que le permite desarrollar componentes personalizados dentro del contenedor de Docker. Le recomendamos que utilice este componente únicamente en entornos de desarrollo, no en entornos de producción. Este componente proporciona acceso a información y operaciones que normalmente no necesitará en un entorno de producción. Siga el principio de privilegios mínimos implementando este componente solo en los dispositivos principales donde lo necesite.

Ejecute el software AWS IoT Greengrass principal en un contenedor

Este tutorial le muestra cómo iniciar la imagen de Docker que creó en un contenedor de Docker. Puede usar la CLI de Docker o la CLI de Docker Compose para ejecutar la imagen del software AWS IoT Greengrass principal en un contenedor de Docker.

Docker

1. Ejecute el siguiente comando para iniciar el contenedor de 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
   ```

   Este comando de ejemplo usa los siguientes argumentos para docker run:

   - `--rm`. Limpia el contenedor al salir.

- `--init`. Utiliza un proceso de inicio en el contenedor.

  > **Note**
  >
  > El `--init` argumento es necesario para cerrar el software AWS IoT Greengrass Core al detener el contenedor de Docker.

- `-it`. (Opcional) Ejecuta el contenedor de Docker en primer plano como un proceso interactivo. En su lugar, puede sustituirlo por el `-d` argumento para ejecutar el contenedor Docker en modo separado. Para obtener más información, consulte Separado frente a primer plano en la documentación de Docker.
- `--name`. Ejecuta un contenedor llamado `aws-iot-greengrass`
- `-v`. Monta un volumen en el contenedor de Docker para que el archivo de configuración y los archivos de certificado estén disponibles para su AWS IoT Greengrass ejecución dentro del contenedor.
- `--env-file`. (Opcional) Especifica el archivo de entorno para establecer las variables de entorno que se pasarán al instalador del software AWS IoT Greengrass principal dentro del contenedor de Docker. Este argumento solo es necesario si ha creado un archivo de entorno para establecer variables de entorno. Si no creó un archivo de entorno, puede usar `--env` argumentos para establecer las variables de entorno directamente en el comando run de Docker.
- `-p`. (Opcional) Publica el puerto contenedor 8883 en la máquina host. Este argumento es obligatorio si desea conectarse y comunicarse a través de MQTT, ya que AWS IoT Greengrass utiliza el puerto 8883 para el tráfico de MQTT. Para abrir otros puertos, utilice argumentos adicionales. `-p`

  > **Note**
  >
  > Para ejecutar su contenedor de Docker con mayor seguridad, puede usar los `--cap-add` argumentos `--cap-drop` y para habilitar de forma selectiva las capacidades de Linux para su contenedor. Para obtener más información, consulta los privilegios de tiempo de ejecución y las capacidades de Linux en la documentación de Docker.

2. Elimine las credenciales `./greengrass-v2-credentials` del dispositivo anfitrión.

```
rm -rf ./greengrass-v2-credentials
```

> **Important**
>
> Vas a eliminar estas credenciales porque proporcionan amplios permisos que el dispositivo principal solo necesita durante la configuración. Si no eliminas estas credenciales, los componentes de Greengrass y otros procesos que se ejecutan en el contenedor podrán acceder a ellas. Si necesita proporcionar AWS credenciales a un componente de Greengrass, utilice el servicio de intercambio de fichas. Para obtener más información, consulte Interactúa con AWS los servicios.

Docker Compose

1. Use un editor de texto para crear un archivo de Docker Compose llamado. `docker-compose.yml`

   Por ejemplo, en un sistema basado en Linux, puede ejecutar el siguiente comando para usar GNU nano y crear el `docker-compose.yml` directorio actual.

   ```
   nano docker-compose.yml
   ```

   > **Note**
   >
   > También puedes descargar y usar la última versión del archivo Compose proporcionado desdeAWS. GitHub

2. Agrega el siguiente contenido al archivo Compose. El archivo debe ser similar al siguiente ejemplo. Sustituya *`docker-image por`* el nombre de su imagen de Docker.

   ```
   version: '3.7'

   services:
     greengrass:
       init: true
       container_name: aws-iot-greengrass
       image: docker-image
       volumes:
   ```

```
      - ./greengrass-v2-credentials:/root/.aws/:ro
    env_file: .env
    ports:
      - "8883:8883"
```

Los siguientes parámetros del archivo Compose de ejemplo son opcionales:

- `ports`—Publica los 8883 puertos del contenedor en la máquina host. Este parámetro es obligatorio si desea conectarse y comunicarse a través de MQTT, ya que AWS IoT Greengrass utiliza el puerto 8883 para el tráfico de MQTT.
- `env_file`—Especifica el archivo de entorno para establecer las variables de entorno que se pasarán al instalador del software AWS IoT Greengrass principal dentro del contenedor de Docker. Este parámetro solo es necesario si ha creado un archivo de entorno para configurar las variables de entorno. Si no has creado un archivo de entorno, puedes usar el parámetro de entorno para configurar las variables directamente en tu archivo de Compose.

> **Note**
>
> Para ejecutar tu contenedor de Docker con mayor seguridad, puedes usar `cap_drop` y `cap_add` en tu archivo de Compose para habilitar de forma selectiva las capacidades de Linux para tu contenedor. Para obtener más información, consulta los privilegios de tiempo de ejecución y las capacidades de Linux en la documentación de Docker.

3. Ejecute el siguiente comando para iniciar el contenedor de Docker.

```
docker-compose -f docker-compose.yml up
```

4. Elimine las credenciales `./greengrass-v2-credentials` del dispositivo anfitrión.

```
rm -rf ./greengrass-v2-credentials
```

> **Important**
>
> Vas a eliminar estas credenciales porque proporcionan amplios permisos que el dispositivo principal solo necesita durante la configuración. Si no eliminas estas

credenciales, los componentes de Greengrass y otros procesos que se ejecutan en el contenedor podrán acceder a ellas. Si necesita proporcionar AWS credenciales a un componente de Greengrass, utilice el servicio de intercambio de fichas. Para obtener más información, consulte Interactúa con AWS los servicios.

Siguientes pasos

AWS IoT GreengrassEl software principal se ejecuta ahora en un contenedor Docker. Ejecute el siguiente comando para recuperar el ID del contenedor que se está ejecutando actualmente.

```
docker ps
```

A continuación, puede ejecutar el siguiente comando para acceder al contenedor y explorar el software AWS IoT Greengrass principal que se ejecuta dentro del contenedor.

```
docker exec -it container-id /bin/bash
```

Para obtener información sobre cómo crear un componente simple, consulte Paso 4: Desarrolle y pruebe un componente en su dispositivo en Tutorial: Introducción a AWS IoT Greengrass V2

> **Note**
>
> Cuando se ejecutan comandos dentro del contenedor de Docker, esos comandos no se registran en los registros de Docker. `docker exec` Para registrar tus comandos en los registros de Docker, adjunta un shell interactivo al contenedor de Docker. Para obtener más información, consulte Adjunte un shell interactivo al contenedor de Docker.

El archivo de registro AWS IoT Greengrass principal se llama `greengrass.log` y se encuentra en. */greengrass/v2*`/logs` Los archivos de registro de los componentes también se encuentran en el mismo directorio. Para copiar los registros de Greengrass a un directorio temporal del host, ejecute el siguiente comando:

```
docker cp container-id:/greengrass/v2/logs /tmp/logs
```

Si desea conservar los registros después de que un contenedor se cierre o se haya eliminado, le recomendamos que monte únicamente el */greengrass/v2*`/logs` directorio en el directorio de

registros temporales del host, en lugar de montar todo el directorio de Greengrass. Para obtener más información, consulte Conserva los registros de Greengrass fuera del contenedor de Docker.

Para detener un contenedor AWS IoT Greengrass Docker en ejecución, ejecute o. `docker stop` `docker-compose -f docker-compose.yml stop` Esta acción se envía `SIGTERM` al proceso de Greengrass y cierra todos los procesos asociados que se iniciaron en el contenedor. El contenedor Docker se inicializa con el `docker-init` ejecutable como PID de proceso 1, lo que ayuda a eliminar cualquier proceso zombi sobrante. Para obtener más información, consulte Especificar un proceso de inicio en la documentación de Docker.

Para obtener información sobre cómo solucionar problemas relacionados con la ejecución AWS IoT Greengrass en un contenedor Docker, consulte. Solución de problemas de AWS IoT Greengrass en un contenedor Docker

Se ejecuta AWS IoT Greengrass en un contenedor Docker con aprovisionamiento manual de recursos

En este tutorial, se muestra cómo instalar y ejecutar el software AWS IoT Greengrass principal en un contenedor Docker con recursos aprovisionados manualmente. AWS

Temas

Requisitos previos

Necesitará lo siguiente para completar este tutorial:

- Una Cuenta de AWS. Si no dispone de una, consulte Configure un Cuenta de AWS.
- Una imagen de AWS IoT Greengrass Docker. Puede crear una imagen a partir del AWS IoT Greengrass Dockerfile.
- El ordenador host en el que se ejecuta el contenedor de Docker debe cumplir los siguientes requisitos:
  - Un sistema operativo basado en Linux con conexión a Internet.
  - Docker Engine versión 18.09 o posterior.
  - (Opcional) Docker Compose versión 1.22 o posterior. Docker Compose solo es necesario si quieres usar la CLI de Docker Compose para ejecutar tus imágenes de Docker.

Recupera los puntos finales AWS IoT

Obtenga sus AWS IoT Cuenta de AWS puntos finales y guárdelos para usarlos más tarde. El dispositivo utiliza estos puntos finales para conectarse a ellos. AWS IoT Haga lo siguiente:

1. Obtenga el punto final AWS IoT de datos para su. Cuenta de AWS

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

   La respuesta es similar a la del siguiente ejemplo, si la solicitud se realiza correctamente.

   ```
   {
     "endpointAddress": "device-data-prefix-ats.iot.us-west-2.amazonaws.com"
   }
   ```

2. Obtenga el punto final de AWS IoT credenciales para su. Cuenta de AWS

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

   La respuesta es similar a la del siguiente ejemplo, si la solicitud se realiza correctamente.

   ```
   {
     "endpointAddress": "device-credentials-prefix.credentials.iot.us-
   west-2.amazonaws.com"
   }
   ```

Creación de un objeto de AWS IoT

AWS IoTlas cosas representan dispositivos y entidades lógicas a AWS IoT los que se conectan. Los dispositivos principales de Greengrass son AWS IoT cosas. Cuando registras un dispositivo como una AWS IoT cosa, ese dispositivo puede usar un certificado digital para autenticarse. AWS

En esta sección, crearás AWS IoT algo que represente tu dispositivo.

Crear un objeto de AWS IoT

1. Crea cualquier AWS IoT cosa para tu dispositivo. En tu ordenador de desarrollo, ejecuta el siguiente comando.

   - *MyGreengrassCore*Sustitúyalo por el nombre de la cosa que se va a utilizar. Este nombre también es el nombre de su dispositivo principal de Greengrass.

     > **Note**
     >
     > El nombre del objeto no puede contener dos puntos (`:`).

   ```
   aws iot create-thing --thing-name MyGreengrassCore
   ```

   Si la solicitud se realiza correctamente, la respuesta es similar a la del siguiente ejemplo.

   ```
   {
     "thingName": "MyGreengrassCore",
     "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/MyGreengrassCore",
     "thingId": "8cb4b6cd-268e-495d-b5b9-1713d71dbf42"
   }
   ```

2. (Opcional) Añada la AWS IoT cosa a un grupo de cosas nuevo o existente. Los grupos de cosas se utilizan para gestionar las flotas de dispositivos principales de Greengrass. Al implementar componentes de software en sus dispositivos, puede dirigirlos a dispositivos individuales o a grupos de dispositivos. Puede añadir un dispositivo a un grupo de cosas con una implementación activa de Greengrass para implementar los componentes de software de ese grupo de cosas en el dispositivo. Haga lo siguiente:

   a. (Opcional) Cree un grupo de AWS IoT cosas.

- *MyGreengrassCoreGroup*Sustitúyalo por el nombre del grupo de cosas que desee crear.

> **Note**
>
> El nombre del grupo de cosas no puede contener dos puntos (`:`).

```
aws iot create-thing-group --thing-group-name MyGreengrassCoreGroup
```

La respuesta es similar a la del siguiente ejemplo, si la solicitud se realiza correctamente.

```
{
  "thingGroupName": "MyGreengrassCoreGroup",
  "thingGroupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/
MyGreengrassCoreGroup",
  "thingGroupId": "4df721e1-ff9f-4f97-92dd-02db4e3f03aa"
}
```

b. Añada la AWS IoT cosa a un grupo de cosas.

- *MyGreengrassCore*Sustitúyala por el nombre de la AWS IoT cosa.
- *MyGreengrassCoreGroup*Sustitúyalo por el nombre del grupo de cosas.

```
aws iot add-thing-to-thing-group --thing-name MyGreengrassCore --thing-group-
name MyGreengrassCoreGroup
```

El comando no tiene ningún resultado si la solicitud se realiza correctamente.

Crea el certificado de la cosa

Cuando registras un dispositivo como una AWS IoT cosa, ese dispositivo puede usar un certificado digital para autenticarseAWS. Este certificado permite que el dispositivo se comunique con AWS IoT yAWS IoT Greengrass.

En esta sección, puede crear y descargar certificados que el dispositivo puede usar para conectarseAWS.

Para crear la cosa: certificado

1. Crea una carpeta donde descargues los certificados de la AWS IoT cosa.

```
mkdir greengrass-v2-certs
```

2. Crea y descarga los certificados de la AWS IoT cosa.

```
aws iot create-keys-and-certificate --set-as-active --certificate-pem-outfile
 greengrass-v2-certs/device.pem.crt --public-key-outfile greengrass-v2-certs/
public.pem.key --private-key-outfile greengrass-v2-certs/private.pem.key
```

La respuesta es similar a la del siguiente ejemplo, si la solicitud se realiza correctamente.

```
{
  "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-----\
"
  }
}
```

Guarde el nombre de recurso de Amazon (ARN) del certificado para configurarlo más adelante.

Configure el certificado del objeto

Adjunte el certificado del AWS IoT objeto al elemento que creó anteriormente y añada una AWS IoT política al certificado para definir los AWS IoT permisos del dispositivo principal.

Para configurar el certificado de la cosa

1. Adjunte el certificado a la AWS IoT cosa.

   - Reemplácelo *MyGreengrassCore*con el nombre de su AWS IoT cosa.
   - Sustituya el certificado Amazon Resource Name (ARN) por el ARN del certificado que creó en el paso anterior.

   ```
   aws iot attach-thing-principal --thing-name MyGreengrassCore
    --principal arn:aws:iot:us-west-2:123456789012:cert/
   aa0b7958770878eabe251d8a7ddd547f4889c524c9b574ab9fbf65f32248b1d4
   ```

   El comando no tiene ningún resultado si la solicitud se realiza correctamente.

2. Cree y adjunte una AWS IoT política que defina los AWS IoT permisos de su dispositivo principal de Greengrass. La siguiente política permite el acceso a todos los temas de MQTT y a las operaciones de Greengrass, de modo que su dispositivo funcione con aplicaciones personalizadas y con cambios futuros que requieran nuevas operaciones de Greengrass. Puede

restringir esta política en función de su caso de uso. Para obtener más información, consulte AWS IoTPolítica mínima para los dispositivos AWS IoT Greengrass V2 principales.

Si ya ha configurado un dispositivo principal de Greengrass, puede adjuntar su AWS IoT política en lugar de crear una nueva.

Haga lo siguiente:

a. Cree un archivo que contenga el documento AWS IoT de política que requieren los dispositivos principales de Greengrass.

   Por ejemplo, en un sistema basado en Linux, puede ejecutar el siguiente comando para usar GNU nano para crear el archivo.

   ```
   nano greengrass-v2-iot-policy.json
   ```

   Copia el siguiente JSON en el archivo.

   ```
   {
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Action": [
           "iot:Publish",
           "iot:Subscribe",
           "iot:Receive",
           "iot:Connect",
           "greengrass:*"
         ],
         "Resource": [
           "*"
         ]
       }
     ]
   }
   ```

b. Cree una AWS IoT política a partir del documento de política.

   - Sustituya *GreengrassV2IoT* por el nombre ThingPolicy de la política que se va a crear.

```
aws iot create-policy --policy-name GreengrassV2IoTThingPolicy --policy-
document file://greengrass-v2-iot-policy.json
```

Si la solicitud se realiza correctamente, la respuesta es similar a la del siguiente ejemplo.

```
{
  "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. Adjunta la AWS IoT política al certificado de la AWS IoT cosa.

- Sustituya *GreengrassV2IoT* por el nombre ThingPolicy de la política que desee adjuntar.
- Sustituya el ARN de destino por el ARN del certificado de su dispositivo. AWS IoT

```
aws iot attach-policy --policy-name GreengrassV2IoTThingPolicy
 --target arn:aws:iot:us-west-2:123456789012:cert/
aa0b7958770878eabe251d8a7ddd547f4889c524c9b574ab9fbf65f32248b1d4
```

El comando no tiene ningún resultado si la solicitud se realiza correctamente.

Crea un rol de intercambio de fichas

Los dispositivos principales de Greengrass utilizan una función de servicio de IAM, denominada función de intercambio de fichas, para autorizar las llamadas a los servicios. AWS El dispositivo utiliza el proveedor de AWS IoT credenciales para obtener AWS credenciales temporales para esta función, lo que permite al dispositivo interactuar con Amazon LogsAWS IoT, enviar registros a Amazon CloudWatch Logs y descargar artefactos de componentes personalizados de Amazon S3. Para obtener más información, consulte Autorizar a los dispositivos principales a interactuar con AWS los servicios.

Se utiliza un alias de AWS IoT rol para configurar el rol de intercambio de fichas para los dispositivos principales de Greengrass. Los alias de rol le permiten cambiar el rol de intercambio de fichas de un dispositivo, pero mantener la configuración del dispositivo igual. Para obtener más información, consulte Autorizar llamadas directas a AWS los servicios en la Guía para AWS IoT Coredesarrolladores.

En esta sección, se crea una función de IAM de intercambio de fichas y un alias de AWS IoT función que apunte a esa función. Si ya ha configurado un dispositivo principal de Greengrass, puede utilizar su función de intercambio de fichas y su alias de función en lugar de crear otros nuevos. A continuación, configura el dispositivo para que utilice ese rol y ese alias. AWS IoT

Para crear un rol de IAM para el intercambio de fichas

1. Cree una función de IAM que su dispositivo pueda utilizar como función de intercambio de fichas. Haga lo siguiente:

   a. Cree un archivo que contenga el documento de política de confianza que requiere la función de intercambio de tokens.

      Por ejemplo, en un sistema basado en Linux, puede ejecutar el siguiente comando para usar GNU nano para crear el archivo.

      ```
      nano device-role-trust-policy.json
      ```

      Copia el siguiente JSON en el archivo.

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

b. Cree la función de intercambio de fichas con el documento de política de confianza.

- Sustituya *GreengrassV2 TokenExchangeRole* por el nombre del rol de IAM que desee crear.

```
aws iam create-role --role-name GreengrassV2TokenExchangeRole --assume-role-
policy-document file://device-role-trust-policy.json
```

Si la solicitud se realiza correctamente, la respuesta es similar a la del siguiente ejemplo.

```
{
  "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. Cree un archivo que contenga el documento de política de acceso que requiere la función de intercambio de tokens.

   Por ejemplo, en un sistema basado en Linux, puede ejecutar el siguiente comando para usar GNU nano para crear el archivo.

```
nano device-role-access-policy.json
```

   Copia el siguiente JSON en el archivo.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:DescribeLogStreams",
        "s3:GetBucketLocation"
      ],
      "Resource": "*"
    }
  ]
}
```

> **Note**
>
> Esta política de acceso no permite el acceso a los artefactos de los componentes de los depósitos de S3. Para implementar componentes personalizados que definan artefactos en Amazon S3, debe añadir permisos al rol para permitir que su dispositivo principal recupere artefactos de componentes. Para obtener más información, consulte Permita el acceso a los depósitos de S3 para los artefactos de los componentes.

Si aún no tiene un depósito de S3 para los artefactos de los componentes, puede añadir estos permisos más adelante, después de crear un depósito.

d. Cree la política de IAM a partir del documento de política.

- Sustituya *GreengrassV2 TokenExchangeRoleAccess* por el nombre de la política de IAM que desee crear.

```
aws iam create-policy --policy-name GreengrassV2TokenExchangeRoleAccess --
policy-document file://device-role-access-policy.json
```

Si la solicitud se realiza correctamente, la respuesta es similar a la del siguiente ejemplo.

```
{
  "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. Adjunte la política de IAM a la función de intercambio de fichas.

- Sustituya *GreengrassV2 TokenExchangeRole* por el nombre de la función de IAM.
- Sustituya el ARN de la política por el ARN de la política de IAM que creó en el paso anterior.

```
aws iam attach-role-policy --role-name GreengrassV2TokenExchangeRole --policy-
arn arn:aws:iam::123456789012:policy/GreengrassV2TokenExchangeRoleAccess
```

El comando no tiene ningún resultado si la solicitud se realiza correctamente.

2. Cree un alias de AWS IoT rol que apunte al rol de intercambio de fichas.

   - *GreengrassCoreTokenExchangeRoleAlias*Sustitúyalo por el nombre del alias del rol que se va a crear.
   - Sustituya el ARN del rol por el ARN del rol de IAM que creó en el paso anterior.

   ```
   aws iot create-role-alias --role-alias GreengrassCoreTokenExchangeRoleAlias --role-
   arn arn:aws:iam::123456789012:role/GreengrassV2TokenExchangeRole
   ```

   Si la solicitud se realiza correctamente, la respuesta es similar a la del ejemplo siguiente.

   ```
   {
     "roleAlias": "GreengrassCoreTokenExchangeRoleAlias",
     "roleAliasArn": "arn:aws:iot:us-west-2:123456789012:rolealias/
   GreengrassCoreTokenExchangeRoleAlias"
   }
   ```

   > **Note**
   >
   > Para crear un alias de rol, debe tener permiso para transferir el rol de IAM de intercambio de fichas. AWS IoT Si recibe un mensaje de error al intentar crear un alias de rol, compruebe que su AWS usuario tiene este permiso. Para obtener más información, consulte Conceder permisos a un usuario para transferir un rol a un AWS servicio en la Guía del AWS Identity and Access Management usuario.

3. Cree y adjunte una AWS IoT política que permita a su dispositivo principal de Greengrass utilizar el alias del rol para asumir el rol de intercambio de fichas. Si ya ha configurado un dispositivo principal de Greengrass, puede adjuntar su AWS IoT política de alias de rol en lugar de crear una nueva. Haga lo siguiente:

   a. (Opcional) Cree un archivo que contenga el documento AWS IoT de política que requiere el alias del rol.

      Por ejemplo, en un sistema basado en Linux, puede ejecutar el siguiente comando para usar GNU nano para crear el archivo.

      ```
      nano greengrass-v2-iot-role-alias-policy.json
      ```

Copia el siguiente JSON en el archivo.

- Sustituya el ARN del recurso por el ARN del alias de su rol.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:AssumeRoleWithCertificate",
      "Resource": "arn:aws:iot:us-west-2:123456789012:rolealias/
GreengrassCoreTokenExchangeRoleAlias"
    }
  ]
}
```

b. Cree una AWS IoT política a partir del documento de política.

- *GreengrassCoreTokenExchangeRoleAliasPolicy*Sustitúyala por el nombre de la AWS IoT política que se va a crear.

```
aws iot create-policy --policy-name GreengrassCoreTokenExchangeRoleAliasPolicy
 --policy-document file://greengrass-v2-iot-role-alias-policy.json
```

Si la solicitud se realiza correctamente, la respuesta es similar a la del ejemplo siguiente.

```
{
  "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. Adjunta la AWS IoT política al certificado de la AWS IoT cosa.

- *GreengrassCoreTokenExchangeRoleAliasPolicy*Sustitúyala por el nombre de la AWS IoT política de alias del rol.
- Sustituya el ARN de destino por el ARN del certificado de su dispositivo. AWS IoT

```
aws iot attach-policy --policy-name GreengrassCoreTokenExchangeRoleAliasPolicy
 --target arn:aws:iot:us-west-2:123456789012:cert/
aa0b7958770878eabe251d8a7ddd547f4889c524c9b574ab9fbf65f32248b1d4
```

El comando no tiene ningún resultado si la solicitud se realiza correctamente.

Descargue los certificados al dispositivo

Anteriormente, descargaste el certificado de tu dispositivo en tu ordenador de desarrollo. En esta sección, descargas el certificado de la autoridad de certificación (CA) raíz de Amazon. Luego, si planea ejecutar el software AWS IoT Greengrass principal de Docker en un equipo diferente al de su equipo de desarrollo, debe copiar los certificados en ese equipo host. El software AWS IoT Greengrass Core usa estos certificados para conectarse al servicio AWS IoT en la nube.

Para descargar los certificados al dispositivo

1. En tu ordenador de desarrollo, descarga el certificado de la autoridad de certificación raíz (CA) de Amazon. AWS IoTlos certificados están asociados al certificado de CA raíz de Amazon de forma predeterminada.

   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. Si planea ejecutar el software AWS IoT Greengrass principal de Docker en un dispositivo diferente al de su ordenador de desarrollo, copie los certificados en el ordenador host. Si SSH y SCP están habilitados en la computadora de desarrollo y la computadora host, puede usar el `scp` comando en su computadora de desarrollo para transferir los certificados. *`device-ip-address`*Sustitúyala por la dirección IP de la computadora host.

```
scp -r greengrass-v2-certs/ device-ip-address:~
```

Cree un archivo de configuración

1. En el ordenador host, cree una carpeta en la que coloque el archivo de configuración.

```
mkdir ./greengrass-v2-config
```

2. Utilice un editor de texto para crear un archivo de configuración con `config.yaml` el nombre de la `./greengrass-v2-config` carpeta.

   Por ejemplo, puede ejecutar el siguiente comando para usar GNU nano para crear el`config.yaml`.

```
nano ./greengrass-v2-config/config.yaml
```

3. Copia el siguiente contenido de YAML en el archivo. Este archivo de configuración parcial especifica los parámetros del sistema y los parámetros del núcleo de 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"
```

A continuación, sustituya los valores siguientes:

- */tmp/certs*. El directorio del contenedor de Docker en el que se montan los certificados descargados al iniciar el contenedor.
- */greengrass/v2*. La carpeta raíz de Greengrass que desea usar para la instalación. Utilice la variable de `GGC_ROOT` entorno para establecer este valor.
- *MyGreengrassCore*. Es el nombre del objeto de AWS IoT.
- *versión núcleo.* La versión del software AWS IoT Greengrass principal que se va a instalar. Este valor debe coincidir con la versión de la imagen de Docker o del Dockerfile que descargó. Si has descargado la imagen de Docker de Greengrass con la `latest` etiqueta, úsala **`docker inspect`** ***`image-id`*** para ver la versión de la imagen.
- *region*. El Región de AWS lugar donde creó sus recursos. AWS IoT También debe especificar el mismo valor para la variable de `AWS_REGION` entorno en el archivo de entorno.
- *GreengrassCoreTokenExchangeRoleAlias*. El alias del rol de intercambio de fichas.
- *device-data-prefix*. El prefijo del punto final AWS IoT de datos.
- *device-credentials-prefix*. El prefijo del punto final de tus AWS IoT credenciales.

Cree un archivo de entorno

Este tutorial utiliza un archivo de entorno para configurar las variables de entorno que se pasarán al instalador del software AWS IoT Greengrass Core dentro del contenedor de Docker. También puede usar el `--env` argumento `-e` o en su `docker run` comando para establecer las variables de

entorno en el contenedor de Docker o puede establecer las variables en un `environment` bloque del `docker-compose.yml` archivo.

1. Usa un editor de texto para crear un archivo de entorno llamado `.env`.

   Por ejemplo, en un sistema basado en Linux, puede ejecutar el siguiente comando para usar GNU nano y crear el directorio actual `.env`.

   ```
   nano .env
   ```

2. Copie el siguiente contenido en el archivo.

   ```
   GGC_ROOT_PATH=/greengrass/v2
   AWS_REGION=region
   PROVISION=false
   COMPONENT_DEFAULT_USER=ggc_user:ggc_group
   INIT_CONFIG=/tmp/config/config.yaml
   ```

   A continuación, sustituya los valores siguientes.

   - */greengrass/v2*. La ruta a la carpeta raíz que se utilizará para instalar el software AWS IoT Greengrass principal.
   - *region*. El Región de AWS lugar donde creaste tus AWS IoT recursos. Debe especificar el mismo valor para el parámetro `awsRegion` de configuración en el archivo de configuración.
   - */tmp/config/*. La carpeta en la que se monta el archivo de configuración al iniciar el contenedor de Docker.

   > **Note**
   >
   > Puede configurar la variable de `DEPLOY_DEV_TOOLS` entorno `true` para implementar el componente CLI de Greengrass, que le permite desarrollar componentes personalizados dentro del contenedor de Docker. Le recomendamos que utilice este componente únicamente en entornos de desarrollo, no en entornos de producción. Este componente proporciona acceso a información y operaciones que normalmente no necesitará en un entorno de producción. Siga el principio de privilegios mínimos implementando este componente solo en los dispositivos principales donde lo necesite.

Ejecute el software AWS IoT Greengrass principal en un contenedor

Este tutorial le muestra cómo iniciar la imagen de Docker que creó en un contenedor de Docker. Puede usar la CLI de Docker o la CLI de Docker Compose para ejecutar la imagen del software AWS IoT Greengrass principal en un contenedor de Docker.

Docker

- Este tutorial le muestra cómo iniciar la imagen de Docker que creó en un contenedor de 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
  ```

  Este comando de ejemplo usa los siguientes argumentos para docker run:

  - `--rm`. Limpia el contenedor al salir.
  - `--init`. Utiliza un proceso de inicio en el contenedor.

    > **Note**
    >
    > El `--init` argumento es necesario para cerrar el software AWS IoT Greengrass Core al detener el contenedor de Docker.

  - `-it`. (Opcional) Ejecuta el contenedor de Docker en primer plano como un proceso interactivo. En su lugar, puede sustituirlo por el `-d` argumento para ejecutar el contenedor Docker en modo separado. Para obtener más información, consulte Separado frente a primer plano en la documentación de Docker.
  - `--name`. Ejecuta un contenedor llamado `aws-iot-greengrass`
  - `-v`. Monta un volumen en el contenedor de Docker para que el archivo de configuración y los archivos de certificado estén disponibles para su AWS IoT Greengrass ejecución dentro del contenedor.
  - `--env-file`. (Opcional) Especifica el archivo de entorno para establecer las variables de entorno que se pasarán al instalador del software AWS IoT Greengrass principal dentro

del contenedor de Docker. Este argumento solo es necesario si ha creado un archivo de entorno para establecer variables de entorno. Si no creó un archivo de entorno, puede usar `--env` argumentos para establecer las variables de entorno directamente en el comando run de Docker.

- `-p`. (Opcional) Publica el puerto contenedor 8883 en la máquina host. Este argumento es obligatorio si desea conectarse y comunicarse a través de MQTT, ya que AWS IoT Greengrass utiliza el puerto 8883 para el tráfico de MQTT. Para abrir otros puertos, utilice argumentos adicionales. `-p`

> **Note**
>
> Para ejecutar su contenedor de Docker con mayor seguridad, puede usar los `--cap-add` argumentos `--cap-drop` y para habilitar de forma selectiva las capacidades de Linux para su contenedor. Para obtener más información, consulta los privilegios de tiempo de ejecución y las capacidades de Linux en la documentación de Docker.

Docker Compose

1. Usa un editor de texto para crear un archivo de Docker Compose llamado. `docker-compose.yml`

   Por ejemplo, en un sistema basado en Linux, puede ejecutar el siguiente comando para usar GNU nano y crear el `docker-compose.yml` directorio actual.

   ```
   nano docker-compose.yml
   ```

   > **Note**
   >
   > También puedes descargar y usar la última versión del archivo Compose proporcionado desdeAWS. GitHub

2. Agrega el siguiente contenido al archivo Compose. El archivo debe ser similar al siguiente ejemplo. *`Sustituye:version your-container-name por el nombre de la imagen de Docker.`*

   ```
   version: '3.7'
   ```

```
services:
  greengrass:
    init: true
    build:
      context: .
    container_name: aws-iot-greengrass
    image: your-container-name:version
    volumes:
      - /path/to/greengrass-v2-config:/tmp/config/:ro
      - /path/to/greengrass-v2-certs:/tmp/certs:ro
    env_file: .env
    ports:
      - "8883:8883"
```

Los siguientes parámetros de este archivo Compose de ejemplo son opcionales:

- `ports`—Publica los 8883 puertos del contenedor en la máquina host. Este parámetro es obligatorio si desea conectarse y comunicarse a través de MQTT, ya que AWS IoT Greengrass utiliza el puerto 8883 para el tráfico de MQTT.
- `env_file`—Especifica el archivo de entorno para establecer las variables de entorno que se pasarán al instalador del software AWS IoT Greengrass principal dentro del contenedor de Docker. Este parámetro solo es necesario si ha creado un archivo de entorno para configurar las variables de entorno. Si no has creado un archivo de entorno, puedes usar el parámetro de entorno para configurar las variables directamente en tu archivo de Compose.

> **Note**
>
> Para ejecutar tu contenedor de Docker con mayor seguridad, puedes usar `cap_drop` y `cap_add` en tu archivo de Compose para habilitar de forma selectiva las capacidades de Linux para tu contenedor. Para obtener más información, consulta los privilegios de tiempo de ejecución y las capacidades de Linux en la documentación de Docker.

3. Ejecute el siguiente comando para iniciar el contenedor.

```
docker-compose -f docker-compose.yml up
```

Siguientes pasos

AWS IoT GreengrassEl software principal se ejecuta ahora en un contenedor Docker. Ejecute el siguiente comando para recuperar el ID del contenedor que se está ejecutando actualmente.

```
docker ps
```

A continuación, puede ejecutar el siguiente comando para acceder al contenedor y explorar el software AWS IoT Greengrass principal que se ejecuta dentro del contenedor.

```
docker exec -it container-id /bin/bash
```

Para obtener información sobre cómo crear un componente simple, consulte Paso 4: Desarrolle y pruebe un componente en su dispositivo en Tutorial: Introducción a AWS IoT Greengrass V2

> **Note**
>
> Cuando se ejecutan comandos dentro del contenedor de Docker, esos comandos no se registran en los registros de Docker. `docker exec` Para registrar tus comandos en los registros de Docker, adjunta un shell interactivo al contenedor de Docker. Para obtener más información, consulte Adjunte un shell interactivo al contenedor de Docker.

El archivo de registro AWS IoT Greengrass principal se llama `greengrass.log` y se encuentra en. */greengrass/v2*`/logs` Los archivos de registro de los componentes también se encuentran en el mismo directorio. Para copiar los registros de Greengrass a un directorio temporal del host, ejecute el siguiente comando:

```
docker cp container-id:/greengrass/v2/logs /tmp/logs
```

Si desea conservar los registros después de que un contenedor se cierre o se haya eliminado, le recomendamos que monte únicamente el */greengrass/v2*`/logs` directorio en el directorio de registros temporales del host, en lugar de montar todo el directorio de Greengrass. Para obtener más información, consulte Conserva los registros de Greengrass fuera del contenedor de Docker.

Para detener un contenedor de AWS IoT Greengrass Docker en ejecución, ejecute o. `docker stop` `docker-compose -f docker-compose.yml stop` Esta acción se envía `SIGTERM` al proceso de Greengrass y cierra todos los procesos asociados que se iniciaron en el contenedor. El contenedor Docker se inicializa con el `docker-init` ejecutable como PID de proceso 1, lo que ayuda a eliminar

cualquier proceso zombi sobrante. Para obtener más información, consulte Especificar un proceso de inicio en la documentación de Docker.

Para obtener información sobre cómo solucionar problemas relacionados con la ejecución AWS IoT Greengrass en un contenedor Docker, consulte. Solución de problemas de AWS IoT Greengrass en un contenedor Docker

Solución de problemas de AWS IoT Greengrass en un contenedor Docker

Usa la siguiente información para ayudarte a solucionar problemas relacionados con la ejecución AWS IoT Greengrass en un contenedor de Docker y para depurar los problemas AWS IoT Greengrass en el contenedor de Docker.

Temas

Solución de problemas relacionados con la ejecución del contenedor Docker

Utilice la siguiente información para ayudar a solucionar problemas con la ejecución de AWS IoT Greengrass en un contenedor de Docker.

Temas

Error: No se puede realizar un inicio de sesión interactivo desde un dispositivo que no sea TTY

Este error puede producirse al ejecutar el `aws ecr get-login-password` comando. Asegúrese de haber instalado la última versión 2 o la versión 1 de AWS CLI. Le recomendamos que utilice la última versión 2 de AWS CLI. Para obtener más información, consulte Instalar la AWS CLI en la Guía del usuario de AWS Command Line Interface.

Error: opciones desconocidas: - no-include-email

Este error puede producirse al ejecutar el `aws ecr get-login` comando. Asegúrese de que tiene la última versión de la AWS CLI instalada (por ejemplo, ejecute: `pip install awscli --upgrade --user`). Para obtener más información, consulte Instalación de la AWS Command Line Interface en Microsoft Windows en la Guía del usuario de AWS Command Line Interface.

Error: A firewall is blocking file Sharing between windows and the containers.

Es posible que reciba este error o un `Firewall Detected` mensaje al ejecutar Docker en un ordenador con Windows. Esto también puede ocurrir si ha iniciado sesión en una red privada virtual (VPN) y su configuración de red impide el montaje de la unidad compartida. En esta situación, desactive la VPN y vuelva a ejecutar el contenedor Docker.

<user-name>Error: se produjo un error (AccessDeniedException) al llamar a la GetAuthorizationToken operación: el usuario: arn:aws:iam:: ***account-id:user/*** no está autorizado a realizar: ecr: on resource: * GetAuthorizationToken

Puede recibir este error al ejecutar el comando `aws ecr get-login-password` si no tiene los permisos suficientes para acceder a un repositorio de Amazon ECR. Para obtener más información, consulte los Ejemplos de políticas de repositorios de Amazon ECR y el Acceso a un repositorio de Amazon ECR en la Guía del usuario de Amazon ECR.

Error: has alcanzado tu límite de tasa de atracción

Docker Hub limita el número de solicitudes de cambios que pueden realizar los usuarios anónimos y gratuitos de Docker Hub. Si superas los límites de frecuencia de solicitudes de cambios de usuario gratuitas o anónimas, recibirás uno de los siguientes errores:

```
ERROR: toomanyrequests: Too Many Requests.
```

```
You have reached your pull rate limit.
```

Para resolver estos errores, puedes esperar unas horas antes de intentar realizar otra solicitud de incorporación de usuarios. Si planeas enviar una gran cantidad de solicitudes de cambios de forma constante, visita el sitio web de Docker Hub para obtener información sobre los límites de frecuencia y las opciones para autenticar y actualizar tu cuenta de Docker.

Depuración de AWS IoT Greengrass en un contenedor Docker

Para depurar problemas con un contenedor de Docker, puede conservar los registros del tiempo de ejecución de Greengrass o asociar un intérprete de comandos interactivo al contenedor de Docker.

Conserva los registros de Greengrass fuera del contenedor de Docker

Tras detener un AWS IoT Greengrass contenedor, puede usar el siguiente `docker cp` comando para copiar los registros de Greengrass del contenedor de Docker a un directorio de registros temporal.

```
docker cp container-id:/greengrass/v2/logs /tmp/logs
```

Para conservar los registros incluso después de que un contenedor salga o se elimine, debe ejecutar el contenedor AWS IoT Greengrass Docker después de montar el directorio en un enlace. */greengrass/v2*/logs

Para montar el */greengrass/v2*/logs directorio en un enlace, realice una de las siguientes acciones cuando ejecute un contenedor Docker nuevo. AWS IoT Greengrass

- Inclúyalo `-v` */tmp/logs*:*/greengrass/v2*/logs:ro en su comando. `docker run`

  Modifique el `volumes` bloque en el archivo de composición para incluir la siguiente línea antes de ejecutar el `docker-compose up` comando.

  ```
  volumes:
   - /tmp/logs:/greengrass/v2/logs:ro
  ```

A continuación, puede comprobar sus registros */tmp/logs* en su host para ver los registros de Greengrass mientras AWS IoT Greengrass se ejecuta dentro del contenedor de Docker.

Para obtener información sobre cómo ejecutar contenedores Docker de Greengrass, consulte y Se ejecuta AWS IoT Greengrass en Docker con aprovisionamiento manual Se ejecuta AWS IoT Greengrass en Docker con aprovisionamiento automático

Adjunte un shell interactivo al contenedor de Docker

Cuando se ejecutan comandos dentro del contenedor de Docker, esos comandos no se capturan en los registros de Docker. `docker exec` Registrar los comandos en los registros de Docker puede

ayudarle a investigar el estado del contenedor Docker de Greengrass. Realice una de las acciones siguientes:

- Ejecute el siguiente comando en un terminal independiente para adjuntar la entrada, la salida y el error estándar del terminal al contenedor en ejecución. Esto te permite ver y controlar el contenedor de Docker desde tu terminal actual.

  ```
  docker attach container-id
  ```

- Ejecute el siguiente comando en una terminal independiente. Esto le permite ejecutar sus comandos en modo interactivo, incluso si el contenedor no está adjunto.

  ```
  docker exec -it container-id sh -c "command > /proc/1/fd/1"
  ```

Para obtener información general AWS IoT Greengrass sobre la solución de problemas, consulteSolución de problemas.

Configurar el software AWS IoT Greengrass principal

El software AWS IoT Greengrass Core ofrece opciones que puede utilizar para configurar el software. Puede crear implementaciones para configurar el software AWS IoT Greengrass principal en cada dispositivo principal.

Temas

- Implemente el componente núcleo de Greengrass
- Configurar el núcleo de Greengrass como un servicio del sistema
- Controle la asignación de memoria con las opciones de JVM
- Configure el usuario que ejecuta los componentes
- Configure los límites de recursos del sistema para los componentes
- Realizar la conexión en el puerto 443 o a través de un proxy de red
- Utilice un certificado de dispositivo firmado por una entidad emisora de certificados privada
- Configure los tiempos de espera y los ajustes de caché de MQTT

Implemente el componente núcleo de Greengrass

AWS IoT Greengrass proporciona el software AWS IoT Greengrass Core como un componente que puede implementar en sus dispositivos principales de Greengrass. Puede crear una implementación para aplicar la misma configuración a varios dispositivos principales de Greengrass. Para obtener más información, consulte Núcleo de Greengrass y Actualice el software AWS IoT Greengrass principal (OTA).

Configurar el núcleo de Greengrass como un servicio del sistema

Debe configurar el software AWS IoT Greengrass Core como un servicio del sistema en el sistema de inicio del dispositivo para hacer lo siguiente:

- Inicie el software AWS IoT Greengrass Core cuando se inicie el dispositivo. Esta es una buena práctica si gestionas grandes flotas de dispositivos.
- Instala y ejecuta los componentes del complemento. Varios AWS de los componentes proporcionados son componentes de complementos, lo que les permite interactuar directamente con el núcleo de Greengrass. Para obtener más información acerca de los tipos de componentes, consulte Tipos de componentes.
- Aplica las actualizaciones over-the-air (OTA) al software principal del dispositivo AWS IoT Greengrass principal. Para obtener más información, consulte Actualice el software AWS IoT Greengrass principal (OTA).
- Permita que los componentes reinicien el software AWS IoT Greengrass principal o el dispositivo principal cuando una implementación actualice el componente a una nueva versión o actualice ciertos parámetros de configuración. Para obtener más información, consulte el paso del ciclo de vida del arranque.

> **Important**
>
> En los dispositivos principales de Windows, debe configurar el software AWS IoT Greengrass Core como un servicio del sistema.

Temas

- Configure el núcleo como un servicio del sistema (Linux)
- Configure el núcleo como un servicio del sistema (Windows)

Configure el núcleo como un servicio del sistema (Linux)

Los dispositivos Linux admiten diferentes sistemas de inicio, como initd, systemd y SystemV. El `--setup-system-service true` argumento se utiliza al instalar el software AWS IoT Greengrass Core para iniciar el núcleo como un servicio del sistema y configurarlo para que se inicie al arrancar el dispositivo. El instalador configura el software AWS IoT Greengrass Core como un servicio del sistema con systemd.

También puede configurar manualmente el núcleo para que se ejecute como un servicio del sistema. En el siguiente ejemplo se muestra un archivo de servicio para 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
```

Tras configurar el servicio del sistema, puede ejecutar los siguientes comandos para configurar el inicio del dispositivo al arrancar y para iniciar o detener el software AWS IoT Greengrass principal.

- Para comprobar el estado del servicio (systemd)

  ```
  sudo systemctl status greengrass.service
  ```

- Para permitir que el núcleo se inicie al arrancar el dispositivo.

  ```
  sudo systemctl enable greengrass.service
  ```

- Para impedir que el núcleo se inicie al arrancar el dispositivo.

  ```
  sudo systemctl disable greengrass.service
  ```

- Para iniciar el software AWS IoT Greengrass Core.

```
sudo systemctl start greengrass.service
```

- Para detener el software AWS IoT Greengrass Core.

```
sudo systemctl stop greengrass.service
```

Configure el núcleo como un servicio del sistema (Windows)

El `--setup-system-service true` argumento se utiliza al instalar el software AWS IoT Greengrass Core para iniciar el núcleo como un servicio de Windows y configurarlo para que se inicie cuando se inicie el dispositivo.

Tras configurar el servicio, puede ejecutar los siguientes comandos para configurar el inicio del dispositivo al arrancar y para iniciar o detener el software AWS IoT Greengrass Core. Debe ejecutar la línea de comandos o PowerShell como administrador para ejecutar estos comandos.

Windows Command Prompt (CMD)

- Para comprobar el estado del servicio

```
sc query "greengrass"
```

- Para permitir que el núcleo se inicie al arrancar el dispositivo.

```
sc config "greengrass" start=auto
```

- Para impedir que el núcleo se inicie al arrancar el dispositivo.

```
sc config "greengrass" start=disabled
```

- Para iniciar el software AWS IoT Greengrass Core.

```
sc start "greengrass"
```

- Para detener el software AWS IoT Greengrass Core.

```
sc stop "greengrass"
```

> **Note**
>
> En los dispositivos Windows, el software AWS IoT Greengrass Core ignora esta señal de apagado mientras cierra los procesos de los componentes de Greengrass. Si el software AWS IoT Greengrass Core ignora la señal de apagado al ejecutar este comando, espere unos segundos e inténtelo de nuevo.

PowerShell

- Para comprobar el estado del servicio

```
Get-Service -Name "greengrass"
```

- Para permitir que el núcleo se inicie al arrancar el dispositivo.

```
Set-Service -Name "greengrass" -Status stopped -StartupType automatic
```

- Para impedir que el núcleo se inicie al arrancar el dispositivo.

```
Set-Service -Name "greengrass" -Status stopped -StartupType disabled
```

- Para iniciar el software AWS IoT Greengrass Core.

```
Start-Service -Name "greengrass"
```

- Para detener el software AWS IoT Greengrass Core.

```
Stop-Service -Name "greengrass"
```

> **Note**
>
> En los dispositivos Windows, el software AWS IoT Greengrass Core ignora esta señal de apagado mientras cierra los procesos de los componentes de Greengrass. Si el software AWS IoT Greengrass Core ignora la señal de apagado al ejecutar este comando, espere unos segundos e inténtelo de nuevo.

Controle la asignación de memoria con las opciones de JVM

Si utilizas un dispositivo con memoria limitada, puedes usar las opciones de la máquina virtual Java (JVM) para controlar el tamaño máximo del montón, los modos de recolección de basura y las opciones del compilador, que controlan la cantidad de memoria que AWS IoT Greengrass utiliza el software Core. AWS IoT Greengrass El tamaño del montón de la JVM determina la cantidad de memoria que puede utilizar una aplicación antes de que se produzca la recolección de elementos no utilizados o antes de que la aplicación se quede sin memoria. El tamaño de montón máximo especifica la cantidad máxima de memoria que la JVM puede asignar al ampliar el montón durante un periodo de actividad intensa.

Para controlar la asignación de memoria, cree una nueva implementación o revise una implementación existente que incluya el componente núcleo y especifique las opciones de JVM en el parámetro de configuración de la `jvmOptions` configuración del componente núcleo.

Según sus requisitos, puede ejecutar el software AWS IoT Greengrass Core con una asignación de memoria reducida o con una asignación de memoria mínima.

Asignación de memoria reducida

Para ejecutar el software AWS IoT Greengrass Core con una asignación de memoria reducida, le recomendamos que utilice el siguiente ejemplo de actualización de combinación de configuraciones para configurar las opciones de JVM en su configuración de núcleo:

```
{
  "jvmOptions": "-Xmx64m -XX:+UseSerialGC -XX:TieredStopAtLevel=1"
}
```

Asignación mínima de memoria

Para ejecutar el software AWS IoT Greengrass Core con una asignación de memoria mínima, le recomendamos que utilice el siguiente ejemplo de actualización de combinación de configuraciones para configurar las opciones de JVM en su configuración de núcleo:

```
{
  "jvmOptions": "-Xmx32m -XX:+UseSerialGC -Xint"
}
```

En estos ejemplos de actualizaciones de combinación de configuraciones se utilizan las siguientes opciones de JVM:

`-Xmx`*NN*`m`

Establece el tamaño máximo del montón de JVM.

Para reducir la asignación de memoria, `-Xmx64m` utilícelo como valor inicial para limitar el tamaño del montón a 64 MB. Para una asignación de memoria mínima, `-Xmx32m` utilícelo como valor inicial para limitar el tamaño del montón a 32 MB.

Puede aumentar o disminuir el `-Xmx` valor en función de sus necesidades reales; sin embargo, le recomendamos encarecidamente que no establezca el tamaño máximo del montón por debajo de 16 MB. Si el tamaño máximo de pila es demasiado bajo para su entorno, es posible que el software AWS IoT Greengrass Core detecte errores inesperados debido a la falta de memoria.

`-XX:+UseSerialGC`

Especifica el uso de la recolección de basura en serie para el espacio de pila de la JVM. El recolector de basura en serie es más lento, pero utiliza menos memoria que otras implementaciones de recolección de basura de JVM.

`-XX:TieredStopAtLevel=1`

Indica a la JVM que utilice el compilador de Java just-in-time (JIT) una vez. Como el código compilado JIT ocupa espacio en la memoria del dispositivo, usar el compilador JIT más de una vez consume más memoria que una sola compilación.

`-Xint`

Indica a la JVM que no utilice el compilador just-in-time (JIT). En su lugar, la JVM se ejecuta en modo de solo interpretación. Este modo es más lento que ejecutar código compilado JIT; sin embargo, el código compilado no ocupa espacio en la memoria.

Para obtener información sobre la creación de actualizaciones de combinaciones de configuraciones, consulteActualizar las configuraciones de los componentes.

Configure el usuario que ejecuta los componentes

El software AWS IoT Greengrass Core puede ejecutar procesos de componentes como un usuario y un grupo del sistema distintos del que ejecuta el software. Esto aumenta la seguridad, ya que puede ejecutar el software AWS IoT Greengrass principal como usuario root o como usuario administrador, sin conceder esos permisos a los componentes que se ejecutan en el dispositivo principal.

En la siguiente tabla se indican los tipos de componentes que el software AWS IoT Greengrass principal puede ejecutar como usuario que especifique. Para obtener más información, consulte Tipos de componentes.

Tipo de componente	Configure el usuario del componente
Nucleus	 No
Complemento	 No
Genérico	 Sí
Lambda (no contenerizada)	 Sí
Lambda (en contenedor)	 Sí

Debe crear el usuario del componente antes de poder especificarlo en una configuración de despliegue. En los dispositivos basados en Windows, también debe almacenar el nombre de usuario y la contraseña del usuario en la instancia del administrador de credenciales de la LocalSystem cuenta. Para obtener más información, consulte Configure un usuario de componentes en dispositivos Windows.

Al configurar el usuario del componente en un dispositivo basado en Linux, también puede especificar un grupo si lo desea. Especifique el usuario y el grupo separados por dos puntos (`:`) en el siguiente formato:. *`user`*`:`*`group`* Si no especifica ningún grupo, el software AWS IoT Greengrass Core utilizará de forma predeterminada el grupo principal del usuario. Puede usar el nombre o el ID para identificar al usuario y al grupo.

En los dispositivos basados en Linux, también puede ejecutar componentes del sistema como un usuario del sistema que no existe (también denominado usuario desconocido) para aumentar la seguridad. Un proceso de Linux puede indicar cualquier otro proceso que ejecute el mismo usuario. Un usuario desconocido no ejecuta otros procesos, por lo que puede ejecutar componentes como un usuario desconocido para evitar que los componentes señalen otros componentes del dispositivo principal. Para ejecutar los componentes como un usuario desconocido, especifique un ID de usuario que no exista en el dispositivo principal. También puede especificar un ID de grupo que no existe para que se ejecute como un grupo desconocido.

Puede configurar el usuario para cada componente y para cada dispositivo principal.

- Configure para un componente

  Puede configurar cada componente para que se ejecute con un usuario específico de ese componente. Al crear una implementación, puede especificar el usuario de cada componente en la `runWith` configuración de ese componente. El software AWS IoT Greengrass principal ejecuta los componentes como el usuario especificado si los configura. De lo contrario, ejecuta los componentes de forma predeterminada como el usuario predeterminado que se configura para el dispositivo principal. Para obtener más información sobre cómo especificar el usuario del componente en la configuración de despliegue, consulte el parámetro `runWith`de configuración enCrear implementaciones.

- Configure el usuario predeterminado para un dispositivo principal

  Puede configurar un usuario predeterminado que el software AWS IoT Greengrass principal utilice para ejecutar los componentes. Cuando el software AWS IoT Greengrass principal ejecuta un componente, comprueba si ha especificado un usuario para ese componente y lo utiliza para ejecutar el componente. Si el componente no especifica un usuario, el software AWS IoT Greengrass principal ejecuta el componente como el usuario predeterminado que configuró para el dispositivo principal. Para obtener más información, consulte Configure el usuario del componente predeterminado.

> **Note**
>
> En los dispositivos basados en Windows, debe especificar al menos un usuario predeterminado para ejecutar los componentes.
>
> En los dispositivos basados en Linux, se deben tener en cuenta las siguientes consideraciones si no se configura un usuario para que ejecute componentes:
>
> - Si ejecuta el software AWS IoT Greengrass principal como root, el software no ejecutará los componentes. Debe especificar un usuario predeterminado para ejecutar los componentes si ejecuta los componentes como root.
> - Si ejecuta el software AWS IoT Greengrass principal como usuario no root, el software ejecuta los componentes como ese usuario.

Temas

Configure un usuario de componentes en dispositivos Windows

Para configurar un usuario de componentes en un dispositivo basado en Windows

1. Cree el usuario del componente en la LocalSystem cuenta del dispositivo.

   ```
   net user /add component-user password
   ```

2. Utilice la PsExec utilidad de Microsoft para almacenar el nombre de usuario y la contraseña del usuario del componente en la instancia de Credential Manager de la LocalSystem cuenta.

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

   > **Note**
   >
   > En los dispositivos basados en Windows, la LocalSystem cuenta ejecuta el núcleo de Greengrass y debe utilizar PsExec la utilidad para almacenar la información del usuario del componente en LocalSystem la cuenta. El uso de la aplicación Credential Manager

almacena esta información en la cuenta de Windows del usuario que ha iniciado sesión actualmente, en lugar de en la cuenta. LocalSystem

Configure el usuario del componente predeterminado

Puede usar una implementación para configurar el usuario predeterminado en un dispositivo principal. En esta implementación, se actualiza la configuración de los componentes del núcleo.

> **Note**
>
> También puede configurar el usuario predeterminado al instalar el software AWS IoT Greengrass Core con la `--component-default-user` opción. Para obtener más información, consulte Instalación del software AWS IoT Greengrass Core.

Cree una implementación que especifique la siguiente actualización de configuración para el `aws.greengrass.Nucleus` componente.

Linux

```
{
  "runWithDefault": {
    "posixUser": "ggc_user:ggc_group"
  }
}
```

Windows

```
{
  "runWithDefault": {
    "windowsUser": "ggc_user"
  }
}
```

> **Note**
>
> El usuario que especifique debe existir y el nombre de usuario y la contraseña de este usuario deben estar almacenados en la instancia del administrador de credenciales de

la LocalSystem cuenta del dispositivo Windows. Para obtener más información, consulte Configure un usuario de componentes en dispositivos Windows.

El siguiente ejemplo define una implementación para un dispositivo basado en Linux que se configura `ggc_user` como usuario y `ggc_group` grupo predeterminados. La actualización de `merge` configuración requiere un objeto JSON serializado.

```
{
  "components": {
    "aws.greengrass.Nucleus": {
      "version": "2.12.6",
      "configurationUpdate": {
        "merge": "{\"runWithDefault\":{\"posixUser\":\"ggc_user:ggc_group\"}}"
      }
    }
  }
}
```

Configure los límites de recursos del sistema para los componentes

> **Note**
>
> Esta función está disponible para la versión 2.4.0 y versiones posteriores del componente núcleo de Greengrass. AWS IoT Greengrass actualmente no admite esta función en los dispositivos principales de Windows.

Puede configurar la cantidad máxima de uso de CPU y RAM que los procesos de cada componente pueden usar en el dispositivo principal.

En la siguiente tabla se muestran los tipos de componentes que admiten los límites de recursos del sistema. Para obtener más información, consulte Tipos de componentes.

Tipo de componente	Configure los límites de recursos del sistema
Nucleus	 No
Complemento	 No
Genérico	 Sí
Lambda (no contenerizada)	 Sí
Lambda (en contenedor)	 No

Important

Los límites de recursos del sistema no se admiten cuando se ejecuta el software AWS IoT Greengrass principal en un contenedor de Docker.

Puede configurar los límites de recursos del sistema para cada componente y para cada dispositivo principal.

- Configure para un componente

  Puede configurar cada componente con los límites de recursos del sistema específicos para ese componente. Al crear una implementación, puede especificar los límites de recursos del sistema para cada componente de la implementación. Si el componente admite los límites de recursos del sistema, el software AWS IoT Greengrass Core aplica los límites a los procesos del componente. Si no especifica los límites de recursos del sistema para un componente, el software AWS IoT Greengrass principal utilizará los valores predeterminados que haya configurado para el dispositivo principal. Para obtener más información, consulte Crear implementaciones.

- Configure los valores predeterminados para un dispositivo principal

  Puede configurar los límites de recursos del sistema predeterminados que el software AWS IoT Greengrass Core aplica a los componentes que admiten estos límites. Cuando el software AWS IoT Greengrass principal ejecuta un componente, aplica los límites de recursos del sistema que especifique para ese componente. Si ese componente no especifica los límites de recursos del sistema, el software AWS IoT Greengrass Core aplica los límites de recursos del sistema predeterminados que usted configure para el dispositivo principal. Si no especificas los límites de recursos del sistema predeterminados, el software AWS IoT Greengrass Core no aplicará ningún límite de recursos del sistema de forma predeterminada. Para obtener más información, consulte Configure los límites de recursos del sistema predeterminados.

Configure los límites de recursos del sistema predeterminados

Puede implementar el componente núcleo de Greengrass para configurar los límites de recursos del sistema predeterminados para un dispositivo principal. Para configurar los límites de recursos del sistema predeterminados, cree una implementación que especifique la siguiente actualización de configuración para el `aws.greengrass.Nucleus` componente.

```
{
  "runWithDefault": {
    "systemResourceLimits": {
      "cpu": cpuTimeLimit,
      "memory": memoryLimitInKb
    }
  }
}
```

El siguiente ejemplo define una implementación que configura el límite de tiempo de la CPU en2, lo que equivale al 50% de uso en un dispositivo con 4 núcleos de CPU. En este ejemplo también se configura el uso de memoria en 100 MB.

```
{
  "components": {
    "aws.greengrass.Nucleus": {
      "version": "2.12.6",
      "configurationUpdate": {
        "merge": "{\"runWithDefault\":{\"systemResourceLimits\":\"cpus\":2,\"memory
\":102400}}}"
      }
    }
  }
}
```

Realizar la conexión en el puerto 443 o a través de un proxy de red

AWS IoT Greengrass los dispositivos principales se comunican AWS IoT Core mediante el protocolo de mensajería MQTT con autenticación de cliente TLS. Convencionalmente, MQTT sobre TLS utiliza el puerto 8883. Sin embargo, como medida de seguridad, los entornos restrictivos podrían limitar el tráfico de entrada y salida a un pequeño rango de puertos TCP. Por ejemplo, el firewall de una compañía podría abrir el puerto 443 para el tráfico HTTPS, pero cerrar otros puertos que se utilizan para protocolos menos frecuentes, como, por ejemplo, el puerto 8883 para tráfico MQTT. Otros entornos restrictivos pueden requerir que todo el tráfico pase por un proxy antes de conectarse a Internet.

> **Note**
>
> Los dispositivos principales de Greengrass que ejecutan el componente núcleo de Greengrass v2.0.3 y versiones anteriores utilizan el puerto 8443 para conectarse al punto final del plano de datos. AWS IoT Greengrass Estos dispositivos deben poder conectarse a este punto final en el puerto 8443. Para obtener más información, consulte Permitir el tráfico del dispositivo a través de un proxy o firewall.

Para habilitar la comunicación en estos escenarios, AWS IoT Greengrass proporciona las siguientes opciones de configuración:

- Comunicación MQTT a través del puerto 443. Si su red permite conexiones al puerto 443, puede configurar el dispositivo principal de Greengrass para que utilice el puerto 443 para el tráfico MQTT en lugar del puerto predeterminado 8883. Esto puede ser una conexión directa al puerto 443 o una conexión a través de un servidor proxy de red. A diferencia de la configuración predeterminada, que utiliza la autenticación de cliente basada en certificados, el MQTT del puerto 443 utiliza la función de servicio del dispositivo para la autenticación.

  Para obtener más información, consulte Configure MQTT a través del puerto 443.

- Comunicación HTTPS a través del puerto 443. El software AWS IoT Greengrass Core envía el tráfico HTTPS a través del puerto 8443 de forma predeterminada, pero puede configurarlo para que utilice el puerto 443. AWS IoT Greengrass utiliza la extensión TLS de Application Layer Protocol Network (ALPN) para habilitar esta conexión. Al igual que con la configuración predeterminada, HTTPS en el puerto 443 utiliza la autenticación de cliente basada en certificados.

  > **Important**
  >
  > Para usar ALPN y habilitar la comunicación HTTPS a través del puerto 443, el dispositivo principal debe ejecutar Java 8, actualización 252 o posterior. Todas las actualizaciones de la versión 9 y posteriores de Java también son compatibles con ALPN.

  Para obtener más información, consulte Configure HTTPS a través del puerto 443.

- Conexión a través de un proxy de red. Puede configurar un servidor proxy de red para que actúe como intermediario para conectarse al dispositivo principal de Greengrass. AWS IoT Greengrass admite la autenticación básica para los proxies HTTP y HTTPS.

  Los dispositivos principales de Greengrass deben ejecutar Greengrass nucleus v2.5.0 o posterior para usar proxies HTTPS.

  El software AWS IoT Greengrass Core transfiere la configuración del proxy a los componentes a través de las variables de entornoALL_PROXY,HTTP_PROXY, HTTPS_PROXY y. NO_PROXY Los componentes deben usar esta configuración para conectarse a través del proxy. Los componentes utilizan bibliotecas comunes (como boto3, cURL y el `requests` paquete python) que suelen utilizar estas variables de entorno de forma predeterminada para establecer conexiones. Si un componente también especifica estas variables de entorno, AWS IoT Greengrass no las anula.

  Para obtener más información, consulte Configure un proxy de red.

Configure MQTT a través del puerto 443

Puede configurar MQTT a través del puerto 443 en los dispositivos principales existentes o al instalar el software AWS IoT Greengrass Core en un dispositivo principal nuevo.

Temas

Configure MQTT a través del puerto 443 en los dispositivos principales existentes

Puede usar una implementación para configurar MQTT a través del puerto 443 en un dispositivo de un solo núcleo o en un grupo de dispositivos principales. En esta implementación, se actualiza la configuración de los componentes del núcleo. El núcleo se reinicia al actualizar su `mqtt` configuración.

Para configurar MQTT a través del puerto 443, cree una implementación que especifique la siguiente actualización de configuración para el `aws.greengrass.Nucleus` componente.

```
{
  "mqtt": {
    "port": 443
  }
}
```

El siguiente ejemplo define una implementación que configura MQTT a través del puerto 443. La actualización `merge` de configuración requiere un objeto JSON serializado.

```
{
  "components": {
    "aws.greengrass.Nucleus": {
      "version": "2.12.6",
      "configurationUpdate": {
        "merge": "{\"mqtt\":{\"port\":443}}"
      }
    }
  }
}
```

Configure MQTT a través del puerto 443 durante la instalación

Puede configurar MQTT a través del puerto 443 al instalar el software AWS IoT Greengrass Core en un dispositivo principal. Utilice el argumento del `--init-config` instalador para configurar MQTT a través del puerto 443. Puede especificar este argumento al realizar la instalación con aprovisionamiento manual, aprovisionamiento de flota o aprovisionamiento personalizado.

Configure HTTPS a través del puerto 443

Esta función requiere la Núcleo de Greengrass versión 2.0.4 o posterior.

Puede configurar HTTPS a través del puerto 443 en los dispositivos principales existentes o al instalar el software AWS IoT Greengrass Core en un dispositivo principal nuevo.

Temas

- Configure HTTPS a través del puerto 443 en los dispositivos principales existentes
- Configure HTTPS a través del puerto 443 durante la instalación

Configure HTTPS a través del puerto 443 en los dispositivos principales existentes

Puede usar una implementación para configurar HTTPS a través del puerto 443 en un dispositivo de un solo núcleo o en un grupo de dispositivos principales. En esta implementación, se actualiza la configuración de los componentes del núcleo.

Para configurar HTTPS a través del puerto 443, cree una implementación que especifique la siguiente actualización de configuración para el `aws.greengrass.Nucleus` componente.

```
{
  "greengrassDataPlanePort": 443
}
```

El siguiente ejemplo define una implementación que configura HTTPS a través del puerto 443. La actualización `merge` de configuración requiere un objeto JSON serializado.

```
{
  "components": {
    "aws.greengrass.Nucleus": {
      "version": "2.12.6",
      "configurationUpdate": {
        "merge": "{\"greengrassDataPlanePort\":443}"
```

```
      }
    }
  }
}
```

Configure HTTPS a través del puerto 443 durante la instalación

Puede configurar HTTPS a través del puerto 443 al instalar el software AWS IoT Greengrass Core en un dispositivo principal. Utilice el argumento del `--init-config` instalador para configurar HTTPS a través del puerto 443. Puede especificar este argumento al realizar la instalación con aprovisionamiento manual, aprovisionamiento de flota o aprovisionamiento personalizado.

Configure un proxy de red

Siga el procedimiento de esta sección para configurar los dispositivos principales de Greengrass para que se conecten a Internet a través de un proxy de red HTTP o HTTPS. Para obtener más información sobre los puntos finales y los puertos que utilizan los dispositivos principales, consulte. Permitir el tráfico del dispositivo a través de un proxy o firewall

Important

Si su dispositivo principal ejecuta una versión del núcleo de Greengrass anterior a la v2.4.0, la función de su dispositivo debe permitir los siguientes permisos para usar un proxy de red:

- `iot:Connect`
- `iot:Publish`
- `iot:Receive`
- `iot:Subscribe`

Esto es necesario porque el dispositivo utiliza las AWS credenciales del servicio de intercambio de fichas para autenticar las conexiones MQTT. AWS IoT El dispositivo utiliza MQTT para recibir e instalar las implementaciones desde allí Nube de AWS, por lo que el dispositivo no funcionará a menos que defina estos permisos en función de su función. Los dispositivos suelen utilizar certificados X.509 para autenticar las conexiones MQTT, pero los dispositivos no pueden hacerlo para autenticarse cuando utilizan un proxy.
Para obtener más información sobre cómo configurar la función del dispositivo, consulte. Autorizar a los dispositivos principales a interactuar con AWS los servicios

Temas

Configure un proxy de red en los dispositivos principales existentes

Puede usar una implementación para configurar un proxy de red en un dispositivo de un solo núcleo o en un grupo de dispositivos principales. En esta implementación, se actualiza la configuración de los componentes del núcleo. El núcleo se reinicia al actualizar su `networkProxy` configuración.

Para configurar un proxy de red, cree una implementación para el `aws.greengrass.Nucleus` componente que combine la siguiente actualización de configuración. Esta actualización de configuración contiene el objeto NetworkProxy.

```
{
  "networkProxy": {
    "noProxyAddresses": "http://192.168.0.1,www.example.com",
    "proxy": {
      "url": "https://my-proxy-server:1100"
    }
  }
}
```

El siguiente ejemplo define una implementación que configura un proxy de red. La actualización `merge` de configuración requiere un objeto JSON serializado.

```
{
  "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\"}}}"
      }
    }
  }
```

```
}
```

Configure un proxy de red durante la instalación

Puede configurar un proxy de red al instalar el software AWS IoT Greengrass Core en un dispositivo principal. Utilice el argumento del `--init-config` instalador para configurar el proxy de red. Puede especificar este argumento al realizar la instalación con aprovisionamiento manual, aprovisionamiento de flota o aprovisionamiento personalizado.

Habilite el dispositivo principal para que confíe en un proxy HTTPS

Al configurar un dispositivo principal para que utilice un proxy HTTPS, debe añadir la cadena de certificados del servidor proxy al dispositivo principal para que pueda confiar en el proxy HTTPS. De lo contrario, el dispositivo principal podría encontrar errores al intentar enrutar el tráfico a través del proxy. Añada el certificado de CA del servidor proxy al archivo de certificado de CA raíz de Amazon del dispositivo principal.

Para permitir que el dispositivo principal confíe en el proxy HTTPS

1. Busque el archivo de certificado de CA raíz de Amazon en el dispositivo principal.

   - Si ha instalado el software AWS IoT Greengrass Core con aprovisionamiento automático, el archivo de certificado de CA raíz de Amazon se encuentra en*/greengrass/v2*/`rootCA.pem`.
   - Si instaló el software AWS IoT Greengrass Core con aprovisionamiento manual o de flota, es posible que el archivo de certificado de CA raíz de Amazon esté en*/greengrass/v2*/`AmazonRootCA1.pem`.

   Si el certificado de CA raíz de Amazon no existe en estas ubicaciones, registra la `system.rootCaPath` propiedad */greengrass/v2*`/config/effectiveConfig.yaml` para encontrar su ubicación.
2. Añada el contenido del archivo de certificado de CA del servidor proxy al archivo de certificado de CA raíz de Amazon.

   El siguiente ejemplo muestra un certificado de CA de un servidor proxy agregado al archivo de certificado de CA raíz de 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-----
```

El objeto NetworkProxy

Utilice el objeto `networkProxy` para especificar información sobre el proxy de red. Este objeto contiene la siguiente información:

`noProxyAddresses`

(Opcional) Una lista separada por comas de direcciones IP o nombres de host que están exentos del proxy.

`proxy`

El proxy al que se va a conectar. Este objeto contiene la siguiente información:

`url`

La URL del servidor proxy en el formato`scheme://userinfo@host:port`.

- `scheme`— El esquema, que debe ser `http` o`https`.

> **Important**
>
> Los dispositivos principales de Greengrass deben ejecutar Greengrass nucleus v2.5.0 o posterior para usar proxies HTTPS.
> Si configura un proxy HTTPS, debe añadir el certificado de CA del servidor proxy al certificado de CA raíz de Amazon del dispositivo principal. Para obtener más

información, consulte Habilite el dispositivo principal para que confíe en un proxy HTTPS.

- `userinfo`— (Opcional) La información del nombre de usuario y la contraseña. Si especifica esta información en`url`, el dispositivo principal de Greengrass ignora los `username` campos y. `password`
- `host`— El nombre de host o la dirección IP del servidor proxy.
- `port`— (Opcional) El número de puerto. Si no especificas el puerto, el dispositivo principal de Greengrass utilizará los siguientes valores predeterminados:
  - `http`— 80
  - `https`— 443

`username`

(Opcional) El nombre de usuario que autentica el servidor proxy.

`password`

(Opcional) La contraseña que autentica el servidor proxy.

Utilice un certificado de dispositivo firmado por una entidad emisora de certificados privada

Si utiliza una autoridad de certificación (CA) privada personalizada, debe establecer el 'núcleo**`greengrassDataPlaneEndpoint`**' de Greengrass en. **`iotdata`** Puede configurar esta opción durante el despliegue o la instalación mediante el argumento del **`--init-config`** instalador.

Puede personalizar el punto final del plano de datos de Greengrass al que se conecta el dispositivo. Puede configurar esta opción de configuración **`iotdata`** para establecer el punto final del plano de datos de Greengrass en el mismo punto final que el punto final de datos de IoT, que puede especificar con. **`iotDataEndpoint`**

Configure los tiempos de espera y los ajustes de caché de MQTT

En el AWS IoT Greengrass entorno, los componentes pueden utilizar MQTT para comunicarse con ellos. AWS IoT Core El software AWS IoT Greengrass Core gestiona los mensajes MQTT de los componentes. Cuando el dispositivo principal pierde la conexión con el Nube de AWS, el software almacena en caché los mensajes MQTT para volver a intentarlo más tarde cuando se restablezca la conexión. Puede configurar ajustes como los tiempos de espera de los mensajes y el tamaño de la

memoria caché. Para obtener más información, consulte `mqtt` los parámetros de `mqtt.spooler` configuración del componente núcleo de Greengrass.

AWS IoT Core impone cuotas de servicio a su agente de mensajes MQTT. Estas cuotas pueden aplicarse a los mensajes que envíe entre los dispositivos principales y. AWS IoT Core Para obtener más información, consulta las cuotas de servicio de AWS IoT Core Message Broker en Referencia general de AWS.

Actualice el software AWS IoT Greengrass principal (OTA)

El software AWS IoT Greengrass principal comprende el componente núcleo de Greengrass y otros componentes opcionales que puede implementar en sus dispositivos para realizar actualizaciones over-the-air (OTA) del software. Esta función está integrada en el software AWS IoT Greengrass Core.

Las actualizaciones OTA hacen que sea más eficiente:

- Corrige vulnerabilidades de seguridad.
- Solucionar problemas de estabilidad del software.
- Implementar características nuevas o mejoradas.

Temas

Requisitos

Para implementar las actualizaciones OTA del software AWS IoT Greengrass Core se aplican los siguientes requisitos:

- El dispositivo principal de Greengrass debe tener una conexión al Nube de AWS para recibir el despliegue.
- El dispositivo principal de Greengrass debe estar correctamente configurado y aprovisionado con certificados y claves para la autenticación con y. AWS IoT Core AWS IoT Greengrass

- El software AWS IoT Greengrass principal debe configurarse y ejecutarse como un servicio del sistema. Las actualizaciones OTA no funcionan si se ejecuta el núcleo desde el archivo JAR,`Greengrass.jar`. Para obtener más información, consulte Configurar el núcleo de Greengrass como un servicio del sistema.

Consideraciones sobre los dispositivos principales

Antes de realizar una actualización OTA, tenga en cuenta el impacto en los dispositivos principales que actualice y en los dispositivos cliente conectados:

- El núcleo de Greengrass se apaga.
- Todos los componentes que se ejecutan en el dispositivo principal también se apagan. Si esos componentes escriben en los recursos locales, es posible que dejen esos recursos en un estado incorrecto a menos que se apaguen correctamente. Los componentes pueden utilizar la comunicación entre procesos para indicar al componente del núcleo que aplace la actualización hasta que agote los recursos que utilizan.
- Mientras el componente del núcleo está apagado, el dispositivo principal pierde sus conexiones con los dispositivos locales Nube de AWS y con los dispositivos locales. El dispositivo principal no enrutará los mensajes de los dispositivos cliente mientras esté apagado.
- Las funciones Lambda de larga duración que se ejecutan como componentes pierden su información de estado dinámico y eliminan todo el trabajo pendiente.

Comportamiento de actualización del núcleo de Greengrass

Al implementar un componente, AWS IoT Greengrass instala las últimas versiones compatibles de todas las dependencias de ese componente. Por este motivo, es posible que las nuevas versiones con parches de los componentes públicos AWS proporcionados se implementen automáticamente en sus dispositivos principales si agrega nuevos dispositivos a un grupo de cosas o si actualiza la implementación destinada a esos dispositivos. Algunas actualizaciones automáticas, como las actualizaciones de Nucleus, pueden provocar que los dispositivos se reinicien de forma inesperada.

Cuando la versión del componente núcleo de Greengrass cambia, el software AWS IoT Greengrass Core, que incluye el núcleo y todos los demás componentes del dispositivo, se reinicia para aplicar los cambios. Debido a que la actualización del componente núcleo afecta a los dispositivos principales, es posible que desee controlar cuándo se implementará una nueva versión del parche de Nucleus en sus dispositivos. Para ello, debe incluir directamente el componente núcleo de

Greengrass en su despliegue. Incluir directamente un componente significa incluir una versión específica de ese componente en la configuración de despliegue y no depender de las dependencias de los componentes para implementar ese componente en sus dispositivos. Para obtener más información sobre cómo definir las dependencias en las recetas de sus componentes, consulte. Formato de receta

Consulte la siguiente tabla para comprender el comportamiento de actualización del componente núcleo de Greengrass en función de sus acciones y configuraciones de despliegue.

Acción de	Configuración de implement ación	Comportamiento de las actualizaciones de
Agregue nuevos dispositi vos a un grupo de cosas al que apunta una implement ación existente sin revisar la implementación.	El despliegue no incluye directamente el núcleo de Greengrass. La implementación incluye directamente al menos un componente AWS proporcio nado o incluye un component e personalizado que depende de un componente AWS proporcionado o del núcleo de Greengrass.	En los dispositivos nuevos, instala la última versión de parche de nucleus que cumple con todos los requisito s de dependencia de los componentes. En los dispositivos existente s, no actualiza la versión instalada del núcleo.
Añada nuevos dispositi vos a un grupo de cosas al que apunta una implement ación existente sin revisar la implementación.	El despliegue incluye directamente una versión específica del núcleo de Greengrass.	En los dispositivos nuevos, instala la versión del núcleo especificada. En los dispositivos existente s, no actualiza la versión instalada del núcleo.
Cree una nueva implement ación o revise una implement ación existente.	El despliegue no incluye directamente el núcleo de Greengrass.	En todos los dispositivos de destino, instala la última versión del parche del núcleo que cumpla con todos los

Acción de	Configuración de implement ación	Comportamiento de las actualizaciones de
	La implementación incluye directamente al menos un componente AWS proporcio nado o incluye un component e personalizado que depende de un componente AWS proporcionado o del núcleo de Greengrass.	requisitos de dependencia de los componentes, incluso en los dispositivos nuevos que añada al grupo de component es de destino.
Cree una nueva implement ación o revise una implement ación existente.	El despliegue incluye directamente una versión específica del núcleo de Greengrass.	En todos los dispositivos de destino, instala la versión de núcleo especificada, incluidos los dispositivos nuevos que añada al grupo de dispositivos de destino.

Realiza una actualización OTA

Para realizar una actualización OTA, cree una implementación que incluya el componente Nucleus y la versión que desee instalar.

Desinstale el software AWS IoT Greengrass principal

Puede desinstalar el software AWS IoT Greengrass principal para eliminarlo de un dispositivo que no desee utilizar como dispositivo principal de Greengrass. También puede seguir estos pasos para limpiar una instalación que no funcione.

Para desinstalar el software AWS IoT Greengrass principal

1. Si ejecuta el software como un servicio del sistema, debe detener, deshabilitar y eliminar el servicio. Ejecute los siguientes comandos según corresponda a su sistema operativo.

Linux

1. Detenga el servicio .

```
sudo systemctl stop greengrass.service
```

2. Deshabilite el servicio.

```
sudo systemctl disable greengrass.service
```

3. Elimine el servicio.

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

4. Compruebe que se ha eliminado el servicio.

```
sudo systemctl daemon-reload && sudo systemctl reset-failed
```

Windows (Command Prompt)

> **Note**
>
> Debe ejecutar Command Prompt como administrador para ejecutar estos comandos.

1. Detenga el servicio .

```
sc stop "greengrass"
```

2. Deshabilite el servicio.

```
sc config "greengrass" start=disabled
```

3. Elimine el servicio.

```
sc delete "greengrass"
```

4. Reinicie el dispositivo.

Windows (PowerShell)

> **Note**
>
> Debe correr PowerShell como administrador para ejecutar estos comandos.

1. Detenga el servicio .

```
Stop-Service -Name "greengrass"
```

2. Deshabilite el servicio.

```
Set-Service -Name "greengrass" -Status stopped -StartupType disabled
```

3. Elimine el servicio.

   - Para PowerShell 6.0 y versiones posteriores:

```
Remove-Service -Name "greengrass" -Confirm:$false -Verbose
```

   - Para PowerShell versiones anteriores a la 6.0:

```
Get-Item HKLM:\SYSTEM\CurrentControlSet\Services\greengrass | Remove-Item
 -Force -Verbose
```

4. Reinicie el dispositivo.

2. Elimine la carpeta raíz del dispositivo. Sustituya */greengrass/v2* o *C:\greengrass\v2* por la ruta de acceso a la carpeta raíz.

   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. Elimine el dispositivo principal del AWS IoT Greengrass servicio. Este paso elimina la información de estado del dispositivo principal delNube de AWS. Asegúrese de completar este paso si planea volver a instalar el software AWS IoT Greengrass Core en un dispositivo principal con el mismo nombre.
   - Para eliminar un dispositivo principal de la AWS IoT Greengrass consola, haga lo siguiente:
     a. Vaya a la consola de AWS IoT Greengrass.
     b. Selecciona Dispositivos principales.
     c. Elige el dispositivo principal que deseas eliminar.
     d. Elija Eliminar.
     e. En el modal de confirmación, selecciona Eliminar.
   - Para eliminar un dispositivo principal con elAWS Command Line Interface, utilice la DeleteCoreDeviceoperación. Ejecute el siguiente comando y *MyGreengrassCore*sustitúyalo por el nombre del dispositivo principal.

     ```
     aws greengrassv2 delete-core-device --core-device-thing-name MyGreengrassCore
     ```

Tutoriales de AWS IoT Greengrass V2

Puede completar los siguientes tutoriales para obtener información sobre AWS IoT Greengrass V2 sus funciones.

Temas

Tutorial: Desarrolle un componente de Greengrass que aplace las actualizaciones de los componentes

Puede completar este tutorial para desarrollar un componente que aplace las actualizaciones de over-the-air implementación. Al implementar actualizaciones en sus dispositivos, es posible que desee retrasar las actualizaciones en función de ciertas condiciones, como las siguientes:

- El nivel de batería del dispositivo es bajo.
- El dispositivo está ejecutando un proceso o un trabajo que no se puede interrumpir.
- El dispositivo tiene una conexión a Internet limitada o cara.

> **Note**
>
> Un componente es un módulo de software que se ejecuta en los dispositivos AWS IoT Greengrass principales. Los componentes le permiten crear y administrar aplicaciones complejas como bloques de construcción discretos que puede reutilizar de un dispositivo principal de Greengrass a otro.

En este tutorial, aprenderá a hacer lo siguiente:

1. Instale la CLI del kit de desarrollo Greengrass (GDK CLI) en su ordenador de desarrollo. La CLI de GDK proporciona funciones que le ayudan a desarrollar componentes personalizados de Greengrass.
2. Desarrolle un componente Hello World que aplace las actualizaciones de los componentes cuando el nivel de batería del dispositivo principal esté por debajo de un umbral. Este componente se suscribe a las notificaciones de actualización mediante la operación SubscribeToComponentUpdatesIPC. Cuando recibe la notificación, comprueba si el nivel de la batería es inferior a un umbral personalizable. Si el nivel de la batería está por debajo del umbral, aplaza la actualización durante 30 segundos mediante la operación DeferComponentUpdateIPC. Este componente se desarrolla en el ordenador de desarrollo mediante la CLI de GDK.

   > **Note**
   >
   > Este componente lee el nivel de la batería a partir de un archivo que usted crea en el dispositivo principal para imitar una batería real, por lo que puede completar este tutorial en un dispositivo principal sin batería.

3. Publique ese componente en el AWS IoT Greengrass servicio.
4. Despliegue ese componente desde Nube de AWS el dispositivo principal de Greengrass para probarlo. A continuación, se modifica el nivel de batería virtual del dispositivo principal y se crean despliegues adicionales para comprobar cómo el dispositivo principal aplaza las actualizaciones cuando el nivel de batería es bajo.

Puedes dedicar de 20 a 30 minutos a este tutorial.

Requisitos previos

Necesitará lo siguiente para completar este tutorial:

- Una Cuenta de AWS. Si no dispone de una, consulte Configure un Cuenta de AWS.
- Un usuario AWS Identity and Access Management (IAM) con permisos de administrador.
- Un dispositivo central de Greengrass con conexión a Internet. Para obtener más información sobre cómo configurar un dispositivo principal, consulteConfiguración de los dispositivos AWS IoT Greengrass principales.
  - Python 3.6 o posterior instalado para todos los usuarios en el dispositivo principal y agregado a la variable de `PATH` entorno. En Windows, también debe tener instalado el Lanzador de Python para Windows para todos los usuarios.

> **Important**
>
> En Windows, Python no se instala para todos los usuarios de forma predeterminada. Al instalar Python, debe personalizar la instalación para configurarla para que el software AWS IoT Greengrass principal ejecute scripts de Python. Por ejemplo, si utiliza el instalador gráfico de Python, haga lo siguiente:
>
> 1. Seleccione Instalar el lanzador para todos los usuarios (recomendado).
> 2. Elija Customize installation.
> 3. Elija Next.
> 4. Seleccione Install for all users.
> 5. Seleccione Add Python to environment variables.
> 6. Elija Instalar.
>
> Para obtener más información, consulte Uso de Python en Windows en la documentación de Python 3.

- Un ordenador de desarrollo similar a Windows, macOS o Unix con conexión a Internet.
  - Python 3.6 o posterior instalado en su ordenador de desarrollo.
  - Git instalado en tu ordenador de desarrollo.
  - AWS Command Line Interface(AWS CLI) instalado y configurado con credenciales en tu ordenador de desarrollo. Para obtener más información, consulte Instalación, actualización y desinstalación AWS CLI y configuración del AWS CLI en la Guía del AWS Command Line Interface usuario.

> **Note**
>
> Si utiliza una Raspberry Pi u otro dispositivo ARM de 32 bits, instale AWS CLI la V1. AWS CLI La versión 2 no está disponible para dispositivos ARM de 32 bits. Para obtener más información, consulte Instalación, actualización y desinstalación de la AWS CLI versión 1.

Paso 1: Instalar la CLI del kit de desarrollo Greengrass

El kit de desarrollo CLI de Greengrass (GDK CLI) proporciona funciones que le ayudan a desarrollar componentes personalizados de Greengrass. Puede usar la CLI de GDK para crear, compilar y publicar componentes personalizados.

Si no ha instalado la CLI de GDK en su equipo de desarrollo, complete los siguientes pasos para instalarla.

Para instalar la versión más reciente de la CLI de GDK

1. En su equipo de desarrollo, ejecute el siguiente comando para instalar la última versión de la CLI de GDK desde su GitHubrepositorio.

   ```
   python3 -m pip install -U git+https://github.com/aws-greengrass/aws-greengrass-gdk-cli.git@v1.6.2
   ```

2. Ejecute el siguiente comando para comprobar que la CLI de GDK se instaló correctamente.

   ```
   gdk --help
   ```

   Si no encuentra el `gdk` comando, añada su carpeta a PATH.

   - En dispositivos Linux, `/home/MyUser/.local/bin` agréguelo a PATH y *MyUser*sustitúyalo por el nombre de su usuario.
   - En dispositivos Windows, añada *PythonPath*`\\Scripts` a PATH y *PythonPath*sustitúyala por la ruta a la carpeta Python de su dispositivo.

Paso 2: Desarrollar un componente que aplace las actualizaciones

En esta sección, desarrollará un componente Hello World en Python que aplaza las actualizaciones de los componentes cuando el nivel de batería del dispositivo principal esté por debajo del umbral que usted configuró al implementar el componente. En este componente, se utiliza la interfaz de comunicación entre procesos (IPC) de la SDK para dispositivos con AWS IoT versión 2 para Python. La operación SubscribeToComponentUpdatesIPC se utiliza para recibir notificaciones cuando el dispositivo principal recibe una implementación. A continuación, se utiliza la operación DeferComponentUpdateIPC para aplazar o confirmar la actualización en función del nivel de batería del dispositivo.

Desarrollar un componente de Hello World que aplace las actualizaciones

1. En su computadora de desarrollo, cree una carpeta para el código fuente del componente.

   ```
   mkdir com.example.BatteryAwareHelloWorld
   cd com.example.BatteryAwareHelloWorld
   ```

2. Utilice un editor de texto para crear un archivo con el nombre`gdk-config.json`. La CLI de GDK lee el archivo de configuración de la CLI de GDK, denominado`gdk-config.json`, para crear y publicar componentes. Este archivo de configuración existe en la raíz de la carpeta del componente.

   Por ejemplo, en un sistema basado en Linux, puede ejecutar el siguiente comando para usar GNU nano para crear el archivo.

   ```
   nano gdk-config.json
   ```

   Copia el siguiente JSON en el archivo.

   - Sustituye *`Amazon`* por tu nombre.
   - Sustituya *`us-west-2`* por el Región de AWS lugar donde funciona su dispositivo principal. La CLI de GDK publica el componente aquíRegión de AWS.
   - *`greengrass-component-artifacts`*Sustitúyalo por el prefijo de bucket S3 que desee utilizar. Cuando utiliza la CLI de GDK para publicar el componente, la CLI de GDK carga los artefactos del componente en el bucket de S3 cuyo nombre se forma a partir de este valorRegión de AWS, el y su Cuenta de AWS ID con el siguiente formato:. *`bucketPrefix`*`-`*`region`*`-`*`accountId`*

     Por ejemplo, si especificas **`greengrass-component-artifacts`** y**`us-west-2`**, y tu Cuenta de AWS ID es**`123456789012`**, la CLI de GDK usa el bucket de S3 denominado`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"
}
```

El archivo de configuración especifica lo siguiente:

- La versión que se utilizará cuando la CLI de GDK publique el componente Greengrass en el servicio en AWS IoT Greengrass la nube. `NEXT_PATCH`especifica que se debe elegir la siguiente versión del parche después de la última versión disponible en el servicio en la AWS IoT Greengrass nube. Si el componente aún no tiene una versión en el servicio en la AWS IoT Greengrass nube, la CLI de GDK la utiliza`1.0.0`.
- El sistema de compilación del componente. Cuando se utiliza el sistema de `zip` compilación, la CLI de GDK empaqueta el código fuente del componente en un archivo ZIP que se convierte en el único artefacto del componente.
- Allí Región de AWS donde la CLI de GDK publica el componente Greengrass.
- El prefijo del bucket de S3 en el que la CLI de GDK carga los artefactos del componente.

3. Utilice un editor de texto para crear el código fuente del componente en un archivo denominado. `main.py`

   Por ejemplo, en un sistema basado en Linux, puede ejecutar el siguiente comando para usar GNU nano para crear el archivo.

   ```
   nano main.py
   ```

   Copia el siguiente código de Python en el archivo.

   ```
   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()
```

Esta aplicación de Python hace lo siguiente:

- Lee el nivel de batería del dispositivo principal a partir de un archivo virtual de nivel de batería que crearás en el dispositivo principal más adelante. Este archivo virtual de nivel de batería imita una batería real, por lo que puedes completar este tutorial en los dispositivos principales que no tienen batería.
- Lee los argumentos de la línea de comandos para el umbral de batería y la ruta al archivo virtual de nivel de batería. La receta del componente establece estos argumentos de la línea de comandos en función de los parámetros de configuración, por lo que puede personalizar estos valores al implementar el componente.
- Utiliza el cliente IPC V2 en la SDK para dispositivos con AWS IoTversión 2 para Python para comunicarse con el software AWS IoT Greengrass Core. En comparación con el cliente IPC original, el cliente IPC V2 reduce la cantidad de código que hay que escribir para usar el IPC en componentes personalizados.
- Se suscribe para actualizar las notificaciones mediante la operación IPC. SubscribeToComponentUpdates El software AWS IoT Greengrass principal envía notificaciones antes y después de cada implementación. El componente llama a la siguiente

función cada vez que recibe una notificación. Si la notificación es para una próxima implementación, el componente comprueba si el nivel de la batería es inferior a un umbral. Si el nivel de la batería está por debajo del umbral, el componente aplaza la actualización durante 30 segundos mediante la operación DeferComponentUpdateIPC. De lo contrario, si el nivel de la batería no está por debajo del umbral, el componente confirma la actualización para que la actualización pueda continuar.

```
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**
>
> El software AWS IoT Greengrass principal no envía notificaciones de actualización para las implementaciones locales, por lo que debe implementar este componente mediante el servicio AWS IoT Greengrass en la nube para probarlo.

4. Utilice un editor de texto para crear la receta del componente en un archivo denominado `recipe.json` o `recipe.yaml`. La receta del componente define los metadatos del componente, los parámetros de configuración predeterminados y los scripts de ciclo de vida específicos de la plataforma.

   JSON

   Por ejemplo, en un sistema basado en Linux, puede ejecutar el siguiente comando para usar GNU nano para crear el archivo.

```
nano recipe.json
```

Copia el siguiente JSON en el archivo.

```
{
  "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

Por ejemplo, en un sistema basado en Linux, puede ejecutar el siguiente comando para usar GNU nano para crear el archivo.

```
nano recipe.yaml
```

Copia el siguiente archivo YAML en el archivo.

```
---
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
```

Esta receta especifica lo siguiente:

- Parámetros de configuración predeterminados para el umbral de la batería, la ruta del archivo de batería virtual en los dispositivos principales de Linux y la ruta del archivo de batería virtual en los dispositivos principales de Windows.
- Un `install` ciclo de vida que instala la última versión de la versión SDK para dispositivos con AWS IoT 2 para Python.
- Un `run` ciclo de vida en el que se ejecuta la aplicación Python`main.py`.
- Marcadores de posición, como `COMPONENT_NAME` y`COMPONENT_VERSION`, donde la CLI de GDK reemplaza la información al crear la receta del componente.

Para obtener más información sobre las recetas de componentes, consulte. AWS IoT Greengrass referencia de recetas de componentes

Paso 3: Publicar el componente en el AWS IoT Greengrass servicio

En esta sección, publica el componente Hello World en el servicio AWS IoT Greengrass en la nube. Una vez que un componente esté disponible en el servicio en la AWS IoT Greengrass nube, puede implementarlo en los dispositivos principales. Utilice la CLI de GDK para publicar el componente

desde su ordenador de desarrollo en el servicio AWS IoT Greengrass en la nube. La CLI de GDK carga la receta y los artefactos del componente por usted.

Para publicar el componente Hello World en el servicio AWS IoT Greengrass

1. Ejecute el siguiente comando para crear el componente mediante la CLI de GDK. El comando component build crea una receta y artefactos basados en el archivo de configuración CLI de GDK. En este proceso, la CLI de GDK crea un archivo ZIP que contiene el código fuente del componente.

   ```
   gdk component build
   ```

   Debería ver mensajes similares a los del siguiente ejemplo.

   ```
   [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. Ejecute el siguiente comando para publicar el componente en el servicio AWS IoT Greengrass en la nube. El comando component publish carga el artefacto del archivo ZIP del componente en un bucket de S3. A continuación, actualiza el URI S3 del archivo ZIP en la receta del componente y carga la receta en el servicio. AWS IoT Greengrass En este proceso, la CLI de GDK comprueba qué versión del componente Hello World ya está disponible en el servicio en la AWS IoT Greengrass nube para poder elegir la siguiente versión del parche después de esa versión. Si el componente aún no existe, la CLI de GDK usa la versión`1.0.0`.

   ```
   gdk component publish
   ```

Deberías ver mensajes similares a los del siguiente ejemplo. El resultado indica la versión del componente que creó la CLI de 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. Copie el nombre del bucket de S3 de la salida. El nombre del depósito se utiliza más adelante para permitir que el dispositivo principal descargue los artefactos componentes de este depósito.
4. (Opcional) Vea el componente en la AWS IoT Greengrass consola para comprobar que se ha cargado correctamente. Haga lo siguiente:

   a. En el menú de navegación de la AWS IoT Greengrassconsola, elija Componentes.

   b. En la página Componentes, seleccione la pestaña Mis componentes y, a continuación, elija com.example.BatteryAwareHelloWorld.

      En esta página, puede ver la receta del componente y otra información sobre el componente.
5. Permita que el dispositivo principal acceda a los artefactos de los componentes del depósito S3.

Cada dispositivo principal tiene una función de IAM del dispositivo principal que le permite interactuar con los registros AWS IoT y enviarlos a la AWS nube. Esta función de dispositivo no permite el acceso a los depósitos de S3 de forma predeterminada, por lo que debe crear y adjuntar una política que permita al dispositivo principal recuperar los artefactos de los componentes del depósito de S3.

Si la función de tu dispositivo ya te permite acceder al bucket de S3, puedes saltarte este paso. De lo contrario, cree una política de IAM que permita el acceso y asójela al rol, de la siguiente manera:

a. En el menú de navegación de la consola de IAM, seleccione Políticas y, a continuación, elija Crear política.

b. En la pestaña JSON, reemplace el contenido del marcador de posición por la política siguiente. Sustituya *greengrass-component-artifacts-us-west-2-123456789012* por el nombre del bucket de S3 en el que la CLI de GDK cargó los artefactos del componente.

   Por ejemplo, si especificó **greengrass-component-artifacts** y **us-west-2** en el archivo de configuración de la CLI de GDK y su Cuenta de AWS ID es**123456789012**, la CLI de GDK utilizará el bucket de S3 denominado. `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. Elija Siguiente.

d. En la sección de detalles de la política, en Nombre, introduzca. **MyGreengrassV2ComponentArtifactPolicy**

e. Elija Crear política.

f. En el menú de navegación de la consola de IAM, seleccione Función y, a continuación, elija el nombre de la función para el dispositivo principal. Especificó este nombre de función al instalar el software AWS IoT Greengrass principal. Si no especificó ningún nombre, el nombre predeterminado es`GreengrassV2TokenExchangeRole`.

g. En Permisos, selecciona Añadir permisos y, a continuación, selecciona Adjuntar políticas.

h. En la página Añadir permisos, selecciona la casilla de verificación situada junto a la `MyGreengrassV2ComponentArtifactPolicy` política que has creado y, a continuación, selecciona Añadir permisos.

Paso 4: implementar y probar el componente en un dispositivo principal

En esta sección, implementará el componente en el dispositivo principal para probar su funcionalidad. En el dispositivo principal, se crea el archivo virtual de nivel de batería para imitar una batería real. A continuación, debe crear despliegues adicionales y observar los archivos de registro de los componentes del dispositivo principal para comprobar si el componente se aplaza y confirma las actualizaciones.

Para implementar y probar el componente Hello World que aplaza las actualizaciones

1. Utilice un editor de texto para crear un archivo virtual del nivel de batería. Este archivo imita una batería real.

   - En los dispositivos principales de Linux, cree un archivo con el nombre`/home/ggc_user/virtual_battery.json`. Ejecute el editor de texto con `sudo` permisos.
   - En los dispositivos principales de Windows, cree un archivo con el nombre`C:\Users\ggc_user\virtual_battery.json`. Ejecute el editor de texto como administrador.

   Por ejemplo, en un sistema basado en Linux, puede ejecutar el siguiente comando para usar GNU nano para crear el archivo.

   ```
   sudo nano /home/ggc_user/virtual_battery.json
   ```

   Copia el siguiente JSON en el archivo.

   ```
   {
   ```

```
  "battery_level": 50
}
```

2. Implemente el componente Hello World en el dispositivo principal. Haga lo siguiente:

   a. En el menú de navegación de la AWS IoT Greengrassconsola, elija Componentes.

   b. En la página Componentes, seleccione la pestaña Mis componentes y, a continuación, elija com.example.BatteryAwareHelloWorld.

   c. En la página com.example.BatteryAwareHelloWorld, elija Implementar.

   d. En Añadir a la implementación, elija una implementación existente para revisarla o cree una nueva y, a continuación, elija Siguiente.

   e. Si opta por crear una nueva implementación, elija el dispositivo principal o el grupo de cosas de destino para la implementación. En la página Especificar el destino, en Destino del despliegue, elija un dispositivo principal o un grupo de cosas y, a continuación, elija Siguiente.

   f. En la página Seleccionar componentes, compruebe que el com.example.BatteryAwareHelloWorldcomponente esté seleccionado y seleccione Siguiente.

   g. En la página Configurar componentes, seleccione y com.example.BatteryAwareHelloWorld, a continuación, haga lo siguiente:

      i. Seleccione Configurar componente.

      ii. En el com.example.BatteryAwareHelloWorld modal Configurar, en Actualización de configuración, en Configuración para fusionar, introduzca la siguiente actualización de configuración.

      ```
      {
        "BatteryThreshold": 70
      }
      ```

      iii. Elija Confirmar para cerrar el modal y, a continuación, elija Siguiente.

   h. En la página Confirmar la configuración avanzada, en la sección Políticas de despliegue, en Política de actualización de componentes, confirme que está seleccionada la opción Notificar componentes. La opción Notificar componentes está seleccionada de forma predeterminada al crear una nueva implementación.

   i. En la página Revisar, elija Implementar.

La implementación puede tardar hasta un minuto en completarse.

3. El software AWS IoT Greengrass Core guarda la salida estándar de los procesos de los componentes en los archivos de registro de la `logs` carpeta. Ejecute el siguiente comando para comprobar que el componente Hello World se ejecuta e imprime los mensajes de estado.

   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
   ```

   Debería ver mensajes similares a los del siguiente ejemplo.

   ```
   Hello, World! Battery level (50) is below threshold (70), so the component will
    defer updates.
   ```

   > **Note**
   >
   > Si el archivo no existe, es posible que la implementación aún no esté completa. Si el archivo no existe en 30 segundos, es probable que la implementación haya fallado. Esto puede ocurrir si el dispositivo principal no tiene permiso para descargar los artefactos del componente desde el depósito de S3, por ejemplo. Ejecute el siguiente comando para ver el archivo de registro del software AWS IoT Greengrass principal. Este archivo incluye registros del servicio de despliegue del dispositivo principal de Greengrass.
   >
   > Linux or Unix
   >
   > ```
   > sudo tail -f /greengrass/v2/logs/greengrass.log
   > ```

Windows Command Prompt (CMD)

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

El `type` comando escribe el contenido del archivo en la terminal. Ejecute este comando varias veces para observar los cambios en el archivo.

PowerShell

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

4. Cree una nueva implementación en el dispositivo principal para comprobar que el componente aplaza la actualización. Haga lo siguiente:

   a. En el menú de navegación de la AWS IoT Greengrassconsola, seleccione Implementaciones.

   b. Elija la implementación que creó o revisó anteriormente.

   c. En la página de despliegue, elija Revisar.

   d. En el modal de despliegue Revisar, seleccione Revisar despliegue.

   e. Elija Siguiente en cada paso y, a continuación, elija Implementar.

5. Ejecute el siguiente comando para volver a ver los registros del componente y comprobar que aplaza la actualización.

   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
   ```

Debería ver mensajes similares a los del siguiente ejemplo. El componente aplaza la actualización durante 30 segundos, por lo que imprime este mensaje repetidamente.

```
Deferred update for deployment 50722a95-a05f-4e2a-9414-da80103269aa.
```

6. Utilice un editor de texto para editar el archivo virtual de nivel de batería y cambie el nivel de batería a un valor superior al umbral, de modo que la implementación pueda continuar.

   - En los dispositivos principales de Linux, edite el nombre del archivo`/home/ggc_user/virtual_battery.json`. Ejecute el editor de texto con `sudo` permisos.
   - En los dispositivos principales de Windows, edite el archivo denominado`C:\Users\ggc_user\virtual_battery.json`. Ejecute el editor de texto como administrador.

   Por ejemplo, en un sistema basado en Linux, puede ejecutar el siguiente comando para usar GNU nano para crear el archivo.

   ```
   sudo nano /home/ggc_user/virtual_battery.json
   ```

   Cambie el nivel de la batería a. `80`

   ```
   {
     "battery_level": 80
   }
   ```

7. Ejecute el siguiente comando para volver a ver los registros del componente y comprobar que reconoce la actualización.

   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
```

Debería ver mensajes similares a los siguientes ejemplos.

```
Hello, World! Battery level (80) is above threshold (70), so the component will
 acknowledge updates.
Acknowledged update for deployment f9499eb2-4a40-40a7-86c1-c89887d859f1.
```

Has completado este tutorial. El componente Hello World aplaza o confirma las actualizaciones en función del nivel de batería del dispositivo principal. Para obtener más información sobre los temas que se exploran en este tutorial, consulte lo siguiente:

- Desarrolle AWS IoT Greengrass componentes
- Implemente AWS IoT Greengrass componentes en los dispositivos
- Úselo SDK para dispositivos con AWS IoT para comunicarse con el núcleo de Greengrass, otros componentes y AWS IoT Core
- AWS IoT GreengrassInterfaz de línea de comandos del kit de desarrollo

Tutorial: Interactúa con dispositivos IoT locales a través de MQTT

Puede completar este tutorial para configurar un dispositivo principal para que interactúe con los dispositivos IoT locales, denominados dispositivos cliente, que se conectan al dispositivo principal a través de MQTT. En este tutorial, configurará AWS IoT las cosas para usar la detección en la nube para conectarse al dispositivo principal como dispositivos cliente. Al configurar la detección en la nube, un dispositivo cliente puede enviar una solicitud al servicio AWS IoT Greengrass en la nube para descubrir los dispositivos principales. La respuesta AWS IoT Greengrass incluye la información de conectividad y los certificados de los dispositivos principales que usted configura para que los detecte el dispositivo cliente. Luego, el dispositivo cliente puede usar esta información para conectarse a un dispositivo principal disponible donde pueda comunicarse a través de MQTT.

En este tutorial, aprenderá a hacer lo siguiente:

1. Revise y actualice los permisos del dispositivo principal, si es necesario.

2. Asocie los dispositivos cliente al dispositivo principal para que puedan detectar el dispositivo principal mediante la detección en la nube.
3. Implemente los componentes de Greengrass en el dispositivo principal para permitir la compatibilidad con los dispositivos del cliente.
4. Conecte los dispositivos cliente al dispositivo principal y pruebe la comunicación con el servicio AWS IoT Core en la nube.
5. Desarrolle un componente Greengrass personalizado que se comunique con los dispositivos cliente.
6. Desarrolle un componente personalizado que interactúe con las sombras de los dispositivos AWS IoT cliente.

Este tutorial utiliza un dispositivo de núcleo único y un dispositivo cliente único. También puede seguir el tutorial para conectar y probar varios dispositivos cliente.

Puedes dedicar entre 30 y 60 minutos a este tutorial.

Requisitos previos

Necesitará lo siguiente para completar este tutorial:

- Una Cuenta de AWS. Si no dispone de una, consulte Configure un Cuenta de AWS.
- Un usuario AWS Identity and Access Management (IAM) con permisos de administrador.
- Un dispositivo central de Greengrass. Para obtener más información sobre cómo configurar un dispositivo principal, consulteConfiguración de los dispositivos AWS IoT Greengrass principales.
  - El dispositivo principal debe ejecutar Greengrass nucleus v2.6.0 o posterior. Esta versión incluye compatibilidad con caracteres comodín en la comunicación local entre publicaciones y suscripciones y compatibilidad con dispositivos cliente ocultos.

    > **Note**
    >
    > La compatibilidad con dispositivos cliente requiere Greengrass nucleus v2.2.0 o posterior. Sin embargo, en este tutorial se analizan las funciones más recientes, como la compatibilidad con los caracteres comodín MQTT en las publicaciones o suscripciones locales y la compatibilidad con los dispositivos cliente ocultos. Estas funciones requieren Greengrass nucleus v2.6.0 o posterior.

  - El dispositivo principal debe estar en la misma red que los dispositivos cliente para poder conectarse.
  - (Opcional) Para completar los módulos en los que se desarrollan componentes personalizados de Greengrass, el dispositivo principal debe ejecutar la CLI de Greengrass. Para obtener más información, consulte Instalación de la CLI de Greengrass.
- En este tutorial, AWS IoT hay algo que conectar como dispositivo cliente. Para obtener más información, consulte Crear AWS IoT recursos en la Guía para AWS IoT Core desarrolladores.
  - La AWS IoT política del dispositivo cliente debe permitir el `greengrass:Discover` permiso. Para obtener más información, consulte AWS IoTPolítica mínima para los dispositivos cliente.
  - El dispositivo cliente debe estar en la misma red que el dispositivo principal.
  - El dispositivo cliente debe ejecutar Python 3.
  - El dispositivo cliente debe ejecutar Git.

Paso 1: Revisa y actualiza la AWS IoT política principal de dispositivos

Para ser compatible con los dispositivos cliente, la AWS IoT política de un dispositivo principal debe permitir los siguientes permisos:

- `greengrass:PutCertificateAuthorities`
- `greengrass:VerifyClientDeviceIdentity`
- `greengrass:VerifyClientDeviceIoTCertificateAssociation`
- `greengrass:GetConnectivityInfo`
- `greengrass:UpdateConnectivityInfo`— (Opcional) Este permiso es necesario para utilizar el componente de detección de IP, que envía la información de conectividad de red del dispositivo principal al servicio AWS IoT Greengrass en la nube.

Para obtener más información sobre estos permisos y AWS IoT políticas para los dispositivos principales, consulte Políticas de AWS IoT para operaciones de plano de datos yAWS IoTPolítica mínima de compatibilidad con los dispositivos cliente.

En esta sección, revisas las AWS IoT políticas de tu dispositivo principal y añades los permisos necesarios que falten. Si utilizaste el instalador del software AWS IoT Greengrass principal para aprovisionar recursos, tu dispositivo principal tiene una AWS IoT política que permite el acceso a todas AWS IoT Greengrass las acciones (`greengrass:*`). En este caso, solo debe actualizar

la AWS IoT política si planea configurar el componente de administrador de sombras con el que sincronizar las sombras de los dispositivosAWS IoT Core. De lo contrario, puede omitir esta sección.

Para revisar y actualizar la AWS IoT política de un dispositivo principal

1. En el menú de navegación de la AWS IoT Greengrassconsola, selecciona Dispositivos principales.
2. En la página de dispositivos principales, selecciona el dispositivo principal que deseas actualizar.
3. En la página de detalles del dispositivo principal, selecciona el enlace al dispositivo principal. Este enlace abre la página de detalles del dispositivo en la AWS IoT consola.
4. En la página de detalles de la cosa, selecciona Certificados.
5. En la pestaña Certificados, selecciona el certificado activo del objeto.
6. En la página de detalles del certificado, selecciona Políticas.
7. En la pestaña Políticas, elija la AWS IoT política que desee revisar y actualizar. Puede añadir los permisos necesarios a cualquier política que esté asociada al certificado activo del dispositivo principal.

   > **Note**
   >
   > Si usó el instalador de software AWS IoT Greengrass Core para aprovisionar recursos, tiene dos AWS IoT políticas. Le recomendamos que elija la política nombrada GreengrassV2IoTThingPolicy, si existe. Los dispositivos principales que cree con el instalador rápido utilizan este nombre de política de forma predeterminada. Si agrega permisos a esta política, también los otorga a otros dispositivos principales que usan esta política.

8. En la descripción general de la política, selecciona Editar la versión activa.
9. Revisa la política para ver los permisos necesarios y añade los permisos necesarios que falten.
10. Para establecer una nueva versión de la política como la versión activa, en Estado de la versión de la política, seleccione Establecer la versión editada como la versión activa de esta política.
11. Seleccione Guardar como versión nueva.

Paso 2: Habilita la compatibilidad con los dispositivos cliente

Para que un dispositivo cliente utilice la detección en la nube para conectarse a un dispositivo principal, debe asociar los dispositivos. Cuando asocias un dispositivo cliente a un dispositivo

principal, permites que ese dispositivo cliente recupere las direcciones IP y los certificados del dispositivo principal para usarlos en la conexión.

Para permitir que los dispositivos cliente se conecten de forma segura a un dispositivo principal y se comuniquen con los componentes de GreengrassAWS IoT Core, debe implementar los siguientes componentes de Greengrass en el dispositivo principal:

- Autenticación del dispositivo cliente (`aws.greengrass.clientdevices.Auth`)

  Implemente el componente de autenticación del dispositivo cliente para autenticar los dispositivos cliente y autorizar las acciones de los dispositivos cliente. Este componente permite que sus AWS IoT cosas se conecten a un dispositivo principal.

  Este componente requiere alguna configuración para poder usarlo. Debe especificar los grupos de dispositivos cliente y las operaciones que cada grupo está autorizado a realizar, como conectarse y comunicarse a través de MQTT. Para obtener más información, consulte la configuración del componente de autenticación del dispositivo cliente.
- Bróker MQTT 3.1.1 (Moquette) (`aws.greengrass.clientdevices.mqtt.Moquette`)

  Implemente el componente de broker MQTT de Moquette para ejecutar un broker MQTT ligero. El broker MQTT de Moquette es compatible con MQTT 3.1.1 e incluye soporte local para QoS 0, QoS 1, QoS 2, mensajes retenidos, mensajes de última voluntad y suscripciones persistentes.

  No es necesario configurar este componente para usarlo. Sin embargo, puede configurar el puerto en el que este componente opera el broker MQTT. De forma predeterminada, utiliza el puerto 8883.
- Puente MQTT (`aws.greengrass.clientdevices.mqtt.Bridge`)

  (Opcional) Implemente el componente de puente MQTT para retransmitir mensajes entre los dispositivos cliente (MQTT local), la publicación/suscripción local y el MQTT. AWS IoT Core Configure este componente para sincronizar los dispositivos cliente AWS IoT Core e interactuar con los dispositivos cliente de los componentes de Greengrass.

  Para poder utilizar este componente, es necesario configurarlo. Debe especificar las asignaciones de temas en las que este componente transmite los mensajes. Para obtener más información, consulte Configuración del componente del puente MQTT.
- Detector de IP (`aws.greengrass.clientdevices.IPDetector`)

(Opcional) Implemente el componente detector de IP para informar automáticamente al servicio en la nube de los puntos finales del agente MQTT del AWS IoT Greengrass dispositivo principal. No puede usar este componente si tiene una configuración de red compleja, como una en la que un router reenvía el puerto intermediario MQTT al dispositivo principal.

No es necesario configurar este componente para usarlo.

En esta sección, utilizará la AWS IoT Greengrass consola para asociar los dispositivos cliente e implementar los componentes del dispositivo cliente en un dispositivo principal.

Para habilitar la compatibilidad con dispositivos cliente

1. Vaya a la consola de AWS IoT Greengrass.
2. En el menú de navegación de la izquierda, selecciona Dispositivos principales.
3. En la página Dispositivos principales, selecciona el dispositivo principal en el que deseas habilitar la compatibilidad con los dispositivos cliente.
4. En la página de detalles del dispositivo principal, selecciona la pestaña Dispositivos cliente.
5. En la pestaña Dispositivos cliente, elija Configurar la detección en la nube.

   Se abre la página Configurar la detección de dispositivos principales. En esta página, puede asociar los dispositivos cliente a un dispositivo principal e implementar los componentes del dispositivo cliente. En esta página, se selecciona el dispositivo principal en el paso 1: Seleccione los dispositivos principales de destino.

   > **Note**
   >
   > También puede utilizar esta página para configurar la detección de dispositivos principales para un grupo de cosas. Si elige esta opción, puede implementar los componentes de los dispositivos cliente en todos los dispositivos principales de un grupo de cosas. Sin embargo, si elige esta opción, deberá asociar manualmente los dispositivos cliente a cada dispositivo principal más adelante, después de crear la implementación. En este tutorial, configurará un dispositivo de un solo núcleo.

6. En el paso 2: Asociar los dispositivos cliente, asocie el AWS IoT dispositivo cliente al dispositivo principal. Esto permite que el dispositivo cliente utilice la detección en la nube para recuperar la información de conectividad y los certificados del dispositivo principal. Haga lo siguiente:

a. Elija Asociar dispositivos cliente.

b. En el modal Asociar dispositivos cliente al dispositivo principal, introduzca el nombre del elemento AWS IoT que desee asociar.

c. Elija Add (Añadir).

d. Elija Associate (Asociar).

7. En el paso 3: configurar e implementar los componentes de Greengrass, despliegue componentes para habilitar la compatibilidad con los dispositivos cliente. Si el dispositivo principal de destino tiene una implementación anterior, esta página revisa esa implementación. De lo contrario, esta página crea una nueva implementación para el dispositivo principal. Haga lo siguiente para configurar e implementar los componentes del dispositivo cliente:

   a. El dispositivo principal debe ejecutar Greengrass nucleus v2.6.0 o posterior para completar este tutorial. Si el dispositivo principal ejecuta una versión anterior, haga lo siguiente:

      i. Seleccione la casilla para implementar el aws.greengrass.Nucleuscomponente.

      ii. Para el aws.greengrass.Nucleuscomponente, elija Editar configuración.

      iii. Para la versión del componente, elija la versión 2.6.0 o posterior.

      iv. Seleccione Confirmar.

      > **Note**
      >
      > Si actualiza el núcleo de Greengrass desde una versión secundaria anterior y el dispositivo principal ejecuta componentes AWS proporcionados que dependen del núcleo, también debe actualizar los componentes AWS proporcionados a versiones más recientes. Puede configurar la versión de estos componentes al revisar la implementación más adelante en este tutorial. Para obtener más información, consulte Actualice el software AWS IoT Greengrass principal (OTA).

   b. Para el aws.greengrass.clientdevices.Authcomponente, elija Editar configuración.

   c. En el modal de edición de configuración del componente de autenticación del dispositivo cliente, configure una política de autorización que permita a los dispositivos cliente publicar y suscribirse al agente MQTT en el dispositivo principal. Haga lo siguiente:

      i. En Configuración, en el bloque Configuración para fusionar códigos, introduzca la siguiente configuración, que contiene una política de autorización de dispositivos

cliente. Cada política de autorización de grupos de dispositivos especifica un conjunto de acciones y los recursos en los que un dispositivo cliente puede realizar esas acciones.

- Esta política permite que los dispositivos cliente cuyos nombres comiencen por se conecten y se comuniquen sobre todos los temas de MQTT. `MyClientDevice` Sustituya *`MyClientDevice*`* por el nombre del dispositivo AWS IoT que se va a conectar como dispositivo cliente. También puede especificar un nombre con un `*` comodín que coincida con el nombre del dispositivo cliente. El `*` comodín debe estar al final del nombre.

  Si tiene que conectar un segundo dispositivo cliente, sustituya *`MyOtherClientDevice*`* por el nombre de ese dispositivo cliente o por un patrón comodín que coincida con el nombre de ese dispositivo cliente. De lo contrario, puede eliminar o conservar esta sección de la regla de selección que permite que los dispositivos cliente con nombres que coincidan `MyOtherClientDevice*` se conecten y se comuniquen.
- Esta política utiliza un `OR` operador para permitir también que los dispositivos cliente cuyos nombres comiencen por «se conecten y se comuniquen en todos los temas de MQTT». `MyOtherClientDevice` Puede eliminar esta cláusula de la regla de selección o modificarla para que coincida con los dispositivos cliente a los que se va a conectar.
- Esta política permite a los dispositivos cliente publicar y suscribirse a todos los temas de MQTT. Para seguir las mejores prácticas de seguridad, `mqtt:publish` limite las `mqtt:subscribe` operaciones al conjunto mínimo de temas que utilizan los dispositivos cliente para comunicarse.

```
{
  "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": [
            "*"
          ]
        }
      }
    }
  }
}
```

Para obtener más información, consulte Configuración del componente de autenticación del dispositivo cliente.

ii. Seleccione Confirmar.

d. Para el aws.greengrass.clientdevices.mqtt.Bridgecomponente, elija Editar configuración.

e. En el modal Editar configuración del componente MQTT bridge, configure un mapeo de temas que transmita los mensajes MQTT de los dispositivos cliente a. AWS IoT Core Haga lo siguiente:

i. En Configuración, en el bloque Configuración para fusionar códigos, introduzca la siguiente configuración. Esta configuración especifica la retransmisión de los mensajes MQTT del filtro de `clients/+/hello/world` temas desde los dispositivos cliente al servicio AWS IoT Core en la nube. Por ejemplo, este filtro de temas coincide con el `clients/MyClientDevice1/hello/world` tema.

```
{
  "mqttTopicMapping": {
    "HelloWorldIotCoreMapping": {
      "topic": "clients/+/hello/world",
      "source": "LocalMqtt",
      "target": "IotCore"
    }
  }
}
```

Para obtener más información, consulte Configuración de los componentes del puente MQTT.

ii. Seleccione Confirmar.

8. Seleccione Revisar e implementar para revisar la implementación que esta página crea para usted.
9. Si no ha configurado previamente el rol de servicio de Greengrass en esta región, la consola abre un modal para configurar el rol de servicio por usted. El componente de autenticación del dispositivo cliente usa esta función de servicio para verificar la identidad de los dispositivos cliente, y el componente detector de IP usa esta función de servicio para administrar la información de conectividad principal de los dispositivos. Elija Grant permissions (Conceder permisos).
10. En la página de revisión, elija Implementar para iniciar la implementación en el dispositivo principal.
11. Para comprobar que la implementación se ha realizado correctamente, compruebe el estado de la implementación y compruebe los registros del dispositivo principal. Para comprobar el estado de la implementación en el dispositivo principal, puede elegir Target en la descripción general de la implementación. Para obtener más información, consulte los siguientes temas:

- Verificar el estado de implementación
- Supervisar AWS IoT Greengrass registros

Paso 3: Conectar los dispositivos cliente

Los dispositivos cliente pueden usarlo SDK para dispositivos con AWS IoT para detectar, conectarse y comunicarse con un dispositivo principal. El dispositivo cliente debe ser una AWS IoT cosa. Para obtener más información, consulte Crear un objeto objeto en la Guía para AWS IoT Core desarrolladores.

En esta sección, instalará la SDK para dispositivos con AWS IoTversión 2 para Python y ejecutará la aplicación de ejemplo Greengrass Discovery desde. SDK para dispositivos con AWS IoT

> **Note**
>
> También SDK para dispositivos con AWS IoT está disponible en otros lenguajes de programación. En este tutorial se usa SDK para dispositivos con AWS IoT la versión 2 para Python, pero puedes explorar los otros SDK para tu caso de uso. Para obtener más información, consulte los SDKs de dispositivos de AWS IoT en la Guía para desarrolladores de AWS IoT Core.

Para conectar un dispositivo cliente a un dispositivo principal

1. Descargue e instale la SDK para dispositivos con AWS IoTversión 2 para Python en el AWS IoT dispositivo para conectarlo como dispositivo cliente.

   En el dispositivo cliente, haga lo siguiente:

   a. Clona el repositorio SDK para dispositivos con AWS IoT v2 para Python para descargarlo.

   ```
   git clone https://github.com/aws/aws-iot-device-sdk-python-v2.git
   ```

   b. Instale la SDK para dispositivos con AWS IoT versión 2 para Python.

   ```
   python3 -m pip install --user ./aws-iot-device-sdk-python-v2
   ```

2. Cambie a la carpeta de muestras de la SDK para dispositivos con AWS IoT versión 2 para Python.

```
cd aws-iot-device-sdk-python-v2/samples
```

3. Ejecute la aplicación Greengrass Discovery de muestra. Esta aplicación espera argumentos que especifiquen el nombre del dispositivo cliente, el tema y el mensaje de MQTT que se van a utilizar y los certificados que autentican y protegen la conexión. En el siguiente ejemplo, se envía un mensaje de Hello World al `clients/`*`MyClientDevice1`*`/hello/world` tema.

> **Note**
>
> Este tema coincide con el tema en el que se configuró AWS IoT Core anteriormente el puente MQTT para retransmitir mensajes.

- Sustituya *`MyClientDevice1`* por el nombre del dispositivo cliente.
- Sustituya *`~/certs/ AmazonRoot CA1.pem`* por la ruta al certificado de CA raíz de Amazon en el dispositivo cliente.
- Sustituya *`~/certs/device.pem.crt por la ruta al certificado del dispositivo cliente`*.
- Sustituya *`~/certs/private.pem.key`* por la ruta al archivo de clave privada del dispositivo cliente.
- Sustituya *`us-east-1`* por la región en la que funcionan el dispositivo cliente y el dispositivo principal. AWS

```
python3 basic_discovery.py \\
  --thing_name MyClientDevice1 \\
  --topic 'clients/MyClientDevice1/hello/world' \\
  --message 'Hello World!' \\
  --ca_file ~/certs/AmazonRootCA1.pem \\
  --cert ~/certs/device.pem.crt \\
  --key ~/certs/private.pem.key \\
  --region us-east-1 \\
  --verbosity Warn
```

La aplicación de ejemplo Discovery envía el mensaje 10 veces y se desconecta. También se suscribe al mismo tema en el que publica los mensajes. Si el resultado indica que la aplicación recibió mensajes MQTT sobre el tema, el dispositivo cliente puede comunicarse correctamente con el dispositivo principal.

```
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}'
```

Si, en cambio, la aplicación genera un error, consulte Solución de problemas de detección de Greengrass.

También puede ver los registros de Greengrass en el dispositivo principal para comprobar si el dispositivo cliente se conecta y envía mensajes correctamente. Para obtener más información, consulte Supervisar AWS IoT Greengrass registros.

4. Compruebe que el puente MQTT retransmita los mensajes del dispositivo cliente a. AWS IoT Core Puede utilizar el cliente de pruebas de MQTT de la AWS IoT Core consola para suscribirse a un filtro de temas de MQTT. Haga lo siguiente:

   a. Vaya a la consola de AWS IoT.

   b. En el menú de navegación de la izquierda, en Probar, elija el cliente de prueba MQTT.

   c. En la pestaña Suscribirse a un tema, en Filtro por tema, escriba para suscribirse `clients/+/hello/world` a los mensajes del dispositivo cliente desde el dispositivo principal.

   d. Elija Suscribirse.

   e. Vuelva a ejecutar la aplicación de publicación/suscripción en el dispositivo cliente.

      El cliente de prueba de MQTT muestra los mensajes que se envían desde el dispositivo cliente sobre temas que coinciden con este filtro de temas.

Paso 4: Desarrolle un componente que se comunique con los dispositivos cliente

Puede desarrollar componentes de Greengrass que se comuniquen con los dispositivos cliente. Los componentes utilizan la comunicación entre procesos (IPC) y la interfaz local de publicación/suscripción para comunicarse en un dispositivo central. Para interactuar con los dispositivos cliente, configure el componente de puente MQTT para retransmitir mensajes entre los dispositivos cliente y la interfaz local de publicación/suscripción.

En esta sección, se actualiza el componente de puente MQTT para retransmitir los mensajes desde los dispositivos cliente a la interfaz local de publicación/suscripción. A continuación, desarrolla un componente que se suscribe a estos mensajes e imprime los mensajes cuando los recibe.

Desarrollar un componente que se comunique con los dispositivos cliente

1. Revise la implementación en el dispositivo principal y configure el componente de puente MQTT para retransmitir los mensajes desde los dispositivos cliente a la publicación o suscripción local. Haga lo siguiente:

a. Vaya a la consola de AWS IoT Greengrass.

b. En el menú de navegación de la izquierda, elija Dispositivos principales.

c. En la página Dispositivos principales, elige el dispositivo principal que vas a utilizar para este tutorial.

d. En la página de detalles del dispositivo principal, selecciona la pestaña Dispositivos cliente.

e. En la pestaña Dispositivos cliente, elija Configurar la detección en la nube.

   Se abre la página Configurar la detección de dispositivos principales. En esta página, puede cambiar o configurar qué componentes del dispositivo cliente se implementan en el dispositivo principal.

f. En el paso 3, para el aws.greengrass.clientdevices.mqtt.Bridgecomponente, elija Editar configuración.

g. En el modal de edición de configuración del componente MQTT bridge, configure un mapeo de temas que transmita los mensajes MQTT desde los dispositivos cliente a la interfaz local de publicación/suscripción. Haga lo siguiente:

   i. En Configuración, en el bloque Configuración para fusionar códigos, introduzca la siguiente configuración. Esta configuración especifica la retransmisión de mensajes MQTT sobre temas que coincidan con el filtro de `clients/+/hello/world` temas desde los dispositivos cliente al servicio AWS IoT Core en la nube y al agente local de publicación/suscripción de Greengrass.

```
{
  "mqttTopicMapping": {
    "HelloWorldIotCoreMapping": {
      "topic": "clients/+/hello/world",
      "source": "LocalMqtt",
      "target": "IotCore"
    },
    "HelloWorldPubsubMapping": {
      "topic": "clients/+/hello/world",
      "source": "LocalMqtt",
      "target": "Pubsub"
    }
  }
}
```

Para obtener más información, consulte Configuración de los componentes del puente MQTT.

ii. Seleccione Confirmar.

h. Seleccione Revisar e implementar para revisar la implementación que esta página crea para usted.

i. En la página de revisión, elija Implementar para iniciar la implementación en el dispositivo principal.

j. Para comprobar que la implementación se ha realizado correctamente, compruebe el estado de la implementación y compruebe los registros del dispositivo principal. Para comprobar el estado de la implementación en el dispositivo principal, puede elegir Target en la descripción general de la implementación. Para obtener más información, consulte los siguientes temas:

- Verificar el estado de implementación
- Supervisar AWS IoT Greengrass registros

2. Desarrolle e implemente un componente de Greengrass que se suscriba a los mensajes de Hello World desde los dispositivos cliente. Haga lo siguiente:

a. Cree carpetas para recetas y artefactos en el dispositivo principal.

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**
>
> Debe utilizar el siguiente formato para la ruta de la carpeta de artefactos. Incluya el nombre y la versión del componente que especifique en la receta.
>
> ```
> artifacts/componentName/componentVersion/
> ```

b. Utilice un editor de texto para crear una receta de componentes con los siguientes contenidos. Esta receta especifica instalar la SDK para dispositivos con AWS IoT versión 2 para Python y ejecutar un script que se suscriba al tema e imprima los mensajes.

Por ejemplo, en un sistema basado en Linux, puede ejecutar el siguiente comando para usar GNU nano para crear el archivo.

```
nano recipes/com.example.clientdevices.MyHelloWorldSubscriber-1.0.0.json
```

Copie la siguiente receta en el archivo.

```
{
  "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. Utilice un editor de texto para crear un artefacto de script de Python denominado `hello_world_subscriber.py` con el siguiente contenido. Esta aplicación utiliza el servicio IPC de publicación/suscripción para suscribirse al `clients/+/hello/world` tema e imprimir los mensajes que recibe.

   Por ejemplo, en un sistema basado en Linux, puede ejecutar el siguiente comando para usar GNU nano para crear el archivo.

   ```
   nano artifacts/com.example.clientdevices.MyHelloWorldSubscriber/1.0.0/
   hello_world_subscriber.py
   ```

   Copia el siguiente código de Python en el archivo.

   ```
   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**
>
> Este componente usa el cliente IPC V2 en la SDK para dispositivos con AWS IoTv2 para Python para comunicarse con el software AWS IoT Greengrass Core. En

comparación con el cliente IPC original, el cliente IPC V2 reduce la cantidad de código que hay que escribir para usar el IPC en componentes personalizados.

d. Utilice la CLI de Greengrass para implementar el componente.

Linux or Unix

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

Windows Command Prompt (CMD)

```
C:\greengrass\v2/bin/greengrass-cli deployment create ^
  --recipeDir recipes ^
  --artifactDir artifacts ^
  --merge "com.example.clientdevices.MyHelloWorldSubscriber=1.0.0"
```

PowerShell

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

3. Consulte los registros de los componentes para comprobar que el componente se ha instalado correctamente y que está suscrito al tema.

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
```

Puede mantener el feed de registro abierto para comprobar que el dispositivo principal recibe los mensajes.

4. En el dispositivo cliente, vuelva a ejecutar la aplicación Greengrass Discovery de muestra para enviar mensajes al dispositivo principal.

```
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. Vuelva a ver los registros de los componentes para comprobar que el componente recibe e imprime los mensajes del dispositivo cliente.

   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
   ```

Paso 5: Desarrolle un componente que interactúe con las sombras de los dispositivos cliente

Puede desarrollar componentes de Greengrass que interactúen con las sombras de los dispositivos del AWS IoTdispositivo cliente. Una sombra es un documento JSON que almacena la información de estado actual o deseada de AWS IoT un objeto, como un dispositivo cliente. Los componentes personalizados pueden acceder a las sombras de los dispositivos cliente para gestionar su estado, incluso cuando el dispositivo cliente no está conectado a ellos. AWS IoT Cada AWS IoT elemento

tiene una sombra sin nombre y también puedes crear varias sombras con nombre para cada elemento.

En esta sección, implementará el componente administrador de sombras para gestionar las sombras en el dispositivo principal. También debe actualizar el componente de puente MQTT para retransmitir mensajes ocultos entre los dispositivos cliente y el componente administrador de sombras. A continuación, desarrolla un componente que actualiza las actualizaciones ocultas de los dispositivos cliente y ejecuta una aplicación de ejemplo en los dispositivos cliente que responde a las actualizaciones ocultas del componente. Este componente representa una aplicación de gestión de la iluminación inteligente, en la que el dispositivo principal gestiona el estado del color de las luces inteligentes que se conectan a él como dispositivos cliente.

Desarrollar un componente que interactúe con las sombras de los dispositivos cliente

1. Revise la implementación en el dispositivo principal para implementar el componente de administrador de sombras y configurar el componente de puente MQTT para retransmitir mensajes instantáneos entre los dispositivos cliente y la publicación o suscripción local, donde se comunica el administrador de sombras. Haga lo siguiente:

   a. Vaya a la consola de AWS IoT Greengrass.

   b. En el menú de navegación de la izquierda, selecciona Dispositivos principales.

   c. En la página Dispositivos principales, elige el dispositivo principal que vas a utilizar para este tutorial.

   d. En la página de detalles del dispositivo principal, selecciona la pestaña Dispositivos cliente.

   e. En la pestaña Dispositivos cliente, elija Configurar la detección en la nube.

      Se abre la página Configurar la detección de dispositivos principales. En esta página, puede cambiar o configurar qué componentes del dispositivo cliente se implementan en el dispositivo principal.

   f. En el paso 3, para el aws.greengrass.clientdevices.mqtt.Bridgecomponente, elija Editar configuración.

   g. En el modal de edición de configuración del componente MQTT bridge, configure un mapeo de temas que retransmita los mensajes MQTT en temas ocultos de los dispositivos entre los dispositivos cliente y la interfaz local de publicación/suscripción. También debe confirmar que la implementación especifica una versión de puente MQTT compatible. La compatibilidad con la sombra de dispositivos cliente requiere MQTT bridge v2.2.0 o posterior. Haga lo siguiente:

i. Para la versión Component, elija la versión 2.2.0 o posterior.

ii. En Configuración, en el bloque de códigos Configuración para fusionar, introduzca la siguiente configuración. Esta configuración especifica la retransmisión de mensajes MQTT sobre temas paralelos.

```
{
  "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"
    }
  }
}
```

Para obtener más información, consulte Configuración de los componentes del puente MQTT.

iii. Seleccione Confirmar.

h. En el paso 3, seleccione el aws.greengrass.ShadowManagercomponente para desplegarlo.

i. Seleccione Revisar e implementar para revisar la implementación que esta página crea para usted.

j. En la página de revisión, elija Implementar para iniciar la implementación en el dispositivo principal.

k. Para comprobar que la implementación se ha realizado correctamente, compruebe el estado de la implementación y compruebe los registros del dispositivo principal. Para comprobar el estado de la implementación en el dispositivo principal, puede elegir Target en la descripción general de la implementación. Para obtener más información, consulte los siguientes temas:

- Verificar el estado de implementación
- Supervisar AWS IoT Greengrass registros

2. Desarrolle e implemente un componente de Greengrass que gestione los dispositivos cliente Smart Light. Haga lo siguiente:

a. Cree una carpeta con los artefactos del componente en el dispositivo principal.

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**
>
> Debe utilizar el siguiente formato para la ruta de la carpeta del artefacto. Incluya el nombre y la versión del componente que especifique en la receta.
>
> ```
> artifacts/componentName/componentVersion/
> ```

b. Utilice un editor de texto para crear una receta de componentes con los siguientes contenidos. Esta receta especifica instalar la SDK para dispositivos con AWS IoT versión 2 para Python y ejecutar un script que interactúe con las sombras de los dispositivos cliente de iluminación inteligente para gestionar sus colores.

Por ejemplo, en un sistema basado en Linux, puede ejecutar el siguiente comando para usar GNU nano para crear el archivo.

```
nano recipes/com.example.clientdevices.MySmartLightManager-1.0.0.json
```

Copie la siguiente receta en el archivo.

```
{
  "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. Utilice un editor de texto para crear un artefacto de script de Python denominado `smart_light_manager.py` con el siguiente contenido. Esta aplicación utiliza el servicio IPC oculto para obtener y actualizar las sombras de los dispositivos cliente y el servicio IPC local de publicación/suscripción para recibir las actualizaciones ocultas notificadas.

Por ejemplo, en un sistema basado en Linux, puede ejecutar el siguiente comando para usar GNU nano para crear el archivo.

```
nano artifacts/com.example.clientdevices.MySmartLightManager/1.0.0/
smart_light_manager.py
```

Copia el siguiente código de Python en el archivo.

```
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()
```

Esta aplicación de Python hace lo siguiente:

- Lee la configuración del componente para obtener la lista de dispositivos cliente de Smart Light que hay que gestionar.
- Se suscribe a las notificaciones de actualización de la configuración mediante la operación SubscribeToConfigurationUpdate IPC. El software AWS IoT Greengrass Core envía notificaciones cada vez que cambia la configuración del componente. Cuando el componente recibe una notificación de actualización de la configuración, actualiza la lista de dispositivos cliente de iluminación inteligente que administra.
- Obtiene la sombra de cada dispositivo cliente de iluminación inteligente para obtener su estado de color inicial.
- Establece el color de cada dispositivo cliente de iluminación inteligente en un color aleatorio cada 15 segundos. El componente actualiza la sombra del dispositivo cliente para cambiar su color. Esta operación envía un evento shadow delta al dispositivo cliente a través de MQTT.
- Se suscribe a la actualización paralela de los mensajes aceptados en la interfaz local de publicación/suscripción mediante la operación IPC. SubscribeToTopic Este componente recibe estos mensajes para rastrear el color de cada dispositivo cliente de Smart Light. Cuando un dispositivo cliente Smart Light recibe una actualización oculta, envía un mensaje MQTT para confirmar que ha recibido la actualización. El puente MQTT transmite este mensaje a la interfaz local de publicación/suscripción.

d. Utilice la CLI de Greengrass para implementar el componente. Al implementar este componente, se especifica la lista de dispositivos clientes`smartLightDeviceNames`, cuyas sombras administra. Sustituya *`MyClientDevice1`* por el nombre del dispositivo cliente.

   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. Consulte los registros de los componentes para comprobar que el componente se instala y ejecuta correctamente.

   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
```

El componente envía solicitudes para cambiar el color del dispositivo cliente de iluminación inteligente. El administrador de sombras recibe la solicitud y establece el `desired` estado de la sombra. Sin embargo, el dispositivo cliente de luz inteligente aún no está funcionando, por lo que el `reported` estado de la sombra no cambia. Los registros del componente incluyen los siguientes mensajes.

```
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}
```

Puede mantener el feed de registro abierto para ver cuándo imprime los mensajes el componente.

4. Descarga y ejecuta una aplicación de muestra que usa Greengrass Discovery y se suscribe a las actualizaciones ocultas de los dispositivos. En el dispositivo cliente, haga lo siguiente:

   a. Cambie a la carpeta de muestras de la SDK para dispositivos con AWS IoT versión 2 para Python. Esta aplicación de ejemplo utiliza un módulo de análisis de línea de comandos en la carpeta de ejemplos.

   ```
   cd aws-iot-device-sdk-python-v2/samples
   ```

   b. Utilice un editor de texto para crear un script de Python denominado `basic_discovery_shadow.py` con el siguiente contenido. Esta aplicación utiliza el descubrimiento y las sombras de Greengrass para mantener sincronizada una propiedad entre el dispositivo cliente y el dispositivo principal.

Por ejemplo, en un sistema basado en Linux, puede ejecutar el siguiente comando para usar GNU nano para crear el archivo.

```
nano basic_discovery_shadow.py
```

Copia el siguiente código de Python en el archivo.

```
# 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()
```

Esta aplicación de Python hace lo siguiente:

- Utiliza el descubrimiento de Greengrass para descubrir el dispositivo principal y conectarse a él.
- Solicita el documento paralelo del dispositivo principal para obtener el estado inicial de la propiedad.
- Se suscribe a los eventos delta ocultos, que el dispositivo principal envía cuando el `desired` valor de la propiedad difiere de su `reported` valor. Cuando la aplicación

recibe un evento delta oculto, cambia el valor de la propiedad y envía una actualización al dispositivo principal para establecer el nuevo valor como su `reported` valor.

Esta aplicación combina el descubrimiento de Greengrass y las muestras de sombras de la SDK para dispositivos con AWS IoT v2.

c. Ejecute la aplicación de muestra. Esta aplicación espera argumentos que especifiquen el nombre del dispositivo cliente, la propiedad oculta que se va a utilizar y los certificados que autentican y protegen la conexión.

- Sustituya *MyClientDevice1* por el nombre de la cosa del dispositivo cliente.
- Sustituya *~/certs/ AmazonRoot CA1.pem* por la ruta al certificado de CA raíz de Amazon en el dispositivo cliente.
- Sustituya *~/certs/device.pem.crt por la ruta al certificado del dispositivo cliente*.
- Sustituya *~/certs/private.pem.key* por la ruta al archivo de clave privada del dispositivo cliente.
- Sustituya *us-east-1* por la región en la que funcionan el dispositivo cliente y el dispositivo principal. AWS

```
python3 basic_discovery_shadow.py \
  --thing_name MyClientDevice1 \
  --shadow_property color \
  --ca_file ~/certs/AmazonRootCA1.pem \
  --cert ~/certs/device.pem.crt \
  --key ~/certs/private.pem.key \
  --region us-east-1 \
  --verbosity Warn
```

La aplicación de ejemplo se suscribe a los temas ocultos y espera a recibir los eventos paralelos del dispositivo principal. Si el resultado indica que la aplicación recibe eventos delta ocultos y responde a ellos, el dispositivo cliente puede interactuar correctamente con su sombra en el dispositivo principal.

```
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.
```

Si, en cambio, la aplicación genera un error, consulte Solución de problemas de detección de Greengrass.

También puede ver los registros de Greengrass en el dispositivo principal para comprobar si el dispositivo cliente se conecta y envía mensajes correctamente. Para obtener más información, consulte Supervisar AWS IoT Greengrass registros.

5. Vuelva a ver los registros de los componentes para comprobar que el componente recibe confirmaciones de actualización clandestinas del dispositivo cliente de Smart Light.

   Linux or Unix

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

   PowerShell

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

El componente registra los mensajes para confirmar que el dispositivo cliente de iluminación inteligente ha cambiado de color.

```
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**
>
> La sombra del dispositivo cliente está sincronizada entre el dispositivo principal y el dispositivo cliente. Sin embargo, el dispositivo principal no sincroniza con la sombra del dispositivo clienteAWS IoT Core. Puedes sincronizar una sombra con, por ejemplo, AWS IoT Core para ver o modificar el estado de todos los dispositivos de tu flota. Para obtener más información sobre cómo configurar el componente de administrador de sombras con el que se sincronizan las sombrasAWS IoT Core, consulteSincronice las sombras de los dispositivos locales con AWS IoT Core.

Ha completado este tutorial. El dispositivo cliente se conecta al dispositivo principal, envía mensajes MQTT a los componentes de Greengrass AWS IoT Core y recibe actualizaciones instantáneas del dispositivo principal. Para obtener más información sobre los temas tratados en este tutorial, consulte lo siguiente:

- Asociar dispositivos cliente
- Administre los puntos finales de los dispositivos principales

- Pruebe las comunicaciones del dispositivo cliente
- API RESTful de Greengrass Discovery
- Retransmitir mensajes MQTT entre dispositivos cliente y AWS IoT Core
- Interactúe con los dispositivos cliente en los componentes
- Interactúa con las sombras de los dispositivos
- Interactúa con las sombras de los dispositivos cliente y sincronízalas

Tutorial: Cómo empezar a usar SageMaker Edge Manager

> **Important**
>
> SageMaker Edge Manager dejará de fabricarse el 26 de abril de 2024. Para obtener más información sobre cómo seguir implementando sus modelos en los dispositivos periféricos, consulte el final del ciclo de vida de SageMaker Edge Manager.

Amazon SageMaker Edge Manager es un agente de software que se ejecuta en dispositivos periféricos. SageMaker Edge Manager proporciona administración de modelos para dispositivos periféricos para que pueda empaquetar y usar modelos SageMaker compilados por Amazon NEO directamente en los dispositivos principales de Greengrass. Al usar SageMaker Edge Manager, también puede muestrear los datos de entrada y salida del modelo de sus dispositivos principales y enviar esos datos a ellos Nube de AWS para su monitoreo y análisis. Para obtener más información sobre cómo funciona SageMaker Edge Manager en los dispositivos principales de Greengrass, consulte. Utilice Amazon SageMaker Edge Manager en los dispositivos principales de Greengrass

En este tutorial, se muestra cómo empezar a utilizar SageMaker Edge Manager con los componentes AWS de muestra proporcionados en un dispositivo principal existente. Estos componentes de ejemplo utilizan el componente SageMaker Edge Manager como dependencia para implementar el agente de Edge Manager y realizar inferencias mediante modelos previamente entrenados que se compilaron con Neo. SageMaker Para obtener más información sobre el agente de SageMaker Edge Manager, consulte SageMaker Edge Manager en la Guía para SageMaker desarrolladores de Amazon.

Para configurar y usar el agente SageMaker Edge Manager en un dispositivo principal de Greengrass existente, AWS proporciona un código de ejemplo que puede usar para crear los siguientes ejemplos de componentes de inferencia y modelo.

- Clasificación de imágenes
  - `com.greengrass.SageMakerEdgeManager.ImageClassification`
  - `com.greengrass.SageMakerEdgeManager.ImageClassification.Model`
- Detección de objetos
  - `com.greengrass.SageMakerEdgeManager.ObjectDetection`
  - `com.greengrass.SageMakerEdgeManager.ObjectDetection.Model`

En este tutorial, se muestra cómo implementar los componentes de muestra y el agente de SageMaker Edge Manager.

Temas

Requisitos previos

Para completar este tutorial, debe cumplir los siguientes requisitos previos:

- Un dispositivo principal de Greengrass que se ejecuta en Amazon Linux 2, una plataforma Linux basada en Debian (x86_64 o Armv8) o Windows (x86_64). Si no dispone de una, consulte Tutorial: Introducción a AWS IoT Greengrass V2.
- Python 3.6 o posterior, incluida `pip` la versión de Python, instalada en el dispositivo principal.
- El tiempo de ejecución GLX de la API OpenGL `libgl1-mesa-glx` () instalado en su dispositivo principal.
- Un usuario AWS Identity and Access Management (IAM) con permisos de administrador.
- Un ordenador de desarrollo tipo Windows, Mac o Unix con acceso a Internet que cumpla los siguientes requisitos:
  - Python 3.6 o posterior instalado.
  - AWS CLI instalado y configurado con sus credenciales de usuario de administrador de IAM. Para obtener más información, consulte Instalación AWS CLI y configuración del AWS CLI.

- Los siguientes depósitos de S3 se crearon en el mismo dispositivo principal de Greengrass Cuenta de AWS y en el Región de AWS mismo que él:
  - Un depósito de S3 para almacenar los artefactos que se incluyen en los componentes de inferencia y del modelo de muestra. En este tutorial se utiliza DOC-EXAMPLE-BUCKET1 para hacer referencia a este depósito.
  - Un bucket de S3 que asocie a su flota de dispositivos periféricos. SageMaker SageMaker Edge Manager requiere un depósito S3 para crear la flota de dispositivos perimetrales y almacenar datos de muestra derivados de la ejecución de inferencias en su dispositivo. En este tutorial, se utiliza el documento DOC-EXAMPLE-BUCKET2 para hacer referencia a este depósito.

  Para obtener información sobre la creación de buckets de S3, consulte Introducción a Amazon S3.
- El rol del dispositivo Greengrass se configuró con lo siguiente:
  - Una relación de confianza que `sagemaker.amazonaws.com` permite `credentials.iot.amazonaws.com` y asume el rol, como se muestra en el siguiente ejemplo de política de IAM.

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

  - La política gestionada por AmazonSageMakerEdgeDeviceFleetPolicyIAM.
  - La política gestionada por AmazonSageMakerFullAccessIAM.

- La `s3:GetObject` acción del depósito de S3 que contiene los artefactos de sus componentes, como se muestra en el siguiente ejemplo de política de IAM.

```
{
  "Version": "2012-10-17",
  "Statement": [
  {
    "Action": [
      "s3:GetObject"
    ],
    "Resource": [
      "arn:aws:s3:::DOC-EXAMPLE-BUCKET1/*"
    ],
    "Effect": "Allow"
  }
  ]
}
```

Configura tu dispositivo principal de Greengrass en Edge Manager SageMaker

Las flotas de dispositivos perimetrales de SageMaker Edge Manager son conjuntos de dispositivos agrupados de forma lógica. Para usar SageMaker Edge Manager con AWS IoT Greengrass, debe crear una flota de dispositivos perimetrales que utilice el mismo alias de AWS IoT rol que el dispositivo principal de Greengrass en el que implementa el agente de SageMaker Edge Manager. A continuación, debe registrar el dispositivo principal como parte de esa flota.

Temas

Cree una flota de dispositivos periféricos

Para crear una flota de dispositivos periféricos (consola)

1. En la SageMaker consola de Amazon, selecciona Edge Manager y, a continuación, selecciona Flotas de dispositivos Edge.

2. En la página Flotas de dispositivos, selecciona Crear flota de dispositivos.
3. En Propiedades de la flota de dispositivos, haz lo siguiente:
   - En Nombre de la flota de dispositivos, introduce un nombre para tu flota de dispositivos.
   - Para el rol de IAM, introduzca el nombre de recurso de Amazon (ARN) del alias AWS IoT del rol que especificó al configurar su dispositivo principal de Greengrass.
   - Desactive la opción Crear alias de rol de IAM.
4. Elija Siguiente.
5. En Configuración de salida, para el URI del bucket de S3, introduzca el URI del bucket de S3 que desee asociar a la flota de dispositivos.
6. Seleccione Submit (Enviar).

Registra tu dispositivo principal de Greengrass

Para registrar su dispositivo principal Greengrass como dispositivo perimetral (consola)

1. En la SageMaker consola de Amazon, selecciona Edge Manager y, a continuación, elige dispositivos Edge.
2. En la página Dispositivos, selecciona Registrar dispositivos.
3. En Propiedades del dispositivo, en Nombre de la flota de dispositivos, introduce el nombre de la flota de dispositivos que has creado y, a continuación, selecciona Siguiente.
4. Elija Siguiente.
5. En Fuente del dispositivo, en Nombre del dispositivo, introduzca el AWS IoT nombre del dispositivo principal de Greengrass.
6. Seleccione Submit (Enviar).

Cree los componentes de muestra

Para ayudarle a empezar a utilizar el componente SageMaker Edge Manager, AWS proporciona un script de Python GitHub que crea los componentes de inferencia y modelo de muestra y los carga en su lugar Nube de AWS . Complete los siguientes pasos en un ordenador de desarrollo.

Para crear los componentes de muestra

1. Descargue el repositorio de ejemplos de AWS IoT Greengrass componentes en GitHub su ordenador de desarrollo.
2. Navegue hasta la `/machine-learning/sagemaker-edge-manager` carpeta descargada.

   ```
   cd download-directory/machine-learning/sagemaker-edge-manager
   ```
3. Ejecute el siguiente comando para crear y cargar los componentes de muestra en Nube de AWS.

   ```
   python3 create_components.py -r region -b DOC-EXAMPLE-BUCKET
   ```

   Sustituya la *región* por la Región de AWS que creó su dispositivo principal de Greengrass y sustituya DOC-EXAMPLE-BUCKET1 por el nombre del depósito S3 para almacenar los artefactos de sus componentes.

   > **Note**
   >
   > De forma predeterminada, el script crea componentes de muestra tanto para la clasificación de imágenes como para la inferencia de detección de objetos. Para crear componentes solo para un tipo específico de inferencia, especifique el `-i ImageClassification | ObjectDetection` argumento.

Los componentes de inferencia y modelo de muestra para usarlos con SageMaker Edge Manager ahora se crean en su. Cuenta de AWS Para ver los componentes de muestra en la AWS IoT Greengrass consola, elija Componentes y, a continuación, en Mis componentes, busque los siguientes componentes:

- `com.greengrass.SageMakerEdgeManager.ImageClassification`
- `com.greengrass.SageMakerEdgeManager.ImageClassification.Model`
- `com.greengrass.SageMakerEdgeManager.ObjectDetection`
- `com.greengrass.SageMakerEdgeManager.ObjectDetection.Model`

Ejecute un ejemplo de inferencia de clasificación de imágenes

Para realizar una inferencia de clasificación de imágenes con los componentes AWS de muestra proporcionados y el agente de SageMaker Edge Manager, debe implementar estos componentes en su dispositivo principal. Al implementar estos componentes, se descarga un modelo Resnet-50 previamente preparado y SageMaker compilado en NEO e se instala el agente Edge Manager en el dispositivo. SageMaker El agente de SageMaker Edge Manager carga el modelo y publica los resultados de las inferencias sobre el tema. `gg/sageMakerEdgeManager/image-classification` Para ver estos resultados de inferencia, utilice el cliente AWS IoT MQTT de la AWS IoT consola para suscribirse a este tema.

Temas

Suscríbase al tema de notificaciones

En este paso, configurará el cliente AWS IoT MQTT de la AWS IoT consola para ver los mensajes MQTT publicados por el componente de inferencia de muestra. De forma predeterminada, el componente publica los resultados de las inferencias sobre el tema. `gg/sageMakerEdgeManager/image-classification` Consulte este tema antes de implementar el componente en su dispositivo principal de Greengrass para ver los resultados de la inferencia cuando el componente se ejecute por primera vez.

Para suscribirse al tema de notificaciones predeterminado

1. En el menú de navegación de la AWS IoT consola, elija Test, MQTT test client.
2. En Suscribirse a un tema, en el cuadro Nombre del tema, escriba**`gg/sageMakerEdgeManager/image-classification`**.
3. Elija Suscribirse.

Implemente los componentes de muestra

En este paso, debe configurar e implementar los siguientes componentes en su dispositivo principal:

- `aws.greengrass.SageMakerEdgeManager`
- `com.greengrass.SageMakerEdgeManager.ImageClassification`
- `com.greengrass.SageMakerEdgeManager.ImageClassification.Model`

Para implementar sus componentes (consola)

1. En el menú de navegación de la AWS IoT Greengrass consola, elija Implementaciones y, a continuación, elija la implementación del dispositivo de destino que desee revisar.
2. En la página de despliegue, elija Revisar y, a continuación, elija Revisar despliegue.
3. En la página Especificar destino, seleccione Siguiente.
4. En la página Seleccionar componentes, haga lo siguiente:
   a. En Mis componentes, seleccione los siguientes componentes:
      - `com.greengrass.SageMakerEdgeManager.ImageClassification`
      - `com.greengrass.SageMakerEdgeManager.ImageClassification.Model`
   b. En Componentes públicos, desactive la opción Mostrar solo los componentes seleccionados y, a continuación, seleccione el `aws.greengrass.SageMakerEdgeManager` componente.
   c. Elija Siguiente.
5. En la página Configurar componentes, seleccione el `aws.greengrass.SageMakerEdgeManager` componente y haga lo siguiente.
   a. Seleccione Configurar componente.
   b. En Actualización de la configuración, en Configuración para fusionar, introduzca la siguiente configuración.

```
{
    "DeviceFleetName": "device-fleet-name",
    "BucketName": "DOC-EXAMPLE-BUCKET"
}
```

   *device-fleet-name*Sustitúyalo por el nombre de la flota de dispositivos perimetrales que has creado y sustituye *DOC-EXAMPLE-BUCKET* por el nombre del depósito de S3 que está asociado a tu flota de dispositivos.
   c. Seleccione Confirmar y, a continuación, elija Siguiente.

6. En la página Configurar ajustes avanzados, mantenga los ajustes de configuración predeterminados y seleccione Siguiente.
7. En la página de revisión, selecciona Implementar

Para implementar sus componentes (AWS CLI)

1. En su computadora de desarrollo, cree un `deployment.json` archivo para definir la configuración de implementación de los componentes de SageMaker Edge Manager. Este archivo debería ser igual al siguiente ejemplo.

```
{
  "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": {
      }
    },
  }
}
```

   - En el campo `targetArn`, sustituya *`targetArn`* por el nombre de recurso de Amazon (ARN) de la cosa o grupo de cosas a la que apunte la implementación, en el siguiente formato:
     - Cosa: `arn:aws:iot:`*`region`*`:`*`account-id`*`:thing/`*`thingName`*
     - Grupo de cosas: `arn:aws:iot:`*`region`*`:`*`account-id`*`:thinggroup/`*`thingGroupName`*

- En el `merge` campo, *`device-fleet-name`*sustitúyalo por el nombre de la flota de dispositivos perimetrales que creaste. A continuación, sustituya *`DOC-EXAMPLE-BUCKET2`* por el nombre del depósito S3 asociado a su flota de dispositivos.
- Sustituya las versiones de cada componente por la última versión disponible.

2. Ejecute el siguiente comando para implementar los componentes en el dispositivo:

```
aws greengrassv2 create-deployment \
    --cli-input-json file://path/to/deployment.json
```

La implementación puede tardar varios minutos en completarse. En el siguiente paso, compruebe el registro de componentes para comprobar que la implementación se ha completado correctamente y para ver los resultados de la inferencia.

Vea los resultados de las inferencias

Tras implementar los componentes, puede ver los resultados de la inferencia en el registro de componentes de su dispositivo principal de Greengrass y en AWS IoT el cliente MQTT de la consola. AWS IoT Para suscribirse al tema sobre el que el componente publica los resultados de las inferencias, consulte. Suscríbase al tema de notificaciones

- AWS IoT Cliente MQTT: para ver los resultados que publica el componente de inferencia sobre el tema de notificaciones predeterminado, complete los siguientes pasos:

  1. En el menú de navegación de la AWS IoT consola, elija Test, MQTT test client.
  2. En Suscripciones, elija**`gg/sageMakerEdgeManager/image-classification`**.

- Registro de componentes: para ver los resultados de la inferencia en el registro de componentes, ejecute el siguiente comando en su dispositivo principal de Greengrass.

```
sudo tail -f /greengrass/v2/logs/
com.greengrass.SageMakerEdgeManager.ImageClassification.log
```

Si no puede ver los resultados de la inferencia en el registro de componentes o en el cliente MQTT, significa que la implementación ha fallado o no ha llegado al dispositivo principal. Esto puede ocurrir si el dispositivo principal no está conectado a Internet o no tiene los permisos adecuados para

ejecutar el componente. Ejecute el siguiente comando en el dispositivo principal para ver el archivo de registro del software AWS IoT Greengrass principal. Este archivo incluye registros del servicio de despliegue del dispositivo principal de Greengrass.

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

Para obtener más información, consulte Solución de problemas de inferencia de aprendizaje automático.

Tutorial: Realice una inferencia de clasificación de imágenes de muestra con Lite TensorFlow

En este tutorial se muestra cómo utilizar el componente de inferencia de clasificación de imágenes de TensorFlow Lite para realizar una inferencia de clasificación de imágenes de muestra en un dispositivo central de Greengrass. Este componente incluye las siguientes dependencias de componentes:

- TensorFlow Componente de tienda de modelos de clasificación de imágenes Lite
- TensorFlow Componente de tiempo de ejecución Lite

Al implementar este componente, descarga un modelo MobileNet v1 previamente entrenado e instala el motor de ejecución de TensorFlow Lite y sus dependencias. Este componente publica los resultados de las inferencias sobre el tema. `ml/tflite/image-classification` Para ver estos resultados de inferencia, utilice el cliente AWS IoT MQTT de la AWS IoT consola para suscribirse a este tema.

En este tutorial, implementará el componente de inferencia de muestra para realizar la clasificación de imágenes en la imagen de muestra proporcionada por. AWS IoT Greengrass Después de completar este tutorial, puede completarloTutorial: Realice una inferencia de clasificación de imágenes de muestra en imágenes de una cámara con Lite TensorFlow , que le muestra cómo modificar el componente de inferencia de muestras para realizar la clasificación de imágenes en las imágenes de una cámara local en un dispositivo central de Greengrass.

Para obtener más información sobre el aprendizaje automático en los dispositivos Greengrass, consulte. Cómo realizar la inferencia de machine learning

Temas

Requisitos previos

Necesitará lo siguiente para completar este tutorial:

- Un dispositivo central de Linux Greengrass. Si no dispone de una, consulte Tutorial: Introducción a AWS IoT Greengrass V2. El dispositivo principal debe cumplir los siguientes requisitos:
  - En los dispositivos principales de Greengrass que ejecutan Amazon Linux 2 o Ubuntu 18.04, se instala en el dispositivo la versión 2.27 o posterior de la Biblioteca C GNU (glibc).
  - En los dispositivos ARMv7L, como Raspberry Pi, las dependencias para OpenCV-Python están instaladas en el dispositivo. Ejecute el siguiente comando para instalar las dependencias.

    ```
    sudo apt-get install libopenjp2-7 libilmbase23 libopenexr-dev libavcodec-dev
     libavformat-dev libswscale-dev libv4l-dev libgtk-3-0 libwebp-dev
    ```
  - Los dispositivos Raspberry Pi que ejecutan el sistema operativo Bullseye de Raspberry Pi deben cumplir los siguientes requisitos:
    - NumPy 1.22.4 o una versión posterior instalada en el dispositivo. Raspberry Pi OS Bullseye incluye una versión anterior de NumPy, por lo que puede ejecutar el siguiente comando para actualizar NumPy el dispositivo.

      ```
      pip3 install --upgrade numpy
      ```
    - La pila de cámaras antigua habilitada en el dispositivo. El Raspberry Pi OS Bullseye incluye una nueva pila de cámaras que está habilitada de forma predeterminada y no es compatible, por lo que debes activar la pila de cámaras antigua.

      Para activar la pila de cámaras antigua

      1. Ejecute el siguiente comando para abrir la herramienta de configuración de Raspberry Pi.

         ```
         sudo raspi-config
         ```

2. Seleccione Opciones de interfaz.
3. Seleccione Cámara antigua para activar la pila de cámaras antiguas.
4. Reinicie el Raspberry Pi.

Paso 1: Suscríbase al tema de notificaciones predeterminado

En este paso, configura el cliente AWS IoT MQTT de la AWS IoT consola para ver los mensajes MQTT publicados por el componente de clasificación de imágenes de TensorFlow Lite. De forma predeterminada, el componente publica los resultados de las inferencias sobre el tema. `ml/tflite/image-classification` Suscríbase a este tema antes de implementar el componente en su dispositivo principal de Greengrass para ver los resultados de la inferencia cuando el componente se ejecute por primera vez.

Para suscribirse al tema de notificaciones predeterminado

1. En el menú de navegación de la AWS IoTconsola, elija Test, MQTT test client.
2. En Suscribirse a un tema, en el cuadro Nombre del tema, escriba**`ml/tflite/image-classification`**.
3. Elija Suscribirse.

Paso 2: Implemente el componente de clasificación de imágenes TensorFlow Lite

En este paso, implementará el componente de clasificación de imágenes TensorFlow Lite en su dispositivo principal:

Para implementar el componente de clasificación de imágenes TensorFlow Lite (consola)

1. En el menú de navegación de la AWS IoT Greengrassconsola, elija Componentes.
2. En la página Componentes, en la pestaña Componentes públicos, elija `aws.greengrass.TensorFlowLiteImageClassification`.
3. En la página aws.greengrass.TensorFlowLiteImageClassification, elija Implementar.
4. En Añadir al despliegue, elija una de las siguientes opciones:

a. Para combinar este componente con una implementación existente en el dispositivo de destino, elija Agregar a la implementación existente y, a continuación, seleccione la implementación que desee revisar.

b. Para crear una nueva implementación en el dispositivo de destino, elija Crear nueva implementación. Si tiene una implementación existente en su dispositivo, al elegir este paso se reemplaza la implementación existente.

5. En la página Especificar detalles, haga lo siguiente:

   a. En Información de implementación, introduzca o modifique el nombre descriptivo de su implementación.

   b. En Objetivos de implementación, seleccione un objetivo para su implementación y elija Siguiente. No puede cambiar el objetivo de implementación si está revisando una implementación existente.

6. En la página Seleccionar componentes, en Componentes públicos, compruebe que el `aws.greengrass.TensorFlowLiteImageClassification` componente esté seleccionado y elija Siguiente.

7. En la página Configurar componentes, conserve los valores de configuración predeterminados y seleccione Siguiente.

8. En la página Configurar ajustes avanzados, mantenga los ajustes de configuración predeterminados y seleccione Siguiente.

9. En la página de revisión, elija Implementar

Para implementar el componente de clasificación de imágenes TensorFlow Lite (AWS CLI)

1. Cree un `deployment.json` archivo para definir la configuración de despliegue del componente de clasificación de imágenes de TensorFlow Lite. Este archivo debería tener el siguiente aspecto:

```
{
  "targetArn":"targetArn",
  "components": {
    "aws.greengrass.TensorFlowLiteImageClassification": {
      "componentVersion": 2.1.0,
      "configurationUpdate": {
      }
    }
```

```
  }
}
```

- En el `targetArn` campo, *`targetArn`*sustitúyalo por el nombre de recurso de Amazon (ARN) de la cosa o grupo de cosas a la que apunte la implementación, en el siguiente formato:
  - Cosa: `arn:aws:iot:`*`region`*`:`*`account-id`*`:thing/`*`thingName`*
  - Grupo de cosas: `arn:aws:iot:`*`region`*`:`*`account-id`*`:thinggroup/`*`thingGroupName`*
- Este tutorial utiliza la versión 2.1.0 del componente. En el objeto `aws.greengrass.TensorFlowLiteObjectDetection` componente, sustituya la *`2.1.0`* por una versión diferente del componente de detección de objetos TensorFlow Lite.

2. Ejecute el siguiente comando para implementar el componente de clasificación de imágenes TensorFlow Lite en el dispositivo:

```
aws greengrassv2 create-deployment \
    --cli-input-json file://path/to/deployment.json
```

La implementación puede tardar varios minutos en completarse. En el siguiente paso, compruebe el registro de componentes para comprobar que la implementación se ha completado correctamente y para ver los resultados de la inferencia.

Paso 3: Ver los resultados de la inferencia

Tras implementar el componente, puede ver los resultados de la inferencia en el registro del componente de su dispositivo principal de Greengrass y en AWS IoT el cliente MQTT de la consola. AWS IoT Para suscribirse al tema sobre el que el componente publica los resultados de las inferencias, consulte. Paso 1: Suscríbase al tema de notificaciones predeterminado

- AWS IoTCliente MQTT: para ver los resultados que publica el componente de inferencia sobre el tema de notificaciones predeterminado, complete los siguientes pasos:

  1. En el menú de navegación de la AWS IoTconsola, elija Test, MQTT test client.
  2. En Suscripciones, elija**`ml/tflite/image-classification`**.

     Deberías ver mensajes similares a los del siguiente ejemplo.

```
{
  "timestamp": "2021-01-01 00:00:00.000000",
```

```
  "inference-type": "image-classification",
  "inference-description": "Top 5 predictions with score 0.3 or above ",
  "inference-results": [
    {
      "Label": "cougar, puma, catamount, mountain lion, painter, panther, Felis
 concolor",
      "Score": "0.5882352941176471"
    },
    {
      "Label": "Persian cat",
      "Score": "0.5882352941176471"
    },
    {
      "Label": "tiger cat",
      "Score": "0.5882352941176471"
    },
    {
      "Label": "dalmatian, coach dog, carriage dog",
      "Score": "0.5607843137254902"
    },
    {
      "Label": "malamute, malemute, Alaskan malamute",
      "Score": "0.5450980392156862"
    }
  ]
}
```

- Registro de componentes: para ver los resultados de la inferencia en el registro de componentes, ejecute el siguiente comando en su dispositivo principal de Greengrass.

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

  Debería ver resultados similares a los del siguiente ejemplo.

  ```
  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}
```

Si no puede ver los resultados de la inferencia en el registro de componentes o en el cliente MQTT, significa que la implementación ha fallado o no ha llegado al dispositivo principal. Esto puede ocurrir si el dispositivo principal no está conectado a Internet o no tiene los permisos adecuados para ejecutar el componente. Ejecute el siguiente comando en el dispositivo principal para ver el archivo de registro del software AWS IoT Greengrass principal. Este archivo incluye registros del servicio de despliegue del dispositivo principal de Greengrass.

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

Para obtener más información, consulte Solución de problemas de inferencia de aprendizaje automático.

Siguientes pasos

Si tiene un dispositivo central Greengrass con una interfaz de cámara compatible, puede completarTutorial: Realice una inferencia de clasificación de imágenes de muestra en imágenes de una cámara con Lite TensorFlow , que le muestra cómo modificar el componente de inferencia de muestras para clasificar las imágenes de una cámara.

Para explorar más a fondo la configuración del componente de inferencia de clasificación de imágenes de TensorFlow Lite de muestra, pruebe lo siguiente:

- Modifique el parámetro `InferenceInterval` de configuración para cambiar la frecuencia con la que se ejecuta el código de inferencia.
- Modifique los parámetros `ImageName` y los parámetros de `ImageDirectory` configuración en la configuración del componente de inferencia para especificar una imagen personalizada que se utilizará en la inferencia.

Para obtener información sobre la personalización de la configuración de los componentes públicos o la creación de componentes de aprendizaje automático personalizados, consulte. Personalice sus componentes de aprendizaje automático

Tutorial: Realice una inferencia de clasificación de imágenes de muestra en imágenes de una cámara con Lite TensorFlow

Este tutorial le muestra cómo utilizar el componente de inferencia de clasificación de imágenes TensorFlow Lite para realizar inferencias de clasificación de imágenes de muestra en imágenes de una cámara local en un dispositivo central de Greengrass. Este componente incluye las siguientes dependencias de componentes:

- TensorFlow Componente de tienda de modelos de clasificación de imágenes Lite
- TensorFlow Componente de tiempo de ejecución Lite

> **Note**
>
> Este tutorial permite acceder al módulo de cámara de los dispositivos Raspberry Pi o NVIDIA Jetson Nano, pero AWS IoT Greengrass es compatible con otros dispositivos en las plataformas ARMv7L, Armv8 o x86_64. Para configurar una cámara para un dispositivo diferente, consulte la documentación correspondiente a su dispositivo.

Para obtener más información sobre el aprendizaje automático en los dispositivos Greengrass, consulte. Cómo realizar la inferencia de machine learning

Temas

Requisitos previos

Para completar este tutorial, primero debe Tutorial: Realice una inferencia de clasificación de imágenes de muestra con Lite TensorFlow completarlo.

También necesitará lo siguiente:

- Un dispositivo central de Linux Greengrass con una interfaz de cámara. Este tutorial permite acceder al módulo de cámara en uno de los siguientes dispositivos compatibles:
  - Raspberry Pi con el sistema operativo Raspberry Pi (anteriormente llamado Raspbian)
  - NVIDIA Jetson Nano

  Para obtener información sobre la configuración de un dispositivo principal de Greengrass, consulte. Tutorial: Introducción a AWS IoT Greengrass V2

  El dispositivo principal debe cumplir los siguientes requisitos:
  - En los dispositivos principales de Greengrass que ejecutan Amazon Linux 2 o Ubuntu 18.04, se instala en el dispositivo la versión 2.27 o posterior de la Biblioteca C GNU (glibc).
  - En los dispositivos ARMv7L, como Raspberry Pi, las dependencias para OpenCV-Python están instaladas en el dispositivo. Ejecute el siguiente comando para instalar las dependencias.

    ```
    sudo apt-get install libopenjp2-7 libilmbase23 libopenexr-dev libavcodec-dev
     libavformat-dev libswscale-dev libv4l-dev libgtk-3-0 libwebp-dev
    ```
  - Los dispositivos Raspberry Pi que ejecutan el sistema operativo Bullseye de Raspberry Pi deben cumplir los siguientes requisitos:
    - NumPy 1.22.4 o una versión posterior instalada en el dispositivo. Raspberry Pi OS Bullseye incluye una versión anterior de NumPy, por lo que puede ejecutar el siguiente comando para actualizar NumPy el dispositivo.

      ```
      pip3 install --upgrade numpy
      ```
    - La pila de cámaras antigua habilitada en el dispositivo. El Raspberry Pi OS Bullseye incluye una nueva pila de cámaras que está habilitada de forma predeterminada y no es compatible, por lo que debes activar la pila de cámaras antigua.

      Para activar la pila de cámaras antigua

      1. Ejecute el siguiente comando para abrir la herramienta de configuración de Raspberry Pi.

```
sudo raspi-config
```

2. Seleccione Opciones de interfaz.
3. Seleccione Cámara antigua para activar la pila de cámaras antiguas.
4. Reinicie el Raspberry Pi.

- Para dispositivos Raspberry Pi o NVIDIA Jetson Nano, módulo de cámara Raspberry Pi V2:8 megapíxeles, 1080p. Para obtener información sobre cómo configurar la cámara, consulte Connecting the camera en la documentación de Raspberry Pi.

Paso 1: Configura el módulo de cámara de tu dispositivo

En este paso, instala y habilita el módulo de cámara para su dispositivo. Ejecute los siguientes comandos en el dispositivo.

Raspberry Pi (Armv7l)

1. Instale la `picamera` interfaz del módulo de cámara. Ejecute el siguiente comando para instalar el módulo de cámara y las demás bibliotecas de Python necesarias para este tutorial.

   ```
   sudo apt-get install -y python3-picamera
   ```

2. Compruebe que Picamera se ha instalado correctamente.

   ```
   sudo -u ggc_user bash -c 'python3 -c "import picamera"'
   ```

   Si el resultado no contiene errores, la validación es correcta.

   > **Note**
   >
   > Si el archivo ejecutable de Python que está instalado en su dispositivo lo es`python3.7`, `python3.7` utilícelo en lugar de `python3` para los comandos de este tutorial. Asegúrese de que la instalación de pip corresponde a la versión `python3` o `python3.7` correcta para evitar errores de dependencia.

3. Reinicie el dispositivo.

```
sudo reboot
```

4. Abra la herramienta de configuración de Raspberry Pi.

   ```
   sudo raspi-config
   ```

5. Utilice las teclas de flecha para abrir Interfacing Options (Opciones de interfaz) y habilitar la interfaz de la cámara. Si se le solicita, permita que el dispositivo se reinicie.
6. Ejecute el siguiente comando para probar la configuración de la cámara.

   ```
   raspistill -v -o test.jpg
   ```

   Se abre una ventana de vista previa en el Raspberry Pi, se guarda una imagen denominada `test.jpg` en el directorio actual y se muestra información sobre la cámara en el terminal de Raspberry Pi.
7. Ejecute el siguiente comando para crear un enlace simbólico que permita que el componente de inferencia acceda a la cámara desde el entorno virtual creado por el componente de tiempo de ejecución.

   ```
   sudo ln -s /usr/lib/python3/dist-packages/picamera "MLRootPath/
   greengrass_ml_tflite_venv/lib/python3.7/site-packages"
   ```

   El valor predeterminado de *ML RootPath* para este tutorial es. */greengrass/v2*/work/`variant.TensorFlowLite/greengrass_ml` La `greengrass_ml_tflite_venv` carpeta de esta ubicación se crea al implementar el componente de inferencia por primera vez enTutorial: Realice una inferencia de clasificación de imágenes de muestra con Lite TensorFlow .

Jetson Nano (Armv8)

1. Ejecute el siguiente comando para probar la configuración de la cámara.

   ```
   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
   ```

   Esto captura y guarda una imagen nombrada `test.jpg` en su directorio actual.

2. (Opcional) Reinicie el dispositivo. Si tienes problemas al ejecutar el `gst-launch` comando del paso anterior, es posible que reinicies el dispositivo para resolverlos.

```
sudo reboot
```

> **Note**
>
> En el caso de los dispositivos Armv8 (AArch64), como un Jetson Nano, no es necesario crear un enlace simbólico para permitir que el componente de inferencia acceda a la cámara desde el entorno virtual creado por el componente de tiempo de ejecución.

Paso 2: Compruebe su suscripción al tema de notificaciones predeterminado

EnTutorial: Realice una inferencia de clasificación de imágenes de muestra con Lite TensorFlow , configuró el cliente AWS IoT MQTT que está configurado en la AWS IoT consola para ver los mensajes MQTT publicados por el componente de clasificación de imágenes de TensorFlow Lite sobre el `ml/tflite/image-classification` tema. En la AWS IoT consola, compruebe que existe esta suscripción. Si no es así, siga los pasos descritos Paso 1: Suscríbase al tema de notificaciones predeterminado para suscribirse a este tema antes de implementar el componente en su dispositivo principal de Greengrass.

Paso 3: Modifique la configuración del componente de clasificación de imágenes de TensorFlow Lite e impleméntelo

En este paso, debe configurar e implementar el componente de clasificación de imágenes de TensorFlow Lite en su dispositivo principal:

Para configurar e implementar el componente de clasificación de imágenes de TensorFlow Lite (consola)

1. En el menú de navegación de la AWS IoT Greengrassconsola, elija Componentes.
2. En la página Componentes, en la pestaña Componentes públicos, elija `aws.greengrass.TensorFlowLiteImageClassification`.
3. En la página aws.greengrass.TensorFlowLiteImageClassification, elija Implementar.

4. En Añadir al despliegue, elija una de las siguientes opciones:

   a. Para combinar este componente con una implementación existente en el dispositivo de destino, elija Agregar a la implementación existente y, a continuación, seleccione la implementación que desee revisar.

   b. Para crear una nueva implementación en el dispositivo de destino, elija Crear nueva implementación. Si tiene una implementación existente en su dispositivo, al elegir este paso se reemplaza la implementación existente.

5. En la página Especificar detalles, haga lo siguiente:

   a. En Información de implementación, introduzca o modifique el nombre descriptivo de su implementación.

   b. En Objetivos de implementación, seleccione un objetivo para su implementación y elija Siguiente. No puede cambiar el objetivo de implementación si está revisando una implementación existente.

6. En la página Seleccionar componentes, en Componentes públicos, compruebe que el `aws.greengrass.TensorFlowLiteImageClassification` componente esté seleccionado y elija Siguiente.

7. En la página Configurar componentes, haga lo siguiente:

   a. Seleccione el componente de inferencia y elija Configurar componente.

   b. En Actualización de configuración, introduzca la siguiente actualización de configuración en el cuadro Configuración que se va a fusionar.

   ```
   {
     "InferenceInterval": "60",
     "UseCamera": "true"
   }
   ```

   Con esta actualización de configuración, el componente accede al módulo de cámara del dispositivo y realiza inferencias a partir de las imágenes tomadas por la cámara. El código de inferencia se ejecuta cada 60 segundos.

   c. Seleccione Confirmar y, a continuación, elija Siguiente.

8. En la página Configurar ajustes avanzados, mantenga los ajustes de configuración predeterminados y seleccione Siguiente.

9. En la página de revisión, elija Implementar

Para configurar e implementar el componente de clasificación de imágenes de TensorFlow Lite (AWS CLI)

1. Cree un `deployment.json` archivo para definir la configuración de despliegue del componente de clasificación de imágenes de TensorFlow Lite. Este archivo debería tener el siguiente aspecto:

```
{
  "targetArn":"targetArn",
  "components": {
    "aws.greengrass.TensorFlowLiteImageClassification": {
      "componentVersion": 2.1.0,
      "configurationUpdate": {
        "InferenceInterval": "60",
        "UseCamera": "true"
      }
    }
  }
}
```

   - En el `targetArn` campo, *`targetArn`*sustitúyalo por el nombre de recurso de Amazon (ARN) de la cosa o grupo de cosas a la que apunte la implementación, en el siguiente formato:
     - Cosa: `arn:aws:iot:`*`region`*`:`*`account-id`*`:thing/`*`thingName`*
     - Grupo de cosas: `arn:aws:iot:`*`region`*`:`*`account-id`*`:thinggroup/`*`thingGroupName`*
   - Este tutorial utiliza la versión 2.1.0 del componente. En el objeto `aws.greengrass.TensorFlowLiteImageClassification` componente, sustituya la *`2.1.0`* por una versión diferente del componente de clasificación de imágenes TensorFlow Lite.

   Con esta actualización de configuración, el componente accede al módulo de cámara del dispositivo y realiza inferencias a partir de las imágenes tomadas por la cámara. El código de inferencia se ejecuta cada 60 segundos. Sustituya los siguientes valores

2. Ejecute el siguiente comando para implementar el componente de clasificación de imágenes TensorFlow Lite en el dispositivo:

```
aws greengrassv2 create-deployment \
    --cli-input-json file://path/to/deployment.json
```

La implementación puede tardar varios minutos en completarse. En el siguiente paso, compruebe el registro de componentes para comprobar que la implementación se ha completado correctamente y para ver los resultados de la inferencia.

Paso 4: Ver los resultados de la inferencia

Tras implementar el componente, puede ver los resultados de la inferencia en el registro del componente de su dispositivo principal de Greengrass y en AWS IoT el cliente MQTT de la consola. AWS IoT Para suscribirse al tema sobre el que el componente publica los resultados de las inferencias, consulte. Paso 2: Compruebe su suscripción al tema de notificaciones predeterminado

- AWS IoTCliente MQTT: para ver los resultados que publica el componente de inferencia sobre el tema de notificaciones predeterminado, complete los siguientes pasos:

  1. En el menú de navegación de la AWS IoTconsola, elija Test, MQTT test client.
  2. En Suscripciones, elija**`ml/tflite/image-classification`**.
- Registro de componentes: para ver los resultados de la inferencia en el registro de componentes, ejecute el siguiente comando en su dispositivo principal de Greengrass.

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

Si no puede ver los resultados de la inferencia en el registro de componentes o en el cliente MQTT, significa que la implementación ha fallado o no ha llegado al dispositivo principal. Esto puede ocurrir si el dispositivo principal no está conectado a Internet o no tiene los permisos necesarios para ejecutar el componente. Ejecute el siguiente comando en el dispositivo principal para ver el archivo de registro del software AWS IoT Greengrass principal. Este archivo incluye registros del servicio de despliegue del dispositivo principal de Greengrass.

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

Para obtener más información, consulte Solución de problemas de inferencia de aprendizaje automático.

Siguientes pasos

Este tutorial le muestra cómo utilizar el componente de clasificación de imágenes TensorFlow Lite, con opciones de configuración personalizadas para clasificar imágenes de muestra en imágenes tomadas por una cámara.

Para obtener más información sobre la personalización de la configuración de los componentes públicos o la creación de componentes de aprendizaje automático personalizados, consulte Personalice sus componentes de aprendizaje automático.

Componentes

AWS IoT Greengrasslos componentes son módulos de software que se implementan en los dispositivos principales de Greengrass. Los componentes pueden representar aplicaciones, instaladores en tiempo de ejecución, bibliotecas o cualquier código que se ejecute en un dispositivo. Puede definir componentes que dependan de otros componentes. Por ejemplo, puede definir un componente que instale Python y, a continuación, definir ese componente como una dependencia de los componentes que ejecutan aplicaciones de Python. Cuando despliega sus componentes en sus flotas de dispositivos, Greengrass despliega solo los módulos de software que requieren sus dispositivos.

Temas

AWS-componentes proporcionados

AWS IoT Greengrass proporciona y mantiene componentes prediseñados que puede implementar en sus dispositivos. Estos componentes incluyen funciones (como el administrador de transmisiones), conectores AWS IoT Greengrass V1 (como CloudWatch métricas) y herramientas de desarrollo local (como la AWS IoT Greengrass CLI). Puede implementar estos componentes en sus dispositivos para que funcionen de forma independiente, o puede usarlos como dependencias en sus componentes personalizados de Greengrass.

> **Note**
>
> Varios AWS de los componentes proporcionados dependen de versiones secundarias específicas del núcleo de Greengrass. Debido a esta dependencia, es necesario actualizar estos componentes al actualizar el núcleo de Greengrass a una nueva versión secundaria. Para obtener información sobre las versiones específicas del núcleo de las que depende cada componente, consulte el tema del componente correspondiente. Para obtener más

información sobre la actualización del núcleo, consulteActualice el software AWS IoT Greengrass principal (OTA).

Cuando un componente tiene un tipo de componente genérico y Lambda, la versión actual del componente es del tipo genérico y la versión anterior del componente es del tipo Lambda.

Componente	Descripción	Tipo de componente	Sistema operativo admitido	Código abierto
Núcleo de Greengrass	El núcleo del software AWS IoT Greengrass Core. Utilice este componente para configurar y actualizar el software de sus dispositivos principales.	Núcleo	Linux, Windows	Sí
Autenticación del dispositivo cliente	Permite que los dispositivos IoT locales, denominados dispositivos cliente, se conecten al dispositivo principal.	Complemento	Linux, Windows	Sí
CloudWatch métricas	Publica métricas personalizadas	Genérico, Lambda	Linux, Windows	Sí

Componente	Descripción	Tipo de componente	Sistema operativo admitido	Código abierto
	en Amazon CloudWatch.			
AWS IoT Device Defender	Notifica a los administradores los cambios en el estado del dispositivo principal de Greengrass para identificar un comportamiento inusual.	Genérico, Lambda	Linux, Windows	Sí

Componente	Descripción	Tipo de componente	Sistema operativo admitido	Código abierto
Bobina de disco	Habilita una opción de almacenamiento persistente para los mensajes enviados desde los dispositivos principales de Greengrass a. AWS IoT Core Este componente almacenará estos mensajes salientes en el disco.	Complemento	Linux, Windows	Sí
Gestor de aplicaciones Docker	Permite AWS IoT Greengrass descargar imágenes de Docker desde Docker Hub y Amazon Elastic Container Registry (Amazon ECR).	Genérico	Linux, Windows	No

Componente	Descripción	Tipo de componente	Sistema operativo admitido	Código abierto
Conector Edge para Kinesis Video Streams	Lee las transmisiones de vídeo de las cámaras locales, publica las transmisiones en Kinesis Video Streams y las muestra en los paneles de Grafana con. AWS IoT TwinMaker	Genérico	Linux	No
Greengrass CLI	Proporciona una interfaz de línea de comandos que puede usar para crear despliegues locales e interactuar con el dispositivo principal de Greengrass y sus componentes.	Complemento	Linux, Windows	Sí

Componente	Descripción	Tipo de componente	Sistema operativo admitido	Código abierto
Detector de IP	Envía la información de conectividad al intermediario MQTT para que AWS IoT Greengrass los dispositivos cliente puedan descubrir cómo conectarse.	Complemento	Linux, Windows	Sí
Firehose	Publica datos a través de las transmisiones de entrega de Amazon Data Firehose a los destinos del. Nube de AWS	Lambda	Linux	No
Lanzador Lambda	Maneja los procesos y la configuración del entorno para las funciones Lambda.	Genérico	Linux	No

Componente	Descripción	Tipo de componente	Sistema operativo admitido	Código abierto
Gestor Lambda	Maneja la comunicación y el escalado entre procesos para las funciones Lambda.	Complemento	Linux	No
Tiempos de ejecución de Lambda	Proporciona artefactos para cada tiempo de ejecución de Lambda.	Genérico	Linux	No
Enrutador de suscripción antiguo	Administra las suscripciones para las funciones de Lambda que se ejecutan en AWS IoT Greengrass la V1.	Genérico	Linux	No

Componente	Descripción	Tipo de componente	Sistema operativo admitido	Código abierto
Consola de depuración local	Proporciona una consola local que puede usar para depurar y administrar el dispositivo principal de Greengrass y sus componentes.	Complemento	Linux, Windows	Sí
Gestor de registros	Recopila y carga registros en el dispositivo principal de Greengrass.	Complemento	Linux, Windows	Sí
Componentes de aprendizaje automático	Proporciona modelos de aprendizaje automático y código de inferencia de muestra que puede usar para realizar inferencias de aprendizaje automático en los dispositivos principales de Greengrass.	Consulte Componentes de aprendizaje automático.		

Componente	Descripción	Tipo de componente	Sistema operativo admitido	Código abierto
Adaptador de protocolo Modbus-RTU	Extrae información de dispositivos Modbus RTU locales.	Lambda	Linux	No
Emisor de telemetría Nucleus	Publica los datos de telemetría del estado del sistema recopilados desde el núcleo hasta un tema local o un tema de MQTT. AWS IoT Core	Complemento	Linux, Windows	Sí
Puente MQTT	Transmite mensajes MQTT entre dispositivos cliente, publica o suscribe localmente y. AWS IoT Greengrass AWS IoT Core	Complemento	Linux, Windows	Sí

Componente	Descripción	Tipo de componente	Sistema operativo admitido	Código abierto
Bróker MQTT 3.1.1 (Moquette)	Ejecuta un intermediario MQTT 3.1.1 que gestiona los mensajes entre los dispositivos cliente y el dispositivo principal.	Complemento	Linux, Windows	Sí
Bróker MQTT 5 (EMQX)	Ejecuta un intermediario MQTT 5 que gestiona los mensajes entre los dispositivos cliente y el dispositivo principal.	Genérico	Linux, Windows	No

Componente	Descripción	Tipo de componente	Sistema operativo admitido	Código abierto
Proveedor PKCS #11	Permite que los componentes de Greengrass accedan a una clave privada y un certificado que se almacenan de forma segura en un módulo de seguridad de hardware (HSM).	Complemento	Linux	Sí

Componente	Descripción	Tipo de componente	Sistema operativo admitido	Código abierto
Gestor secreto	Despliega secretos a partir de AWS Secrets Manager secretos para que pueda utilizar de forma segura las credenciales, como las contraseñas, en componentes personalizados del dispositivo principal de Greengrass.	Complemento	Linux, Windows	Sí

Componente	Descripción	Tipo de componente	Sistema operativo admitido	Código abierto
Tunelización segura	Permite conexiones de tunelización AWS IoT seguras que puede utilizar para establecer comunicaciones bidireccionales con los dispositivos principales de Greengrass que se encuentran detrás de firewalls restringidos.	Genérico	Linux	No

Componente	Descripción	Tipo de componente	Sistema operativo admitido	Código abierto
Gestor en la sombra	Permite la interacción con las sombras del dispositivo principal. Gestiona el almacenamiento de documentos ocultos y también la sincronización de los estados ocultos locales con el servicio AWS IoT Device Shadow.	Complemento	Linux, Windows	Sí
Amazon SNS	Publica mensajes en temas de Amazon SNS.	Lambda	Linux	No
Administrador de transmisiones	Transmite un gran volumen de datos de fuentes locales al. Nube de AWS	Genérico	Linux, Windows	No

Componente	Descripción	Tipo de componente	Sistema operativo admitido	Código abierto
Agente de Systems Manager	Administre el dispositivo principal con AWS Systems Manager, lo que le permite parchear dispositivos, ejecutar comandos y mucho más.	Genérico	Linux	No
Servicio de intercambio de fichas	Proporciona AWS credenciales que puede utilizar para interactuar con AWS los servicios.	Genérico	Linux, Windows	No
Colector IoT SiteWise OPC-UA	Recopila datos de los servidores OPC-UA.	Genérico	Linux, Windows	No
Simulador de fuente de datos IoT SiteWise OPC-UA	Ejecuta un servidor OPC-UA local que genera datos de muestra.	Genérico	Linux, Windows	No

Componente	Descripción	Tipo de componente	Sistema operativo admitido	Código abierto
SiteWise Publicador de IoT	Publica datos en la AWS nube.	Genérico	Linux, Windows	No
SiteWise Procesador IoT	Procesa los datos de los dispositivos principales de Greengrass.	Genérico	Linux, Windows	No

Núcleo de Greengrass

El componente núcleo de Greengrass (`aws.greengrass.Nucleus`) es un componente obligatorio y el requisito mínimo para ejecutar el software AWS IoT Greengrass Core en un dispositivo. Puede configurar este componente para personalizar y actualizar el software AWS IoT Greengrass Core de forma remota. Implemente este componente para configurar ajustes como el proxy, la función del dispositivo y la configuración de las AWS IoT cosas en sus dispositivos principales.

Important

Cuando la versión del componente núcleo cambia, o cuando cambias determinados parámetros de configuración, el software AWS IoT Greengrass Core, que incluye el núcleo y todos los demás componentes del dispositivo, se reinicia para aplicar los cambios.
Al implementar un componente, AWS IoT Greengrass instala las últimas versiones compatibles de todas las dependencias de ese componente. Por este motivo, es posible que las nuevas versiones con parches de los componentes públicos AWS proporcionados se implementen automáticamente en sus dispositivos principales si agrega nuevos dispositivos a un grupo de cosas o si actualiza la implementación destinada a esos dispositivos. Algunas actualizaciones automáticas, como las actualizaciones de Nucleus, pueden provocar que los dispositivos se reinicien de forma inesperada.
Para evitar actualizaciones no deseadas de un componente que se ejecuta en su dispositivo, le recomendamos que incluya directamente la versión que prefiera de ese componente al crear una implementación. Para obtener más información sobre el comportamiento de las

actualizaciones AWS IoT Greengrass del software principal, consulteActualice el software AWS IoT Greengrass principal (OTA).

Temas

Versiones

Este componente tiene las siguientes versiones:

- 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

Sistema operativo

Este componente se puede instalar en los dispositivos principales que ejecutan los siguientes sistemas operativos:

- Linux
- Windows

Para obtener más información, consulte Plataformas admitidas.

Requisitos

Los dispositivos deben cumplir ciertos requisitos para instalar y ejecutar el núcleo de Greengrass y el software AWS IoT Greengrass Core. Para obtener más información, consulte Requisitos de los dispositivos.

Se admite la ejecución del componente núcleo de Greengrass en una VPC. Para implementar este componente en una VPC, se requiere lo siguiente.

- El componente núcleo de Greengrass debe tener conectividad con AWS IoT data, AWS IoT Credentials y Amazon S3.

Dependencias

El núcleo de Greengrass no incluye ninguna dependencia de componentes. Sin embargo, varios componentes AWS proporcionados incluyen el núcleo como dependencia. Para obtener más información, consulte AWS-componentes proporcionados.

Para obtener más información sobre las dependencias de los componentes, consulte la referencia de recetas de componentes.

Descarga e instalación

Puede descargar un instalador que configura el componente núcleo de Greengrass en su dispositivo. Este instalador configura su dispositivo como un dispositivo principal de Greengrass. Hay dos tipos de instalaciones que puede realizar: una instalación rápida que crea AWS los recursos necesarios para usted o una instalación manual en la que usted mismo crea los AWS recursos. Para obtener más información, consulte Instalación del software AWS IoT Greengrass Core.

También puede seguir un tutorial para instalar el núcleo de Greengrass y explorar el desarrollo de componentes de Greengrass. Para obtener más información, consulte Tutorial: Introducción a AWS IoT Greengrass V2.

Configuración

Este componente proporciona los siguientes parámetros de configuración que puede personalizar al implementar el componente. Algunos parámetros requieren que el software AWS IoT Greengrass principal se reinicie para que surta efecto. Para obtener más información sobre por qué y cómo configurar este componente, consulteConfigurar el software AWS IoT Greengrass principal.

`iotRoleAlias`

El alias del AWS IoT rol que apunta a un rol de IAM de intercambio de tokens. El proveedor de AWS IoT credenciales asume esta función para permitir que el dispositivo principal de Greengrass interactúe con AWS los servicios. Para obtener más información, consulte Autorizar a los dispositivos principales a interactuar con AWS los servicios.

Al ejecutar el software AWS IoT Greengrass Core con la `--provision true` opción, el software proporciona un alias de rol y establece su valor en el componente core.

`interpolateComponentConfiguration`

(Opcional) Puede habilitar el núcleo de Greengrass para interpolar las variables de las recetas de los componentes en las configuraciones de los componentes y combinar las actualizaciones de configuración. Se recomienda configurar esta opción para `true` que el dispositivo principal pueda ejecutar componentes de Greengrass que usen variables de receta en sus configuraciones.

Esta función está disponible para la versión 2.6.0 y versiones posteriores de este componente.

Valor predeterminado: `false`

`networkProxy`

(Opcional) El proxy de red que se utilizará en todas las conexiones. Para obtener más información, consulte Realizar la conexión en el puerto 443 o a través de un proxy de red.

> **Important**
>
> Al implementar un cambio en este parámetro de configuración, el software AWS IoT Greengrass principal se reinicia para que el cambio surta efecto.

Este objeto contiene la siguiente información:

`noProxyAddresses`

(Opcional) Una lista separada por comas de direcciones IP o nombres de host que están exentos del proxy.

`proxy`

El proxy al que se va a conectar. Este objeto contiene la siguiente información:

`url`

La URL del servidor proxy en el formato`scheme://userinfo@host:port`.

- `scheme`— El esquema, que debe ser `http` o`https`.

  > **Important**
  >
  > Los dispositivos principales de Greengrass deben ejecutar Greengrass nucleus v2.5.0 o posterior para usar proxies HTTPS.
  > Si configura un proxy HTTPS, debe añadir el certificado de CA del servidor proxy al certificado de CA raíz de Amazon del dispositivo principal. Para obtener más información, consulte Habilite el dispositivo principal para que confíe en un proxy HTTPS.

- `userinfo`— (Opcional) La información del nombre de usuario y la contraseña. Si especifica esta información en`url`, el dispositivo principal de Greengrass ignora los `username` campos y. `password`
- `host`— El nombre de host o la dirección IP del servidor proxy.
- `port`— (Opcional) El número de puerto. Si no especificas el puerto, el dispositivo principal de Greengrass utilizará los siguientes valores predeterminados:
  - `http`— 80
  - `https`— 443

`username`

(Opcional) El nombre de usuario que autentica el servidor proxy.

`password`

(Opcional) La contraseña que autentica el servidor proxy.

`mqtt`

(Opcional) La configuración MQTT para el dispositivo principal de Greengrass. Para obtener más información, consulte Realizar la conexión en el puerto 443 o a través de un proxy de red.

> **Important**
>
> Al implementar un cambio en este parámetro de configuración, el software AWS IoT Greengrass principal se reinicia para que el cambio surta efecto.

Este objeto contiene la siguiente información:

`port`

(Opcional) El puerto que se utilizará para las conexiones MQTT.

Valor predeterminado: `8883`

`keepAliveTimeoutMs`

(Opcional) El tiempo en milisegundos que transcurre entre cada `PING` mensaje que envía el cliente para mantener activa la conexión MQTT. Este valor debe ser superior a. `pingTimeoutMs`

Predeterminado: `60000` (60 segundos)

`pingTimeoutMs`

(Opcional) La cantidad de tiempo en milisegundos que el cliente espera para recibir un `PINGACK` mensaje del servidor. Si la espera supera el tiempo de espera, el dispositivo principal cierra y vuelve a abrir la conexión MQTT. Este valor debe ser inferior a. `keepAliveTimeoutMs`

Predeterminado: `30000` (30 segundos)

`operationTimeoutMs`

(Opcional) El tiempo en milisegundos que el cliente espera a que se completen las operaciones de MQTT (como `CONNECT` o`PUBLISH`). Esta opción no se aplica a los mensajes de MQTT `PING` ni a los de Keep Live.

Predeterminado: `30000` (30 segundos)

`maxInFlightPublishes`

(Opcional) El número máximo de mensajes QoS 1 de MQTT no confirmados que pueden estar en vuelo al mismo tiempo.

Esta función está disponible para la versión 2.1.0 y versiones posteriores de este componente.

Valor predeterminado: `5`

Rango válido: valor máximo de 100

`maxMessageSizeInBytes`

(Opcional) El tamaño máximo de un mensaje MQTT. Si un mensaje supera este tamaño, el núcleo de Greengrass lo rechaza con un error.

Esta función está disponible para la versión 2.1.0 y versiones posteriores de este componente.

Predeterminado: `131072` (128 KB)

Rango válido: valor máximo de `2621440` (2,5 MB)

`maxPublishRetry`

(Opcional) El número máximo de veces que se puede volver a intentar un mensaje que no se publica. Puede especificar que se vuelva `-1` a intentar un número ilimitado de veces.

Esta función está disponible para la versión 2.1.0 y versiones posteriores de este componente.

Valor predeterminado: `100`

`spooler`

(Opcional) La configuración del spooler MQTT para el dispositivo central Greengrass. Este objeto contiene la siguiente información:

`storageType`

El tipo de almacenamiento para almacenar mensajes. Si `storageType` está establecido en`Disk`, se `pluginName` puede configurar. Puede especificar `Memory` o `Disk`.

Esta función está disponible para la versión 2.11.0 y versiones posteriores del componente núcleo de Greengrass.

> **Important**
>
> Si el spooler de MQTT `storageType` está configurado en `Disk` y desea degradar Greengrass Nucleus de la versión 2.11.x a una versión anterior, debe volver a cambiar la configuración a. `Memory` La única configuración compatible con las `storageType` versiones 2.10.x y anteriores del núcleo de Greengrass es. `Memory` Si no se siguen estas instrucciones, se puede romper la bobina. Esto provocaría que su dispositivo principal de Greengrass no pudiera enviar mensajes MQTT al. Nube de AWS

Valor predeterminado: `Memory`

`pluginName`

(Opcional) El nombre del componente del complemento. Este componente solo se usará si `storageType` está configurado en`Disk`. Esta opción está predeterminada `aws.greengrass.DiskSpooler` y usará la proporcionada por Bobina de disco Greengrass.

Esta función está disponible para la versión 2.11.0 y versiones posteriores del componente núcleo de Greengrass.

Valor predeterminado: `"aws.greengrass.DiskSpooler"`

`maxSizeInBytes`

(Opcional) El tamaño máximo de la memoria caché en la que el dispositivo principal almacena los mensajes MQTT sin procesar. Si la caché está llena, se rechazan los mensajes nuevos.

Predeterminado: `2621440` (2,5 MB)

`keepQos0WhenOffline`

(Opcional) Puede agrupar los mensajes QoS 0 de MQTT que el dispositivo principal recibe mientras está fuera de línea. Si estableces esta opción`true`, el dispositivo principal almacena los mensajes de QoS 0 que no puede enviar mientras está desconectado. Si estableces esta opción`false`, el dispositivo principal descartará estos mensajes. El dispositivo principal siempre envía los mensajes de QoS 1 a menos que la bobina esté llena.

Valor predeterminado: `false`

`version`

(Opcional) La versión de MQTT. Puede especificar `mqtt3` o `mqtt5`.

Esta función está disponible para la versión 2.10.0 y versiones posteriores del componente núcleo de Greengrass.

Valor predeterminado: `mqtt5`

`receiveMaximum`

(Opcional) El número máximo de paquetes QoS1 no confirmados que el bróker puede enviar.

Esta función está disponible para la versión 2.10.0 y versiones posteriores del componente núcleo de Greengrass.

Valor predeterminado: `100`

`sessionExpirySeconds`

(Opcional) La cantidad de tiempo en segundos que puede solicitar para que dure una sesión desde IoT Core. El valor predeterminado es el tiempo máximo admitido por AWS IoT Core.

Esta función está disponible para la versión 2.10.0 y versiones posteriores del componente núcleo de Greengrass.

Valor predeterminado: `604800 (7 days)`

`minimumReconnectDelaySeconds`

(Opcional) Una opción para el comportamiento de reconexión. El tiempo mínimo en segundos para que MQTT se vuelva a conectar.

Esta función está disponible para la versión 2.10.0 y versiones posteriores del componente núcleo de Greengrass.

Valor predeterminado: `1`

`maximumReconnectDelaySeconds`

(Opcional) Una opción para el comportamiento de reconexión. El tiempo máximo en segundos que tarda MQTT en volver a conectarse.

Esta función está disponible para la versión 2.10.0 y versiones posteriores del componente núcleo de Greengrass.

Valor predeterminado: `120`

`minimumConnectedTimeBeforeRetryResetSeconds`

(Opcional) Una opción para el comportamiento de reconexión. Cantidad de tiempo en segundos que una conexión debe estar activa antes de que el retraso en el reintento se restablezca al mínimo.

Esta función está disponible para la versión 2.10.0 y versiones posteriores del componente núcleo de Greengrass.

Valor predeterminado: `30`

`jvmOptions`

(Opcional) Las opciones de JVM que se utilizarán para ejecutar el software principal. AWS IoT Greengrass Para obtener información sobre las opciones de JVM recomendadas para ejecutar el software AWS IoT Greengrass Core, consulte. Controle la asignación de memoria con las opciones de JVM

> **Important**
>
> Al implementar un cambio en este parámetro de configuración, el software AWS IoT Greengrass principal se reinicia para que el cambio surta efecto.

`iotDataEndpoint`

El punto final AWS IoT de datos para su. Cuenta de AWS

Cuando ejecuta el software AWS IoT Greengrass Core con la `--provision true` opción, el software obtiene sus datos y credenciales de los puntos finales AWS IoT y los coloca en el componente núcleo.

`iotCredEndpoint`

El punto final de AWS IoT credenciales para su. Cuenta de AWS

Cuando ejecuta el software AWS IoT Greengrass Core con la `--provision true` opción, el software obtiene sus datos y credenciales de los puntos finales AWS IoT y los establece en el componente núcleo.

`greengrassDataPlaneEndpoint`

Esta función está disponible en la versión 2.7.0 y versiones posteriores de este componente.

Para obtener más información, consulte Utilice un certificado de dispositivo firmado por una entidad emisora de certificados privada.

`greengrassDataPlanePort`

Esta función está disponible en la versión 2.0.4 y versiones posteriores de este componente.

(Opcional) El puerto que se utilizará para las conexiones del plano de datos. Para obtener más información, consulte Realizar la conexión en el puerto 443 o a través de un proxy de red.

> **Important**
>
> Debe especificar un puerto en el que el dispositivo pueda realizar conexiones salientes. Si especificas un puerto que está bloqueado, el dispositivo no podrá conectarse AWS IoT Greengrass para recibir despliegues.

Puede elegir entre las siguientes opciones:

- `443`
- `8443`

Valor predeterminado: `8443`

`awsRegion`

El que se Región de AWS debe usar.

`runWithDefault`

El usuario del sistema que se utilizará para ejecutar los componentes.

> **Important**
>
> Al implementar un cambio en este parámetro de configuración, el software AWS IoT Greengrass principal se reinicia para que el cambio surta efecto.

Este objeto contiene la siguiente información:

`posixUser`

El nombre o ID del usuario del sistema y, opcionalmente, del grupo de sistemas que el dispositivo principal utiliza para ejecutar los componentes genéricos y de Lambda. Especifique el usuario y el grupo separados por dos puntos (`:`) con el siguiente formato: `user:group`. El grupo es opcional. Si no especifica un grupo, el software AWS IoT Greengrass Core utiliza el grupo principal para el usuario. Por ejemplo, puede especificar `ggc_user` o `ggc_user:ggc_group`. Para obtener más información, consulte Configure el usuario que ejecuta los componentes.

Al ejecutar el instalador del software AWS IoT Greengrass Core con la `--component-default-user` *`ggc_user`*`:`*`ggc_group`* opción, el software establece este parámetro en el componente core.

`windowsUser`

Esta función está disponible en la versión 2.5.0 y versiones posteriores de este componente.

El nombre del usuario de Windows que se va a utilizar para ejecutar este componente en los dispositivos principales de Windows. El usuario debe estar en todos los dispositivos principales de Windows y su nombre y contraseña deben almacenarse en la instancia del administrador de credenciales de la LocalSystem cuenta. Para obtener más información, consulte Configure el usuario que ejecuta los componentes.

Al ejecutar el instalador de software AWS IoT Greengrass Core con la `--component-default-user` *`ggc_user`* opción, el software establece este parámetro en el componente core.

`systemResourceLimits`

Esta función está disponible en la versión 2.4.0 y versiones posteriores de este componente. AWS IoT Greengrass actualmente no admite esta función en los dispositivos principales de Windows.

Los límites de recursos del sistema se aplicarán de forma predeterminada a los procesos de componentes de Lambda genéricos y no contenerizados. Puede anular los límites de recursos del sistema para los componentes individuales al crear una implementación. Para obtener más información, consulte Configure los límites de recursos del sistema para los componentes.

Este objeto contiene la siguiente información:

`cpus`

La cantidad máxima de tiempo de CPU que los procesos de cada componente pueden utilizar en el dispositivo principal. El tiempo total de CPU de un dispositivo principal equivale a la cantidad de núcleos de CPU del dispositivo. Por ejemplo, en un dispositivo principal con 4 núcleos de CPU, puede establecer este valor 2 para limitar los procesos de cada componente al 50 por ciento de uso de cada núcleo de CPU. En un dispositivo con 1 núcleo de CPU, puede establecer este valor `0.25` para limitar los procesos de cada componente al 25 por ciento de uso de la CPU. Si estableces este valor en un número mayor que el número de núcleos de la CPU, el software AWS IoT Greengrass Core no limita el uso de la CPU de los componentes.

`memory`

La cantidad máxima de RAM (en kilobytes) que los procesos de cada componente pueden utilizar en el dispositivo principal.

`s3EndpointType`

(Opcional) El tipo de punto final S3. Este parámetro solo se aplicará a la región EE.UU. Este (Virginia del Norte`us-east-1`) (). Se ignorará la configuración de este parámetro desde cualquier otra región. Puede elegir entre las siguientes opciones:

- `REGIONAL`— El cliente S3 y la URL prefirmada utilizan el punto final regional.
- `GLOBAL`— El cliente S3 y la URL prefirmada utilizan el punto final heredado.

Valor predeterminado: `GLOBAL`

`logging`

(Opcional) La configuración de registro del dispositivo principal. Para obtener más información sobre cómo configurar y usar los registros de Greengrass, consulte. Supervisar AWS IoT Greengrass registros

Este objeto contiene la siguiente información:

`level`

(Opcional) El nivel mínimo de mensajes de registro que se van a generar.

Elija uno de los siguientes niveles de registro, que se muestran aquí en orden de niveles:

- `DEBUG`

- `INFO`
- `WARN`
- `ERROR`

Valor predeterminado: `INFO`

`format`

(Opcional) El formato de datos de los registros. Puede elegir entre las siguientes opciones:

- `TEXT`— Elija esta opción si desea ver los registros en forma de texto.
- `JSON`— Elija esta opción si desea ver los registros con el comando logs de la CLI de Greengrass o interactuar con los registros mediante programación.

Valor predeterminado: `TEXT`

`outputType`

(Opcional) El tipo de salida de los registros. Puede elegir entre las siguientes opciones:

- `FILE`— El software AWS IoT Greengrass Core envía los registros a los archivos del directorio que especifiqueoutputDirectory.
- `CONSOLE`— El software AWS IoT Greengrass Core imprime los registros enstdout. Seleccione esta opción para ver los registros a medida que los imprime el dispositivo principal.

Valor predeterminado: `FILE`

`fileSizeKB`

(Opcional) El tamaño máximo de cada archivo de registro (en kilobytes). Cuando un archivo de registro supera este tamaño máximo de archivo, el software AWS IoT Greengrass principal crea un nuevo archivo de registro.

Este parámetro solo se aplica cuando se especifica `FILE` paraoutputType.

Valor predeterminado: `1024`

`totalLogsSizeKB`

(Opcional) El tamaño total máximo de los archivos de registro (en kilobytes) de cada componente, incluido el núcleo de Greengrass. Los archivos de registro del núcleo de Greengrass también incluyen registros de los componentes del complemento. Cuando el

tamaño total de los archivos de registro de un componente supera este tamaño máximo, el software AWS IoT Greengrass Core elimina los archivos de registro más antiguos de ese componente.

Este parámetro equivale al parámetro de límite de espacio en disco del componente del administrador de registros (`diskSpaceLimit`), que puede especificar para el núcleo (sistema) de Greengrass y para cada componente. El software AWS IoT Greengrass Core utiliza el mínimo de los dos valores como tamaño de registro total máximo para el núcleo de Greengrass y cada componente.

Este parámetro se aplica solo cuando se especifica `FILE` para`outputType`.

Valor predeterminado: `10240`

`outputDirectory`

(Opcional) El directorio de salida de los archivos de registro.

Este parámetro solo se aplica cuando se especifica `FILE` para`outputType`.

Predeterminado:*`/greengrass/v2`*`/logs`, donde *`/greengrass/v2`* está la carpeta AWS IoT Greengrass raíz.

`fleetstatus`

Este parámetro está disponible en la versión 2.1.0 y versiones posteriores de este componente.

(Opcional) La configuración del estado de la flota para el dispositivo principal.

Este objeto contiene la siguiente información:

`periodicStatusPublishIntervalSeconds`

(Opcional) El tiempo (en segundos) transcurrido desde que el dispositivo principal publica el estado del dispositivo en el Nube de AWS.

Mínimo: `86400` (24 horas)

Predeterminado: `86400` (24 horas)

`telemetry`

(Opcional) La configuración de telemetría del estado del sistema para el dispositivo principal. Para obtener más información sobre las métricas de telemetría y cómo actuar en función de los datos

de telemetría, consulte. Recopile datos de telemetría del estado del sistema de los dispositivos principales AWS IoT Greengrass

Este objeto contiene la siguiente información:

`enabled`

(Opcional) Puede activar o desactivar la telemetría.

Valor predeterminado: `true`

`periodicAggregateMetricsIntervalSeconds`

(Opcional) El intervalo (en segundos) durante el que el dispositivo principal agrega las métricas.

Si establece este valor por debajo del valor mínimo admitido, el núcleo utilizará el valor predeterminado en su lugar.

Mínimo: `3600`

Valor predeterminado: `3600`

`periodicPublishMetricsIntervalSeconds`

(Opcional) El intervalo de tiempo (en segundos) entre el que el dispositivo principal publica las métricas de telemetría en el. Nube de AWS

Si establece este valor por debajo del valor mínimo admitido, el núcleo utilizará el valor predeterminado en su lugar.

Mínimo: `86400`

Valor predeterminado: `86400`

`deploymentPollingFrequencySeconds`

(Opcional) El período en segundos durante el que se deben sondear las notificaciones de despliegue.

Valor predeterminado: `15`

`componentStoreMaxSizeBytes`

(Opcional) El tamaño máximo en disco del almacén de componentes, que incluye las recetas y los artefactos de los componentes.

Predeterminado: `10000000000` (10 GB)

`platformOverride`

(Opcional) Un diccionario de atributos que identifica la plataforma del dispositivo principal. Úselo para definir los atributos de plataforma personalizados que las recetas de componentes pueden usar para identificar el ciclo de vida y los artefactos correctos del componente. Por ejemplo, puede definir un atributo de capacidad de hardware para implementar solo el conjunto mínimo de artefactos necesarios para la ejecución de un componente. Para obtener más información, consulte el parámetro de plataforma del manifiesto en la receta del componente.

También puede usar este parámetro para anular los atributos `os` y de `architecture` plataforma del dispositivo principal.

`httpClient`

Este parámetro está disponible en la versión 2.5.0 y versiones posteriores de este componente.

(Opcional) La configuración del cliente HTTP para el dispositivo principal. Estas opciones de configuración se aplican a todas las solicitudes HTTP realizadas por este componente. Si un dispositivo principal funciona en una red más lenta, puede aumentar estos tiempos de espera para evitar que se agoten los tiempos de espera de las solicitudes HTTP.

Este objeto contiene la siguiente información:

`connectionTimeoutMs`

(Opcional) El tiempo (en milisegundos) que se debe esperar a que se abra una conexión antes de que se agote el tiempo de espera de la solicitud de conexión.

Predeterminado: `2000` (2 segundos)

`socketTimeoutMs`

(Opcional) La cantidad de tiempo (en milisegundos) que se debe esperar a que los datos se transfieran a través de una conexión abierta antes de que se agote el tiempo de espera de la conexión.

Predeterminado: `30000` (30 segundos)

Example Ejemplo: actualización de la combinación de configuraciones

```
{
```

```
  "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"
  }
}
```

Archivo de registro local

Este componente utiliza el siguiente archivo de registro.

Linux

```
/greengrass/v2/logs/greengrass.log
```

Windows

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

Para ver los registros de este componente

- Ejecute el siguiente comando en el dispositivo principal para ver el archivo de registro de este componente en tiempo real. Sustituya */greengrass/v2* o *C:\greengrass\v2* por la ruta a la carpeta AWS IoT Greengrass raíz.

  Linux

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

Windows (PowerShell)

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

Registros de cambios

En la siguiente tabla se describen los cambios en cada versión del componente.

Versión	Cambios
2.12.6	Mejoras y correcciones de errores • Corrige un problema que provocaba un bloqueo al arrancar algunos procesadores ARMv8, incluido el Jetson Nano.
2.12.5	Mejoras y correcciones de errores • Soluciona un problema que provocaba que, en ocasiones, la reversión de la implementación se bloqueara al revertir un componente previamen te dañado con dependencias sólidas. • Soluciona un problema por el que el núcleo no publicaba actualiza ciones de estado tras el aprovisionamiento de la flota. • Añade reintentos a la `GetDeploymentConfiguration` API después de recibir 404 errores.
2.12.4	Mejoras y correcciones de errores • Soluciona un problema por el que el núcleo entra en un punto muerto durante el arranque en algunos dispositivos Linux.
2.12.3	**Warning** Esta versión ya no está disponible. Las mejoras de esta versión están disponibles en versiones posteriores de este componente. Mejoras y correcciones de errores

Versión	Cambios
	• Soluciona un problema que provocaba que el núcleo no mostrara el estado correcto de los componentes después del relanzamiento del núcleo y durante la recuperación de los componentes. • Corrección de errores y mejoras generales.
2.12.2	Mejoras y correcciones de errores • Soluciona un problema por el que los registros antiguos no se limpiaban correctamente. • Corrección de errores y mejoras generales.
2.12.1	Mejoras y correcciones de errores • Soluciona un problema por el que el núcleo podía duplicar las suscripci ones de MQTT por temas de despliegue, lo que provocaba más registros y publicaciones de MQTT.
2.12.0	Nuevas características • Le permite ejecutar los pasos del ciclo de vida del arranque como parte de una implementación de reversión.
2.11.3	Mejoras y correcciones de errores • Soluciona un problema en el núcleo que podía iniciar incorrectamente un componente cuando fallaban sus dependencias. Nuevas características • Añade un tipo de punto final s3 configurable.
2.11.2	Mejoras y correcciones de errores • Soluciona un problema en el cliente Core MQTT 5 que podía aparecer desconectado cuando se utilizaban un gran número de suscripciones (> 50). • Añade un reintento por el error TCP del docker dial.

Versión	Cambios
2.11.1	Mejoras y correcciones de errores • Soluciona un problema por el que el núcleo no se iniciaba si se producía un error en una tarea de arranque y el archivo de metadatos de despliegue estaba dañado. • Soluciona un problema por el que los componentes Lambda bajo demanda no se incluyen en las actualizaciones del estado de la implementación. • Añade compatibilidad con los identificadores de política de autorización duplicados.
2.11.0	Nuevas características • Permite cancelar una implementación local. • Le permite configurar una política de gestión de errores para una implementación local. • Añade compatibilidad con un complemento de spooler de discos.
2.10.3	Mejoras y correcciones de errores • Soluciona un problema por el que Greengrass no se suscribe a las notificaciones de despliegue cuando utiliza el proveedor PKCS #11.
2.10.2	Mejoras y correcciones de errores • Permite analizar los ciclos de vida de los componentes sin distinguir entre mayúsculas y minúsculas. • Soluciona un problema por el que la variable PATH del entorno no se recreaba correctamente. • Corrige la codificación URI del proxy para los componentes, incluido el administrador de flujos para los nombres de usuario con caracteres especiales.

Versión	Cambios
2.10.1	Mejoras y correcciones de errores • Corrige un problema que podía provocar un bloqueo al arrancar algunos procesadores ARMv8, incluido el Jetson Nano. • Greengrass ya no cierra el estándar de un componente, lo que revierte el comportamiento anterior a la versión 2.10.0
2.10.0	Nuevas características • Añade `interpolateComponentConfiguration` compatibilidad con la expresión regular vacía. Greengrass ahora interpola desde el objeto de configuración raíz. • Añade compatibilidad con MQTT5. • Añade un mecanismo para cargar los componentes del plugin rápidamente sin necesidad de escanearlos. • Permite a Greengrass ahorrar espacio en disco al eliminar las imágenes de Docker no utilizadas. Mejoras y correcciones de errores • Soluciona un problema por el que la reversión dejaba determinados valores de configuración de una implementación en su lugar. • Soluciona un problema por el que el núcleo de Greengrass valida una secuencia de AWS dominios en puntos finales de AWS datos y sin credenciales personalizados. • Actualiza la resolución de dependencias multigrupo para volver a resolver todas las dependencias de los grupos mediante la Nube de AWS negociación, en lugar de limitarlas a la versión activa. Esta actualización también elimina el código de error de implementación. `INSTALLED_COMPONENT_NOT_FOUND` • Actualiza el núcleo de Greengrass para omitir la descarga de imágenes de Docker cuando ya existen localmente. • Actualiza el núcleo de Greengrass para reiniciar el paso de instalación de un componente antes de que se agote el tiempo de espera. • Correcciones y mejoras menores adicionales.

Versión	Cambios
2.9.6	Mejoras y correcciones de errores • Corrige un problema por el que una implementación de Greengras s fallaba con el error LAUNCH_DIRECTORY_CORRUPTED y un posterior reinicio del dispositivo no podía iniciar Greengrass. Este error puede producirse al mover el dispositivo Greengrass entre varios grupos de cosas con implementaciones que requieren el reinicio de Greengras s.
2.9.5	Nuevas características • Añade compatibilidad con la verificación de firmas del software Greengrass Nucleus. Mejoras y correcciones de errores • Soluciona un problema por el que una implementación falla cuando la región de metadatos de la receta local no coincide con la región de lanzamiento del núcleo de Greengrass. El núcleo de Greengrass ahora renegocia con la nube cuando esto ocurre. • Soluciona un problema por el que el spooler de mensajes de MQTT se llenaba y nunca eliminaba los mensajes. • Correcciones y mejoras menores adicionales.
2.9.4	Mejoras y correcciones de errores • Comprueba si hay un mensaje nulo antes de eliminar los mensajes de QOS 0. • Trunca los valores de detalle del estado del trabajo si superan el límite de 1024 caracteres. • Actualiza el script de arranque para Windows para que lea correctam ente la ruta raíz de Greengrass si esa ruta incluye espacios. • Actualiza la suscripción para AWS IoT Core que se eliminen los mensajes de los clientes si no se envió la respuesta de la suscripción. • Garantiza que el núcleo cargue su configuración a partir de archivos de respaldo cuando el archivo de configuración principal esté dañado o falte.

Versión	Cambios
2.9.3	Mejoras y correcciones de errores • Garantiza que los ID de los clientes de MQTT no estén duplicados. • Añade una lectura y escritura de archivos más robustas para evitar la corrupción y recuperarse de ella. • Reintenta extraer la imagen del docker en caso de errores específicos relacionados con la red. • Añade la `noProxyAddresses` opción de conexión MQTT.
2.9.2	Mejoras y correcciones de errores • Soluciona un problema por el que la configuración `interpolateComponentConfiguration` no se aplicaba a una implementación en curso. • Utiliza OSHI para enumerar todos los procesos secundarios.
2.9.1	Mejoras y correcciones de errores • Añade una corrección por la que Greengrass se reinicia si una implementación elimina un componente del complemento.
2.9.0	Nuevas características • Añade la posibilidad de crear subdespliegues que reintenten los despliegues con un subconjunto de dispositivos más pequeño. Esta función crea una forma más eficiente de probar y resolver las implementaciones fallidas. Mejoras y correcciones de errores • Mejora la compatibilidad con los sistemas que no tienen `useraddgroupadd`, y`usermod`. • Mejoras y correcciones menores adicionales.

Versión	Cambios
2.8.1	Mejoras y correcciones de errores • Soluciona un problema por el que los códigos de error de despliegue no se generaban correctamente a partir de errores de la API de Greengras s. • Soluciona un problema que provocaba que las actualizaciones del estado de la flota enviaran información inexacta cuando un componente alcanzaba un `ERRORED` estado determinado durante un despliegue. • Soluciona un problema por el que las implementaciones no se podían completar cuando Greengrass tenía más de 50 suscripciones existente s.

Versión	Cambios
2.8.0	Nuevas características • Actualiza el núcleo de Greengrass para informar sobre una respuesta sobre el estado del despliegue que incluye códigos de error detallados cuando se produce un problema al implementar los componentes en un dispositivo central. Para obtener más información, consulte Códigos de error de implementación detallados. • Actualiza el núcleo de Greengrass para informar de una respuesta al estado de salud de un componente que incluye códigos de error detallados cuando un componente entra en el estado `BROKEN` o`ERRORED`. Para obtener más información, consulte Códigos de estado detallados de los componentes. • Amplía los campos de los mensajes de estado para mejorar la informaci ón de disponibilidad de los dispositivos en la nube. • Mejora la solidez del servicio de estado de la flota. Mejoras y correcciones de errores • Permite volver a instalar un componente dañado cuando cambia su configuración. • Soluciona un problema por el que el reinicio del núcleo durante la implementación de bootstrap provoca un error en la implementación. • Soluciona un problema en Windows por el que la instalación falla cuando una ruta raíz contiene espacios. • Soluciona un problema por el que un componente que se apagaba durante una implementación utilizaba el script de apagado de la nueva versión. • Varias mejoras de apagado. • Correcciones y mejoras menores adicionales.

Versión	Cambios
2.7.0	Nuevas características • Actualiza el núcleo de Greengrass para enviar actualizaciones de estado a la AWS IoT Greengrass nube cuando el dispositivo principal aplica un despliegue local. • Añade compatibilidad con los certificados de cliente firmados por una autoridad de certificación (CA) personalizada, en la que la CA no esté registrada. AWS IoT Para utilizar esta función, puede establecer la nueva opción `greengrassDataPlaneEndpoint` de configuración en `iotdata`. Para obtener más información, consulte Utilice un certificado de dispositivo firmado por una entidad emisora de certificados privada. Mejoras y correcciones de errores • Soluciona un problema por el que el núcleo de Greengrass retrasa un despliegue en determinadas situaciones cuando el núcleo se detiene o se reinicia. El núcleo ahora reanuda su despliegue después de que se reinicie. • Actualiza el instalador de Greengrass para que respete el `--start` argumento cuando se especifica configurar el software como un servicio del sistema. • Actualiza el comportamiento SubscribeToComponentUpdates para establecer el ID de despliegue en los casos en los que el núcleo actualiza un componente. • Correcciones y mejoras menores adicionales.

Versión	Cambios
2.6.0	Nuevas características • Añade compatibilidad con los caracteres comodín de MQTT al suscribirse a temas locales de publicación/suscripción. Para obtener más información, consulte Publicar/suscribir mensajes locales y Subscribe ToTopic. • Añade compatibilidad con variables de receta en las configuraciones de componentes, distintas de la variable de receta. *component_dependency_name* :configuration: *json_pointer* Puede usar estas variables de receta al definir un componente DefaultConfiguration en una receta o al configurar un componente en una implementación. Para habilitar esta función, defina la opción de ComponentConfiguration configuración de interpolación en. true Para obtener más información, consulte Variables de receta y Utilice variables de receta en las actualizaciones de fusión. • Añade una compatibilidad total con el * carácter comodín en las políticas de autorización de la comunicación entre procesos (IPC). Ahora puede especificar el * carácter de una cadena de recursos para que coincida con cualquier combinación de caracteres. Para obtener más información, consulte Comodín en las políticas de autorización. • Añade compatibilidad con componentes personalizados para llamar a las operaciones de IPC que utiliza la CLI de Greengrass. Puede usar estas operaciones de IPC para administrar las implementaciones locales, ver los detalles de los componentes y generar una contraseña que podrá usar para iniciar sesión en la consola de depuración local. Para obtener más información, consulte IPC: administrar las implementaciones y los componentes locales. Mejoras y correcciones de errores • Soluciona un problema por el que los componentes dependientes no reaccionaban cuando sus dependencias principales se reiniciaban o cambiaban de estado en determinadas situaciones. • Mejora los mensajes de error que el dispositivo principal envía al servicio AWS IoT Greengrass en la nube cuando se produce un error en la implementación.

Versión	Cambios
	• Soluciona un problema por el que el núcleo de Greengrass desplegab a una cosa dos veces en determinadas situaciones cuando el núcleo se reiniciaba. • Correcciones y mejoras menores adicionales. Para obtener más información, consulte las versiones en GitHub.
2.5.6	Nuevas características • Añade compatibilidad con los módulos de seguridad de hardware que utilizan claves ECC. Puede utilizar un módulo de seguridad de hardware (HSM) para almacenar de forma segura la clave privada y el certificado del dispositivo. Para obtener más información, consulte Integración de la seguridad de hardware. Mejoras y correcciones de errores • Soluciona un problema por el que la implementación nunca se completa cuando se implementa un componente con un script de instalación defectuoso en determinadas situaciones. • Mejora el rendimiento durante el inicio. • Mejoras y correcciones menores adicionales.

Versión	Cambios
2.5.5	Nuevas características • Añade la variable de `GG_ROOT_CA_PATH` entorno para los componentes, de forma que pueda acceder al certificado raíz de la entidad emisora de certificados (CA) en los componentes personalizados. Mejoras y correcciones de errores • Añade compatibilidad con dispositivos Windows que utilizan un idioma de visualización distinto del inglés. • Actualiza la forma en que el núcleo de Greengrass analiza los argumentos booleanos del instalador, de modo que puede especificar un argumento booleano sin un valor booleano para especificar un valor. `true` Por ejemplo, ahora puede especificar en `--provision` lugar de instalar con el aprovisionamiento automático de recursos. `--provision true` • Soluciona un problema por el que el dispositivo principal no informaba de su estado al servicio en la AWS IoT Greengrass nube después del aprovisionamiento en determinadas situaciones. • Correcciones y mejoras menores adicionales.
2.5.4	Mejoras y correcciones de errores • Corrección de errores y mejoras generales.
2.5.3	Nuevas características • Añade compatibilidad con la integración de la seguridad del hardware. Puede utilizar un módulo de seguridad de hardware (HSM) para almacenar de forma segura la clave privada y el certificado del dispositivo. Para obtener más información, consulte Integración de la seguridad de hardware. Mejoras y correcciones de errores • Corrige un problema con las excepciones de tiempo de ejecución cuando el núcleo establece conexiones con AWS IoT Core MQTT.

Versión	Cambios
2.5.2	Mejoras y correcciones de errores • Soluciona un problema por el que, una vez actualizado el núcleo de Greengrass, el servicio de Windows no se puede iniciar de nuevo después de detenerlo o reiniciar el dispositivo.
2.5.1	**Warning** Esta versión ya no está disponible. Las mejoras de esta versión están disponibles en versiones posteriores de este componente. Mejoras y correcciones de errores • Añade compatibilidad con las versiones de 32 bits del entorno de ejecución de Java (JRE) en Windows. • Cambia el comportamiento de eliminación de grupos de cosas en los dispositivos principales cuya AWS IoT política no concede el `greengrass:ListThingGroupsForCoreDevice` permiso. Con esta versión, la implementación continúa, registra una advertencia y no elimina componentes al quitar el dispositivo principal de un grupo de cosas. Para obtener más información, consulte Implemente AWS IoT Greengrass componentes en los dispositivos. • Corrige un problema con las variables de entorno del sistema que el núcleo de Greengrass pone a disposición de los procesos de los componentes de Greengrass. Ahora puede reiniciar un componente para que utilice las variables de entorno del sistema más recientes.

Versión	Cambios
2.5.0	Nuevas características • Añade compatibilidad con los dispositivos principales que ejecutan Windows. • Cambia el comportamiento de la eliminación de grupos de cosas. Con esta versión, puede eliminar un dispositivo principal de un grupo de cosas para desinstalar los componentes de ese grupo de cosas en la siguiente implementación. Como resultado de este cambio, la AWS IoT política de un dispositivo principal debe tener el `greengrass:ListThingGroupsForCoreDevice` permiso. Si usó el instalador de software AWS IoT Greengrass Core para aprovisionar recursos, la AWS IoT política predeterminada lo permite `greengrass:*`, e incluye este permiso. Para obtener más información, consulte Autenticación y autorización de dispositivos en AWS IoT Greengrass. • Añade compatibilidad con las configuraciones de proxy HTTPS. Para obtener más información, consulte Realizar la conexión en el puerto 443 o a través de un proxy de red. • Añade el nuevo parámetro `windowsUser` de configuración. Puede usar este parámetro para especificar el usuario predeterminado que se utilizará para ejecutar los componentes en un dispositivo principal de Windows. Para obtener más información, consulte Configure el usuario que ejecuta los componentes. • Agrega las nuevas opciones de `httpClient` configuración que puede usar para personalizar los tiempos de espera de las solicitudes HTTP a fin de mejorar el rendimiento en redes lentas. Para obtener más información, consulte el parámetro de configuración HttpClient. Mejoras y correcciones de errores • Corrige la opción del ciclo de vida de arranque para reiniciar el dispositivo principal desde un componente. • Añade compatibilidad con guiones en las variables de receta. • Corrige la autorización de IPC para los componentes de la función Lambda bajo demanda.

Versión	Cambios
	• Mejora los mensajes de registro y cambia los registros no críticos de un `DEBUG` nivel `INFO` a otro, por lo que los registros son más útiles. • Elimina el `iot:DescribeCertificate` permiso de la función de intercambio de fichas predeterminada que el núcleo de Greengrass crea al instalar el software AWS IoT Greengrass Core con aprovisionamiento automático. El núcleo de Greengrass no utiliza este permiso. • Corrige un problema por el que el script de aprovisionamiento automático no requería el `iam:GetPolicy` permiso si estaba `iam:CreatePolicy` disponible para la misma política. • Correcciones y mejoras menores adicionales.

Versión	Cambios
2.4.0	Nuevas características • Añade compatibilidad con los límites de recursos del sistema. Puede configurar la cantidad máxima de uso de CPU y RAM que los procesos de cada componente pueden usar en el dispositivo principal. Para obtener más información, consulte Configure los límites de recursos del sistema para los componentes. • Añade operaciones de IPC para pausar y reanudar los component es. Para obtener más información, consulte PauseComponent y ResumeComponent. • Añade compatibilidad con el aprovisionamiento de complementos. Puede especificar un archivo JAR para que se ejecute durante la instalación a fin de aprovisionar AWS los recursos necesarios para un dispositivo principal de Greengrass. El núcleo de Greengrass incluye una interfaz que puede implementar para desarrollar complementos de aprovisionamiento personalizados. Para obtener más información, consulte Instale el software AWS IoT Greengrass principal con aprovisio namiento de recursos personalizado. • Agrega el `thing-name-policy` argumento opcional al instalador del software AWS IoT Greengrass Core. Puede usar esta opción para especificar una AWS IoT política existente o personalizada al instalar el software AWS IoT Greengrass Core con aprovisionamiento automático de recursos. Mejoras y correcciones de errores • Actualiza la configuración de registro al inicio. Esto corrige un problema por el que la configuración de registro no se aplicaba al inicio. • Actualiza el enlace simbólico del cargador de núcleos para que apunte al almacén de componentes de la carpeta raíz de Greengrass durante la instalación. Esta actualización le permite eliminar el archivo JAR y otros artefactos del núcleo que se descargan al instalar el software AWS IoT Greengrass Core. • Correcciones y mejoras menores adicionales. Para obtener más información, consulte las versiones en GitHub.

Versión	Cambios
2.3.0	Nuevas características • Añade soporte para documentos de configuración de despliegues de hasta 10 MB, en lugar de 7 KB (para despliegues orientados a objetos) o 31 KB (para despliegues dirigidos a grupos de objetos). Para utilizar esta función, la AWS IoT política de un dispositivo principal debe permitir el `greengrass:GetDeploymentConfiguration` permiso. Si utilizaste el instalador del software AWS IoT Greengrass principal para aprovisionar recursos, la AWS IoT política de tu dispositivo principal lo permite `greengrass:*` , e incluye este permiso. Para obtener más información, consulte Autenticación y autorización de dispositivos en AWS IoT Greengrass. • Añade la variable de `iot:thingName` receta. Puedes usar esta variable de receta para obtener el nombre del dispositivo AWS IoT principal en una receta. Para obtener más información, consulte Variables de receta. Mejoras y correcciones de errores • Correcciones y mejoras menores adicionales. Para obtener más información, consulte las versiones en GitHub.
2.2.0	Nuevas características • Añade operaciones de IPC para la gestión clandestina local. Mejoras y correcciones de errores • Reduce el tamaño del archivo JAR. • Reduce el uso de memoria. • Soluciona problemas por los que la configuración del registro no se actualizaba en determinados casos. • Correcciones y mejoras menores adicionales. Para obtener más información, consulte las versiones en GitHub.

Versión	Cambios

2.1.0	Nuevas características
	• Admite la descarga de imágenes de Docker desde repositorios privados en Amazon ECR.
	• Añade los siguientes parámetros para personalizar la configuración de MQTT en los dispositivos principales:
	  • `maxInFlightPublishes` — El número máximo de mensajes QoS 1 de MQTT no confirmados que pueden estar en vuelo al mismo tiempo.
	  • `maxPublishRetry` — El número máximo de veces que se puede volver a intentar un mensaje que no se publica.
	• Añade el parámetro `fleetstatusservice` de configuración para configurar el intervalo en el que el dispositivo principal publica el estado del Nube de AWS dispositivo en.
	• Correcciones y mejoras menores adicionales. Para obtener más información, consulte las versiones en GitHub.
	Mejoras y correcciones de errores
	• Corrige un problema que provocaba que los despliegues ocultos se duplicaran al reiniciarse el núcleo.
	• Corrige un problema que provocaba que el núcleo se bloqueara cuando detectaba una excepción de carga de servicio.
	• Mejora la resolución de dependencias de componentes para evitar que se produzca un error en una implementación que incluya una dependencia circular.
	• Corrige un problema que impedía volver a implementar un componente de un complemento si ese componente se había eliminado previamente del dispositivo principal.
	• Se ha corregido un problema que provocaba que la variable de `HOME` entorno se estableciera en el */greengrass/v2* `/work` directorio de los componentes de Lambda o de los componentes que se ejecutan como root. La `HOME` variable ahora está correctamente configurada en el directorio principal del usuario que ejecuta el componente.

Versión	Cambios
	• Correcciones y mejoras menores adicionales. Para obtener más información, consulte las versiones en GitHub.
2.0.5	Mejoras y correcciones de errores • Enruta correctamente el tráfico a través de un proxy de red configurado al descargar los componentes AWS proporcionados. • Utilice el punto final correcto del plano de datos de Greengrass en las regiones de AWS China.

Versión	Cambios
2.0.4	Nuevas características • Habilita el tráfico HTTPS a través del puerto 443. Puede usar el nuevo parámetro de `greengrassDataPlanePort` configura ción de la versión 2.0.4 del componente núcleo para configurar la comunicación HTTPS para que viaje a través del puerto 443 en lugar del puerto predeterminado 8443. Para obtener más información, consulte Configure HTTPS a través del puerto 443. • Añade la variable de receta de ruta de trabajo. Puede utilizar esta variable de receta para obtener la ruta a las carpetas de trabajo de los componentes, que puede utilizar para compartir archivos entre los componentes y sus dependencias. Para obtener más información, consulte la variable de receta de ruta de trabajo. Mejoras y correcciones de errores • Impide la creación de la política de roles de intercambio de fichas AWS Identity and Access Management (IAM) si ya existe una política de roles. Como resultado de este cambio, el instalador ahora necesita la `iam:GetPolicy` y `sts:GetCallerIdentity` cuando se ejecuta con `--provision true` ella. Para obtener más información, consulte Política de IAM mínima para que el instalador aprovisione recursos. • Gestiona correctamente la cancelación de una implementación que aún no se ha registrado correctamente. • Actualiza la configuración para eliminar las entradas antiguas con marcas de tiempo más recientes al anular una implementación. • Correcciones y mejoras menores adicionales. Para obtener más información, consulte las versiones en GitHub.
2.0.3	Versión inicial.

Autenticación del dispositivo cliente

El componente de autenticación del dispositivo cliente (`aws.greengrass.clientdevices.Auth`) autentica los dispositivos cliente y autoriza las acciones de los dispositivos cliente.

> **Note**
>
> Los dispositivos cliente son dispositivos IoT locales que se conectan a un dispositivo central de Greengrass para enviar mensajes MQTT y datos para su procesamiento. Para obtener más información, consulte Interactúa con dispositivos IoT locales.

Temas

Versiones

> **Note**
>
> La versión 2.3.0 de autenticación de dispositivos cliente ha dejado de fabricarse. Se recomienda encarecidamente que actualice a la versión 2.3.1 o posterior de la autenticación del dispositivo cliente.

Este componente tiene las siguientes versiones:

- 2.4.x
- 2.3.x

- 2.2.x
- 2.1.x
- 2.0.x

Tipo

Este componente es un componente de complemento ()`aws.greengrass.plugin`. El núcleo de Greengrass ejecuta este componente en la misma máquina virtual Java (JVM) que el núcleo. El núcleo se reinicia al cambiar la versión de este componente en el dispositivo principal.

Este componente utiliza el mismo archivo de registro que el núcleo de Greengrass. Para obtener más información, consulte Supervisar AWS IoT Greengrass registros.

Para obtener más información, consulte Tipos de componentes.

Sistema operativo

Este componente se puede instalar en los dispositivos principales que ejecutan los siguientes sistemas operativos:

- Linux
- Windows

Requisitos

Este componente tiene los siguientes requisitos:

- El rol de servicio de Greengrass debe estar asociado a usted Cuenta de AWS y permitir el `iot:DescribeCertificate` permiso.
- La AWS IoT política del dispositivo principal debe permitir los siguientes permisos:
  - `greengrass:GetConnectivityInfo`, donde los recursos incluyen el ARN del dispositivo principal que ejecuta este componente
  - `greengrass:VerifyClientDeviceIoTCertificateAssociation`, donde los recursos incluyen el nombre de recurso de Amazon (ARN) de cada dispositivo cliente que se conecta al dispositivo principal
  - `greengrass:VerifyClientDeviceIdentity`

- `greengrass:PutCertificateAuthorities`
- `iot:Publish`, donde los recursos incluyen el ARN del siguiente tema de MQTT:
  - `$aws/things/coreDeviceThingName*-gci/shadow/get`
- `iot:Subscribe`, donde los recursos incluyen los ARN de los siguientes filtros de temas de MQTT:
  - `$aws/things/coreDeviceThingName*-gci/shadow/update/delta`
  - `$aws/things/coreDeviceThingName*-gci/shadow/get/accepted`
- `iot:Receive`, donde los recursos incluyen los ARN de los siguientes temas de MQTT:
  - `$aws/things/coreDeviceThingName*-gci/shadow/update/delta`
  - `$aws/things/coreDeviceThingName*-gci/shadow/get/accepted`

  Para obtener más información, consulte Políticas de AWS IoT para operaciones de plano de datos y AWS IoTPolítica mínima de compatibilidad con los dispositivos cliente.

- (Opcional) Para utilizar la autenticación sin conexión, la función AWS Identity and Access Management (IAM) utilizada por el AWS IoT Greengrass servicio debe contener el siguiente permiso:
  - `greengrass:ListClientDevicesAssociatedWithCoreDevice`para permitir que el dispositivo principal enumere los clientes para la autenticación sin conexión.
- Se admite la ejecución del componente de autenticación del dispositivo cliente en una VPC. Para implementar este componente en una VPC, se requiere lo siguiente.
  - El componente de autenticación del dispositivo cliente debe tener conectividad con AWS IoT data AWS IoT Credentials y Amazon S3.

Puntos finales y puertos

Este componente debe poder realizar solicitudes salientes a los siguientes puntos finales y puertos, además de a los puntos finales y puertos necesarios para el funcionamiento básico. Para obtener más información, consulte Permitir el tráfico del dispositivo a través de un proxy o firewall.

punto de enlace	Puerto	Obligatoria	Descripción
`iot.region.amazonaws.com`	443	Sí	Se utiliza para

punto de enlace	Puerto	Obligatoria	Descripción
			obtener información sobre AWS IoT los certificados de cosas.

Dependencias

Cuando se implementa un componente, AWS IoT Greengrass también se implementan versiones compatibles de sus dependencias. Esto significa que debe cumplir los requisitos del componente y de todas sus dependencias para poder implementarlo correctamente. En esta sección se enumeran las dependencias de las versiones publicadas de este componente y las restricciones de las versiones semánticas que definen las versiones de los componentes para cada dependencia. También puede ver las dependencias de cada versión del componente en la consola.AWS IoT Greengrass En la página de detalles del componente, busque la lista de dependencias.

2.4.4

En la siguiente tabla se enumeran las dependencias de la versión 2.4.4 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.6.0 <2.13.0	Flexible

2.4.3

La siguiente tabla muestra las dependencias de la versión 2.4.3 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.6.0 <2.12.0	Flexible

2.4.1 and 2.4.2

La siguiente tabla muestra las dependencias de las versiones 2.4.1 y 2.4.2 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.6.0 <2.11.0	Flexible

2.3.0 – 2.4.0

La siguiente tabla muestra las dependencias de las versiones 2.3.0 a 2.4.0 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.6.0 <2.10.0	Flexible

2.3.0

La siguiente tabla muestra las dependencias de la versión 2.3.0 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.6.0 <2.10.0	Flexible

2.2.3

La siguiente tabla muestra las dependencias de la versión 2.2.3 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.6.0 <=2.9.0	Flexible

2.2.2

La siguiente tabla muestra las dependencias de la versión 2.2.2 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.6.0 <=2.8.0	Flexible

2.2.1

La siguiente tabla muestra las dependencias de la versión 2.2.1 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.6.0 <2.8.0	Flexible

2.2.0

La siguiente tabla muestra las dependencias de la versión 2.2.0 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.6.0 <2.7.0	Flexible

2.1.0

La siguiente tabla muestra las dependencias de la versión 2.1.0 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.2.0 <2.7.0	Flexible

2.0.4

La siguiente tabla muestra las dependencias de la versión 2.0.4 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.2.0 <2.6.0	Flexible

2.0.2 and 2.0.3

La siguiente tabla muestra las dependencias de las versiones 2.0.2 y 2.0.3 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.2.0 <2.5.0	Flexible

2.0.1

La siguiente tabla muestra las dependencias de la versión 2.0.1 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.2.0 <2.4.0	Flexible

2.0.0

La siguiente tabla muestra las dependencias de la versión 2.0.0 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.2.0 <2.3.0	Flexible

Para obtener más información sobre las dependencias de los componentes, consulta la referencia de recetas de componentes.

Configuración

Este componente proporciona los siguientes parámetros de configuración que puede personalizar al implementar el componente.

> **Note**
>
> El permiso de suscripción se evalúa durante una solicitud de suscripción del cliente al broker MQTT local. Si se revoca el permiso de suscripción existente del cliente, el cliente ya no podrá suscribirse a un tema. Sin embargo, seguirá recibiendo mensajes de cualquier tema al

que se haya suscrito anteriormente. Para evitar este comportamiento, el broker MQTT local debe reiniciarse tras revocar el permiso de suscripción para forzar la reautorización de los clientes.
Para el componente MQTT 5 broker (EMQX), actualice la configuración para reiniciar el `restartIdentifier` broker MQTT 5. Para obtener más información, consulte la configuración del componente del broker MQTT 5.
En el caso del componente broker MQTT 3.1.1 (Moquette), se reinicia semanalmente de forma predeterminada cuando el certificado del servidor cambia, lo que obliga a los clientes a volver a autorizar. Puede forzar un reinicio cambiando la información de conectividad (direcciones IP) del dispositivo principal o realizando un despliegue para eliminar el componente intermediario y volver a desplegarlo posteriormente.

v2.5.0

`deviceGroups`

Los grupos de dispositivos son grupos de dispositivos cliente que tienen permisos para conectarse y comunicarse con un dispositivo principal. Utilice reglas de selección para identificar grupos de dispositivos cliente y defina políticas de autorización de dispositivos cliente que especifiquen los permisos para cada grupo de dispositivos.

Este objeto contiene la siguiente información:

`formatVersion`

La versión de formato de este objeto de configuración.

Puede elegir entre las siguientes opciones:

- `2021-03-05`

`definitions`

Los grupos de dispositivos de este dispositivo principal. Cada definición especifica una regla de selección para evaluar si un dispositivo cliente es miembro del grupo. Cada definición también especifica la política de permisos que se aplicará a los dispositivos cliente que coincidan con la regla de selección. Si un dispositivo cliente es miembro de varios grupos de dispositivos, los permisos del dispositivo se componen de la política de permisos de cada grupo.

Este objeto contiene la siguiente información:

groupNameKey

El nombre de este grupo de dispositivos. *groupNameKey*Sustitúyalo por un nombre que ayude a identificar este grupo de dispositivos.

Este objeto contiene la siguiente información:

`selectionRule`

La consulta que especifica qué dispositivos cliente son miembros de este grupo de dispositivos. Cuando un dispositivo cliente se conecta, el dispositivo principal evalúa esta regla de selección para determinar si el dispositivo cliente es miembro de este grupo de dispositivos. Si el dispositivo cliente es miembro, el dispositivo principal utiliza la política de este grupo de dispositivos para autorizar las acciones del dispositivo cliente.

Cada regla de selección incluye al menos una cláusula de regla de selección, que es una consulta de expresión única que puede coincidir con los dispositivos del cliente. Las reglas de selección utilizan la misma sintaxis de consulta que la indexación de AWS IoT flotas. Para obtener más información sobre la sintaxis de las reglas de selección, consulte la sintaxis de las consultas de indexación de AWS IoT flotas en la Guía AWS IoT Core para desarrolladores.

Utilice el * comodín para hacer coincidir varios dispositivos cliente con una cláusula de regla de selección. Puede usar este comodín al principio y al final del nombre del elemento para hacer coincidir los dispositivos cliente cuyos nombres comiencen o terminen con la cadena que especifique. También puede usar este comodín para hacer coincidir todos los dispositivos cliente.

> **Note**
>
> Para seleccionar un valor que contenga dos puntos (`:`), evite los dos puntos y coloque un carácter de barra invertida (). \ En formatos como JSON, debe evitar los caracteres de barra invertida, por lo que debe escribir dos caracteres de barra invertida antes del carácter de dos puntos. Por ejemplo, especifique seleccionar un `thingName: MyTeam\\:ClientDevice1` elemento cuyo nombre sea. `MyTeam:ClientDevice1`

Puede especificar el siguiente selector:

- `thingName`— El nombre del dispositivo de un AWS IoT cliente.

Example Ejemplo de regla de selección

La siguiente regla de selección coincide con los dispositivos cliente cuyos nombres son `MyClientDevice1` o`MyClientDevice2`.

```
thingName: MyClientDevice1 OR thingName: MyClientDevice2
```

Example Ejemplo de regla de selección (utilice caracteres comodín)

La siguiente regla de selección coincide con los dispositivos cliente cuyos nombres comiencen por. `MyClientDevice`

```
thingName: MyClientDevice*
```

Example Ejemplo de regla de selección (utilice caracteres comodín)

La siguiente regla de selección coincide con los dispositivos cliente cuyos nombres terminan en. `MyClientDevice`

```
thingName: *MyClientDevice
```

Example Ejemplo de regla de selección (hacer coincidir todos los dispositivos)

La siguiente regla de selección coincide con todos los dispositivos cliente.

```
thingName: *
```

`policyName`

La política de permisos que se aplica a los dispositivos cliente de este grupo de dispositivos. Especifique el nombre de la política que defina en el `policies` objeto.

`policies`

Las políticas de autorización de los dispositivos cliente para los dispositivos cliente que se conectan al dispositivo principal. Cada política de autorización especifica un conjunto de acciones y los recursos en los que un dispositivo cliente puede realizar esas acciones.

Este objeto contiene la siguiente información:

policyNameKey

El nombre de esta política de autorización. *policyNameKey*Sustitúyala por un nombre que ayude a identificar esta política de autorización. Este nombre de política se utiliza para definir qué política se aplica a un grupo de dispositivos.

Este objeto contiene la siguiente información:

statementNameKey

El nombre de esta declaración de política. *statementNameKey*Sustitúyala por un nombre que le ayude a identificar esta declaración de política.

Este objeto contiene la siguiente información:

`operations`

La lista de operaciones que permiten utilizar los recursos de esta política.

Puede incluir cualquiera de las siguientes operaciones:

- `mqtt:connect`— Otorga permiso para conectarse al dispositivo principal. Los dispositivos cliente deben tener este permiso para conectarse a un dispositivo principal.

  Esta operación admite los siguientes recursos:

  - `mqtt:clientId:`*deviceClientId*— Restrinja el acceso en función del ID de cliente que utilice el dispositivo cliente para conectarse al intermediario MQTT del dispositivo principal. *deviceClientId*Sustitúyalo por el ID de cliente que se va a utilizar.
- `mqtt:publish`— Otorga permiso para publicar mensajes de MQTT en los temas.

  Esta operación admite los siguientes recursos:

  - `mqtt:topic:`*mqttTopic*— Restrinja el acceso en función del tema de MQTT en el que un dispositivo cliente publica un mensaje. Sustituya *MQTTopic por el tema* que desee utilizar.

    Este recurso no admite los caracteres comodín de los temas MQTT.
- `mqtt:subscribe`— Otorga permiso para suscribirse a los filtros de temas de MQTT para recibir mensajes.

Esta operación admite los siguientes recursos:

- `mqtt:topicfilter:`*`mqttTopicFilter`*— Restrinja el acceso en función de los temas de MQTT en los que un dispositivo cliente puede suscribirse a los mensajes. *`mqttTopicFilter`*Sustitúyalo por el filtro de temas que desee utilizar.

  Este recurso admite los caracteres comodín de los temas # MQTT + y MQTT. Para obtener más información, consulte los temas de MQTT en la AWS IoT Core Guía para desarrolladores.

  El dispositivo cliente puede suscribirse a los filtros de temas exactos que usted permita. Por ejemplo, si permites que el dispositivo cliente se suscriba al `mqtt:topicfilter:client/+/status` recurso, el dispositivo cliente puede suscribirse, `client/+/status` pero no`client/client1/status`.

Puede especificar el * comodín para permitir el acceso a todas las acciones.

`resources`

La lista de recursos que permiten las operaciones de esta política. Especifique los recursos que corresponden a las operaciones de esta política. Por ejemplo, puede especificar una lista de recursos de temas de MQTT (`mqtt:topic:`*`mqttTopic`*) en una política que especifique la `mqtt:publish` operación.

Puede especificar el * comodín en cualquier lugar de la variable de recurso para permitir el acceso a todos los recursos. Por ejemplo, puede especificar que se permita **`mqtt:topic:my*`** el acceso a los recursos que coincidan con esa entrada.

Se admite la siguiente variable de recurso:

- `mqtt:topic:${iot:Connection.Thing.ThingName}`

  Esto se traduce en el nombre del elemento del AWS IoT Core registro para el que se está evaluando la política. AWS IoT Core usa el certificado que presenta el dispositivo cuando se autentica para determinar qué se debe usar para verificar la conexión. Esta variable de política solo está disponible cuando un dispositivo se conecta a través de MQTT o MQTT a través del protocolo. WebSocket

`statementDescription`

(Opcional) Una descripción de esta declaración de política.

`certificates`

(Opcional) Las opciones de configuración del certificado para este dispositivo principal. Este objeto contiene la siguiente información:

`serverCertificateValiditySeconds`

(Opcional) Cantidad de tiempo (en segundos) tras la cual caduca el certificado del servidor MQTT local. Puede configurar esta opción para personalizar la frecuencia con la que los dispositivos cliente se desconectan y se vuelven a conectar al dispositivo principal.

Este componente rota el certificado del servidor MQTT local 24 horas antes de que caduque. El broker MQTT, como el componente Broker MQTT de Moquette, genera un nuevo certificado y se reinicia. Cuando esto ocurre, todos los dispositivos cliente conectados a este dispositivo principal se desconectan. Los dispositivos cliente se pueden volver a conectar al dispositivo principal tras un breve período de tiempo.

Predeterminado: `604800` (7 días)

Valor mínimo: `172800` (2 días)

Valor máximo: `864000` (10 días)

`performance`

(Opcional) Las opciones de configuración del rendimiento de este dispositivo principal. Este objeto contiene la siguiente información:

`maxActiveAuthTokens`

(Opcional) El número máximo de tokens de autorización de los dispositivos cliente activos. Puede aumentar este número para permitir que un mayor número de dispositivos cliente se conecten a un dispositivo de un solo núcleo, sin tener que volver a autenticarlos.

Valor predeterminado: `2500`

`cloudRequestQueueSize`

(Opcional) El número máximo de Nube de AWS solicitudes que se van a poner en cola antes de que este componente las rechace.

Valor predeterminado: `100`

`maxConcurrentCloudRequests`

(Opcional) El número máximo de solicitudes simultáneas que se van a enviar al. Nube de AWS Puede aumentar este número para mejorar el rendimiento de la autenticación en los dispositivos principales a los que se conecta un gran número de dispositivos cliente.

Valor predeterminado: `1`

`certificateAuthority`

(Opcional) Opciones de configuración de la autoridad de certificación para reemplazar la autoridad intermedia del dispositivo principal por su propia autoridad de certificación intermedia.

> **Note**
>
> Si configura su dispositivo principal de Greengrass con una autoridad de certificación (CA) personalizada y utiliza la misma CA para emitir los certificados de los dispositivos cliente, Greengrass omite las comprobaciones de las políticas de autorización para las operaciones de MQTT de los dispositivos cliente. El componente de autenticación del dispositivo cliente confía plenamente en los clientes que utilizan certificados firmados por la CA para la que está configurado.
> Para restringir este comportamiento al utilizar una CA personalizada, cree y firme los dispositivos cliente con una CA diferente o intermedia y, a continuación, ajuste los `certificateChainUri` campos `certificateUri` y para que apunten a la CA intermedia correcta.

Este objeto contiene la siguiente información.

URI del certificado

La ubicación del certificado. Puede ser un URI del sistema de archivos o un URI que apunte a un certificado almacenado en un módulo de seguridad de hardware.

`certificateChainUri`

La ubicación de la cadena de certificados de la CA del dispositivo principal. Debe ser la cadena de certificados completa que lleva a su CA raíz. Puede ser un URI del sistema de

archivos o un URI que apunte a una cadena de certificados almacenada en un módulo de seguridad de hardware.

`privateKeyUri`

La ubicación de la clave privada del dispositivo principal. Puede ser un URI del sistema de archivos o un URI que apunta a una clave privada de certificado almacenada en un módulo de seguridad de hardware.

`security`

(Opcional) Opciones de configuración de seguridad para este dispositivo principal. Este objeto contiene la siguiente información.

`clientDeviceTrustDurationMinutes`

El tiempo en minutos durante el que se puede confiar en la información de autenticación de un dispositivo cliente antes de que sea necesario volver a autenticarse con el dispositivo principal. El valor predeterminado es 1.

`metrics`

(Opcional) Las opciones de métricas de este dispositivo principal. Las métricas de error solo se mostrarán si hay un error en la autenticación del dispositivo cliente. Este objeto contiene la siguiente información:

`disableMetrics`

Si el `disableMetrics` campo está establecido como`true`, la autenticación del dispositivo cliente no recopilará métricas.

Valor predeterminado: `false`

`aggregatePeriodSeconds`

El período de agregación en segundos que determina la frecuencia con la que la autenticación del dispositivo cliente agrega las métricas y las envía al agente de telemetría. Esto no cambia la frecuencia con la que se publican las métricas, ya que el agente de telemetría sigue publicándolas una vez al día.

Valor predeterminado: `3600`

startupTimeoutSeconds

(Opcional) El tiempo máximo en segundos para que se inicie el componente. El estado del componente cambia a `BROKEN` si supera este tiempo de espera.

Valor predeterminado: `120`

Example Ejemplo: actualización de la combinación de configuraciones (mediante una política restrictiva)

El siguiente ejemplo de configuración especifica que se permita a los dispositivos cliente cuyos nombres comiencen por `MyClientDevice` conectarse y publicar o suscribirse en todos los temas.

```
{
  "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 Ejemplo: actualización de la combinación de configuraciones (mediante una política permisiva)

El siguiente ejemplo de configuración especifica permitir que todos los dispositivos cliente se conecten y publiquen o se suscriban sobre todos los temas.

```
{
  "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 Ejemplo: actualización de combinación de configuraciones (mediante una política de nombres de cosas)

El siguiente ejemplo de configuración permite a los dispositivos cliente publicar temas que comiencen con el nombre del dispositivo cliente y terminen con la cadena`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`

Los grupos de dispositivos son grupos de dispositivos cliente que tienen permisos para conectarse y comunicarse con un dispositivo principal. Utilice reglas de selección para identificar grupos de dispositivos cliente y defina políticas de autorización de dispositivos cliente que especifiquen los permisos para cada grupo de dispositivos.

Este objeto contiene la siguiente información:

`formatVersion`

La versión de formato de este objeto de configuración.

Puede elegir entre las siguientes opciones:

- `2021-03-05`

`definitions`

Los grupos de dispositivos de este dispositivo principal. Cada definición especifica una regla de selección para evaluar si un dispositivo cliente es miembro del grupo. Cada definición también especifica la política de permisos que se aplicará a los dispositivos cliente que coincidan con la regla de selección. Si un dispositivo cliente es miembro de varios grupos de dispositivos, los permisos del dispositivo se componen de la política de permisos de cada grupo.

Este objeto contiene la siguiente información:

`groupNameKey`

El nombre de este grupo de dispositivos. *`groupNameKey`*Sustitúyalo por un nombre que ayude a identificar este grupo de dispositivos.

Este objeto contiene la siguiente información:

`selectionRule`

La consulta que especifica qué dispositivos cliente son miembros de este grupo de dispositivos. Cuando un dispositivo cliente se conecta, el dispositivo principal evalúa esta regla de selección para determinar si el dispositivo cliente es miembro de este grupo de dispositivos. Si el dispositivo cliente es miembro, el dispositivo principal utiliza la política de este grupo de dispositivos para autorizar las acciones del dispositivo cliente.

Cada regla de selección incluye al menos una cláusula de regla de selección, que es una consulta de expresión única que puede coincidir con los dispositivos del cliente. Las reglas de selección utilizan la misma sintaxis de consulta que la indexación de AWS IoT flotas. Para obtener más información sobre la sintaxis de las reglas de selección, consulte la sintaxis de las consultas de indexación de AWS IoT flotas en la Guía AWS IoT Core para desarrolladores.

Utilice el * comodín para hacer coincidir varios dispositivos cliente con una cláusula de regla de selección. Puede usar este comodín al principio y al final del nombre del elemento para hacer coincidir los dispositivos cliente cuyos nombres comiencen o terminen con la cadena que especifique. También puede usar este comodín para hacer coincidir todos los dispositivos cliente.

> **Note**
>
> Para seleccionar un valor que contenga dos puntos (`:`), evite los dos puntos y coloque un carácter de barra invertida (). \ En formatos como JSON, debe evitar los caracteres de barra invertida, por lo que debe escribir dos caracteres de barra invertida antes del carácter de dos puntos. Por ejemplo, especifique seleccionar un `thingName: MyTeam\\:ClientDevice1` elemento cuyo nombre sea. `MyTeam:ClientDevice1`

Puede especificar el siguiente selector:

- `thingName`— El nombre del dispositivo de un AWS IoT cliente.

Example Ejemplo de regla de selección

La siguiente regla de selección coincide con los dispositivos cliente cuyos nombres son `MyClientDevice1` o`MyClientDevice2`.

```
thingName: MyClientDevice1 OR thingName: MyClientDevice2
```

Example Ejemplo de regla de selección (utilice caracteres comodín)

La siguiente regla de selección coincide con los dispositivos cliente cuyos nombres comiencen por. `MyClientDevice`

```
thingName: MyClientDevice*
```

Example Ejemplo de regla de selección (utilice caracteres comodín)

La siguiente regla de selección coincide con los dispositivos cliente cuyos nombres terminan en. `MyClientDevice`

```
thingName: *MyClientDevice
```

Example Ejemplo de regla de selección (hacer coincidir todos los dispositivos)

La siguiente regla de selección coincide con todos los dispositivos cliente.

```
thingName: *
```

`policyName`

La política de permisos que se aplica a los dispositivos cliente de este grupo de dispositivos. Especifique el nombre de la política que defina en el `policies` objeto.

`policies`

Las políticas de autorización de los dispositivos cliente para los dispositivos cliente que se conectan al dispositivo principal. Cada política de autorización especifica un conjunto de acciones y los recursos en los que un dispositivo cliente puede realizar esas acciones.

Este objeto contiene la siguiente información:

`policyNameKey`

El nombre de esta política de autorización. *`policyNameKey`*Sustitúyala por un nombre que ayude a identificar esta política de autorización. Este nombre de política se utiliza para definir qué política se aplica a un grupo de dispositivos.

Este objeto contiene la siguiente información:

`statementNameKey`

El nombre de esta declaración de política. *`statementNameKey`*Sustitúyala por un nombre que le ayude a identificar esta declaración de política.

Este objeto contiene la siguiente información:

`operations`

La lista de operaciones que permiten utilizar los recursos de esta política.

Puede incluir cualquiera de las siguientes operaciones:

- `mqtt:connect`— Otorga permiso para conectarse al dispositivo principal. Los dispositivos cliente deben tener este permiso para conectarse a un dispositivo principal.

Esta operación admite los siguientes recursos:

- `mqtt:clientId:`*`deviceClientId`*— Restrinja el acceso en función del ID de cliente que utilice el dispositivo cliente para conectarse al intermediario MQTT del dispositivo principal. *`deviceClientId`*Sustitúyalo por el ID de cliente que se va a utilizar.

- `mqtt:publish`— Otorga permiso para publicar mensajes de MQTT en los temas.

  Esta operación admite los siguientes recursos:

  - `mqtt:topic:`*`mqttTopic`*— Restrinja el acceso en función del tema de MQTT en el que un dispositivo cliente publica un mensaje. Sustituya *`MQTTTopic por el tema`* que desee utilizar.

    Este recurso no admite los caracteres comodín de los temas MQTT.

- `mqtt:subscribe`— Otorga permiso para suscribirse a los filtros de temas de MQTT para recibir mensajes.

  Esta operación admite los siguientes recursos:

  - `mqtt:topicfilter:`*`mqttTopicFilter`*— Restrinja el acceso en función de los temas de MQTT en los que un dispositivo cliente puede suscribirse a los mensajes. *`mqttTopicFilter`*Sustitúyalo por el filtro de temas que desee utilizar.

    Este recurso admite los caracteres comodín de los temas # MQTT + y MQTT. Para obtener más información, consulte los temas de MQTT en la AWS IoT Core Guía para desarrolladores.

    El dispositivo cliente puede suscribirse a los filtros de temas exactos que usted permita. Por ejemplo, si permites que el dispositivo cliente se suscriba al `mqtt:topicfilter:client/+/status` recurso, el dispositivo cliente puede suscribirse, `client/+/status` pero no`client/client1/status`.

Puede especificar el * comodín para permitir el acceso a todas las acciones.

`resources`

La lista de recursos que permiten las operaciones de esta política. Especifique los recursos que corresponden a las operaciones de esta política. Por

ejemplo, puede especificar una lista de recursos de temas de MQTT (`mqtt:topic:`*`mqttTopic`*) en una política que especifique la `mqtt:publish` operación.

Puede especificar el * comodín para permitir el acceso a todos los recursos. No puedes usar el * comodín para hacer coincidir los identificadores de recursos parciales. Por ejemplo, puede especificar**`"resources": "*"`**, pero no puede especificar. **`"resources": "mqtt:clientId:*"`**

`statementDescription`

(Opcional) Una descripción de esta declaración de política.

`certificates`

(Opcional) Las opciones de configuración del certificado para este dispositivo principal. Este objeto contiene la siguiente información:

`serverCertificateValiditySeconds`

(Opcional) Cantidad de tiempo (en segundos) tras la cual caduca el certificado del servidor MQTT local. Puede configurar esta opción para personalizar la frecuencia con la que los dispositivos cliente se desconectan y se vuelven a conectar al dispositivo principal.

Este componente rota el certificado del servidor MQTT local 24 horas antes de que caduque. El broker MQTT, como el componente Broker MQTT de Moquette, genera un nuevo certificado y se reinicia. Cuando esto ocurre, todos los dispositivos cliente conectados a este dispositivo principal se desconectan. Los dispositivos cliente se pueden volver a conectar al dispositivo principal tras un breve período de tiempo.

Predeterminado: `604800` (7 días)

Valor mínimo: `172800` (2 días)

Valor máximo: `864000` (10 días)

`performance`

(Opcional) Las opciones de configuración del rendimiento de este dispositivo principal. Este objeto contiene la siguiente información:

`maxActiveAuthTokens`

(Opcional) El número máximo de tokens de autorización de los dispositivos cliente activos. Puede aumentar este número para permitir que un mayor número de dispositivos cliente se conecten a un dispositivo de un solo núcleo, sin tener que volver a autenticarlos.

Valor predeterminado: `2500`

`cloudRequestQueueSize`

(Opcional) El número máximo de Nube de AWS solicitudes que se van a poner en cola antes de que este componente las rechace.

Valor predeterminado: `100`

`maxConcurrentCloudRequests`

(Opcional) El número máximo de solicitudes simultáneas que se van a enviar al. Nube de AWS Puede aumentar este número para mejorar el rendimiento de la autenticación en los dispositivos principales a los que se conecta un gran número de dispositivos cliente.

Valor predeterminado: `1`

`certificateAuthority`

(Opcional) Opciones de configuración de la autoridad de certificación para reemplazar la autoridad intermedia del dispositivo principal por su propia autoridad de certificación intermedia.

> **Note**
>
> Si configura su dispositivo principal de Greengrass con una autoridad de certificación (CA) personalizada y utiliza la misma CA para emitir los certificados de los dispositivos cliente, Greengrass omite las comprobaciones de las políticas de autorización para las operaciones de MQTT de los dispositivos cliente. El componente de autenticación del dispositivo cliente confía plenamente en los clientes que utilizan certificados firmados por la CA para la que está configurado.
> Para restringir este comportamiento al utilizar una CA personalizada, cree y firme los dispositivos cliente con una CA diferente o intermedia y, a continuación, ajuste los `certificateChainUri` campos `certificateUri` y para que apunten a la CA intermedia correcta.

Este objeto contiene la siguiente información.

URI del certificado

La ubicación del certificado. Puede ser un URI del sistema de archivos o un URI que apunte a un certificado almacenado en un módulo de seguridad de hardware.

`certificateChainUri`

La ubicación de la cadena de certificados de la CA del dispositivo principal. Debe ser la cadena de certificados completa que lleva a su CA raíz. Puede ser un URI del sistema de archivos o un URI que apunte a una cadena de certificados almacenada en un módulo de seguridad de hardware.

`privateKeyUri`

La ubicación de la clave privada del dispositivo principal. Puede ser un URI del sistema de archivos o un URI que apunta a una clave privada de certificado almacenada en un módulo de seguridad de hardware.

`security`

(Opcional) Opciones de configuración de seguridad para este dispositivo principal. Este objeto contiene la siguiente información.

`clientDeviceTrustDurationMinutes`

El tiempo en minutos durante el que se puede confiar en la información de autenticación de un dispositivo cliente antes de que sea necesario volver a autenticarse con el dispositivo principal. El valor predeterminado es 1.

`metrics`

(Opcional) Las opciones de métricas de este dispositivo principal. Las métricas de error solo se mostrarán si hay un error en la autenticación del dispositivo cliente. Este objeto contiene la siguiente información:

`disableMetrics`

Si el `disableMetrics` campo está establecido como`true`, la autenticación del dispositivo cliente no recopilará métricas.

Valor predeterminado: `false`

`aggregatePeriodSeconds`

El período de agregación en segundos que determina la frecuencia con la que la autenticación del dispositivo cliente agrega las métricas y las envía al agente de telemetría. Esto no cambia la frecuencia con la que se publican las métricas, ya que el agente de telemetría sigue publicándolas una vez al día.

Valor predeterminado: `3600`

startupTimeoutSeconds

(Opcional) El tiempo máximo en segundos para que se inicie el componente. El estado del componente cambia a `BROKEN` si supera este tiempo de espera.

Valor predeterminado: `120`

Example Ejemplo: actualización de la combinación de configuraciones (mediante una política restrictiva)

El siguiente ejemplo de configuración especifica que se permita a los dispositivos cliente cuyos nombres comiencen por `MyClientDevice` conectarse y publicar o suscribirse en todos los temas.

```
{
  "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 Ejemplo: actualización de la combinación de configuraciones (mediante una política permisiva)

El siguiente ejemplo de configuración especifica permitir que todos los dispositivos cliente se conecten y publiquen o se suscriban sobre todos los temas.

```
{
  "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`

Los grupos de dispositivos son grupos de dispositivos cliente que tienen permisos para conectarse y comunicarse con un dispositivo principal. Utilice reglas de selección para identificar grupos de dispositivos cliente y defina políticas de autorización de dispositivos cliente que especifiquen los permisos para cada grupo de dispositivos.

Este objeto contiene la siguiente información:

`formatVersion`

La versión de formato de este objeto de configuración.

Puede elegir entre las siguientes opciones:

- `2021-03-05`

`definitions`

Los grupos de dispositivos de este dispositivo principal. Cada definición especifica una regla de selección para evaluar si un dispositivo cliente es miembro del grupo. Cada definición también especifica la política de permisos que se aplicará a los dispositivos cliente que coincidan con la regla de selección. Si un dispositivo cliente es miembro de varios grupos de dispositivos, los permisos del dispositivo se componen de la política de permisos de cada grupo.

Este objeto contiene la siguiente información:

groupNameKey

El nombre de este grupo de dispositivos. *groupNameKey*Sustitúyalo por un nombre que ayude a identificar este grupo de dispositivos.

Este objeto contiene la siguiente información:

`selectionRule`

La consulta que especifica qué dispositivos cliente son miembros de este grupo de dispositivos. Cuando un dispositivo cliente se conecta, el dispositivo principal evalúa esta regla de selección para determinar si el dispositivo cliente es miembro de este grupo de dispositivos. Si el dispositivo cliente es miembro, el dispositivo principal utiliza la política de este grupo de dispositivos para autorizar las acciones del dispositivo cliente.

Cada regla de selección incluye al menos una cláusula de regla de selección, que es una consulta de expresión única que puede coincidir con los dispositivos del cliente. Las reglas de selección utilizan la misma sintaxis de consulta que la indexación de AWS IoT flotas. Para obtener más información sobre la sintaxis de las reglas de selección, consulte la sintaxis de las consultas de indexación de AWS IoT flotas en la Guía AWS IoT Core para desarrolladores.

Utilice el * comodín para hacer coincidir varios dispositivos cliente con una cláusula de regla de selección. Puede utilizar este comodín al final del nombre del elemento para hacer coincidir los dispositivos cliente cuyos nombres comiencen por la cadena que especifique. También puede usar este comodín para que coincidan con todos los dispositivos cliente.

> **Note**
>
> Para seleccionar un valor que contenga dos puntos (`:`), evite los dos puntos y coloque un carácter de barra invertida (). \\ En formatos como JSON, debe evitar los caracteres de barra invertida, por lo que debe escribir dos caracteres de barra invertida antes del carácter de dos puntos. Por ejemplo, especifique seleccionar un `thingName: MyTeam\\\\:ClientDevice1` elemento cuyo nombre sea. `MyTeam:ClientDevice1`

Puede especificar el siguiente selector:

- `thingName`— El nombre del dispositivo de un AWS IoT cliente.

Example Ejemplo de regla de selección

La siguiente regla de selección coincide con los dispositivos cliente cuyos nombres son `MyClientDevice1` o`MyClientDevice2`.

```
thingName: MyClientDevice1 OR thingName: MyClientDevice2
```

Example Ejemplo de regla de selección (utilice caracteres comodín)

La siguiente regla de selección coincide con los dispositivos cliente cuyos nombres comiencen por. `MyClientDevice`

```
thingName: MyClientDevice*
```

Example Ejemplo de regla de selección (hacer coincidir todos los dispositivos)

La siguiente regla de selección coincide con todos los dispositivos cliente.

```
thingName: *
```

`policyName`

La política de permisos que se aplica a los dispositivos cliente de este grupo de dispositivos. Especifique el nombre de la política que defina en el `policies` objeto.

`policies`

Las políticas de autorización de los dispositivos cliente para los dispositivos cliente que se conectan al dispositivo principal. Cada política de autorización especifica un conjunto de acciones y los recursos en los que un dispositivo cliente puede realizar esas acciones.

Este objeto contiene la siguiente información:

`policyNameKey`

El nombre de esta política de autorización. *`policyNameKey`*Sustitúyala por un nombre que ayude a identificar esta política de autorización. Este nombre de política se utiliza para definir qué política se aplica a un grupo de dispositivos.

Este objeto contiene la siguiente información:

statementNameKey

El nombre de esta declaración de política. *statementNameKey*Sustitúyala por un nombre que le ayude a identificar esta declaración de política.

Este objeto contiene la siguiente información:

operations

La lista de operaciones que permiten utilizar los recursos de esta política.

Puede incluir cualquiera de las siguientes operaciones:

- `mqtt:connect`— Otorga permiso para conectarse al dispositivo principal. Los dispositivos cliente deben tener este permiso para conectarse a un dispositivo principal.

  Esta operación admite los siguientes recursos:

  - `mqtt:clientId:`*deviceClientId*— Restrinja el acceso en función del ID de cliente que utilice el dispositivo cliente para conectarse al intermediario MQTT del dispositivo principal. *deviceClientId*Sustitúyalo por el ID de cliente que se va a utilizar.
- `mqtt:publish`— Otorga permiso para publicar mensajes de MQTT en los temas.

  Esta operación admite los siguientes recursos:

  - `mqtt:topic:`*mqttTopic*— Restrinja el acceso en función del tema de MQTT en el que un dispositivo cliente publica un mensaje. Sustituya *MQTTopic por el tema* que desee utilizar.

    Este recurso no admite los caracteres comodín de los temas MQTT.
- `mqtt:subscribe`— Otorga permiso para suscribirse a los filtros de temas de MQTT para recibir mensajes.

  Esta operación admite los siguientes recursos:

  - `mqtt:topicfilter:`*mqttTopicFilter*— Restrinja el acceso en función de los temas de MQTT en los que un dispositivo cliente puede suscribirse a los mensajes. *mqttTopicFilter*Sustitúyalo por el filtro de temas que desee utilizar.

Este recurso admite los caracteres comodín de los temas # MQTT + y MQTT. Para obtener más información, consulte los temas de MQTT en la AWS IoT Core Guía para desarrolladores.

El dispositivo cliente puede suscribirse a los filtros de temas exactos que usted permita. Por ejemplo, si permites que el dispositivo cliente se suscriba al `mqtt:topicfilter:client/+/status` recurso, el dispositivo cliente puede suscribirse, `client/+/status` pero no`client/client1/status`.

Puede especificar el * comodín para permitir el acceso a todas las acciones.

`resources`

La lista de recursos que permiten las operaciones de esta política. Especifique los recursos que corresponden a las operaciones de esta política. Por ejemplo, puede especificar una lista de recursos de temas de MQTT (`mqtt:topic:`*`mqttTopic`*) en una política que especifique la `mqtt:publish` operación.

Puede especificar el * comodín para permitir el acceso a todos los recursos. No puedes usar el * comodín para hacer coincidir los identificadores de recursos parciales. Por ejemplo, puede especificar**`"resources": "*"`**, pero no puede especificar. **`"resources": "mqtt:clientId:*"`**

`statementDescription`

(Opcional) Una descripción de esta declaración de política.

`certificates`

(Opcional) Las opciones de configuración del certificado para este dispositivo principal. Este objeto contiene la siguiente información:

`serverCertificateValiditySeconds`

(Opcional) Cantidad de tiempo (en segundos) tras la cual caduca el certificado del servidor MQTT local. Puede configurar esta opción para personalizar la frecuencia con la que los dispositivos cliente se desconectan y se vuelven a conectar al dispositivo principal.

Este componente rota el certificado del servidor MQTT local 24 horas antes de que caduque. El broker MQTT, como el componente Broker MQTT de Moquette, genera

un nuevo certificado y se reinicia. Cuando esto ocurre, todos los dispositivos cliente conectados a este dispositivo principal se desconectan. Los dispositivos cliente se pueden volver a conectar al dispositivo principal tras un breve período de tiempo.

Predeterminado: `604800` (7 días)

Valor mínimo: `172800` (2 días)

Valor máximo: `864000` (10 días)

`performance`

(Opcional) Las opciones de configuración del rendimiento de este dispositivo principal. Este objeto contiene la siguiente información:

`maxActiveAuthTokens`

(Opcional) El número máximo de tokens de autorización de los dispositivos cliente activos. Puede aumentar este número para permitir que un mayor número de dispositivos cliente se conecten a un dispositivo de un solo núcleo, sin tener que volver a autenticarlos.

Valor predeterminado: `2500`

`cloudRequestQueueSize`

(Opcional) El número máximo de Nube de AWS solicitudes que se van a poner en cola antes de que este componente las rechace.

Valor predeterminado: `100`

`maxConcurrentCloudRequests`

(Opcional) El número máximo de solicitudes simultáneas que se van a enviar al. Nube de AWS Puede aumentar este número para mejorar el rendimiento de la autenticación en los dispositivos principales a los que se conecta un gran número de dispositivos cliente.

Valor predeterminado: `1`

`certificateAuthority`

(Opcional) Opciones de configuración de la autoridad de certificación para reemplazar la autoridad intermedia del dispositivo principal por su propia autoridad de certificación intermedia.

> **Note**
>
> Si configura su dispositivo principal de Greengrass con una autoridad de certificación (CA) personalizada y utiliza la misma CA para emitir los certificados de los dispositivos cliente, Greengrass omite las comprobaciones de las políticas de autorización para las operaciones de MQTT de los dispositivos cliente. El componente de autenticación del dispositivo cliente confía plenamente en los clientes que utilizan certificados firmados por la CA para la que está configurado.
> Para restringir este comportamiento al utilizar una CA personalizada, cree y firme los dispositivos cliente con una CA diferente o intermedia y, a continuación, ajuste los `certificateChainUri` campos `certificateUri` y para que apunten a la CA intermedia correcta.

Este objeto contiene la siguiente información.

URI del certificado

La ubicación del certificado. Puede ser un URI del sistema de archivos o un URI que apunte a un certificado almacenado en un módulo de seguridad de hardware.

`certificateChainUri`

La ubicación de la cadena de certificados de la CA del dispositivo principal. Debe ser la cadena de certificados completa que lleva a su CA raíz. Puede ser un URI del sistema de archivos o un URI que apunte a una cadena de certificados almacenada en un módulo de seguridad de hardware.

`privateKeyUri`

La ubicación de la clave privada del dispositivo principal. Puede ser un URI del sistema de archivos o un URI que apunta a una clave privada de certificado almacenada en un módulo de seguridad de hardware.

`security`

(Opcional) Opciones de configuración de seguridad para este dispositivo principal. Este objeto contiene la siguiente información.

`clientDeviceTrustDurationMinutes`

El tiempo en minutos durante el que se puede confiar en la información de autenticación de un dispositivo cliente antes de que sea necesario volver a autenticarse con el dispositivo principal. El valor predeterminado es 1.

`metrics`

(Opcional) Las opciones de métricas de este dispositivo principal. Las métricas de error solo se mostrarán si hay un error en la autenticación del dispositivo cliente. Este objeto contiene la siguiente información:

`disableMetrics`

Si el `disableMetrics` campo está establecido como`true`, la autenticación del dispositivo cliente no recopilará métricas.

Valor predeterminado: `false`

`aggregatePeriodSeconds`

El período de agregación en segundos que determina la frecuencia con la que la autenticación del dispositivo cliente agrega las métricas y las envía al agente de telemetría. Esto no cambia la frecuencia con la que se publican las métricas, ya que el agente de telemetría sigue publicándolas una vez al día.

Valor predeterminado: `3600`

startupTimeoutSeconds

(Opcional) El tiempo máximo en segundos para que se inicie el componente. El estado del componente cambia a `BROKEN` si supera este tiempo de espera.

Valor predeterminado: `120`

Example Ejemplo: actualización de la combinación de configuraciones (mediante una política restrictiva)

El siguiente ejemplo de configuración especifica que se permita a los dispositivos cliente cuyos nombres comiencen por `MyClientDevice` conectarse y publicar o suscribirse en todos los temas.

```
{
```

```
  "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 Ejemplo: actualización de la combinación de configuraciones (mediante una política permisiva)

El siguiente ejemplo de configuración especifica permitir que todos los dispositivos cliente se conecten y publiquen o se suscriban sobre todos los temas.

```
{
  "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`

Los grupos de dispositivos son grupos de dispositivos cliente que tienen permisos para conectarse y comunicarse con un dispositivo principal. Utilice reglas de selección para identificar grupos de dispositivos cliente y defina políticas de autorización de dispositivos cliente que especifiquen los permisos para cada grupo de dispositivos.

Este objeto contiene la siguiente información:

`formatVersion`

La versión de formato de este objeto de configuración.

Puede elegir entre las siguientes opciones:

- `2021-03-05`

`definitions`

Los grupos de dispositivos de este dispositivo principal. Cada definición especifica una regla de selección para evaluar si un dispositivo cliente es miembro del grupo. Cada definición también especifica la política de permisos que se aplicará a los dispositivos cliente que coincidan con la regla de selección. Si un dispositivo cliente es miembro de varios grupos de dispositivos, los permisos del dispositivo se componen de la política de permisos de cada grupo.

Este objeto contiene la siguiente información:

`groupNameKey`

El nombre de este grupo de dispositivos. *`groupNameKey`*Sustitúyalo por un nombre que ayude a identificar este grupo de dispositivos.

Este objeto contiene la siguiente información:

`selectionRule`

La consulta que especifica qué dispositivos cliente son miembros de este grupo de dispositivos. Cuando un dispositivo cliente se conecta, el dispositivo principal evalúa esta regla de selección para determinar si el dispositivo cliente es miembro de este grupo de dispositivos. Si el dispositivo cliente es miembro, el dispositivo principal utiliza la política de este grupo de dispositivos para autorizar las acciones del dispositivo cliente.

Cada regla de selección incluye al menos una cláusula de regla de selección, que es una consulta de expresión única que puede coincidir con los dispositivos del cliente. Las reglas de selección utilizan la misma sintaxis de consulta que la indexación de AWS IoT flotas. Para obtener más información sobre la sintaxis de las reglas de selección, consulte la sintaxis de las consultas de indexación de AWS IoT flotas en la Guía AWS IoT Core para desarrolladores.

Utilice el * comodín para hacer coincidir varios dispositivos cliente con una cláusula de regla de selección. Puede utilizar este comodín al final del nombre del elemento

para hacer coincidir los dispositivos cliente cuyos nombres comiencen por la cadena que especifique. También puede usar este comodín para que coincidan con todos los dispositivos cliente.

> **Note**
>
> Para seleccionar un valor que contenga dos puntos (`:`), evite los dos puntos y coloque un carácter de barra invertida (). \\ En formatos como JSON, debe evitar los caracteres de barra invertida, por lo que debe escribir dos caracteres de barra invertida antes del carácter de dos puntos. Por ejemplo, especifique seleccionar un `thingName: MyTeam\\\\:ClientDevice1` elemento cuyo nombre sea. `MyTeam:ClientDevice1`

Puede especificar el siguiente selector:

- `thingName`— El nombre del dispositivo de un AWS IoT cliente.

Example Ejemplo de regla de selección

La siguiente regla de selección coincide con los dispositivos cliente cuyos nombres son `MyClientDevice1` o`MyClientDevice2`.

```
thingName: MyClientDevice1 OR thingName: MyClientDevice2
```

Example Ejemplo de regla de selección (utilice caracteres comodín)

La siguiente regla de selección coincide con los dispositivos cliente cuyos nombres comiencen por. `MyClientDevice`

```
thingName: MyClientDevice*
```

Example Ejemplo de regla de selección (hacer coincidir todos los dispositivos)

La siguiente regla de selección coincide con todos los dispositivos cliente.

```
thingName: *
```

`policyName`

La política de permisos que se aplica a los dispositivos cliente de este grupo de dispositivos. Especifique el nombre de la política que defina en el `policies` objeto.

`policies`

Las políticas de autorización de los dispositivos cliente para los dispositivos cliente que se conectan al dispositivo principal. Cada política de autorización especifica un conjunto de acciones y los recursos en los que un dispositivo cliente puede realizar esas acciones.

Este objeto contiene la siguiente información:

`policyNameKey`

El nombre de esta política de autorización. *`policyNameKey`*Sustitúyala por un nombre que ayude a identificar esta política de autorización. Este nombre de política se utiliza para definir qué política se aplica a un grupo de dispositivos.

Este objeto contiene la siguiente información:

`statementNameKey`

El nombre de esta declaración de política. *`statementNameKey`*Sustitúyala por un nombre que le ayude a identificar esta declaración de política.

Este objeto contiene la siguiente información:

`operations`

La lista de operaciones que permiten utilizar los recursos de esta política.

Puede incluir cualquiera de las siguientes operaciones:

- `mqtt:connect`— Otorga permiso para conectarse al dispositivo principal. Los dispositivos cliente deben tener este permiso para conectarse a un dispositivo principal.

  Esta operación admite los siguientes recursos:

  - `mqtt:clientId:`*`deviceClientId`*— Restrinja el acceso en función del ID de cliente que utilice el dispositivo cliente para conectarse al intermediario MQTT del dispositivo principal. *`deviceClientId`*Sustitúyalo por el ID de cliente que se va a utilizar.

- `mqtt:publish`— Otorga permiso para publicar mensajes de MQTT en los temas.

  Esta operación admite los siguientes recursos:

  - `mqtt:topic:`*`mqttTopic`*— Restrinja el acceso en función del tema de MQTT en el que un dispositivo cliente publica un mensaje. Sustituya *`MQTTopic por el tema`* que desee utilizar.

    Este recurso no admite los caracteres comodín de los temas MQTT.

- `mqtt:subscribe`— Otorga permiso para suscribirse a los filtros de temas de MQTT para recibir mensajes.

  Esta operación admite los siguientes recursos:

  - `mqtt:topicfilter:`*`mqttTopicFilter`*— Restrinja el acceso en función de los temas de MQTT en los que un dispositivo cliente puede suscribirse a los mensajes. *`mqttTopicFilter`*Sustitúyalo por el filtro de temas que desee utilizar.

    Este recurso admite los caracteres comodín de los temas # MQTT + y MQTT. Para obtener más información, consulte los temas de MQTT en la AWS IoT Core Guía para desarrolladores.

    El dispositivo cliente puede suscribirse a los filtros de temas exactos que usted permita. Por ejemplo, si permites que el dispositivo cliente se suscriba al `mqtt:topicfilter:client/+/status` recurso, el dispositivo cliente puede suscribirse, `client/+/status` pero no`client/client1/status`.

  Puede especificar el * comodín para permitir el acceso a todas las acciones.

`resources`

La lista de recursos que permiten las operaciones de esta política. Especifique los recursos que corresponden a las operaciones de esta política. Por ejemplo, puede especificar una lista de recursos de temas de MQTT (`mqtt:topic:`*`mqttTopic`*) en una política que especifique la `mqtt:publish` operación.

Puede especificar el * comodín para permitir el acceso a todos los recursos. No puedes usar el * comodín para hacer coincidir los identificadores de recursos

parciales. Por ejemplo, puede especificar**`"resources": "*"`**, pero no puede especificar. **`"resources": "mqtt:clientId:*"`**

`statementDescription`

(Opcional) Una descripción de esta declaración de política.

`certificates`

(Opcional) Las opciones de configuración del certificado para este dispositivo principal. Este objeto contiene la siguiente información:

`serverCertificateValiditySeconds`

(Opcional) Cantidad de tiempo (en segundos) tras la cual caduca el certificado del servidor MQTT local. Puede configurar esta opción para personalizar la frecuencia con la que los dispositivos cliente se desconectan y se vuelven a conectar al dispositivo principal.

Este componente rota el certificado del servidor MQTT local 24 horas antes de que caduque. El broker MQTT, como el componente Broker MQTT de Moquette, genera un nuevo certificado y se reinicia. Cuando esto ocurre, todos los dispositivos cliente conectados a este dispositivo principal se desconectan. Los dispositivos cliente se pueden volver a conectar al dispositivo principal tras un breve período de tiempo.

Predeterminado: `604800` (7 días)

Valor mínimo: `172800` (2 días)

Valor máximo: `864000` (10 días)

`performance`

(Opcional) Las opciones de configuración del rendimiento de este dispositivo principal. Este objeto contiene la siguiente información:

`maxActiveAuthTokens`

(Opcional) El número máximo de tokens de autorización de los dispositivos cliente activos. Puede aumentar este número para permitir que un mayor número de dispositivos cliente se conecten a un dispositivo de un solo núcleo, sin tener que volver a autenticarlos.

Valor predeterminado: `2500`

`cloudRequestQueueSize`

(Opcional) El número máximo de Nube de AWS solicitudes que se van a poner en cola antes de que este componente las rechace.

Valor predeterminado: `100`

`maxConcurrentCloudRequests`

(Opcional) El número máximo de solicitudes simultáneas que se van a enviar al. Nube de AWS Puede aumentar este número para mejorar el rendimiento de la autenticación en los dispositivos principales a los que se conecta un gran número de dispositivos cliente.

Valor predeterminado: `1`

`certificateAuthority`

(Opcional) Opciones de configuración de la autoridad de certificación para reemplazar la autoridad intermedia del dispositivo principal por su propia autoridad de certificación intermedia. Este objeto contiene la siguiente información.

Este objeto contiene la siguiente información:

URI del certificado

La ubicación del certificado. Puede ser un URI del sistema de archivos o un URI que apunte a un certificado almacenado en un módulo de seguridad de hardware.

`certificateChainUri`

La ubicación de la cadena de certificados de la CA del dispositivo principal. Debe ser la cadena de certificados completa que lleva a su CA raíz. Puede ser un URI del sistema de archivos o un URI que apunte a una cadena de certificados almacenada en un módulo de seguridad de hardware.

`privateKeyUri`

La ubicación de la clave privada del dispositivo principal. Puede ser un URI del sistema de archivos o un URI que apunta a una clave privada de certificado almacenada en un módulo de seguridad de hardware.

`security`

(Opcional) Opciones de configuración de seguridad para este dispositivo principal. Este objeto contiene la siguiente información.

`clientDeviceTrustDurationMinutes`

El tiempo en minutos durante el que se puede confiar en la información de autenticación de un dispositivo cliente antes de que sea necesario volver a autenticarse con el dispositivo principal. El valor predeterminado es 1.

`metrics`

(Opcional) Las opciones de métricas de este dispositivo principal. Las métricas de error solo se mostrarán si hay un error en la autenticación del dispositivo cliente. Este objeto contiene la siguiente información:

`disableMetrics`

Si el `disableMetrics` campo está establecido como`true`, la autenticación del dispositivo cliente no recopilará métricas.

Valor predeterminado: `false`

`aggregatePeriodSeconds`

El período de agregación en segundos que determina la frecuencia con la que la autenticación del dispositivo cliente agrega las métricas y las envía al agente de telemetría. Esto no cambia la frecuencia con la que se publican las métricas, ya que el agente de telemetría sigue publicándolas una vez al día.

Valor predeterminado: `3600`

Example Ejemplo: actualización de la combinación de configuraciones (mediante una política restrictiva)

El siguiente ejemplo de configuración especifica que se permita a los dispositivos cliente cuyos nombres comiencen por `MyClientDevice` conectarse y publicar o suscribirse en todos los temas.

```
{
  "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 Ejemplo: actualización de la combinación de configuraciones (mediante una política permisiva)

El siguiente ejemplo de configuración especifica permitir que todos los dispositivos cliente se conecten y publiquen o se suscriban sobre todos los temas.

```
{
  "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`

Los grupos de dispositivos son grupos de dispositivos cliente que tienen permisos para conectarse y comunicarse con un dispositivo principal. Utilice reglas de selección para identificar grupos de dispositivos cliente y defina políticas de autorización de dispositivos cliente que especifiquen los permisos para cada grupo de dispositivos.

Este objeto contiene la siguiente información:

`formatVersion`

La versión de formato de este objeto de configuración.

Puede elegir entre las siguientes opciones:

- `2021-03-05`

`definitions`

Los grupos de dispositivos de este dispositivo principal. Cada definición especifica una regla de selección para evaluar si un dispositivo cliente es miembro del grupo. Cada definición también especifica la política de permisos que se aplicará a los dispositivos cliente que coincidan con la regla de selección. Si un dispositivo cliente es miembro de varios grupos de dispositivos, los permisos del dispositivo se componen de la política de permisos de cada grupo.

Este objeto contiene la siguiente información:

`groupNameKey`

El nombre de este grupo de dispositivos. *`groupNameKey`*Sustitúyalo por un nombre que ayude a identificar este grupo de dispositivos.

Este objeto contiene la siguiente información:

`selectionRule`

La consulta que especifica qué dispositivos cliente son miembros de este grupo de dispositivos. Cuando un dispositivo cliente se conecta, el dispositivo principal evalúa esta regla de selección para determinar si el dispositivo cliente es miembro de este grupo de dispositivos. Si el dispositivo cliente es miembro, el dispositivo principal utiliza la política de este grupo de dispositivos para autorizar las acciones del dispositivo cliente.

Cada regla de selección incluye al menos una cláusula de regla de selección, que es una consulta de expresión única que puede coincidir con los dispositivos del cliente. Las reglas de selección utilizan la misma sintaxis de consulta que la indexación de AWS IoT flotas. Para obtener más información sobre la sintaxis de las reglas de selección, consulte la sintaxis de las consultas de indexación de AWS IoT flotas en la Guía AWS IoT Core para desarrolladores.

Utilice el * comodín para hacer coincidir varios dispositivos cliente con una cláusula de regla de selección. Puede utilizar este comodín al final del nombre del elemento para hacer coincidir los dispositivos cliente cuyos nombres comiencen por la cadena que especifique. También puede usar este comodín para que coincidan con todos los dispositivos cliente.

> **Note**
>
> Para seleccionar un valor que contenga dos puntos (`:`), evite los dos puntos y coloque un carácter de barra invertida (). \\ En formatos como JSON, debe evitar los caracteres de barra invertida, por lo que debe escribir dos caracteres de barra invertida antes del carácter de dos puntos. Por ejemplo, especifique seleccionar un `thingName: MyTeam\\\\:ClientDevice1` elemento cuyo nombre sea. `MyTeam:ClientDevice1`

Puede especificar el siguiente selector:

- `thingName`— El nombre del dispositivo de un AWS IoT cliente.

Example Ejemplo de regla de selección

La siguiente regla de selección coincide con los dispositivos cliente cuyos nombres son `MyClientDevice1` o`MyClientDevice2`.

```
thingName: MyClientDevice1 OR thingName: MyClientDevice2
```

Example Ejemplo de regla de selección (utilice caracteres comodín)

La siguiente regla de selección coincide con los dispositivos cliente cuyos nombres comiencen por. `MyClientDevice`

```
thingName: MyClientDevice*
```

Example Ejemplo de regla de selección (hacer coincidir todos los dispositivos)

La siguiente regla de selección coincide con todos los dispositivos cliente.

```
thingName: *
```

`policyName`

La política de permisos que se aplica a los dispositivos cliente de este grupo de dispositivos. Especifique el nombre de la política que defina en el `policies` objeto.

`policies`

Las políticas de autorización de los dispositivos cliente para los dispositivos cliente que se conectan al dispositivo principal. Cada política de autorización especifica un conjunto de acciones y los recursos en los que un dispositivo cliente puede realizar esas acciones.

Este objeto contiene la siguiente información:

`policyNameKey`

El nombre de esta política de autorización. *`policyNameKey`*Sustitúyala por un nombre que ayude a identificar esta política de autorización. Este nombre de política se utiliza para definir qué política se aplica a un grupo de dispositivos.

Este objeto contiene la siguiente información:

`statementNameKey`

El nombre de esta declaración de política. *`statementNameKey`*Sustitúyala por un nombre que le ayude a identificar esta declaración de política.

Este objeto contiene la siguiente información:

`operations`

La lista de operaciones que permiten utilizar los recursos de esta política.

Puede incluir cualquiera de las siguientes operaciones:

- `mqtt:connect`— Otorga permiso para conectarse al dispositivo principal. Los dispositivos cliente deben tener este permiso para conectarse a un dispositivo principal.

  Esta operación admite los siguientes recursos:

  - `mqtt:clientId:`*`deviceClientId`*— Restrinja el acceso en función del ID de cliente que utilice el dispositivo cliente para conectarse al intermediario MQTT del dispositivo principal. *`deviceClientId`*Sustitúyalo por el ID de cliente que se va a utilizar.
- `mqtt:publish`— Otorga permiso para publicar mensajes de MQTT en los temas.

  Esta operación admite los siguientes recursos:

- `mqtt:topic:`*`mqttTopic`*— Restrinja el acceso en función del tema de MQTT en el que un dispositivo cliente publica un mensaje. Sustituya *`MQTTopic por el tema`* que desee utilizar.

  Este recurso no admite los caracteres comodín de los temas MQTT.

- `mqtt:subscribe`— Otorga permiso para suscribirse a los filtros de temas de MQTT para recibir mensajes.

  Esta operación admite los siguientes recursos:

  - `mqtt:topicfilter:`*`mqttTopicFilter`*— Restrinja el acceso en función de los temas de MQTT en los que un dispositivo cliente puede suscribirse a los mensajes. *`mqttTopicFilter`*Sustitúyalo por el filtro de temas que desee utilizar.

    Este recurso admite los caracteres comodín de los temas # MQTT + y MQTT. Para obtener más información, consulte los temas de MQTT en la AWS IoT Core Guía para desarrolladores.

    El dispositivo cliente puede suscribirse a los filtros de temas exactos que usted permita. Por ejemplo, si permites que el dispositivo cliente se suscriba al `mqtt:topicfilter:client/+/status` recurso, el dispositivo cliente puede suscribirse, `client/+/status` pero no`client/client1/status`.

Puede especificar el * comodín para permitir el acceso a todas las acciones.

`resources`

La lista de recursos que permiten las operaciones de esta política. Especifique los recursos que corresponden a las operaciones de esta política. Por ejemplo, puede especificar una lista de recursos de temas de MQTT (`mqtt:topic:`*`mqttTopic`*) en una política que especifique la `mqtt:publish` operación.

Puede especificar el * comodín para permitir el acceso a todos los recursos. No puedes usar el * comodín para hacer coincidir los identificadores de recursos parciales. Por ejemplo, puede especificar**`"resources": "*"`**, pero no puede especificar. **`"resources": "mqtt:clientId:*"`**

`statementDescription`

(Opcional) Una descripción de esta declaración de política.

`certificates`

(Opcional) Las opciones de configuración del certificado para este dispositivo principal. Este objeto contiene la siguiente información:

`serverCertificateValiditySeconds`

(Opcional) Cantidad de tiempo (en segundos) tras la cual caduca el certificado del servidor MQTT local. Puede configurar esta opción para personalizar la frecuencia con la que los dispositivos cliente se desconectan y se vuelven a conectar al dispositivo principal.

Este componente rota el certificado del servidor MQTT local 24 horas antes de que caduque. El broker MQTT, como el componente Broker MQTT de Moquette, genera un nuevo certificado y se reinicia. Cuando esto ocurre, todos los dispositivos cliente conectados a este dispositivo principal se desconectan. Los dispositivos cliente se pueden volver a conectar al dispositivo principal tras un breve período de tiempo.

Predeterminado: `604800` (7 días)

Valor mínimo: `172800` (2 días)

Valor máximo: `864000` (10 días)

`performance`

(Opcional) Las opciones de configuración del rendimiento de este dispositivo principal. Este objeto contiene la siguiente información:

`maxActiveAuthTokens`

(Opcional) El número máximo de tokens de autorización de los dispositivos cliente activos. Puede aumentar este número para permitir que un mayor número de dispositivos cliente se conecten a un dispositivo de un solo núcleo sin volver a autenticarlos.

Valor predeterminado: `2500`

`cloudRequestQueueSize`

(Opcional) El número máximo de Nube de AWS solicitudes que se pueden poner en cola antes de que este componente las rechace.

Valor predeterminado: `100`

`maxConcurrentCloudRequests`

(Opcional) El número máximo de solicitudes simultáneas que se van a enviar al. Nube de AWS Puede aumentar este número para mejorar el rendimiento de la autenticación en los dispositivos principales a los que se conecta un gran número de dispositivos cliente.

Valor predeterminado: 1

`certificateAuthority`

(Opcional) Opciones de configuración de la autoridad de certificación para reemplazar la autoridad intermedia del dispositivo principal por su propia autoridad de certificación intermedia. Este objeto contiene la siguiente información.

URI del certificado

La ubicación del certificado. Puede ser un URI del sistema de archivos o un URI que apunte a un certificado almacenado en un módulo de seguridad de hardware.

`certificateChainUri`

La ubicación de la cadena de certificados de la CA del dispositivo principal. Debe ser la cadena de certificados completa que lleva a su CA raíz. Puede ser un URI del sistema de archivos o un URI que apunte a una cadena de certificados almacenada en un módulo de seguridad de hardware.

`privateKeyUri`

La ubicación de la clave privada del dispositivo principal. Puede ser un URI del sistema de archivos o un URI que apunta a una clave privada de certificado almacenada en un módulo de seguridad de hardware.

`security`

(Opcional) Opciones de configuración de seguridad para este dispositivo principal. Este objeto contiene la siguiente información.

`clientDeviceTrustDurationMinutes`

El tiempo en minutos durante el que se puede confiar en la información de autenticación de un dispositivo cliente antes de que sea necesario volver a autenticarse con el dispositivo principal. El valor predeterminado es 1.

Example Ejemplo: actualización de la combinación de configuraciones (mediante una política restrictiva)

El siguiente ejemplo de configuración especifica que se permita a los dispositivos cliente cuyos nombres comiencen por `MyClientDevice` conectarse y publicar o suscribirse en todos los temas.

```
{
  "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 Ejemplo: actualización de la combinación de configuraciones (mediante una política permisiva)

El siguiente ejemplo de configuración especifica permitir que todos los dispositivos cliente se conecten y publiquen o se suscriban sobre todos los temas.

```
{
  "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`

Los grupos de dispositivos son grupos de dispositivos cliente que tienen permisos para conectarse y comunicarse con un dispositivo principal. Utilice reglas de selección para identificar grupos de dispositivos cliente y defina políticas de autorización de dispositivos cliente que especifiquen los permisos para cada grupo de dispositivos.

Este objeto contiene la siguiente información:

`formatVersion`

La versión de formato de este objeto de configuración.

Puede elegir entre las siguientes opciones:

- `2021-03-05`

`definitions`

Los grupos de dispositivos de este dispositivo principal. Cada definición especifica una regla de selección para evaluar si un dispositivo cliente es miembro del grupo. Cada definición también especifica la política de permisos que se aplicará a los dispositivos cliente que coincidan con la regla de selección. Si un dispositivo cliente es miembro de varios grupos de dispositivos, los permisos del dispositivo se componen de la política de permisos de cada grupo.

Este objeto contiene la siguiente información:

`groupNameKey`

El nombre de este grupo de dispositivos. *`groupNameKey`*Sustitúyalo por un nombre que ayude a identificar este grupo de dispositivos.

Este objeto contiene la siguiente información:

`selectionRule`

La consulta que especifica qué dispositivos cliente son miembros de este grupo de dispositivos. Cuando un dispositivo cliente se conecta, el dispositivo principal evalúa esta regla de selección para determinar si el dispositivo cliente es miembro de este grupo de dispositivos. Si el dispositivo cliente es miembro, el dispositivo principal utiliza la política de este grupo de dispositivos para autorizar las acciones del dispositivo cliente.

Cada regla de selección incluye al menos una cláusula de regla de selección, que es una consulta de expresión única que puede coincidir con los dispositivos del cliente. Las reglas de selección utilizan la misma sintaxis de consulta que la indexación de AWS IoT flotas. Para obtener más información sobre la sintaxis de las reglas de selección, consulte la sintaxis de las consultas de indexación de AWS IoT flotas en la Guía AWS IoT Core para desarrolladores.

Utilice el * comodín para hacer coincidir varios dispositivos cliente con una cláusula de regla de selección. Puede utilizar este comodín al final del nombre del elemento para hacer coincidir los dispositivos cliente cuyos nombres comiencen por la cadena que especifique. También puede usar este comodín para que coincidan con todos los dispositivos cliente.

> **Note**
>
> Para seleccionar un valor que contenga dos puntos (`:`), evite los dos puntos y coloque un carácter de barra invertida (). \\ En formatos como JSON, debe evitar los caracteres de barra invertida, por lo que debe escribir dos caracteres de barra invertida antes del carácter de dos puntos. Por ejemplo, especifique seleccionar un `thingName: MyTeam\\\\:ClientDevice1` elemento cuyo nombre sea. `MyTeam:ClientDevice1`

Puede especificar el siguiente selector:

- `thingName`— El nombre del dispositivo de un AWS IoT cliente.

Example Ejemplo de regla de selección

La siguiente regla de selección coincide con los dispositivos cliente cuyos nombres son `MyClientDevice1` o`MyClientDevice2`.

```
thingName: MyClientDevice1 OR thingName: MyClientDevice2
```

Example Ejemplo de regla de selección (utilice caracteres comodín)

La siguiente regla de selección coincide con los dispositivos cliente cuyos nombres comiencen por. `MyClientDevice`

```
thingName: MyClientDevice*
```

Example Ejemplo de regla de selección (hacer coincidir todos los dispositivos)

La siguiente regla de selección coincide con todos los dispositivos cliente.

```
thingName: *
```

`policyName`

La política de permisos que se aplica a los dispositivos cliente de este grupo de dispositivos. Especifique el nombre de la política que defina en el `policies` objeto.

`policies`

Las políticas de autorización de los dispositivos cliente para los dispositivos cliente que se conectan al dispositivo principal. Cada política de autorización especifica un conjunto de acciones y los recursos en los que un dispositivo cliente puede realizar esas acciones.

Este objeto contiene la siguiente información:

`policyNameKey`

El nombre de esta política de autorización. *`policyNameKey`*Sustitúyala por un nombre que ayude a identificar esta política de autorización. Este nombre de política se utiliza para definir qué política se aplica a un grupo de dispositivos.

Este objeto contiene la siguiente información:

`statementNameKey`

El nombre de esta declaración de política. *`statementNameKey`*Sustitúyala por un nombre que le ayude a identificar esta declaración de política.

Este objeto contiene la siguiente información:

`operations`

La lista de operaciones que permiten utilizar los recursos de esta política.

Puede incluir cualquiera de las siguientes operaciones:

- `mqtt:connect`— Otorga permiso para conectarse al dispositivo principal. Los dispositivos cliente deben tener este permiso para conectarse a un dispositivo principal.

Esta operación admite los siguientes recursos:

- `mqtt:clientId:`*`deviceClientId`*— Restrinja el acceso en función del ID de cliente que utilice el dispositivo cliente para conectarse al intermediario MQTT del dispositivo principal. *`deviceClientId`*Sustitúyalo por el ID de cliente que se va a utilizar.

- `mqtt:publish`— Otorga permiso para publicar mensajes de MQTT en los temas.

  Esta operación admite los siguientes recursos:

  - `mqtt:topic:`*`mqttTopic`*— Restrinja el acceso en función del tema de MQTT en el que un dispositivo cliente publica un mensaje. Sustituya *`MQTTTopic por el tema`* que desee utilizar.

    Este recurso no admite los caracteres comodín de los temas MQTT.

- `mqtt:subscribe`— Otorga permiso para suscribirse a los filtros de temas de MQTT para recibir mensajes.

  Esta operación admite los siguientes recursos:

  - `mqtt:topicfilter:`*`mqttTopicFilter`*— Restrinja el acceso en función de los temas de MQTT en los que un dispositivo cliente puede suscribirse a los mensajes. *`mqttTopicFilter`*Sustitúyalo por el filtro de temas que desee utilizar.

    Este recurso admite los caracteres comodín de los temas # MQTT + y MQTT. Para obtener más información, consulte los temas de MQTT en la AWS IoT Core Guía para desarrolladores.

    El dispositivo cliente puede suscribirse a los filtros de temas exactos que usted permita. Por ejemplo, si permites que el dispositivo cliente se suscriba al `mqtt:topicfilter:client/+/status` recurso, el dispositivo cliente puede suscribirse, `client/+/status` pero no`client/client1/status`.

Puede especificar el * comodín para permitir el acceso a todas las acciones.

`resources`

La lista de recursos que permiten las operaciones de esta política. Especifique los recursos que corresponden a las operaciones de esta política. Por

ejemplo, puede especificar una lista de recursos de temas de MQTT (`mqtt:topic:`*`mqttTopic`*) en una política que especifique la `mqtt:publish` operación.

Puede especificar el * comodín para permitir el acceso a todos los recursos. No puedes usar el * comodín para hacer coincidir los identificadores de recursos parciales. Por ejemplo, puede especificar**`"resources": "*"`**, pero no puede especificar. **`"resources": "mqtt:clientId:*"`**

`statementDescription`

(Opcional) Una descripción de esta declaración de política.

`certificates`

(Opcional) Las opciones de configuración del certificado para este dispositivo principal. Este objeto contiene la siguiente información:

`serverCertificateValiditySeconds`

(Opcional) Cantidad de tiempo (en segundos) tras la cual caduca el certificado del servidor MQTT local. Puede configurar esta opción para personalizar la frecuencia con la que los dispositivos cliente se desconectan y se vuelven a conectar al dispositivo principal.

Este componente rota el certificado del servidor MQTT local 24 horas antes de que caduque. El broker MQTT, como el componente Broker MQTT de Moquette, genera un nuevo certificado y se reinicia. Cuando esto ocurre, todos los dispositivos cliente conectados a este dispositivo principal se desconectan. Los dispositivos cliente se pueden volver a conectar al dispositivo principal tras un breve período de tiempo.

Predeterminado: `604800` (7 días)

Valor mínimo: `172800` (2 días)

Valor máximo: `864000` (10 días)

`performance`

(Opcional) Las opciones de configuración del rendimiento de este dispositivo principal. Este objeto contiene la siguiente información:

`maxActiveAuthTokens`

(Opcional) El número máximo de tokens de autorización de los dispositivos cliente activos. Puede aumentar este número para permitir que un mayor número de dispositivos cliente se conecten a un dispositivo de un solo núcleo sin volver a autenticarlos.

Valor predeterminado: `2500`

`cloudRequestQueueSize`

(Opcional) El número máximo de Nube de AWS solicitudes que se pueden poner en cola antes de que este componente las rechace.

Valor predeterminado: `100`

`maxConcurrentCloudRequests`

(Opcional) El número máximo de solicitudes simultáneas que se van a enviar al. Nube de AWS Puede aumentar este número para mejorar el rendimiento de la autenticación en los dispositivos principales a los que se conecta un gran número de dispositivos cliente.

Valor predeterminado: `1`

Example Ejemplo: actualización de la combinación de configuraciones (mediante una política restrictiva)

El siguiente ejemplo de configuración especifica que se permita a los dispositivos cliente cuyos nombres comiencen por `MyClientDevice` conectarse y publicar o suscribirse en todos los temas.

```
{
  "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 Ejemplo: actualización de la combinación de configuraciones (mediante una política permisiva)

El siguiente ejemplo de configuración especifica permitir que todos los dispositivos cliente se conecten y publiquen o se suscriban sobre todos los temas.

```
{
  "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`

Los grupos de dispositivos son grupos de dispositivos cliente que tienen permisos para conectarse y comunicarse con un dispositivo principal. Utilice reglas de selección para identificar grupos de dispositivos cliente y defina políticas de autorización de dispositivos cliente que especifiquen los permisos para cada grupo de dispositivos.

Este objeto contiene la siguiente información:

`formatVersion`

La versión de formato de este objeto de configuración.

Puede elegir entre las siguientes opciones:

- `2021-03-05`

`definitions`

Los grupos de dispositivos de este dispositivo principal. Cada definición especifica una regla de selección para evaluar si un dispositivo cliente es miembro del grupo. Cada definición también especifica la política de permisos que se aplicará a los dispositivos

cliente que coincidan con la regla de selección. Si un dispositivo cliente es miembro de varios grupos de dispositivos, los permisos del dispositivo se componen de la política de permisos de cada grupo.

Este objeto contiene la siguiente información:

groupNameKey

El nombre de este grupo de dispositivos. *groupNameKey*Sustitúyalo por un nombre que ayude a identificar este grupo de dispositivos.

Este objeto contiene la siguiente información:

`selectionRule`

La consulta que especifica qué dispositivos cliente son miembros de este grupo de dispositivos. Cuando un dispositivo cliente se conecta, el dispositivo principal evalúa esta regla de selección para determinar si el dispositivo cliente es miembro de este grupo de dispositivos. Si el dispositivo cliente es miembro, el dispositivo principal utiliza la política de este grupo de dispositivos para autorizar las acciones del dispositivo cliente.

Cada regla de selección incluye al menos una cláusula de regla de selección, que es una consulta de expresión única que puede coincidir con los dispositivos del cliente. Las reglas de selección utilizan la misma sintaxis de consulta que la indexación de AWS IoT flotas. Para obtener más información sobre la sintaxis de las reglas de selección, consulte la sintaxis de las consultas de indexación de AWS IoT flotas en la Guía AWS IoT Core para desarrolladores.

Utilice el * comodín para hacer coincidir varios dispositivos cliente con una cláusula de regla de selección. Puede utilizar este comodín al final del nombre del elemento para hacer coincidir los dispositivos cliente cuyos nombres comiencen por la cadena que especifique. También puede usar este comodín para que coincidan con todos los dispositivos cliente.

> **Note**
>
> Para seleccionar un valor que contenga dos puntos (`:`), evite los dos puntos y coloque un carácter de barra invertida (). \\ En formatos como JSON, debe evitar los caracteres de barra invertida, por lo que debe escribir dos caracteres de barra invertida antes del carácter de dos puntos. Por ejemplo,

especifique seleccionar un `thingName: MyTeam\\\\:ClientDevice1` elemento cuyo nombre sea. `MyTeam:ClientDevice1`

Puede especificar el siguiente selector:

- `thingName`— El nombre del dispositivo de un AWS IoT cliente.

Example Ejemplo de regla de selección

La siguiente regla de selección coincide con los dispositivos cliente cuyos nombres son `MyClientDevice1` o`MyClientDevice2`.

```
thingName: MyClientDevice1 OR thingName: MyClientDevice2
```

Example Ejemplo de regla de selección (utilice caracteres comodín)

La siguiente regla de selección coincide con los dispositivos cliente cuyos nombres comiencen por. `MyClientDevice`

```
thingName: MyClientDevice*
```

Example Ejemplo de regla de selección (hacer coincidir todos los dispositivos)

La siguiente regla de selección coincide con todos los dispositivos cliente.

```
thingName: *
```

`policyName`

La política de permisos que se aplica a los dispositivos cliente de este grupo de dispositivos. Especifique el nombre de la política que defina en el `policies` objeto.

`policies`

Las políticas de autorización de los dispositivos cliente para los dispositivos cliente que se conectan al dispositivo principal. Cada política de autorización especifica un conjunto de acciones y los recursos en los que un dispositivo cliente puede realizar esas acciones.

Este objeto contiene la siguiente información:

policyNameKey

El nombre de esta política de autorización. *policyNameKey*Sustitúyala por un nombre que ayude a identificar esta política de autorización. Este nombre de política se utiliza para definir qué política se aplica a un grupo de dispositivos.

Este objeto contiene la siguiente información:

statementNameKey

El nombre de esta declaración de política. *statementNameKey*Sustitúyala por un nombre que le ayude a identificar esta declaración de política.

Este objeto contiene la siguiente información:

`operations`

La lista de operaciones que permiten utilizar los recursos de esta política.

Puede incluir cualquiera de las siguientes operaciones:

- `mqtt:connect`— Otorga permiso para conectarse al dispositivo principal. Los dispositivos cliente deben tener este permiso para conectarse a un dispositivo principal.

  Esta operación admite los siguientes recursos:

  - `mqtt:clientId:`*deviceClientId*— Restrinja el acceso en función del ID de cliente que utilice el dispositivo cliente para conectarse al intermediario MQTT del dispositivo principal. *deviceClientId*Sustitúyalo por el ID de cliente que se va a utilizar.
- `mqtt:publish`— Otorga permiso para publicar mensajes de MQTT en los temas.

  Esta operación admite los siguientes recursos:

  - `mqtt:topic:`*mqttTopic*— Restrinja el acceso en función del tema de MQTT en el que un dispositivo cliente publica un mensaje. Sustituya *MQTTopic por el tema* que desee utilizar.

    Este recurso no admite los caracteres comodín de los temas MQTT.
- `mqtt:subscribe`— Otorga permiso para suscribirse a los filtros de temas de MQTT para recibir mensajes.

Esta operación admite los siguientes recursos:

- `mqtt:topicfilter:`*`mqttTopicFilter`*— Restrinja el acceso en función de los temas de MQTT en los que un dispositivo cliente puede suscribirse a los mensajes. *`mqttTopicFilter`*Sustitúyalo por el filtro de temas que desee utilizar.

  Este recurso admite los caracteres comodín de los temas # MQTT + y MQTT. Para obtener más información, consulte los temas de MQTT en la AWS IoT Core Guía para desarrolladores.

  El dispositivo cliente puede suscribirse a los filtros de temas exactos que usted permita. Por ejemplo, si permites que el dispositivo cliente se suscriba al `mqtt:topicfilter:client/+/status` recurso, el dispositivo cliente puede suscribirse, `client/+/status` pero no`client/client1/status`.

Puede especificar el * comodín para permitir el acceso a todas las acciones.

`resources`

La lista de recursos que permiten las operaciones de esta política. Especifique los recursos que corresponden a las operaciones de esta política. Por ejemplo, puede especificar una lista de recursos de temas de MQTT (`mqtt:topic:`*`mqttTopic`*) en una política que especifique la `mqtt:publish` operación.

Puede especificar el * comodín para permitir el acceso a todos los recursos. No puedes usar el * comodín para hacer coincidir los identificadores de recursos parciales. Por ejemplo, puede especificar**`"resources": "*"`**, pero no puede especificar. **`"resources": "mqtt:clientId:*"`**

`statementDescription`

(Opcional) Una descripción de esta declaración de política.

`certificates`

(Opcional) Las opciones de configuración del certificado para este dispositivo principal. Este objeto contiene la siguiente información:

`serverCertificateValiditySeconds`

(Opcional) Cantidad de tiempo (en segundos) tras la cual caduca el certificado del servidor MQTT local. Puede configurar esta opción para personalizar la frecuencia con la que los dispositivos cliente se desconectan y se vuelven a conectar al dispositivo principal.

Este componente rota el certificado del servidor MQTT local 24 horas antes de que caduque. El broker MQTT, como el componente Broker MQTT de Moquette, genera un nuevo certificado y se reinicia. Cuando esto ocurre, todos los dispositivos cliente conectados a este dispositivo principal se desconectan. Los dispositivos cliente se pueden volver a conectar al dispositivo principal tras un breve período de tiempo.

Predeterminado: `604800` (7 días)

Valor mínimo: `172800` (2 días)

Valor máximo: `864000` (10 días)

Example Ejemplo: actualización de la combinación de configuraciones (mediante una política restrictiva)

El siguiente ejemplo de configuración especifica que se permita a los dispositivos cliente cuyos nombres comiencen por `MyClientDevice` conectarse y publicar o suscribirse en todos los temas.

```
{
  "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 Ejemplo: actualización de la combinación de configuraciones (mediante una política permisiva)

El siguiente ejemplo de configuración especifica permitir que todos los dispositivos cliente se conecten y publiquen o se suscriban sobre todos los temas.

```
{
  "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`

Los grupos de dispositivos son grupos de dispositivos cliente que tienen permisos para conectarse y comunicarse con un dispositivo principal. Utilice reglas de selección para identificar grupos de dispositivos cliente y defina políticas de autorización de dispositivos cliente que especifiquen los permisos para cada grupo de dispositivos.

Este objeto contiene la siguiente información:

`formatVersion`

La versión de formato de este objeto de configuración.

Puede elegir entre las siguientes opciones:

- `2021-03-05`

`definitions`

Los grupos de dispositivos de este dispositivo principal. Cada definición especifica una regla de selección para evaluar si un dispositivo cliente es miembro del grupo. Cada definición también especifica la política de permisos que se aplicará a los dispositivos cliente que coincidan con la regla de selección. Si un dispositivo cliente es miembro de varios grupos de dispositivos, los permisos del dispositivo se componen de la política de permisos de cada grupo.

Este objeto contiene la siguiente información:

groupNameKey

El nombre de este grupo de dispositivos. *groupNameKey*Sustitúyalo por un nombre que ayude a identificar este grupo de dispositivos.

Este objeto contiene la siguiente información:

`selectionRule`

La consulta que especifica qué dispositivos cliente son miembros de este grupo de dispositivos. Cuando un dispositivo cliente se conecta, el dispositivo principal evalúa esta regla de selección para determinar si el dispositivo cliente es miembro de este grupo de dispositivos. Si el dispositivo cliente es miembro, el dispositivo principal utiliza la política de este grupo de dispositivos para autorizar las acciones del dispositivo cliente.

Cada regla de selección incluye al menos una cláusula de regla de selección, que es una consulta de expresión única que puede coincidir con los dispositivos del cliente. Las reglas de selección utilizan la misma sintaxis de consulta que la indexación de AWS IoT flotas. Para obtener más información sobre la sintaxis de las reglas de selección, consulte la sintaxis de las consultas de indexación de AWS IoT flotas en la Guía AWS IoT Core para desarrolladores.

Utilice el * comodín para hacer coincidir varios dispositivos cliente con una cláusula de regla de selección. Puede utilizar este comodín al final del nombre del elemento para hacer coincidir los dispositivos cliente cuyos nombres comiencen por la cadena que especifique. También puede usar este comodín para que coincidan con todos los dispositivos cliente.

> **Note**
>
> Para seleccionar un valor que contenga dos puntos (`:`), evite los dos puntos y coloque un carácter de barra invertida (). \\ En formatos como JSON, debe evitar los caracteres de barra invertida, por lo que debe escribir dos caracteres de barra invertida antes del carácter de dos puntos. Por ejemplo, especifique seleccionar un `thingName: MyTeam\\\\:ClientDevice1` elemento cuyo nombre sea. `MyTeam:ClientDevice1`

Puede especificar el siguiente selector:

- `thingName`— El nombre del dispositivo de un AWS IoT cliente.

Example Ejemplo de regla de selección

La siguiente regla de selección coincide con los dispositivos cliente cuyos nombres son `MyClientDevice1` o`MyClientDevice2`.

```
thingName: MyClientDevice1 OR thingName: MyClientDevice2
```

Example Ejemplo de regla de selección (utilice caracteres comodín)

La siguiente regla de selección coincide con los dispositivos cliente cuyos nombres comiencen por. `MyClientDevice`

```
thingName: MyClientDevice*
```

Example Ejemplo de regla de selección (hacer coincidir todos los dispositivos)

La siguiente regla de selección coincide con todos los dispositivos cliente.

```
thingName: *
```

`policyName`

La política de permisos que se aplica a los dispositivos cliente de este grupo de dispositivos. Especifique el nombre de la política que defina en el `policies` objeto.

`policies`

Las políticas de autorización de los dispositivos cliente para los dispositivos cliente que se conectan al dispositivo principal. Cada política de autorización especifica un conjunto de acciones y los recursos en los que un dispositivo cliente puede realizar esas acciones.

Este objeto contiene la siguiente información:

`policyNameKey`

El nombre de esta política de autorización. *`policyNameKey`*Sustitúyala por un nombre que ayude a identificar esta política de autorización. Este nombre de política se utiliza para definir qué política se aplica a un grupo de dispositivos.

Este objeto contiene la siguiente información:

statementNameKey

El nombre de esta declaración de política. *statementNameKey*Sustitúyala por un nombre que le ayude a identificar esta declaración de política.

Este objeto contiene la siguiente información:

`operations`

La lista de operaciones que permiten utilizar los recursos de esta política.

Puede incluir cualquiera de las siguientes operaciones:

- `mqtt:connect`— Otorga permiso para conectarse al dispositivo principal. Los dispositivos cliente deben tener este permiso para conectarse a un dispositivo principal.

  Esta operación admite los siguientes recursos:

  - `mqtt:clientId:`*deviceClientId*— Restrinja el acceso en función del ID de cliente que utilice el dispositivo cliente para conectarse al intermediario MQTT del dispositivo principal. *deviceClientId*Sustitúyalo por el ID de cliente que se va a utilizar.
- `mqtt:publish`— Otorga permiso para publicar mensajes de MQTT en los temas.

  Esta operación admite los siguientes recursos:

  - `mqtt:topic:`*mqttTopic*— Restrinja el acceso en función del tema de MQTT en el que un dispositivo cliente publica un mensaje. Sustituya *MQTTopic por el tema* que desee utilizar.

    Este recurso no admite los caracteres comodín de los temas MQTT.
- `mqtt:subscribe`— Otorga permiso para suscribirse a los filtros de temas de MQTT para recibir mensajes.

  Esta operación admite los siguientes recursos:

  - `mqtt:topicfilter:`*mqttTopicFilter*— Restrinja el acceso en función de los temas de MQTT en los que un dispositivo cliente puede suscribirse a los mensajes. *mqttTopicFilter*Sustitúyalo por el filtro de temas que desee utilizar.

Este recurso admite los caracteres comodín de los temas # MQTT + y MQTT. Para obtener más información, consulte los temas de MQTT en la AWS IoT Core Guía para desarrolladores.

El dispositivo cliente puede suscribirse a los filtros de temas exactos que usted permita. Por ejemplo, si permites que el dispositivo cliente se suscriba al `mqtt:topicfilter:client/+/status` recurso, el dispositivo cliente puede suscribirse, `client/+/status` pero no`client/client1/status`.

Puede especificar el * comodín para permitir el acceso a todas las acciones.

`resources`

La lista de recursos que permiten las operaciones de esta política. Especifique los recursos que corresponden a las operaciones de esta política. Por ejemplo, puede especificar una lista de recursos de temas de MQTT (`mqtt:topic:`*`mqttTopic`*) en una política que especifique la `mqtt:publish` operación.

Puede especificar el * comodín para permitir el acceso a todos los recursos. No puedes usar el * comodín para hacer coincidir los identificadores de recursos parciales. Por ejemplo, puede especificar**`"resources": "*"`**, pero no puede especificar. **`"resources": "mqtt:clientId:*"`**

`statementDescription`

(Opcional) Una descripción de esta declaración de política.

Example Ejemplo: actualización de la combinación de configuraciones (mediante una política restrictiva)

El siguiente ejemplo de configuración especifica que se permita a los dispositivos cliente cuyos nombres comiencen por `MyClientDevice` conectarse y publicar o suscribirse en todos los temas.

```
{
  "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 Ejemplo: actualización de la combinación de configuraciones (mediante una política permisiva)

El siguiente ejemplo de configuración especifica permitir que todos los dispositivos cliente se conecten y publiquen o se suscriban sobre todos los temas.

```
{
  "deviceGroups": {
    "formatVersion": "2021-03-05",
    "definitions": {
      "MyPermissiveDeviceGroup": {
        "selectionRule": "thingName: *",
        "policyName": "MyPermissivePolicy"
      }
    },
    "policies": {
      "MyPermissivePolicy": {
        "AllowAll": {
          "statementDescription": "Allow client devices to perform all actions.",
          "operations": [
            "*"
          ],
          "resources": [
            "*"
          ]
        }
      }
    }
  }
}
```

Archivo de registro local

Este componente utiliza el mismo archivo de registro que el componente núcleo de Greengrass.

Linux

```
/greengrass/v2/logs/greengrass.log
```

Windows

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

Para ver los registros de este componente

- Ejecute el siguiente comando en el dispositivo principal para ver el archivo de registro de este componente en tiempo real. Sustituya */greengrass/v2* o *C:\greengrass\v2* por la ruta a la carpeta AWS IoT Greengrass raíz.

  Linux

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

  Windows (PowerShell)

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

Registros de cambios

En la siguiente tabla se describen los cambios en cada versión del componente.

Versión	Cambios
2.5.0	Nuevas características • Permite la sustitución de los recursos de políticas por `${iot:Connection.Thing.ThingName}` variables. • Permite los recursos de políticas con caracteres comodín como `mqtt:topic:my*` .
2.4.5	Nuevas características Añade compatibilidad con prefijos comodín para seleccionar nombres de cosas con el parámetro. `selectionRule` Mejoras y correcciones de errores Soluciona un problema por el que, en algunos casos, los certificados no se actualizaban con nueva información de conectividad.
2.4.4	Versión actualizada para la versión 2.12.0 de Greengrass nucleus.
2.4.3	Versión actualizada para la versión 2.11.0 de Greengrass nucleus.

Versión	Cambios
2.4.2	Nuevas características Añade una nueva opción de configuración`startupTimeoutSeconds` .
2.4.1	Versión actualizada para la versión 2.10.0 de Greengrass nucleus.
2.4.0	Nuevas características • Añade compatibilidad con la autenticación del dispositivo cliente para emitir métricas operativas que publicará el agente de telemetría. Mejoras y correcciones de errores • Soluciona un problema por el que la autenticación del dispositivo cliente tarda más de 10 segundos en verificar la identidad de un dispositivo cliente. • Correcciones y mejoras menores adicionales.
2.3.2	Mejoras y correcciones de errores • Añade compatibilidad con el almacenamiento en caché de la informaci ón del nombre de host, de modo que el componente genere correctam ente los sujetos de los certificados cuando se reinicie sin conexión a Internet.
2.3.1	Mejoras y correcciones de errores • Corrige una pérdida de memoria.

Versión	Cambios
2.3.0	**Warning** Esta versión ya no está disponible. Las mejoras de esta versión están disponibles en versiones posteriores de este componente. Nuevas características • Añade compatibilidad con la autenticación sin conexión de los dispositivos cliente para que puedan seguir conectándose al dispositivo principal cuando este no esté conectado a Internet. • Añade compatibilidad con la autoridad de certificación proporcionada por el cliente, que el dispositivo principal utiliza como certificado raíz para generar certificados de intermediario de MQTT.
2.2.3	Versión actualizada para la versión 2.8.0 de Greengrass nucleus.
2.2.2	Mejoras y correcciones de errores • Soluciona un problema por el que el certificado del servidor MQTT local rota con más frecuencia de lo previsto en determinadas situaciones.
2.2.1	Versión actualizada para la versión 2.7.0 de Greengrass nucleus.
2.2.0	Nuevas características • Añade compatibilidad con componentes personalizados para llamar a las operaciones de comunicación entre procesos (IPC) para autenticar y autorizar los dispositivos cliente. Puede utilizar estas operaciones en un componente de intermediario MQTT personalizado, por ejemplo. Para obtener más información, consulte IPC: Autenticar y autorizar dispositivos cliente. • Agrega las `threadPoolSize` opciones `maxActiveAuthTokens` `cloudQueueSize` , y que puede configurar para ajustar el rendimiento de este componente.

Versión	Cambios
2.1.0	Nuevas características • Añade la `serverCertificateValiditySeconds` opción que puede configurar para personalizar la fecha de caducidad del certificado del servidor MQTT broker. Puede configurar el certificado del servidor para que caduque entre 2 y 10 días. Mejoras y correcciones de errores • Soluciona problemas relacionados con la forma en que este component e gestiona las actualizaciones de restablecimiento de la configuración. • Soluciona un problema por el que el certificado del servidor MQTT local rota con más frecuencia de lo previsto en determinadas situaciones. Para aplicar esta corrección, también debe utilizar la versión 2.1.0 o posterior del componente broker MQTT de Moquette. • Mejora los mensajes que este componente registra cuando rota los certificados. • Versión actualizada para la versión 2.6.0 de Greengrass nucleus.
2.0.4	Versión actualizada para la versión 2.5.0 de Greengrass nucleus.
2.0.3	Mejoras y correcciones de errores • Las credenciales ahora se actualizan si se rota la clave privada del dispositivo principal. • Actualizaciones para que los mensajes de registro sean más claros.
2.0.2	Versión actualizada para la versión 2.4.0 de Greengrass nucleus.
2.0.1	Versión actualizada para la versión 2.3.0 de Greengrass nucleus.
2.0.0	Versión inicial.

CloudWatch métricas

El componente de CloudWatch métricas de Amazon (`aws.greengrass.Cloudwatch`) publica métricas personalizadas de los dispositivos principales de Greengrass en Amazon. CloudWatch

El componente permite a los componentes publicar CloudWatch métricas, que puede utilizar para supervisar y analizar el entorno del dispositivo principal de Greengrass. Para obtener más información, consulta Uso de CloudWatch las métricas de Amazon en la Guía del CloudWatch usuario de Amazon.

Para publicar una CloudWatch métrica con este componente, publica un mensaje en un tema al que esté suscrito este componente. De forma predeterminada, este componente se suscribe al tema `cloudwatch/metric/put` local de publicación/suscripción. Puede especificar otros temas, incluidos los temas de AWS IoT Core MQTT, al implementar este componente.

Este componente agrupa las métricas que se encuentran en el mismo espacio de nombres y las publica a CloudWatch intervalos regulares.

> **Note**
>
> Este componente proporciona una funcionalidad similar a la del conector de CloudWatch métricas de la versión 1. AWS IoT Greengrass Para obtener más información, consulte el conector de CloudWatch métricas en la Guía para AWS IoT Greengrass desarrolladores de la versión 1.

Temas

Versiones

Este componente tiene las siguientes versiones:

- 3.1.x
- 3.0.x
- 2.1.x
- 2.0.x

Para obtener información sobre los cambios en cada versión del componente, consulte el registro de cambios.

Tipo

v3.x

Este componente es un componente genérico () `aws.greengrass.generic`. El núcleo de Greengrass ejecuta los scripts del ciclo de vida del componente.

v2.x

Este componente es un componente Lambda () `aws.greengrass.lambda`. El núcleo de Greengrass ejecuta la función Lambda de este componente mediante el componente Lambda launcher.

Para obtener más información, consulte Tipos de componentes.

Sistema operativo

v3.x

Este componente se puede instalar en los dispositivos principales que ejecutan los siguientes sistemas operativos:

- Linux
- Windows

v2.x

Este componente solo se puede instalar en los dispositivos principales de Linux.

Requisitos

Este componente tiene los siguientes requisitos:

3.x

- Versión 3.7 de Python instalada en el dispositivo principal y añadida a la variable de entorno PATH.
- El rol de dispositivo de Greengrass debe permitir la `cloudwatch:PutMetricData` acción, como se muestra en el siguiente ejemplo de política de IAM.

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

Para obtener más información, consulta la referencia de CloudWatch permisos de Amazon en la Guía del CloudWatch usuario de Amazon.

2.x

- El dispositivo principal debe cumplir los requisitos para ejecutar las funciones de Lambda. Si desea que el dispositivo principal ejecute funciones Lambda en contenedores, el dispositivo debe cumplir los requisitos para hacerlo. Para obtener más información, consulte Requisitos de la función de Lambda.
- Versión 3.7 de Python instalada en el dispositivo principal y añadida a la variable de entorno PATH.
- El rol de dispositivo de Greengrass debe permitir la `cloudwatch:PutMetricData` acción, como se muestra en el siguiente ejemplo de política de IAM.

```
{
```

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

Para obtener más información, consulta la referencia de CloudWatch permisos de Amazon en la Guía del CloudWatch usuario de Amazon.

- Para recibir los datos de salida de este componente, debe combinar la siguiente actualización de configuración para el componente antiguo del router de suscripciones (`aws.greengrass.LegacySubscriptionRouter`) al implementar este componente. Esta configuración especifica el tema en el que este componente publica las respuestas.

  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"
  ```

```
    }
  }
}
```

- Sustituya la *región* por la Región de AWS que utilice.
- Sustituya la *versión* por la versión de la función Lambda que ejecuta este componente. Para encontrar la versión de la función Lambda, debe ver la receta de la versión de este componente que desee implementar. Abra la página de detalles de este componente en la AWS IoT Greengrass consola y busque el par clave-valor de la función Lambda. Este par clave-valor contiene el nombre y la versión de la función Lambda.

> **Important**
>
> Debe actualizar la versión de la función Lambda en el router de suscripción anterior cada vez que implemente este componente. Esto garantiza que utilice la versión correcta de la función Lambda para la versión del componente que implemente.

Para obtener más información, consulte Crear implementaciones.

Puntos finales y puertos

Este componente debe poder realizar solicitudes salientes a los siguientes puntos finales y puertos, además de a los puntos finales y puertos necesarios para el funcionamiento básico. Para obtener más información, consulte Permitir el tráfico del dispositivo a través de un proxy o firewall.

punto de enlace	Puerto	Obligatoria	Descripción
`monitoring. region.amazonaws.com`	443	Sí	Cargue métricas. CloudWatch

Dependencias

Al implementar un componente, AWS IoT Greengrass también despliega versiones compatibles de sus dependencias. Esto significa que debe cumplir los requisitos del componente y de todas sus dependencias para poder implementarlo correctamente. En esta sección se enumeran las dependencias de las versiones publicadas de este componente y las restricciones de las versiones semánticas que definen las versiones de los componentes para cada dependencia. También puede ver las dependencias de cada versión del componente en la consola. AWS IoT Greengrass En la página de detalles del componente, busque la lista de dependencias.

3.0.0 - 3.1.0

La siguiente tabla muestra las dependencias de las versiones 3.0.0 a 3.1.0 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <3.0.0	Flexible
Servicio de intercambio de fichas	>=0.0.0	Rígido

2.1.2 and 2.1.3

La siguiente tabla muestra las dependencias de las versiones 2.1.2 y 2.1.3 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.8.0	Rígido
Lanzador Lambda	^2.0.0	Rígido
Tiempos de ejecución de Lambda	^2.0.0	Flexible
Servicio de intercambio de fichas	^2.0.0	Rígido

2.1.1

La siguiente tabla muestra las dependencias de la versión 2.1.1 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.7.0	Rígido
Lanzador Lambda	^2.0.0	Rígido
Tiempos de ejecución de Lambda	^2.0.0	Flexible
Servicio de intercambio de fichas	^2.0.0	Rígido

2.0.8 - 2.1.0

La siguiente tabla muestra las dependencias de las versiones 2.0.8 a 2.1.0 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.6.0	Rígido
Lanzador Lambda	^2.0.0	Rígido
Tiempos de ejecución de Lambda	^2.0.0	Flexible
Servicio de intercambio de fichas	^2.0.0	Rígido

2.0.7

La siguiente tabla muestra las dependencias de la versión 2.0.7 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.5.0	Rígido
Lanzador Lambda	^2.0.0	Rígido
Tiempos de ejecución de Lambda	^2.0.0	Flexible
Servicio de intercambio de fichas	^2.0.0	Rígido

2.0.6

La siguiente tabla muestra las dependencias de la versión 2.0.6 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.4.0	Rígido
Lanzador Lambda	^2.0.0	Rígido
Tiempos de ejecución de Lambda	^2.0.0	Flexible
Servicio de intercambio de fichas	^2.0.0	Rígido

2.0.5

La siguiente tabla muestra las dependencias de la versión 2.0.5 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.3.0	Rígido
Lanzador Lambda	^2.0.0	Rígido

Dependencia	Versiones compatibles	Tipo de dependencia
Tiempos de ejecución de Lambda	^2.0.0	Flexible
Servicio de intercambio de fichas	^2.0.0	Rígido

2.0.4

La siguiente tabla muestra las dependencias de la versión 2.0.4 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.2.0	Rígido
Lanzador Lambda	^2.0.0	Rígido
Tiempos de ejecución de Lambda	^2.0.0	Flexible
Servicio de intercambio de fichas	^2.0.0	Rígido

2.0.3

La siguiente tabla muestra las dependencias de la versión 2.0.3 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.3 <2.1.0	Rígido
Lanzador Lambda	>=1.0.0	Rígido
Tiempos de ejecución de Lambda	>=1.0.0	Flexible

Dependencia	Versiones compatibles	Tipo de dependencia
Servicio de intercambio de fichas	>=1.0.0	Rígido

Para obtener más información sobre las dependencias de los componentes, consulta la referencia de recetas de componentes.

Configuración

Este componente proporciona los siguientes parámetros de configuración que puede personalizar al implementar el componente.

v3.x

`PublishInterval`

(Opcional) El número máximo de segundos que se deben esperar antes de que el componente publique métricas por lotes para un espacio de nombres determinado. Para configurar el componente para que publique las métricas a medida que las reciba, es decir, sin procesamiento por lotes, especifique. `0`

El componente publica CloudWatch después de recibir 20 métricas en el mismo espacio de nombres o después del intervalo que especifique.

> **Note**
>
> El componente no especifica el orden en el que se publican los eventos.

Este valor puede ser de 900 segundos como máximo.

Predeterminado: 10 segundos

`MaxMetricsToRetain`

(Opcional) El número máximo de métricas en todos los espacios de nombres que se deben guardar en la memoria antes de que el componente las sustituya por métricas más recientes.

Este límite se aplica cuando el dispositivo principal no tiene conexión a Internet, por lo que el componente almacena las métricas en un búfer para publicarlas más adelante. Cuando el

búfer está lleno, el componente reemplaza las métricas más antiguas por otras más nuevas. Las métricas de un espacio de nombres determinado solo sustituyen a las métricas del mismo espacio de nombres.

> **Note**
>
> Si el proceso anfitrión del componente se interrumpe, el componente no guarda las métricas. Esto puede ocurrir durante una implementación o cuando el dispositivo principal se reinicia, por ejemplo.

Este valor debe ser de al menos 2000 métricas.

Predeterminado: 5000 métricas

`InputTopic`

(Opcional) El tema al que se suscribe el componente para recibir mensajes. Si lo especifica `truePubSubToIoTCore`, puede usar los comodines MQTT (+ y #) en este tema.

Predeterminado: `cloudwatch/metric/put`

`OutputTopic`

(Opcional) El tema en el que el componente publica las respuestas de estado.

Predeterminado: `cloudwatch/metric/put/status`

`PubSubToIoTCore`

(Opcional) Valor de cadena que define si se deben publicar o suscribirse a los temas de AWS IoT Core MQTT. Los valores admitidos son `true` y `false`.

Predeterminado: `false`

`UseInstaller`

(Opcional) Valor booleano que define si se debe utilizar el script de instalación de este componente para instalar las dependencias del SDK de este componente.

Establezca este valor `false` si desea utilizar un script personalizado para instalar las dependencias o si desea incluir las dependencias en tiempo de ejecución en una imagen de Linux prediseñada. Para usar este componente, debe instalar las siguientes bibliotecas,

incluidas las dependencias, y ponerlas a disposición del usuario predeterminado del sistema Greengrass.

- SDK para dispositivos con AWS IoT v2 para Python
- AWS SDK for Python (Boto3)

Predeterminado: `true`

`PublishRegion`

(Opcional) El lugar en el Región de AWS que se van a publicar CloudWatch las métricas. Este valor anula la región predeterminada del dispositivo principal. Este parámetro solo es necesario para las métricas entre regiones.

`accessControl`

(Opcional) El objeto que contiene la política de autorización que permite al componente publicar y suscribirse a los temas especificados. Si especifica valores personalizados para `InputTopic` y`OutputTopic`, debe actualizar los valores de los recursos de este objeto.

Predeterminado:

```
{
  "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 Ejemplo: actualización de combinación de configuraciones

```
{
  "PublishInterval": 0,
  "PubSubToIoTCore": true
}
```

v2.x

> **Note**
>
> La configuración predeterminada de este componente incluye los parámetros de la función Lambda. Le recomendamos que edite solo los siguientes parámetros para configurar este componente en sus dispositivos.

`lambdaParams`

Objeto que contiene los parámetros de la función Lambda de este componente. Este objeto contiene la siguiente información:

`EnvironmentVariables`

Objeto que contiene los parámetros de la función Lambda. Este objeto contiene la siguiente información:

`PUBLISH_INTERVAL`

(Opcional) El número máximo de segundos que se deben esperar antes de que el componente publique métricas por lotes para un espacio de nombres determinado. Para configurar el componente para que publique las métricas a medida que las reciba, es decir, sin procesamiento por lotes, especifique. `0`

El componente publica CloudWatch después de recibir 20 métricas en el mismo espacio de nombres o después del intervalo que especifique.

> **Note**
>
> El componente no garantiza el orden en el que se publican los eventos.

Este valor puede ser como máximo de 900 segundos.

Predeterminado: 10 segundos

`MAX_METRICS_TO_RETAIN`

(Opcional) El número máximo de métricas en todos los espacios de nombres que se deben guardar en la memoria antes de que el componente las sustituya por métricas más recientes.

Este límite se aplica cuando el dispositivo principal no tiene conexión a Internet, por lo que el componente almacena las métricas en un búfer para publicarlas más adelante. Cuando el búfer está lleno, el componente reemplaza las métricas más antiguas por otras más nuevas. Las métricas de un espacio de nombres determinado solo sustituyen a las métricas del mismo espacio de nombres.

> **Note**
>
> Si el proceso anfitrión del componente se interrumpe, el componente no guarda las métricas. Esto puede ocurrir durante una implementación o cuando el dispositivo principal se reinicia, por ejemplo.

Este valor debe ser de al menos 2000 métricas.

Predeterminado: 5000 métricas

`PUBLISH_REGION`

(Opcional) El lugar en el Región de AWS que se van a publicar CloudWatch las métricas. Este valor anula la región predeterminada del dispositivo principal. Este parámetro solo es necesario para las métricas entre regiones.

`containerMode`

(Opcional) El modo de contenedorización de este componente. Puede elegir entre las siguientes opciones:

- `NoContainer`— El componente no se ejecuta en un entorno de ejecución aislado.
- `GreengrassContainer`— El componente se ejecuta en un entorno de ejecución aislado dentro del AWS IoT Greengrass contenedor.

Predeterminado: `GreengrassContainer`

`containerParams`

(Opcional) Un objeto que contiene los parámetros del contenedor de este componente. El componente utiliza estos parámetros si se especifica `GreengrassContainer` para`containerMode`.

Este objeto contiene la siguiente información:

`memorySize`

(Opcional) La cantidad de memoria (en kilobytes) que se va a asignar al componente.

El valor predeterminado es 64 MB (65.535 KB).

`pubsubTopics`

(Opcional) Objeto que contiene los temas a los que el componente se suscribe para recibir mensajes. Puede especificar cada tema y si el componente se suscribe a los temas de MQTT AWS IoT Core o a los temas de publicación/suscripción locales.

Este objeto contiene la siguiente información:

`0`— Se trata de un índice matricial en forma de cadena.

Objeto que contiene la siguiente información:

`type`

(Opcional) El tipo de mensajes de publicación/suscripción que utiliza este componente para suscribirse a los mensajes. Puede elegir entre las siguientes opciones:

- `PUB_SUB` — Suscribirse a mensajes locales de publicación/suscripción. Si elige esta opción, el tema no puede contener caracteres comodín de MQTT. Para obtener más información sobre cómo enviar mensajes desde un componente personalizado al especificar esta opción, consulte. Publicar/suscribir mensajes locales
- `IOT_CORE`— Suscríbase a los mensajes AWS IoT Core MQTT. Si elige esta opción, el tema puede contener caracteres comodín de MQTT. Para obtener más información sobre cómo enviar mensajes desde componentes personalizados al especificar esta opción, consulte. Publicar/suscribir mensajes MQTT AWS IoT Core

Predeterminado: `PUB_SUB`

`topic`

(Opcional) El tema al que se suscribe el componente para recibir mensajes. Si lo especifica `IotCoretype`, puede usar los comodines MQTT (+y#) en este tema.

Example Ejemplo: actualización de la combinación de configuraciones (modo contenedor)

```
{
  "containerMode": "GreengrassContainer"
}
```

Example Ejemplo: actualización de la combinación de configuraciones (sin modo contenedor)

```
{
  "containerMode": "NoContainer"
}
```

Datos de entrada

Este componente acepta métricas del tema siguiente y las publica en CloudWatch. De forma predeterminada, este componente se suscribe a los mensajes locales de publicación/suscripción. Para obtener más información sobre cómo publicar mensajes en este componente desde sus componentes personalizados, consulte. Publicar/suscribir mensajes locales

A partir de la versión 3.0.0 del componente, si lo desea, puede configurar este componente para que se suscriba a un tema de MQTT estableciendo el parámetro de `PubSubToIoTCore` configuración en. `true` Para obtener más información sobre la publicación de mensajes en un tema de MQTT en sus componentes personalizados, consulte. Publicar/suscribir mensajes MQTT AWS IoT Core

Tema predeterminado: `cloudwatch/metric/put`

El mensaje acepta las siguientes propiedades. Los mensajes de entrada deben tener un formato JSON válido.

`request`

La métrica de este mensaje.

El objeto de la solicitud contiene los datos de métricas que se publicarán en CloudWatch. Los valores métricos deben cumplir las especificaciones de la `PutMetricData`operación.

Tipo: `object` que contiene la siguiente información:

`namespace`

El espacio de nombres definido por el usuario para los datos de las métricas de esta solicitud. CloudWatch utiliza los espacios de nombres como contenedores para los puntos de datos métricos.

> **Note**
>
> No se puede especificar un espacio de nombres que comience la cadena reservada por `AWS/`.

Tipo: `string`

Patrón válido: `[^:].*`

`metricData`

Los datos de la métrica.

Tipo: `object` que contiene la siguiente información:

`metricName`

El nombre de la métrica.

Tipo: `string`

`value`

El valor de la métrica.

> **Note**
>
> CloudWatch rechaza los valores demasiado pequeños o demasiado grandes. El valor debe estar comprendido entre `8.515920e-109` y `1.174271e+108` (Base 10) o `2e-360` y `2e360` (Base 2). CloudWatch no admite valores especiales como `NaN`, `+Infinity`, `y-Infinity`.

Tipo: `double`

`dimensions`

(Opcional) Las dimensiones de la métrica. Las dimensiones proporcionan información adicional acerca de la métrica y sus datos. Una métrica puede definir hasta 10 dimensiones.

Este componente incluye automáticamente una dimensión con un nombre`coreName`, donde el valor es el nombre del dispositivo principal.

Tipo: `array` de objetos, cada uno de los cuales contiene la siguiente información:

`name`

(Opcional) El nombre de la dimensión.

Tipo: `string`

`value`

(Opcional) El valor de la dimensión.

Tipo: `string`

`timestamp`

(Opcional) La hora a la que se recibieron los datos de la métrica, expresada en segundos en el tiempo de época de Unix.

El valor predeterminado es la hora a la que el componente recibe el mensaje.

Tipo: `double`

> **Note**
>
> Si utiliza las versiones 2.0.3 y 2.0.7 de este componente, le recomendamos que recupere la marca de tiempo por separado para cada métrica cuando envíe varias métricas desde una sola fuente. No utilice una variable para almacenar la marca de tiempo.

`unit`

(Opcional) La unidad de la métrica.

Tipo: `string`

Valores válidos:`Seconds`,`Microseconds`,`Milliseconds`,`Bytes`,`Kilobytes`,`Megabytes`,`Gigabytes`,`T`
`Second`,`Kilobytes/Second`,`Megabytes/Second`,`Gigabytes/Second`,`Terabytes/`
`Second`,`Bits/Second`,`Kilobits/Second`,`Megabits/Second`,`Gigabits/`
`Second`,`Terabits/Second`,`Count/Second`, `None`

El valor predeterminado es `None`.

> **Note**
>
> Todas las cuotas que se aplican a la CloudWatch `PutMetricData` API se aplican a las métricas que publiques con este componente. Las cuotas siguientes son especialmente importantes:
>
> - Límite de 40 KB en la carga útil de la API
> - 20 métricas por solicitud de API
> - 150 transacciones por segundo (TPS) para la API de `PutMetricData`
>
> Para obtener más información, consulte las cuotas CloudWatch de servicio en la Guía del CloudWatch usuario.

Example Ejemplo de entrada

```
{
  "request": {
    "namespace": "Greengrass",
    "metricData": {
      "metricName": "latency",
      "dimensions": [
        {
          "name": "hostname",
          "value": "test_hostname"
        }
      ],
      "timestamp": 1539027324,
      "value": 123.0,
      "unit": "Seconds"
    }
  }
}
```

Datos de salida

De forma predeterminada, este componente publica las respuestas como datos de salida sobre el siguiente tema local de publicación/suscripción. Para obtener más información sobre cómo suscribirse a los mensajes sobre este tema en sus componentes personalizados, consulte. Publicar/suscribir mensajes locales

Si lo desea, puede configurar este componente para que se publique en un tema de MQTT estableciendo el parámetro `PubSubToIoTCore` de configuración en. `true` Para obtener más información sobre cómo suscribirse a mensajes sobre un tema de MQTT en sus componentes personalizados, consulte. Publicar/suscribir mensajes MQTT AWS IoT Core

> **Note**
>
> Las versiones 2.0.x de los componentes publican las respuestas como datos de salida sobre un tema de MQTT de forma predeterminada. Debe especificar el tema como parte de la `subject` configuración del componente del router de suscripciones antiguo.

Tema predeterminado: `cloudwatch/metric/put/status`

Example Ejemplo de salida: Correcto

La respuesta incluye el espacio de nombres de los datos de la métrica y el `RequestId` campo de la CloudWatch respuesta.

```
{
  "response": {
    "cloudwatch_rid": "70573243-d723-11e8-b095-75ff2EXAMPLE",
    "namespace": "Greengrass",
    "status": "success"
  }
}
```

Example Ejemplo de salida: Error

```
{
  "response" : {
    "namespace": "Greengrass",
    "error": "InvalidInputException",
    "error_message": "cw metric is invalid",
    "status": "fail"
  }
}
```

> **Note**
>
> Si el componente detecta un error que se puede volver a intentar, como un error de conexión, volverá a intentar la publicación en el siguiente lote.

Licencias

Este componente incluye el siguiente software o licencia de terceros:

- AWS SDK for Python (Boto3)/Apache License 2.0
- botocore/Apache License 2.0
- dateutil/PSF License
- docutils/BSD License, GNU General Public License (GPL), Python Software Foundation License, Public Domain

- jmespath/MIT License
- s3transfer/Apache License 2.0
- urllib3/MIT License

Este componente se publica en virtud del contrato de licencia de software principal de Greengrass.

Archivo de registro local

Este componente utiliza el siguiente archivo de registro.

Linux

```
/greengrass/v2/logs/aws.greengrass.Cloudwatch.log
```

Windows

```
C:\greengrass\v2\logs\aws.greengrass.Cloudwatch.log
```

Para ver los registros de este componente

- Ejecute el siguiente comando en el dispositivo principal para ver el archivo de registro de este componente en tiempo real. Sustituya */greengrass/v2* o *C:\greengrass\v2* por la ruta a la carpeta AWS IoT Greengrass raíz.

  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
  ```

Registros de cambios

En la siguiente tabla se describen los cambios en cada versión del componente.

v3.x

Versión	Cambios
3.1.0	Mejoras y correcciones de errores • Añade compatibilidad con las configuraciones de proxy de red HTTPS. Para obtener más información, consulte Realizar la conexión en el puerto 443 o a través de un proxy de red y Habilite el dispositivo principal para que confíe en un proxy HTTPS.
3.0.0	Esta versión del componente de CloudWatch métricas espera parámetros de configuración diferentes a los de la versión 2.x. Si utiliza una configuración no predeterminada para la versión 2.x y desea actualizar de la v2.x a la v3.x, debe actualizar la configuración del componente. Para obtener más información, consulte la configuración de los componentes de métricas. CloudWatch Nuevas características • Añade compatibilidad con los dispositivos principales que ejecutan Windows. • Cambia el tipo de componente de componente Lambda a componente genérico. Este componente ya no depende del componente antiguo del router de suscripciones para crear suscripciones. • Agrega un nuevo parámetro de `InputTopic` configuración para especificar el tema al que se suscribe el componente para recibir mensajes. • Agrega un nuevo parámetro `OutputTopic` de configuración para especificar el tema en el que el componente publica las respuestas de estado. • Añade un nuevo parámetro `PubSubToIoTCore` de configuración para especificar si se deben publicar o suscribirse a los temas de AWS IoT Core MQTT. • Añade el nuevo parámetro `UseInstaller` de configuración que permite deshabilitar opcionalmente el script de instalación que instala las dependencias de los componentes.

Versión	Cambios
	Mejoras y correcciones de errores Añade compatibilidad con marcas de tiempo duplicadas en los datos de entrada.

v2.x

Versión	Cambios
2.1.3	Versión actualizada para la versión 2.11.0 de Greengrass nucleus.
2.1.2	Versión actualizada para la versión 2.7.0 de Greengrass nucleus.
2.1.1	Versión actualizada para la versión 2.6.0 de Greengrass nucleus.
2.1.0	Nuevas características • Añade compatibilidad con las configuraciones de proxy de red HTTPS. Para obtener más información, consulte Realizar la conexión en el puerto 443 o a través de un proxy de red y Habilite el dispositivo principal para que confíe en un proxy HTTPS.
2.0.8	Mejoras y correcciones de errores • Añade compatibilidad con marcas de tiempo duplicadas en los datos de entrada. • Versión actualizada para la versión 2.5.0 de Greengrass nucleus.
2.0.7	Versión actualizada para la versión 2.4.0 de Greengrass nucleus.
2.0.6	Versión actualizada para la versión 2.3.0 de Greengrass nucleus.
2.0.5	Versión actualizada para la versión 2.2.0 de Greengrass nucleus.
2.0.4	Versión actualizada para la versión 2.1.0 de Greengrass nucleus.
2.0.3	Versión inicial.

Véase también

- Uso de CloudWatch las métricas de Amazon en la Guía del CloudWatch usuario de Amazon
- PutMetricDataen la referencia de la CloudWatch API de Amazon

AWS IoT Device Defender

El AWS IoT Device Defender componente (`aws.greengrass.DeviceDefender`) notifica a los administradores los cambios en el estado de los dispositivos principales de Greengrass. Esto puede ayudar a identificar comportamiento inusual que podrían indicar un dispositivo en riesgo. Para obtener más información, consulte AWS IoT Device Defender en la Guía para desarrolladores de AWS IoT Core .

Este componente lee las métricas del sistema del dispositivo principal. A continuación, publica las métricas en AWS IoT Device Defender. Para obtener más información sobre cómo leer e interpretar las métricas que informa este componente, consulte las especificaciones del documento sobre las métricas de los dispositivos en la Guía para AWS IoT Core desarrolladores.

> **Note**
>
> Este componente proporciona una funcionalidad similar a la del conector Device Defender incorporado AWS IoT Greengrass V1. Para obtener más información, consulte el conector Device Defender en la Guía para AWS IoT Greengrass V1 desarrolladores.

Temas

- Versiones
- Tipo
- Sistema operativo
- Requisitos
- Dependencias
- Configuración
- Datos de entrada
- Datos de salida
- Archivo de registro local

- Licencias
- Registros de cambios

Versiones

Este componente tiene las siguientes versiones:

- 3.1.x
- 3.0.x
- 2.0.x

Para obtener información sobre los cambios en cada versión del componente, consulte el registro de cambios.

Tipo

v3.x

Este componente es un componente genérico () `aws.greengrass.generic`. El núcleo de Greengrass ejecuta los scripts del ciclo de vida del componente.

v2.x

Este componente es un componente Lambda () `aws.greengrass.lambda`. El núcleo de Greengrass ejecuta la función Lambda de este componente mediante el componente Lambda launcher.

Para obtener más información, consulte Tipos de componentes.

Sistema operativo

v3.x

Este componente se puede instalar en los dispositivos principales que ejecutan los siguientes sistemas operativos:

- Linux
- Windows

v2.x

Este componente solo se puede instalar en los dispositivos principales de Linux.

Requisitos

Este componente tiene los siguientes requisitos:

v3.x

- Versión 3.7 de Python instalada en el dispositivo principal y añadida a la variable de entorno PATH.
- AWS IoT Device Defender configurado para utilizar la función de detección para supervisar las infracciones. Para obtener más información, consulte Detectar en la Guía para desarrolladores de AWS IoT Core .

v2.x

- El dispositivo principal debe cumplir los requisitos para ejecutar las funciones de Lambda. Si desea que el dispositivo principal ejecute funciones Lambda en contenedores, el dispositivo debe cumplir los requisitos para hacerlo. Para obtener más información, consulte Requisitos de la función de Lambda.
- Versión 3.7 de Python instalada en el dispositivo principal y añadida a la variable de entorno PATH.
- AWS IoT Device Defender configurado para utilizar la función de detección para supervisar las infracciones. Para obtener más información, consulte Detectar en la Guía para desarrolladores de AWS IoT Core .
- La biblioteca psutil instalada en el dispositivo principal. La versión 5.7.0 es la última versión que se ha comprobado que funciona con el componente.
- La biblioteca cbor instalada en el dispositivo principal. La versión 1.0.0 es la última versión que se ha comprobado que funciona con el componente.
- Para recibir los datos de salida de este componente, debe combinar la siguiente actualización de configuración para el componente antiguo del enrutador de suscripciones (`aws.greengrass.LegacySubscriptionRouter`) al implementar este componente. Esta configuración especifica el tema en el que este componente publica las respuestas.

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"
    }
  }
}
```

- Sustituya la *región* por la Región de AWS que utilice.
- Sustituya la *versión* por la versión de la función Lambda que ejecuta este componente. Para encontrar la versión de la función Lambda, debe ver la receta de la versión de este componente que desee implementar. Abra la página de detalles de este componente en la AWS IoT Greengrass consola y busque el par clave-valor de la función Lambda. Este par clave-valor contiene el nombre y la versión de la función Lambda.

> **Important**
>
> Debe actualizar la versión de la función Lambda en el router de suscripción anterior cada vez que implemente este componente. Esto garantiza que utilice la versión correcta de la función Lambda para la versión del componente que implemente.

Para obtener más información, consulte Crear implementaciones.

Dependencias

Al implementar un componente, AWS IoT Greengrass también implementa versiones compatibles de sus dependencias. Esto significa que debe cumplir los requisitos del componente y de todas sus dependencias para poder implementarlo correctamente. En esta sección se enumeran las dependencias de las versiones publicadas de este componente y las restricciones de las versiones semánticas que definen las versiones de los componentes para cada dependencia. También puede ver las dependencias de cada versión del componente en la consola. AWS IoT Greengrass En la página de detalles del componente, busque la lista de dependencias.

3.1.1

La siguiente tabla muestra las dependencias de la versión 3.1.1 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <3.0.0	Flexible
Servicio de intercambio de fichas	>=0.0.0	Rígido

3.0.0 - 3.0.2

La siguiente tabla muestra las dependencias de las versiones 3.0.0 a 3.0.2 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <3.0.0	Flexible
Servicio de intercambio de fichas	>=0.0.0	Rígido

2.0.10 and 2.0.11

La siguiente tabla muestra las dependencias de las versiones 2.0.10 y 2.0.11 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.8.0	Rígido
Lanzador Lambda	^2.0.0	Rígido
Tiempos de ejecución de Lambda	^2.0.0	Flexible
Servicio de intercambio de fichas	^2.0.0	Rígido

2.0.9

La siguiente tabla muestra las dependencias de la versión 2.0.9 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.7.0	Rígido
Lanzador Lambda	^2.0.0	Rígido
Tiempos de ejecución de Lambda	^2.0.0	Flexible
Servicio de intercambio de fichas	^2.0.0	Rígido

2.0.8

La siguiente tabla muestra las dependencias de la versión 2.0.8 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.6.0	Rígido
Lanzador Lambda	^2.0.0	Rígido
Tiempos de ejecución de Lambda	^2.0.0	Flexible
Servicio de intercambio de fichas	^2.0.0	Rígido

2.0.7

La siguiente tabla muestra las dependencias de la versión 2.0.7 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.5.0	Rígido
Lanzador Lambda	^2.0.0	Rígido
Tiempos de ejecución de Lambda	^2.0.0	Flexible
Servicio de intercambio de fichas	^2.0.0	Rígido

2.0.6

La siguiente tabla muestra las dependencias de la versión 2.0.6 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.4.0	Rígido
Lanzador Lambda	^2.0.0	Rígido

Dependencia	Versiones compatibles	Tipo de dependencia
Tiempos de ejecución de Lambda	^2.0.0	Flexible
Servicio de intercambio de fichas	^2.0.0	Rígido

2.0.5

La siguiente tabla muestra las dependencias de la versión 2.0.5 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.3.0	Rígido
Lanzador Lambda	^2.0.0	Rígido
Tiempos de ejecución de Lambda	^2.0.0	Flexible
Servicio de intercambio de fichas	^2.0.0	Rígido

2.0.4

La siguiente tabla muestra las dependencias de la versión 2.0.4 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.2.0	Rígido
Lanzador Lambda	^2.0.0	Rígido
Tiempos de ejecución de Lambda	^2.0.0	Flexible

Dependencia	Versiones compatibles	Tipo de dependencia
Servicio de intercambio de fichas	^2.0.0	Rígido

2.0.3

La siguiente tabla muestra las dependencias de la versión 2.0.3 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.3 <2.1.0	Rígido
Lanzador Lambda	>=1.0.0	Rígido
Tiempos de ejecución de Lambda	>=1.0.0	Flexible
Servicio de intercambio de fichas	>=1.0.0	Rígido

Para obtener más información sobre las dependencias de los componentes, consulta la referencia de recetas de componentes.

Configuración

Este componente proporciona los siguientes parámetros de configuración que puede personalizar al implementar el componente.

v3.x

`PublishRetryCount`

El número de veces que se volverá a intentar la publicación. Esta función está disponible en la versión 3.1.1.

El mínimo es 0.

El máximo es 72.

Predeterminado: 5

`SampleIntervalSeconds`

(Opcional) La cantidad de tiempo en segundos entre cada ciclo en el que el componente recopila las métricas e informa de ellas.

El valor mínimo es de 300 segundos (5 minutos).

Predeterminado: 300 segundos

`UseInstaller`

(Opcional) Valor booleano que define si se debe utilizar el script de instalación de este componente para instalar las dependencias de este componente.

Establezca este valor `false` si desea utilizar un script personalizado para instalar las dependencias o si desea incluir las dependencias en tiempo de ejecución en una imagen de Linux prediseñada. Para usar este componente, debe instalar las siguientes bibliotecas, incluidas las dependencias, y ponerlas a disposición del usuario predeterminado del sistema Greengrass.

- SDK para dispositivos con AWS IoT v2 para Python
- biblioteca cbor. La versión 1.0.0 es la última versión que se ha comprobado que funciona con el componente.
- biblioteca psutil. La versión 5.7.0 es la última versión que se ha comprobado que funciona con el componente.

> **Note**
>
> Si utiliza la versión 3.0.0 o 3.0.1 de este componente en los dispositivos principales que configura para usar un proxy HTTPS, debe establecer este valor en. `false` El script de instalación no admite el funcionamiento detrás de un proxy HTTPS en estas versiones de este componente.

Predeterminado: `true`

v2.x

> **Note**
>
> La configuración predeterminada de este componente incluye los parámetros de la función Lambda. Le recomendamos que edite solo los siguientes parámetros para configurar este componente en sus dispositivos.

`lambdaParams`

Objeto que contiene los parámetros de la función Lambda de este componente. Este objeto contiene la siguiente información:

`EnvironmentVariables`

Objeto que contiene los parámetros de la función Lambda. Este objeto contiene la siguiente información:

`PROCFS_PATH`

(Opcional) La ruta a la `/proc` carpeta.

- Para ejecutar este componente en un contenedor, utilice el valor predeterminado,`/host-proc`. El componente se ejecuta en un contenedor de forma predeterminada.
- Para ejecutar este componente en modo sin contenedor, especifique `/proc` este parámetro.

Predeterminado: `/host-proc`. Esta es la ruta por defecto en la que este componente monta la `/proc` carpeta en el contenedor.

> **Note**
>
> Este componente tiene acceso de solo lectura a esta carpeta.

`SAMPLE_INTERVAL_SECONDS`

(Opcional) La cantidad de tiempo en segundos entre cada ciclo en el que el componente recopila las métricas e informa de ellas.

El valor mínimo es de 300 segundos (5 minutos).

Predeterminado: 300 segundos

`containerMode`

(Opcional) El modo de contenerización de este componente. Puede elegir entre las siguientes opciones:

- `GreengrassContainer`— El componente se ejecuta en un entorno de ejecución aislado dentro del AWS IoT Greengrass contenedor.
- `NoContainer`— El componente no se ejecuta en un entorno de ejecución aislado.

  Si especifica esta opción, debe especificar el parámetro `/proc` de la variable de `PROCFS_PATH` entorno.

Predeterminado: `GreengrassContainer`

`containerParams`

(Opcional) Un objeto que contiene los parámetros del contenedor de este componente. El componente utiliza estos parámetros si se especifica `GreengrassContainer` para`containerMode`.

Este objeto contiene la siguiente información:

`memorySize`

(Opcional) La cantidad de memoria (en kilobytes) que se va a asignar al componente.

El valor predeterminado es 50 000 KB.

`pubsubTopics`

(Opcional) Objeto que contiene los temas a los que el componente se suscribe para recibir mensajes. Puede especificar cada tema y si el componente se suscribe a los temas de MQTT AWS IoT Core o a los temas de publicación/suscripción locales.

Este objeto contiene la siguiente información:

`0`— Se trata de un índice matricial en forma de cadena.

Objeto que contiene la siguiente información:

`type`

(Opcional) El tipo de mensajes de publicación/suscripción que utiliza este componente para suscribirse a los mensajes. Puede elegir entre las siguientes opciones:

- PUB_SUB — Suscribirse a mensajes locales de publicación/suscripción. Si elige esta opción, el tema no puede contener caracteres comodín de MQTT. Para obtener más información sobre cómo enviar mensajes desde un componente personalizado al especificar esta opción, consulte. Publicar/suscribir mensajes locales
- `IOT_CORE`— Suscríbase a los mensajes de AWS IoT Core MQTT. Si elige esta opción, el tema puede contener caracteres comodín de MQTT. Para obtener más información sobre cómo enviar mensajes desde componentes personalizados al especificar esta opción, consulte. Publicar/suscribir mensajes MQTT AWS IoT Core

Predeterminado: PUB_SUB

`topic`

(Opcional) El tema al que se suscribe el componente para recibir mensajes. Si lo especifica `IotCoretype`, puede usar los comodines MQTT (+y#) en este tema.

Example Ejemplo: actualización de la combinación de configuraciones (modo contenedor)

```
{
  "lambdaExecutionParameters": {
    "EnvironmentVariables": {
      "PROCFS_PATH": "/host_proc"
    }
  },
  "containerMode": "GreengrassContainer"
}
```

Example Ejemplo: actualización de la combinación de configuraciones (sin modo contenedor)

```
{
  "lambdaExecutionParameters": {
    "EnvironmentVariables": {
      "PROCFS_PATH": "/proc"
    }
  },
  "containerMode": "NoContainer"
```

```
}
```

Datos de entrada

Este componente no acepta mensajes como datos de entrada.

Datos de salida

Este componente publica las métricas de seguridad en el siguiente tema reservado para AWS IoT Device Defender. Este componente se *coreDeviceName* sustituye por el nombre del dispositivo principal cuando publica las métricas.

Tema (AWS IoT Core MQTT): `$aws/things/coreDeviceName/defender/metrics/json`

Example Ejemplo de resultados

```
{
  "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
      }
    }
  }
}
```


Para obtener más información sobre las métricas que informa este componente, consulte las especificaciones del documento sobre las métricas de los dispositivos en la Guía para AWS IoT Core desarrolladores.

Archivo de registro local

Este componente utiliza el siguiente archivo de registro.

Linux

```
/greengrass/v2/logs/aws.greengrass.DeviceDefender.log
```

Windows

```
C:\greengrass\v2\logs\aws.greengrass.DeviceDefender.log
```

Para ver los registros de este componente

- Ejecute el siguiente comando en el dispositivo principal para ver el archivo de registro de este componente en tiempo real. Sustituya */greengrass/v2* o *C:\greengrass\v2* por la ruta a la carpeta AWS IoT Greengrass raíz.

  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
  ```

Licencias

Este componente se publica en virtud del contrato de licencia de software principal de Greengrass.

Registros de cambios

En la siguiente tabla se describen los cambios en cada versión del componente.

v3.x

Versión	Cambios
3.1.1	Mejoras y correcciones de errores • Añade reintentos para la conexión del cliente cuando la conexión no se recupera tras una interrupción de la red.

Versión	Cambios
	• Añade un reintento configurable para publicar métricas.
3.1.0	Mejoras y correcciones de errores • Añade compatibilidad con las configuraciones de proxy de red HTTPS. Para obtener más información, consulte Realizar la conexión en el puerto 443 o a través de un proxy de red y Habilite el dispositivo principal para que confíe en un proxy HTTPS.
3.0.1	Soluciona un problema relacionado con la forma en que el componente calcula los valores delta de las métricas.
3.0.0	⚠ Warning Esta versión ya no está disponible. Las mejoras de esta versión están disponibles en versiones posteriores de este componente. Versión inicial.

v2.x

Versión	Cambios
2.0.11	Versión actualizada para la versión 2.11.0 de Greengrass nucleus.
2.0.10	Versión actualizada para la versión 2.7.0 de Greengrass Nucleus.
2.0.9	Versión actualizada para la versión 2.6.0 de Greengrass nucleus.
2.0.8	Versión actualizada para la versión 2.5.0 de Greengrass nucleus.
2.0.7	Versión actualizada para la versión 2.4.0 de Greengrass nucleus.
2.0.6	Versión actualizada para la versión 2.3.0 de Greengrass nucleus.

Versión	Cambios
2.0.5	Versión actualizada para el lanzamiento de la versión 2.2.0 de Greengrass nucleus.
2.0.4	Versión actualizada para el lanzamiento de la versión 2.1.0 de Greengrass nucleus.
2.0.3	Versión inicial.

Bobina de disco

El componente disk spooler (`aws.greengrass.DiskSpooler`) ofrece una opción de almacenamiento persistente para los mensajes enviados desde los dispositivos principales de Greengrass a. AWS IoT Core Este componente almacenará estos mensajes salientes en el disco.

Temas

- Versiones
- Tipo
- Sistema operativo
- Requisitos
- Dependencias
- Uso
- Archivo de registro local
- Registros de cambios

Versiones

Este componente tiene las siguientes versiones:

- 1.0.x

Tipo

Este componente es un componente de complemento (`aws.greengrass.plugin`). El núcleo de Greengrass ejecuta este componente en la misma máquina virtual Java (JVM) que el núcleo. El núcleo se reinicia al cambiar la versión de este componente en el dispositivo principal.

Este componente utiliza el mismo archivo de registro que el núcleo de Greengrass. Para obtener más información, consulte Supervisar AWS IoT Greengrass registros.

Para obtener más información, consulte Tipos de componentes.

Sistema operativo

Este componente se puede instalar en los dispositivos principales que ejecutan los siguientes sistemas operativos:

- Linux
- Windows

Requisitos

Este componente tiene los siguientes requisitos:

- `storageType`debe configurarse `Disk` para usar este componente. Puede configurarlo en la configuración del núcleo de Greengrass.
- `maxSizeInBytes`no debe configurarse para que supere el espacio disponible en el dispositivo. Puede configurarlo en la configuración del núcleo de Greengrass.
- Se admite la ejecución del componente disk spooler en una VPC.

Dependencias

Al implementar un componente, AWS IoT Greengrass también despliega versiones compatibles de sus dependencias. Esto significa que debe cumplir los requisitos del componente y de todas sus dependencias para poder implementarlo correctamente. En esta sección se enumeran las dependencias de las versiones publicadas de este componente y las restricciones de las versiones semánticas que definen las versiones de los componentes para cada dependencia. También puede ver las dependencias de cada versión del componente en la consola. AWS IoT Greengrass En la página de detalles del componente, busque la lista de dependencias.

1.0.1 – 1.0.3

La siguiente tabla muestra las dependencias de las versiones 1.0.1 a 1.0.3 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.11.0 <2.13.0	Rígido

1.0.0

La siguiente tabla muestra las dependencias de la versión 1.0.0 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.11.0 <2.12.0	Rígido

Para obtener más información sobre las dependencias de los componentes, consulte la referencia de recetas de componentes.

Uso

Para utilizar el componente disk spooler, `aws.greengrass.DiskSpooler` debe estar implementado.

Para configurar y utilizar este componente, debe establecer en. `pluginName` `aws.greengrass.DiskSpooler`

Archivo de registro local

Este componente utiliza el mismo archivo de registro que el componente núcleo de Greengrass.

Linux

```
/greengrass/v2/logs/greengrass.log
```

Windows

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

Para ver los registros de este componente

- Ejecute el siguiente comando en el dispositivo principal para ver el archivo de registro de este componente en tiempo real. Sustituya */greengrass/v2* o *C:\greengrass\v2* por la ruta a la carpeta AWS IoT Greengrass raíz.

  Linux

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

  Windows (PowerShell)

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

Registros de cambios

En la siguiente tabla se describen los cambios en cada versión del componente.

Versión	Cambios
1.0.3	Mejoras y correcciones de errores Mejora el rendimiento al reutilizar las conexiones de bases de datos.
1.0.2	Mejoras y correcciones de errores Soluciona un problema por el que el campo de formato de mensaje MQTT no se conserva en algunos casos.
1.0.1	Versión actualizada para la versión 2.12.0 de Greengrass Nucleus.
1.0.0	Versión inicial.

Gestor de aplicaciones Docker

El componente gestor de aplicaciones de Docker (`aws.greengrass.DockerApplicationManager`) AWS IoT Greengrass permite descargar imágenes de Docker de registros de imágenes públicos y registros privados alojados en

Amazon Elastic Container Registry (Amazon ECR). También permite gestionar las credenciales automáticamente AWS IoT Greengrass para descargar imágenes de forma segura desde repositorios privados en Amazon ECR.

Cuando desarrolle un componente personalizado que ejecute un contenedor de Docker, incluya el administrador de aplicaciones de Docker como dependencia para descargar las imágenes de Docker especificadas como artefactos en su componente. Para obtener más información, consulte Ejecute un contenedor Docker.

Temas

Versiones

Este componente tiene las siguientes versiones:

- 2.0.x

Tipo

Este componente es un componente genérico () `aws.greengrass.generic`. El núcleo de Greengrass ejecuta los scripts del ciclo de vida del componente.

Para obtener más información, consulte Tipos de componentes.

Sistema operativo

Este componente se puede instalar en los dispositivos principales que ejecutan los siguientes sistemas operativos:

- Linux
- Windows

Requisitos

Este componente tiene los siguientes requisitos:

- Docker Engine 1.9.1 o posterior instalado en el dispositivo principal de Greengrass. Se ha comprobado que la versión 20.10 es la última versión que funciona con el software Core. AWS IoT Greengrass Debe instalar Docker directamente en el dispositivo principal antes de implementar los componentes que ejecutan contenedores de Docker.
- El daemon de Docker se inició y se ejecutó en el dispositivo principal antes de implementar este componente.
- Imágenes de Docker almacenadas en una de las siguientes fuentes de imágenes compatibles:
  - Repositorios de imágenes públicos y privados en Amazon Elastic Container Registry (Amazon ECR)
  - Repositorio público de Docker Hub
  - Registro público de confianza de Docker
- Las imágenes de Docker se incluyen como artefactos en sus componentes de contenedor Docker personalizados. Usa los siguientes formatos de URI para especificar tus imágenes de Docker:
  - Imagen privada de Amazon ECR: `docker:`*`account-id`*`.dkr.ecr.`*`region`*`.amazonaws.com/`*`repository`*`/`*`image`*`[`*`:tag|@digest`*`]`
  - Imagen pública de Amazon ECR: `docker:public.ecr.aws/`*`repository`*`/`*`image`*`[`*`:tag|@digest`*`]`
  - Imagen pública de Docker Hub: `docker:`*`name`*`[`*`:tag|@digest`*`]`

  Para obtener más información, consulte Ejecute un contenedor Docker.

  > **Note**
  >
  > Si no especificas la etiqueta de la imagen o el resumen de la imagen en el URI del artefacto de una imagen, el administrador de aplicaciones de Docker extrae la última versión disponible de esa imagen al implementar tu componente de contenedor de Docker personalizado. Para asegurarte de que todos tus dispositivos principales ejecuten la misma

versión de una imagen, te recomendamos que incluyas la etiqueta o el resumen de la imagen en el URI del artefacto.

- El usuario del sistema que ejecute un componente contenedor de Docker debe tener permisos de raíz o administrador, o bien debe configurar Docker para que se ejecute como un usuario no de raíz o no administrador.
  - En los dispositivos Linux, puedes añadir un usuario al `docker` grupo sin `sudo` el cual ejecutar `docker` comandos.
  - En los dispositivos Windows, puede añadir un usuario al `docker-users` grupo para que invoque `docker` comandos sin privilegios de administrador.

  Linux or Unix

  Para añadir `ggc_user` al `docker` grupo el usuario no root que utiliza para ejecutar los componentes del contenedor de Docker, ejecute el siguiente comando.

  ```
  sudo usermod -aG docker ggc_user
  ```

  Para obtener más información, consulta Administrar Docker como usuario no root.

  Windows Command Prompt (CMD)

  Para añadir `ggc_user` al `docker-users` grupo o el usuario que utiliza para ejecutar los componentes del contenedor de Docker, ejecute el siguiente comando como administrador.

  ```
  net localgroup docker-users ggc_user /add
  ```

  Windows PowerShell

  Para añadir `ggc_user` al `docker-users` grupo o al usuario que utiliza para ejecutar los componentes del contenedor de Docker, ejecute el siguiente comando como administrador.

  ```
  Add-LocalGroupMember -Group docker-users -Member ggc_user
  ```

- Si configura el software AWS IoT Greengrass Core para usar un proxy de red, debe configurar Docker para que use el mismo servidor proxy.
- Si sus imágenes de Docker están almacenadas en un registro privado de Amazon ECR, debe incluir el componente del servicio de intercambio de tokens como una dependencia en el componente contenedor de Docker. Además, el rol de dispositivo

de Greengrass debe permitir las `ecr:GetDownloadUrlForLayer` acciones, y `ecr:GetAuthorizationTokenecr:BatchGetImage`, como se muestra en el siguiente ejemplo de política de IAM.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "ecr:GetAuthorizationToken",
        "ecr:BatchGetImage",
        "ecr:GetDownloadUrlForLayer"
      ],
      "Resource": [
        "*"
      ],
      "Effect": "Allow"
    }
  ]
}
```

- Se admite la ejecución del componente docker Application Manager en una VPC. Para implementar este componente en una VPC, se requiere lo siguiente.
  - El componente docker application manager debe tener conectividad para descargar imágenes. Por ejemplo, si usa ECR, debe tener conectividad con los siguientes puntos finales.
    - `*.dkr.ecr.region.amazonaws.com`(punto final de VPC) `com.amazonaws.region.ecr.dkr`
    - `api.ecr.region.amazonaws.com`(punto final de VPC) `com.amazonaws.region.ecr.api`

Puntos finales y puertos

Este componente debe poder realizar solicitudes salientes a los siguientes puntos finales y puertos, además de a los puntos finales y puertos necesarios para el funcionamiento básico. Para obtener más información, consulte Permitir el tráfico del dispositivo a través de un proxy o firewall.

punto de enlace	Puerto	Obligatoria	Descripción
`ecr.region.amazonaws.com`	443	No	Obligatorio si descarga imágenes de Docker de Amazon ECR.

`hub.docker.com`
 	
`registry.hub.docker.com/v1`
	443	No	Necesario si descargas imágenes de Docker desde Docker Hub.

Dependencias

Al implementar un componente, AWS IoT Greengrass también despliega versiones compatibles de sus dependencias. Esto significa que debe cumplir los requisitos del componente y de todas sus dependencias para poder implementarlo correctamente. En esta sección se enumeran las dependencias de las versiones publicadas de este componente y las restricciones de las versiones semánticas que definen las versiones de los componentes para cada dependencia. También puede ver las dependencias de cada versión del componente en la consola. AWS IoT Greengrass En la página de detalles del componente, busque la lista de dependencias.

2.0.11

La siguiente tabla muestra las dependencias de la versión 2.0.11 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.1.0 <2.13.0	Flexible

2.0.10

La siguiente tabla muestra las dependencias de la versión 2.0.10 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.1.0 <2.12.0	Flexible

2.0.9

La siguiente tabla muestra las dependencias de la versión 2.0.9 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.1.0 <2.11.0	Flexible

2.0.8

La siguiente tabla muestra las dependencias de la versión 2.0.8 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.1.0 <2.10.0	Flexible

2.0.7

La siguiente tabla muestra las dependencias de la versión 2.0.7 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.1.0 <2.9.0	Flexible

2.0.6

La siguiente tabla muestra las dependencias de la versión 2.0.6 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.1.0 <2.8.0	Flexible

2.0.5

La siguiente tabla muestra las dependencias de la versión 2.0.5 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.1.0 <2.7.0	Flexible

2.0.4

La siguiente tabla muestra las dependencias de la versión 2.0.4 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.1.0 <2.6.0	Flexible

2.0.3

La siguiente tabla muestra las dependencias de la versión 2.0.3 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.1.0 <2.5.0	Flexible

2.0.2

La siguiente tabla muestra las dependencias de la versión 2.0.2 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.1.0 <2.4.0	Flexible

2.0.1

La siguiente tabla muestra las dependencias de la versión 2.0.1 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.1.0 <2.3.0	Flexible

2.0.0

La siguiente tabla muestra las dependencias de la versión 2.0.0 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.1.0 <2.2.0	Flexible

Para obtener más información sobre las dependencias de los componentes, consulta la referencia de recetas de componentes.

Configuración

Este componente no tiene ningún parámetro de configuración.

Archivo de registro local

Este componente utiliza el mismo archivo de registro que el componente núcleo de Greengrass.

Linux

```
/greengrass/v2/logs/greengrass.log
```

Windows

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

Para ver los registros de este componente

- Ejecute el siguiente comando en el dispositivo principal para ver el archivo de registro de este componente en tiempo real. Sustituya */greengrass/v2* o *C:\greengrass\v2* por la ruta a la carpeta AWS IoT Greengrass raíz.

  Linux

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

  Windows (PowerShell)

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

Registros de cambios

En la siguiente tabla se describen los cambios en cada versión del componente.

Versión	Cambios
2.0.11	Versión actualizada para la versión 2.12.0 de Greengrass nucleus.
2.0.10	Versión actualizada para el lanzamiento de la versión 2.11.0 de Greengrass nucleus.
2.0.9	Versión actualizada para la versión 2.10.0 de Greengrass nucleus.
2.0.8	Versión actualizada para la versión 2.9.0 de Greengrass Nucleus.
2.0.7	Versión actualizada para el lanzamiento de la versión 2.8.0 de Greengrass nucleus.
2.0.6	Versión actualizada para el lanzamiento de la versión 2.7.0 de Greengrass nucleus.
2.0.5	Versión actualizada para la versión 2.6.0 de Greengrass Nucleus.
2.0.4	Versión actualizada para el lanzamiento de la versión 2.5.0 de Greengrass nucleus.

Versión	Cambios
2.0.3	Versión actualizada para la versión 2.4.0 de Greengrass Nucleus.
2.0.2	Versión actualizada para el lanzamiento de la versión 2.3.0 de Greengrass nucleus.
2.0.1	Versión actualizada para el lanzamiento de la versión 2.2.0 de Greengrass nucleus.
2.0.0	Versión inicial.

Véase también

- Ejecute un contenedor Docker

Conector Edge para Kinesis Video Streams

El conector perimetral para el componente Kinesis Video Streams `aws.iot.EdgeConnectorForKVS` () lee las transmisiones de vídeo de las cámaras locales y publica las transmisiones en Kinesis Video Streams. Puede configurar este componente para leer las transmisiones de vídeo de las cámaras de protocolo de Internet (IP) mediante el protocolo de transmisión en tiempo real (RTSP). A continuación, puede configurar paneles en los servidores Grafana gestionados por Amazon o Grafana locales para supervisar las transmisiones de vídeo e interactuar con ellas.

Puede integrar este componente AWS IoT TwinMaker para mostrar y controlar las transmisiones de vídeo en los paneles de Grafana. AWS IoT TwinMaker es un AWS servicio que le permite crear gemelos digitales operativos de sistemas físicos. Puede utilizarlo AWS IoT TwinMaker para visualizar los datos de los sensores, las cámaras y las aplicaciones empresariales con el fin de realizar un seguimiento de sus fábricas, edificios o plantas industriales físicas. También puede utilizar estos datos para supervisar las operaciones, diagnosticar errores y repararlos. Para obtener más información, consulte ¿Qué es AWS IoT TwinMaker? en la Guía AWS IoT TwinMaker del usuario.

Este componente almacena su configuración en AWS IoT SiteWise, que es un AWS servicio que modela y almacena datos industriales. En AWS IoT SiteWise, los activos representan objetos como dispositivos, equipos o grupos de otros objetos. Para configurar y usar este componente, debe crear un AWS IoT SiteWise activo para cada dispositivo principal de Greengrass y para cada cámara

IP conectada a cada dispositivo principal. Cada activo tiene propiedades que se configuran para controlar funciones, como la transmisión en directo, la carga a pedido y el almacenamiento en caché local. Para especificar la URL de cada cámara, debe crear un archivo secreto AWS Secrets Manager que contenga la URL de la cámara. Si la cámara requiere autenticación, también debe especificar un nombre de usuario y una contraseña en la URL. A continuación, especifique ese secreto en una propiedad de activo de la cámara IP.

Este componente carga la transmisión de vídeo de cada cámara a una transmisión de vídeo de Kinesis. Debe especificar el nombre de la transmisión de vídeo de Kinesis de destino en la configuración de AWS IoT SiteWise activos de cada cámara. Si la transmisión de vídeo de Kinesis no existe, este componente la crea automáticamente.

AWS IoT TwinMaker proporciona un script que puede ejecutar para crear estos AWS IoT SiteWise activos y los secretos de Secrets Manager. Para obtener más información sobre cómo crear estos recursos y cómo instalar, configurar y usar este componente, consulte la integración de AWS IoT TwinMaker vídeo en la Guía del AWS IoT TwinMaker usuario.

> **Note**
>
> El conector perimetral para el componente Kinesis Video Streams solo está disponible en las Regiones de AWS siguientes ubicaciones:
>
> - Este de EE. UU. (Norte de Virginia)
> - Oeste de EE. UU. (Oregón)
> - Europa (Fráncfort)
> - Europa (Irlanda)
> - Asia-Pacífico (Singapur)

Temas

- Licencias
- Uso
- Archivo de registro local
- Registros de cambios
- Véase también

Versiones

Este componente tiene las siguientes versiones:

- 1.0.x

Tipo

Este componente es un componente genérico (`aws.greengrass.generic`). El núcleo de Greengrass ejecuta los scripts del ciclo de vida del componente.

Para obtener más información, consulte Tipos de componentes.

Sistema operativo

Este componente solo se puede instalar en los dispositivos principales de Linux.

Requisitos

Este componente tiene los siguientes requisitos:

- Puede implementar este componente solo en dispositivos de un solo núcleo, ya que la configuración del componente debe ser única para cada dispositivo principal. No puede implementar este componente en grupos de dispositivos principales.
- GStreamer 1.18.4 o una versión posterior instalada en el dispositivo principal. Para obtener más información, consulte Instalación de GStreamer.

  En un dispositivo con `apt`, puede ejecutar los siguientes comandos para instalar 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
```

- Un AWS IoT SiteWise activo para cada dispositivo principal. Este AWS IoT SiteWise activo representa el dispositivo principal. Para obtener más información sobre cómo crear este recurso, consulte la integración de AWS IoT TwinMaker vídeo en la Guía del AWS IoT TwinMaker usuario.
- Un AWS IoT SiteWise recurso para cada cámara IP que se conecte a cada dispositivo principal. Estos AWS IoT SiteWise activos representan las cámaras que transmiten vídeo a cada dispositivo principal. El activo de cada cámara debe estar asociado al activo del dispositivo principal que se conecta a la cámara. Los activos de cámara tienen propiedades que puede configurar para especificar una transmisión de vídeo de Kinesis, un secreto de autenticación y parámetros de transmisión de vídeo. Para obtener más información sobre cómo crear y configurar los activos de la cámara, consulte la integración de AWS IoT TwinMaker vídeo en la Guía del AWS IoT TwinMaker usuario.
- Un AWS Secrets Manager secreto para cada cámara IP. Este secreto debe definir un par clave-valor, donde está `RTSPStreamUrl` la clave y el valor es la URL de la cámara. Si la cámara requiere autenticación, incluya el nombre de usuario y la contraseña en esta URL. Puede usar un script para crear un secreto al crear los recursos que requiere este componente. Para obtener más información, consulte la integración de AWS IoT TwinMaker vídeo en la Guía del AWS IoT TwinMaker usuario.

  También puedes usar la consola y la API de Secrets Manager para crear secretos adicionales. Para obtener más información, consulte Crear un secreto en la Guía del AWS Secrets Manager usuario.
- La función de intercambio de tokens de Greengrass debe permitir las siguientes acciones y las de Kinesis Video Streams AWS Secrets Manager AWS IoT SiteWise, como se muestra en el siguiente ejemplo de política de IAM.

> **Note**
>
> Este ejemplo de política permite al dispositivo obtener el valor de los secretos denominados y. **`IPCamera1Url`** **`IPCamera2Url`** Al configurar cada cámara IP, se especifica un secreto que contiene la URL de esa cámara. Si la cámara requiere autenticación, también debe especificar un nombre de usuario y una contraseña en la URL. La función de intercambio de fichas del dispositivo principal debe permitir el acceso al secreto de cada cámara IP a la que se conecte.

```
{
  "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**
>
> Si utilizas una AWS Key Management Service clave gestionada por el cliente para cifrar los secretos, la función del dispositivo también debe permitir la `kms:Decrypt` acción.

Puntos finales y puertos

Este componente debe poder realizar solicitudes salientes a los siguientes puntos finales y puertos, además de a los puntos finales y puertos necesarios para el funcionamiento básico. Para obtener más información, consulte Permitir el tráfico del dispositivo a través de un proxy o firewall.

punto de enlace	Puerto	Obligatoria	Descripción
`kinesisvideo.` *`region`*`.amazonaws.com`	443	Sí	Cargue datos a Kinesis Video Streams.
`data.iotsitewise.` *`region`*`.amazonaws.com`	443	Sí	Publique los metadatos de la transmisión de vídeo en AWS IoT SiteWise.
`secretsmanager.` *`region`*`.amazonaws.com`	443	Sí	Descarga los secretos de las URL de la cámara en

punto de enlace	Puerto	Obligatoria	Descripción
			el dispositivo principal.

Dependencias

Al implementar un componente, AWS IoT Greengrass también despliega versiones compatibles de sus dependencias. Esto significa que debe cumplir los requisitos del componente y de todas sus dependencias para poder implementarlo correctamente. En esta sección se enumeran las dependencias de las versiones publicadas de este componente y las restricciones de las versiones semánticas que definen las versiones de los componentes para cada dependencia. También puede ver las dependencias de cada versión del componente en la consola.AWS IoT Greengrass En la página de detalles del componente, busque la lista de dependencias.

La siguiente tabla muestra las dependencias de las versiones 1.0.0 a 1.0.5 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Servicio de intercambio de fichas	>=2.0.3	Rígido
Gestor de transmisiones	>=2.0.9	Rígido

Para obtener más información sobre las dependencias de los componentes, consulta la referencia de recetas de componentes.

Configuración

Este componente proporciona los siguientes parámetros de configuración que puede personalizar al implementar el componente.

`SiteWiseAssetIdForHub`

El ID del AWS IoT SiteWise activo que representa este dispositivo principal. Para obtener más información sobre cómo crear este recurso y usarlo para interactuar con este componente, consulte la integración de AWS IoT TwinMaker vídeo en la Guía del AWS IoT TwinMaker usuario.

Example Ejemplo: actualización de combinación de configuraciones

```
{
  "SiteWiseAssetIdForHub": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
}
```

Licencias

Este componente incluye el siguiente software o licencias de terceros:

- Quartz Job Scheduler/Licencia Apache 2.0
- Enlaces de Java para GStreamer 1.x/GNU Lesser General Public License v3.0

Uso

Para configurar este componente e interactuar con él, puede configurar las propiedades de AWS IoT SiteWise los activos que representan el dispositivo principal y las cámaras IP a las que se conecta. También puede visualizar e interactuar con las transmisiones de vídeo en los paneles de Grafana a través de. AWS IoT TwinMaker Para obtener más información, consulte la integración de AWS IoT TwinMaker vídeo en la Guía del AWS IoT TwinMaker usuario.

Archivo de registro local

Este componente utiliza el siguiente archivo de registro.

```
/greengrass/v2/logs/aws.iot.EdgeConnectorForKVS.log
```

Para ver los registros de este componente

- Ejecute el siguiente comando en el dispositivo principal para ver el archivo de registro de este componente en tiempo real. */greengrass/v2*Sustitúyalo por la ruta a la carpeta AWS IoT Greengrass raíz.

```
sudo tail -f /greengrass/v2/logs/aws.iot.EdgeConnectorForKVS.log
```

Registros de cambios

En la siguiente tabla se describen los cambios en cada versión del componente.

Versión	Cambios
1.0.5	Corrección de errores y mejoras generales.
1.0.4	Mejoras y correcciones de errores • Corrige un problema que provocaba que se detuviera la carga en directo.
1.0.3	Corrección de errores y mejoras generales.
1.0.1	Corrección de errores y mejoras generales.
1.0.0	Versión inicial.

Véase también

- ¿Qué es? AWS IoT TwinMaker en la Guía AWS IoT TwinMaker del usuario
- AWS IoT TwinMaker integración de vídeo en la Guía AWS IoT TwinMaker del usuario
- ¿Qué es AWS IoT SiteWise? en la Guía AWS IoT SiteWise del usuario
- Actualización de los valores de los atributos en la Guía AWS IoT SiteWise del usuario
- ¿Qué es AWS Secrets Manager? en la Guía del usuario de AWS Secrets Manager
- Cree y gestione secretos en la Guía AWS Secrets Manager del usuario

Greengrass CLI

El componente CLI de Greengrass (`aws.greengrass.Cli`) proporciona una interfaz de línea de comandos local que puede usar en los dispositivos principales para desarrollar y depurar componentes localmente. La CLI de Greengrass le permite crear despliegues locales y reiniciar componentes en el dispositivo principal, por ejemplo.

Puede instalar este componente al instalar el software AWS IoT Greengrass principal. Para obtener más información, consulte Tutorial: Introducción a AWS IoT Greengrass V2.

> **Important**
>
> Se recomienda utilizar este componente únicamente en entornos de desarrollo, no en entornos de producción. Este componente proporciona acceso a información y operaciones que normalmente no necesitará en un entorno de producción. Siga el principio de privilegios mínimos implementando este componente solo en los dispositivos principales donde lo necesite.

Tras instalar este componente, ejecute el siguiente comando para ver su documentación de ayuda. Cuando se instala este componente, añade un enlace simbólico a `greengrass-cli` la */greengrass/v2*`/bin` carpeta. Puede ejecutar la CLI de Greengrass desde esta ruta o añadirla a su variable de `PATH` entorno para que se ejecute `greengrass-cli` sin su ruta absoluta.

Linux or Unix

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

Windows

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

El siguiente comando reinicia un componente denominado`com.example.HelloWorld`, por ejemplo.

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"
```

Para obtener más información, consulte Interfaz de línea de comandos Greengrass.

Temas

Versiones

Este componente tiene las siguientes versiones:

- 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

Tipo

Este componente es un componente de complemento ()`aws.greengrass.plugin`. El núcleo de Greengrass ejecuta este componente en la misma máquina virtual Java (JVM) que el núcleo. El núcleo se reinicia al cambiar la versión de este componente en el dispositivo principal.

Este componente utiliza el mismo archivo de registro que el núcleo de Greengrass. Para obtener más información, consulte Supervisar AWS IoT Greengrass registros.

Para obtener más información, consulte Tipos de componentes.

Sistema operativo

Este componente se puede instalar en los dispositivos principales que ejecutan los siguientes sistemas operativos:

- Linux
- Windows

Requisitos

Este componente tiene los siguientes requisitos:

- Debe estar autorizado a utilizar la CLI de Greengrass para interactuar con el software AWS IoT Greengrass principal. Realice una de las siguientes acciones para usar la CLI de Greengrass:
  - Utilice el usuario del sistema que ejecuta el software AWS IoT Greengrass Core.
  - Utilice un usuario con permisos root o administrativos. En los dispositivos principales de Linux, puede utilizarlos `sudo` para obtener permisos de root.
  - Utilice un usuario del sistema que esté en un grupo que especifique en los parámetros de `AuthorizedWindowsGroups` configuración `AuthorizedPosixGroups` o al implementar el componente. Para obtener más información, consulte Configuración de componentes CLI de Greengrass.
- Se admite la ejecución del componente CLI de Greengrass en una VPC.

Dependencias

Al implementar un componente, AWS IoT Greengrass también despliega versiones compatibles de sus dependencias. Esto significa que debe cumplir los requisitos del componente y de todas

sus dependencias para poder implementarlo correctamente. En esta sección se enumeran las dependencias de las versiones publicadas de este componente y las restricciones de las versiones semánticas que definen las versiones de los componentes para cada dependencia. También puede ver las dependencias de cada versión del componente en la consola.AWS IoT Greengrass En la página de detalles del componente, busque la lista de dependencias.

2.12.0 – 2.12.6

En la siguiente tabla se enumeran las dependencias de las versiones 2.12.0 a 2.12.6 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.12.0 <2.13.0	Flexible

2.11.0 – 2.11.3

La siguiente tabla muestra las dependencias de las versiones 2.11.0 a 2.11.3 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.11.0 <2.12.0	Flexible

2.10.0 – 2.10.3

La siguiente tabla muestra las dependencias de las versiones 2.10.0 a 2.10.3 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.5.0 <2.11.0	Flexible

2.9.0 – 2.9.6

La siguiente tabla muestra las dependencias de las versiones 2.9.0 a 2.9.6 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.5.0 <2.10.0	Flexible

2.8.0 – 2.8.1

La siguiente tabla muestra las dependencias de las versiones 2.8.0 y 2.8.1 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.5.0 <2.9.0	Flexible

2.7.0

La siguiente tabla muestra las dependencias de la versión 2.7.0 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.5.0 <2.8.0	Flexible

2.6.0

La siguiente tabla muestra las dependencias de la versión 2.6.0 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.5.0 <2.7.0	Flexible

2.5.0 – 2.5.6

La siguiente tabla muestra las dependencias de las versiones 2.5.0 a 2.5.6 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.5.0 <2.6.0	Flexible

2.4.0

La siguiente tabla muestra las dependencias de la versión 2.4.0 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.1.0 <2.5.0	Flexible

2.3.0

La siguiente tabla muestra las dependencias de la versión 2.3.0 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.1.0 <2.4.0	Flexible

2.2.0

La siguiente tabla muestra las dependencias de la versión 2.2.0 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.1.0 <2.3.0	Flexible

2.1.0

La siguiente tabla muestra las dependencias de la versión 2.1.0 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.1.0 <2.2.0	Flexible

2.0.x

La siguiente tabla muestra las dependencias de la versión 2.0.x de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.1.0	Flexible

> **Note**
>
> La versión mínima compatible del núcleo de Greengrass corresponde a la versión de parche del componente CLI de Greengrass.

Para obtener más información sobre las dependencias de los componentes, consulte la referencia de recetas de componentes.

Configuración

Este componente proporciona los siguientes parámetros de configuración que puede personalizar al implementar el componente.

2.5.x - 2.12.x

`AuthorizedPosixGroups`

(Opcional) Cadena que contiene una lista de grupos de sistemas separados por comas. Usted autoriza a estos grupos de sistemas a utilizar la CLI de Greengrass para interactuar con el software AWS IoT Greengrass principal. Puede especificar los nombres o los ID de los grupos. Por ejemplo, `group1,1002,group3` autoriza a tres grupos de sistemas (`group1`1002, `group3`) a utilizar la CLI de Greengrass.

Si no especifica ningún grupo para autorizarlo, puede usar la CLI de Greengrass como usuario root (`sudo`) o como usuario del sistema que ejecuta el software AWS IoT Greengrass Core.

`AuthorizedWindowsGroups`

(Opcional) Cadena que contiene una lista de grupos de sistemas separados por comas. Usted autoriza a estos grupos de sistemas a utilizar la CLI de Greengrass para interactuar con el software AWS IoT Greengrass principal. Puede especificar los nombres o los ID de los grupos. Por ejemplo, `group1,1002,group3` autoriza a tres grupos de sistemas (`group1`1002, `group3`) a utilizar la CLI de Greengrass.

Si no especifica ningún grupo para autorizarlo, puede usar la CLI de Greengrass como administrador o como usuario del sistema que ejecuta el software AWS IoT Greengrass principal.

Example Ejemplo: actualización de combinación de configuraciones

El siguiente ejemplo de configuración especifica que se debe autorizar a tres grupos del sistema POSIX (`group11002`, `ygroup3`) y dos grupos de usuarios de Windows (`Device OperatorsyQA Engineers`) a utilizar la CLI de Greengrass.

```
{
  "AuthorizedPosixGroups": "group1,1002,group3",
  "AuthorizedWindowsGroups": "Device Operators,QA Engineers"
}
```

2.4.x - 2.0.x

`AuthorizedPosixGroups`

(Opcional) Cadena que contiene una lista de grupos de sistemas separados por comas. Usted autoriza a estos grupos de sistemas a utilizar la CLI de Greengrass para interactuar con el software AWS IoT Greengrass principal. Puede especificar los nombres o los ID de los grupos. Por ejemplo, `group1,1002,group3` autoriza a tres grupos de sistemas (`group11002`, `ygroup3`) a utilizar la CLI de Greengrass.

Si no especifica ningún grupo para autorizarlo, puede usar la CLI de Greengrass como usuario root (`sudo`) o como usuario del sistema que ejecuta el software AWS IoT Greengrass Core.

Example Ejemplo: actualización de combinación de configuraciones

El siguiente ejemplo de configuración especifica que se autorice a tres grupos de sistemas (`group11002`, `ygroup3`) a utilizar la CLI de Greengrass.

```
{
  "AuthorizedPosixGroups": "group1,1002,group3"
}
```

Archivo de registro local

Este componente utiliza el mismo archivo de registro que el componente núcleo de Greengrass.

Linux

```
/greengrass/v2/logs/greengrass.log
```

Windows

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

Para ver los registros de este componente

- Ejecute el siguiente comando en el dispositivo principal para ver el archivo de registro de este componente en tiempo real. Sustituya */greengrass/v2* o *C:\greengrass\v2* por la ruta a la carpeta AWS IoT Greengrass raíz.

  Linux

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

  Windows (PowerShell)

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

Registros de cambios

En la siguiente tabla se describen los cambios en cada versión del componente.

Versión	Cambios
2.12.6	Versión actualizada para la versión 2.12.6 de Greengrass Nucleus.
2.12.5	Versión actualizada para el lanzamiento de la versión 2.12.5 de Greengrass nucleus.
2.12.4	Versión actualizada para la versión 2.12.4 de Greengrass Nucleus.

Versión	Cambios
2.12.3	**Warning** Esta versión ya no está disponible. Las mejoras de esta versión están disponibles en versiones posteriores de este componente. Versión actualizada para la versión 2.12.3 de Greengrass Nucleus.
2.12.2	Versión actualizada para el lanzamiento de la versión 2.12.2 de Greengrass Nucleus.
2.12.1	Versión actualizada para la versión 2.12.1 de Greengrass Nucleus.
2.12.0	Versión actualizada para la versión 2.12.0 de Greengrass Nucleus.
2.11.3	Versión actualizada para la versión 2.11.3 de Greengrass Nucleus.
2.11.2	Versión actualizada para el lanzamiento de la versión 2.11.2 de Greengrass nucleus.
2.11.1	Versión actualizada para la versión 2.11.1 de Greengrass Nucleus.
2.11.0	Nuevas características • Permite cancelar una implementación local. • Le permite configurar una política de gestión de errores para una implementación local. • Mejora los informes detallados del estado de la implementación.
2.10.3	Versión actualizada para el lanzamiento de la versión 2.10.3 de Greengrass nucleus.
2.10.2	Versión actualizada para el lanzamiento de la versión 2.10.2 de Greengrass nucleus.
2.10.1	Versión actualizada para el lanzamiento de la versión 2.10.1 de Greengrass nucleus.

Versión	Cambios
2.10.0	Versión actualizada para el lanzamiento de la versión 2.10.0 de Greengrass nucleus.
2.9.6	Versión actualizada para la versión 2.9.6 de Greengrass nucleus.
2.9.5	Versión actualizada para el lanzamiento de la versión 2.9.5 de Greengrass nucleus.
2.9.4	Versión actualizada para el lanzamiento de la versión 2.9.4 de Greengrass nucleus.
2.9.3	Versión actualizada para la versión 2.9.3 de Greengrass nucleus.
2.9.2	Versión actualizada para el lanzamiento de la versión 2.9.2 de Greengrass nucleus.
2.9.1	Versión actualizada para la versión 2.9.1 de Greengrass nucleus.
2.9.0	Versión actualizada para la versión 2.9.0 de Greengrass nucleus.
2.8.1	Versión actualizada para la versión 2.8.1 de Greengrass nucleus.
2.8.0	Versión actualizada para la versión 2.8.0 de Greengrass nucleus.
2.7.0	Versión actualizada para la versión 2.7.0 de Greengrass nucleus.
2.6.0	Nuevas características • Añade compatibilidad con componentes personalizados para llamar a las operaciones de comunicación entre procesos (IPC) que utiliza la CLI de Greengrass. Puede usar estas operaciones de IPC para administrar las implementaciones locales, ver los detalles de los componentes y generar una contraseña que pueda usar para iniciar sesión en la consola de depuración local. Para obtener más información, consulte IPC: administrar las implementaciones y los componentes locales. Mejoras y correcciones de errores • Correcciones y mejoras menores adicionales.

Versión	Cambios
2.5.6	Versión actualizada para la versión 2.5.6 de Greengrass nucleus.
2.5.5	Versión actualizada para el lanzamiento de la versión 2.5.5 de Greengrass nucleus.
2.5.4	Versión actualizada para el lanzamiento de la versión 2.5.4 de Greengrass nucleus.
2.5.3	Versión actualizada para el lanzamiento de la versión 2.5.3 de Greengrass nucleus.
2.5.2	Versión actualizada para el lanzamiento de la versión 2.5.2 de Greengrass Nucleus.
2.5.1	Versión actualizada para el lanzamiento de la versión 2.5.1 de Greengrass Nucleus.
2.5.0	Nuevas características • Añade compatibilidad con los dispositivos principales que ejecutan Windows. • Añade el nuevo parámetro `AuthorizedWindowsGroups` de configuración que puede especificar para autorizar a los grupos del sistema a utilizar la CLI de Greengrass en dispositivos Windows. • Agrega el `windowsUser` parámetro para las implementaciones locales. Puede usar este parámetro para especificar el usuario que se utilizará para ejecutar los componentes en un dispositivo principal de Windows.
2.4.0	Nuevas características • Añade compatibilidad con los límites de recursos del sistema. Al crear una implementación local, puede configurar la cantidad máxima de uso de CPU y RAM que los procesos de cada componente pueden usar en el dispositivo principal. Para obtener más información, consulte Configure los límites de recursos del sistema para los componentes y el comando deploy create.

Versión	Cambios
2.3.0	Versión actualizada para el lanzamiento de la versión 2.3.0 de Greengrass nucleus.
2.2.0	Versión actualizada para el lanzamiento de la versión 2.2.0 de Greengrass nucleus.
2.1.0	Versión actualizada para el lanzamiento de la versión 2.1.0 de Greengrass nucleus.
2.0.5	Versión actualizada para el lanzamiento de la versión 2.0.5 de Greengrass nucleus.
2.0.4	Versión actualizada para el lanzamiento de la versión 2.0.4 de Greengrass nucleus.
2.0.3	Versión inicial.

Detector de IP

El componente detector de IP (`aws.greengrass.clientdevices.IPDetector`) hace lo siguiente:

- Supervisa la información de conectividad de red del dispositivo principal de Greengrass. Esta información incluye los puntos finales de red del dispositivo principal y el puerto en el que opera un intermediario de MQTT.
- Actualiza la información de conectividad del dispositivo principal en el servicio en la AWS IoT Greengrass nube.

Los dispositivos cliente pueden usar el descubrimiento en la nube de Greengrass para recuperar la información de conectividad de los dispositivos principales asociados. Luego, los dispositivos cliente pueden intentar conectarse a cada dispositivo principal hasta que se conecten correctamente.

> **Note**
>
> Los dispositivos cliente son dispositivos IoT locales que se conectan a un dispositivo central de Greengrass para enviar mensajes MQTT y datos para su procesamiento. Para obtener más información, consulte Interactúa con dispositivos IoT locales.

El componente detector de IP reemplaza la información de conectividad existente de un dispositivo central por la información que detecta. Como este componente elimina la información existente, puede utilizar el componente detector de IP o administrar manualmente la información de conectividad.

> **Note**
>
> El componente detector de IP detecta únicamente las direcciones IPv4.

Temas

Versiones

Este componente tiene las siguientes versiones:

- 2.1.x
- 2.0.x

Tipo

Este componente es un componente de complemento ()`aws.greengrass.plugin`. El núcleo de Greengrass ejecuta este componente en la misma máquina virtual Java (JVM) que el núcleo. El núcleo se reinicia al cambiar la versión de este componente en el dispositivo principal.

Este componente utiliza el mismo archivo de registro que el núcleo de Greengrass. Para obtener más información, consulte Supervisar AWS IoT Greengrass registros.

Para obtener más información, consulte Tipos de componentes.

Sistema operativo

Este componente se puede instalar en los dispositivos principales que ejecutan los siguientes sistemas operativos:

- Linux
- Windows

Requisitos

Este componente tiene los siguientes requisitos:

- El rol de servicio de Greengrass debe estar asociado a sus permisos Cuenta de AWS y permitir los permisos `iot:GetThingShadow` `yiot:UpdateThingShadow`.
- La AWS IoT política del dispositivo principal debe permitir el `greengrass:UpdateConnectivityInfo` permiso. Para obtener más información, consulte Políticas de AWS IoT para operaciones de plano de datos y AWS IoTPolítica mínima de compatibilidad con los dispositivos cliente.
- Si configura el componente intermediario MQTT del dispositivo principal para que utilice un puerto distinto del puerto 8883 predeterminado, debe utilizar la versión 2.1.0 o posterior del detector de IP. Configúrelo para que indique el puerto en el que opera el intermediario.
- Si tiene una configuración de red compleja, es posible que el componente detector de IP no pueda identificar los puntos finales en los que los dispositivos cliente se pueden conectar al dispositivo principal. Si el componente detector de IP no puede administrar los puntos finales, debe administrar manualmente los puntos finales del dispositivo principal. Por ejemplo, si el dispositivo principal está detrás de un router que le reenvía el puerto intermediario MQTT, debe especificar la

dirección IP del enrutador como punto final del dispositivo principal. Para obtener más información, consulte Administre los puntos finales de los dispositivos principales.

- Se admite que el componente detector de IP se ejecute en una VPC.

Dependencias

Al implementar un componente, AWS IoT Greengrass también despliega versiones compatibles de sus dependencias. Esto significa que debe cumplir los requisitos del componente y de todas sus dependencias para poder implementarlo correctamente. En esta sección se enumeran las dependencias de las versiones publicadas de este componente y las restricciones de las versiones semánticas que definen las versiones de los componentes para cada dependencia. También puede ver las dependencias de cada versión del componente en la consola.AWS IoT Greengrass En la página de detalles del componente, busque la lista de dependencias.

2.1.8 – 2.1.9

La siguiente tabla muestra las dependencias de las versiones 2.1.8 y 2.1.9 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.2.0 <2.13.0	Flexible

2.1.7

La siguiente tabla muestra las dependencias de la versión 2.1.7 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.2.0 <2.12.0	Flexible

2.1.6

La siguiente tabla muestra las dependencias de la versión 2.1.6 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.2.0 <2.11.0	Flexible

2.1.5

La siguiente tabla muestra las dependencias de la versión 2.1.5 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.2.0 <2.10.0	Flexible

2.1.4

La siguiente tabla muestra las dependencias de la versión 2.1.4 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.2.0 <2.9.0	Flexible

2.1.3

La siguiente tabla muestra las dependencias de la versión 2.1.3 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.2.0 <2.8.0	Flexible

2.1.2

La siguiente tabla muestra las dependencias de la versión 2.1.2 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.2.0 <2.7.0	Flexible

2.1.1

La siguiente tabla muestra las dependencias de la versión 2.1.1 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.2.0 <2.6.0	Flexible

2.1.0 and 2.0.2

La siguiente tabla muestra las dependencias de las versiones 2.1.0 y 2.0.2 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.2.0 <2.5.0	Flexible

2.0.1

La siguiente tabla muestra las dependencias de la versión 2.0.1 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.2.0 <2.4.0	Flexible

2.0.0

La siguiente tabla muestra las dependencias de la versión 2.0.0 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.2.0 <2.3.0	Flexible

Para obtener más información sobre las dependencias de los componentes, consulta la referencia de recetas de componentes.

Configuración

Este componente proporciona los siguientes parámetros de configuración que puede personalizar al implementar el componente.

2.1.x

`defaultPort`

(Opcional) El puerto del broker MQTT para informar cuando este componente detecta direcciones IP. Debe especificar este parámetro si configura el broker MQTT para que utilice un puerto diferente al puerto predeterminado 8883.

Valor predeterminado: `8883`

`includeIPv4LoopbackAddrs`

(Opcional) Puede activar esta opción para detectar e informar sobre las direcciones de bucle invertido de IPv4. Se trata de direcciones IP, por ejemplo, en las `localhost` que un dispositivo puede comunicarse consigo mismo. Utilice esta opción en entornos de prueba en los que el dispositivo principal y el dispositivo cliente se ejecuten en el mismo sistema.

Valor predeterminado: `false`

`includeIPv4LinkLocalAddrs`

(Opcional) Puede habilitar esta opción para detectar e informar sobre las direcciones IPv4 de enlace local. Use esta opción si la red del dispositivo principal no tiene el Protocolo de configuración dinámica de host (DHCP) ni direcciones IP asignadas de forma estática.

Valor predeterminado: `false`

2.0.x

`includeIPv4LoopbackAddrs`

(Opcional) Puede habilitar esta opción para detectar e informar sobre las direcciones de bucle invertido de IPv4. Se trata de direcciones IP, por ejemplo, en las `localhost` que un dispositivo puede comunicarse consigo mismo. Utilice esta opción en entornos de prueba en los que el dispositivo principal y el dispositivo cliente se ejecuten en el mismo sistema.

Valor predeterminado: `false`

`includeIPv4LinkLocalAddrs`

(Opcional) Puede habilitar esta opción para detectar e informar sobre las direcciones IPv4 de enlace local. Use esta opción si la red del dispositivo principal no tiene el Protocolo de configuración dinámica de host (DHCP) ni direcciones IP asignadas de forma estática.

Valor predeterminado: `false`

Archivo de registro local

Este componente utiliza el mismo archivo de registro que el componente núcleo de Greengrass.

Linux

```
/greengrass/v2/logs/greengrass.log
```

Windows

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

Para ver los registros de este componente

- Ejecute el siguiente comando en el dispositivo principal para ver el archivo de registro de este componente en tiempo real. Sustituya */greengrass/v2* o *C:\greengrass\v2* por la ruta a la carpeta AWS IoT Greengrass raíz.

  Linux

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

  Windows (PowerShell)

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

Registros de cambios

En la siguiente tabla se describen los cambios en cada versión del componente.

Versión	Cambios
2.1.9	Mejoras y correcciones de errores • Ajusta el paso de IP adquirida para enviar solo los registros a nivel del registro de depuración.

Versión	Cambios
2.1.8	Versión actualizada para la versión 2.12.0 de Greengrass nucleus.
2.1.7	Versión actualizada para el lanzamiento de la versión 2.11.0 de Greengrass nucleus.
2.1.6	Versión actualizada para la versión 2.10.0 de Greengrass nucleus.
2.1.5	Versión actualizada para la versión 2.9.0 de Greengrass Nucleus.
2.1.4	Versión actualizada para el lanzamiento de la versión 2.8.0 de Greengrass nucleus.
2.1.3	Versión actualizada para el lanzamiento de la versión 2.7.0 de Greengrass nucleus.
2.1.2	Mejoras y correcciones de errores • Mejora los mensajes de error que este componente registra en determinados escenarios. • Versión actualizada para la versión 2.6.0 de Greengrass Nucleus.
2.1.1	Versión actualizada para el lanzamiento de la versión 2.5.0 de Greengrass nucleus.
2.1.0	Mejoras • Añade el `defaultPort` parámetro, que permite utilizar un puerto de broker MQTT no predeterminado. • Actualizaciones para que los mensajes de registro sean más claros.
2.0.2	Versión actualizada para la versión 2.4.0 de Greengrass Nucleus.
2.0.1	Versión actualizada para el lanzamiento de la versión 2.3.0 de Greengrass nucleus.
2.0.0	Versión inicial.

Firehose

El componente Firehose (`aws.greengrass.KinesisFirehose`) publica datos a través de las transmisiones de entrega de Amazon Data Firehose a destinos, como Amazon S3, Amazon Redshift y Amazon Service. OpenSearch Para obtener más información, consulte ¿Qué es Amazon Data Firehose? en la Guía para desarrolladores de Amazon Data Firehose.

Para publicar en una transmisión de entrega de Kinesis con este componente, publique un mensaje en un tema al que se suscriba este componente. De forma predeterminada, este componente se suscribe a los temas de publicación `kinesisfirehose/message` o suscripción `kinesisfirehose/message/binary/#` locales. Puede especificar otros temas, incluidos los temas de AWS IoT Core MQTT, al implementar este componente.

> **Note**
>
> Este componente proporciona una funcionalidad similar a la del conector Firehose de la V1. AWS IoT Greengrass Para obtener más información, consulte el conector Firehose en la Guía para desarrolladores de la AWS IoT Greengrass V1.

Temas

Versiones

Este componente tiene las siguientes versiones:

- 2.1.x
- 2.0.x

Tipo

Este componente es un componente Lambda () `aws.greengrass.lambda`. El núcleo de Greengrass ejecuta la función Lambda de este componente mediante el componente Lambda launcher.

Para obtener más información, consulte Tipos de componentes.

Sistema operativo

Este componente solo se puede instalar en los dispositivos principales de Linux.

Requisitos

Este componente tiene los siguientes requisitos:

- El dispositivo principal debe cumplir los requisitos para ejecutar las funciones de Lambda. Si desea que el dispositivo principal ejecute funciones Lambda en contenedores, el dispositivo debe cumplir los requisitos para hacerlo. Para obtener más información, consulte Requisitos de la función de Lambda.
- Versión 3.7 de Python instalada en el dispositivo principal y añadida a la variable de entorno PATH.
- El rol de dispositivo de Greengrass debe permitir las `firehose:PutRecordBatch` acciones `firehose:PutRecord` y, como se muestra en el siguiente ejemplo de política de IAM.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "firehose:PutRecord",
        "firehose:PutRecordBatch"
      ],
```

```
      "Effect": "Allow",
      "Resource": [
        "arn:aws:firehose:region:account-id:deliverystream/stream-name"
      ]
    }
  ]
}
```

Puede anular dinámicamente el flujo de entrega predeterminado en la carga útil del mensaje de entrada para este componente. Si la aplicación utiliza esta función, la política de IAM debe incluir todas las transmisiones de destino como recursos. Puede conceder acceso granular o condicional a recursos (por ejemplo, utilizando un esquema de nomenclatura con comodín *)

- Para recibir los datos de salida de este componente, debe combinar la siguiente actualización de configuración para el componente antiguo del enrutador de suscripciones (`aws.greengrass.LegacySubscriptionRouter`) al implementar este componente. Esta configuración especifica el tema en el que este componente publica las respuestas.

  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"
      }
  ```

```
  }
}
```

- Sustituya la *región* por la Región de AWS que utilice.
- Sustituya la *versión* por la versión de la función Lambda que ejecuta este componente. Para encontrar la versión de la función Lambda, debe ver la receta de la versión de este componente que desee implementar. Abra la página de detalles de este componente en la AWS IoT Greengrass consola y busque el par clave-valor de la función Lambda. Este par clave-valor contiene el nombre y la versión de la función Lambda.

> **Important**
>
> Debe actualizar la versión de la función Lambda en el router de suscripción anterior cada vez que implemente este componente. Esto garantiza que utilice la versión correcta de la función Lambda para la versión del componente que implemente.

Para obtener más información, consulte Crear implementaciones.

- Se admite la ejecución del componente Firehose en una VPC. Para implementar este componente en una VPC, se requiere lo siguiente.
  - El componente Firehose debe tener una conectividad con la `firehose.region.amazonaws.com` que tenga el punto final de la VPC de. `com.amazonaws.region.kinesis-firehose`

Puntos finales y puertos

Este componente debe poder realizar solicitudes salientes a los siguientes puntos finales y puertos, además de a los puntos finales y puertos necesarios para el funcionamiento básico. Para obtener más información, consulte Permitir el tráfico del dispositivo a través de un proxy o firewall.

punto de enlace	Puerto	Obligatoria	Descripción
`firehose. region .amazonaws.com`	443	Sí	Sube datos a Firehose.

Dependencias

Al implementar un componente, AWS IoT Greengrass también despliega versiones compatibles de sus dependencias. Esto significa que debe cumplir los requisitos del componente y de todas sus dependencias para poder implementarlo correctamente. En esta sección se enumeran las dependencias de las versiones publicadas de este componente y las restricciones de las versiones semánticas que definen las versiones de los componentes para cada dependencia. También puede ver las dependencias de cada versión del componente en la consola. AWS IoT Greengrass En la página de detalles del componente, busque la lista de dependencias.

2.1.7

La siguiente tabla muestra las dependencias de la versión 2.1.7 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.13.0	Rígido
Lanzador Lambda	^2.0.0	Rígido
Tiempos de ejecución de Lambda	^2.0.0	Flexible
Servicio de intercambio de fichas	^2.0.0	Rígido

2.1.6

La siguiente tabla muestra las dependencias de la versión 2.1.6 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.12.0	Rígido
Lanzador Lambda	^2.0.0	Rígido
Tiempos de ejecución de Lambda	^2.0.0	Flexible

Dependencia	Versiones compatibles	Tipo de dependencia
Servicio de intercambio de fichas	^2.0.0	Rígido

2.1.5

La siguiente tabla muestra las dependencias de la versión 2.1.5 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.11.0	Rígido
Lanzador Lambda	^2.0.0	Rígido
Tiempos de ejecución de Lambda	^2.0.0	Flexible
Servicio de intercambio de fichas	^2.0.0	Rígido

2.1.4

La siguiente tabla muestra las dependencias de la versión 2.1.4 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.10.0	Rígido
Lanzador Lambda	^2.0.0	Rígido
Tiempos de ejecución de Lambda	^2.0.0	Flexible
Servicio de intercambio de fichas	^2.0.0	Rígido

2.1.3

La siguiente tabla muestra las dependencias de la versión 2.1.3 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.9.0	Rígido
Lanzador Lambda	^2.0.0	Rígido
Tiempos de ejecución de Lambda	^2.0.0	Flexible
Servicio de intercambio de fichas	^2.0.0	Rígido

2.1.2

La siguiente tabla muestra las dependencias de la versión 2.1.2 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.8.0	Rígido
Lanzador Lambda	^2.0.0	Rígido
Tiempos de ejecución de Lambda	^2.0.0	Flexible
Servicio de intercambio de fichas	^2.0.0	Rígido

2.1.1

La siguiente tabla muestra las dependencias de la versión 2.1.1 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.7.0	Rígido
Lanzador Lambda	^2.0.0	Rígido
Tiempos de ejecución de Lambda	^2.0.0	Flexible
Servicio de intercambio de fichas	^2.0.0	Rígido

2.0.8 - 2.1.0

La siguiente tabla muestra las dependencias de las versiones 2.0.8 y 2.1.0 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.6.0	Rígido
Lanzador Lambda	^2.0.0	Rígido
Tiempos de ejecución de Lambda	^2.0.0	Flexible
Servicio de intercambio de fichas	^2.0.0	Rígido

2.0.7

La siguiente tabla muestra las dependencias de la versión 2.0.7 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.5.0	Rígido
Lanzador Lambda	^2.0.0	Rígido

Dependencia	Versiones compatibles	Tipo de dependencia
Tiempos de ejecución de Lambda	^2.0.0	Flexible
Servicio de intercambio de fichas	^2.0.0	Rígido

2.0.6

La siguiente tabla muestra las dependencias de la versión 2.0.6 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.4.0	Rígido
Lanzador Lambda	^2.0.0	Rígido
Tiempos de ejecución de Lambda	^2.0.0	Flexible
Servicio de intercambio de fichas	^2.0.0	Rígido

2.0.5

La siguiente tabla muestra las dependencias de la versión 2.0.5 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.3.0	Rígido
Lanzador Lambda	^2.0.0	Rígido
Tiempos de ejecución de Lambda	^2.0.0	Flexible

Dependencia	Versiones compatibles	Tipo de dependencia
Servicio de intercambio de fichas	^2.0.0	Rígido

2.0.4

La siguiente tabla muestra las dependencias de la versión 2.0.4 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.2.0	Rígido
Lanzador Lambda	^2.0.0	Rígido
Tiempos de ejecución de Lambda	^2.0.0	Flexible
Servicio de intercambio de fichas	^2.0.0	Rígido

2.0.3

La siguiente tabla muestra las dependencias de la versión 2.0.3 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.3 <2.1.0	Rígido
Lanzador Lambda	>=1.0.0	Rígido
Tiempos de ejecución de Lambda	>=1.0.0	Flexible
Servicio de intercambio de fichas	>=1.0.0	Rígido

Para obtener más información sobre las dependencias de los componentes, consulta la referencia de recetas de componentes.

Configuración

Este componente proporciona los siguientes parámetros de configuración que puede personalizar al implementar el componente.

> **Note**
>
> La configuración predeterminada de este componente incluye los parámetros de la función Lambda. Le recomendamos que edite solo los siguientes parámetros para configurar este componente en sus dispositivos.

`lambdaParams`

Objeto que contiene los parámetros de la función Lambda de este componente. Este objeto contiene la siguiente información:

`EnvironmentVariables`

Objeto que contiene los parámetros de la función Lambda. Este objeto contiene la siguiente información:

`DEFAULT_DELIVERY_STREAM_ARN`

El ARN del flujo de entrega predeterminado de Firehose al que el componente envía los datos. Puede anular la transmisión de destino con la `delivery_stream_arn` propiedad en la carga útil del mensaje de entrada.

> **Note**
>
> La función de dispositivo principal debe permitir las acciones necesarias en todas las transmisiones de entrega de destino. Para obtener más información, consulte Requisitos.

`PUBLISH_INTERVAL`

(Opcional) El número máximo de segundos que se deben esperar antes de que el componente publique los datos por lotes en Firehose. Para configurar el componente para

que publique las métricas a medida que las reciba, es decir, sin procesamiento por lotes, especifique. `0`

Este valor puede ser como máximo de 900 segundos.

Predeterminado: 10 segundos

`DELIVERY_STREAM_QUEUE_SIZE`

(Opcional) El número máximo de registros que se deben conservar en la memoria antes de que el componente rechace nuevos registros para el mismo flujo de entrega.

Este valor debe ser de al menos 2000 registros.

Predeterminado: 5.000 registros

`containerMode`

(Opcional) El modo de contenedorización de este componente. Puede elegir entre las siguientes opciones:

- `NoContainer`— El componente no se ejecuta en un entorno de ejecución aislado.
- `GreengrassContainer`— El componente se ejecuta en un entorno de ejecución aislado dentro del AWS IoT Greengrass contenedor.

Predeterminado: `GreengrassContainer`

`containerParams`

(Opcional) Un objeto que contiene los parámetros del contenedor de este componente. El componente utiliza estos parámetros si se especifica `GreengrassContainer` para`containerMode`.

Este objeto contiene la siguiente información:

`memorySize`

(Opcional) La cantidad de memoria (en kilobytes) que se va a asignar al componente.

El valor predeterminado es 64 MB (65.535 KB).

`pubsubTopics`

(Opcional) Objeto que contiene los temas a los que el componente se suscribe para recibir mensajes. Puede especificar cada tema y si el componente se suscribe a los temas de MQTT AWS IoT Core o a los temas de publicación/suscripción locales.

Este objeto contiene la siguiente información:

`0`— Se trata de un índice matricial en forma de cadena.

Objeto que contiene la siguiente información:

`type`

(Opcional) El tipo de mensajes de publicación/suscripción que utiliza este componente para suscribirse a los mensajes. Puede elegir entre las siguientes opciones:

- `PUB_SUB` — Suscribirse a mensajes locales de publicación/suscripción. Si elige esta opción, el tema no puede contener caracteres comodín de MQTT. Para obtener más información sobre cómo enviar mensajes desde un componente personalizado al especificar esta opción, consulte. Publicar/suscribir mensajes locales
- `IOT_CORE`— Suscríbase a los mensajes de AWS IoT Core MQTT. Si elige esta opción, el tema puede contener caracteres comodín de MQTT. Para obtener más información sobre cómo enviar mensajes desde componentes personalizados al especificar esta opción, consulte. Publicar/suscribir mensajes MQTT AWS IoT Core

Predeterminado: `PUB_SUB`

`topic`

(Opcional) El tema al que se suscribe el componente para recibir mensajes. Si lo especifica `IotCoretype`, puede usar los comodines MQTT (+y#) en este tema.

Example Ejemplo: actualización de la combinación de configuraciones (modo contenedor)

```
{
  "lambdaExecutionParameters": {
    "EnvironmentVariables": {
      "DEFAULT_DELIVERY_STREAM_ARN": "arn:aws:firehose:us-
west-2:123456789012:deliverystream/mystream"
    }
  },
  "containerMode": "GreengrassContainer"
}
```

Example Ejemplo: actualización de la combinación de configuraciones (sin modo contenedor)

```
{
```

```
  "lambdaExecutionParameters": {
    "EnvironmentVariables": {
      "DEFAULT_DELIVERY_STREAM_ARN": "arn:aws:firehose:us-
west-2:123456789012:deliverystream/mystream"
    }
  },
  "containerMode": "NoContainer"
}
```

Datos de entrada

Este componente acepta contenido de transmisión sobre los siguientes temas y lo envía a la transmisión de entrega de destino. El componente acepta dos tipos de datos de entrada:

- Datos JSON en el tema `kinesisfirehose/message`.
- Datos binarios en el tema `kinesisfirehose/message/binary/#`.

Tema predeterminado para los datos JSON (publicación/suscripción local): `kinesisfirehose/message`

El mensaje acepta las siguientes propiedades. Los mensajes de entrada deben tener un formato JSON válido.

`request`

Los datos para enviar a la transmisión de entrega y a la transmisión de entrega de destino, si no es la transmisión predeterminada.

Tipo: `object` que contiene la siguiente información:

`data`

Los datos que enviar a la transmisión de entrega.

Tipo: `string`

`delivery_stream_arn`

(Opcional) El ARN de la transmisión de entrega de Firehose objetivo. Especifique esta propiedad para anular el flujo de entrega predeterminado.

Tipo: `string`

`id`

Un ID arbitrario para la solicitud. Utilice esta propiedad para asignar una solicitud de entrada a una respuesta de salida. Al especificar esta propiedad, el componente establece la `id` propiedad del objeto de respuesta en este valor.

Tipo: `string`

Example Ejemplo de entrada

```
{
  "request": {
    "delivery_stream_arn": "arn:aws:firehose:region:account-id:deliverystream/
stream2-name",
    "data": "Data to send to the delivery stream."
  },
  "id": "request123"
}
```

Tema predeterminado para datos binarios (publicación/suscripción local): `kinesisfirehose/message/binary/#`

Consulte este tema para enviar un mensaje que contenga datos binarios. El componente no analiza los datos binarios. El componente transmite los datos tal cual.

Para asignar la solicitud de entrada a una respuesta de salida, sustituya el comodín # en el tema del mensaje con un ID de solicitud arbitrario. Por ejemplo, si publica un mensaje en `kinesisfirehose/message/binary/request123`, la propiedad `id` en el objeto de respuesta se establece en `request123`.

Si no desea asignar una solicitud a una respuesta, puede publicar sus mensajes en `kinesisfirehose/message/binary/`. Asegúrese de incluir la barra final ()/.

Datos de salida

Este componente publica las respuestas como datos de salida sobre el siguiente tema de MQTT de forma predeterminada. Debe especificar este tema como parte de `subject` la configuración del componente antiguo del router de suscripciones. Para obtener más información sobre cómo suscribirse a los mensajes sobre este tema en sus componentes personalizados, consultePublicar/suscribir mensajes MQTT AWS IoT Core.

Tema predeterminado (AWS IoT Core MQTT): `kinesisfirehose/message/status`

Example Ejemplo de resultados

La respuesta contiene el estado de cada registro de datos enviado en el lote.

```
{
  "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**
>
> Si el componente detecta un error que se puede volver a intentar, como un error de conexión, volverá a intentar la publicación en el siguiente lote.

Archivo de registro local

Este componente utiliza el siguiente archivo de registro.

```
/greengrass/v2/logs/aws.greengrass.KinesisFirehose.log
```

Para ver los registros de este componente

- Ejecute el siguiente comando en el dispositivo principal para ver el archivo de registro de este componente en tiempo real. */greengrass/v2*Sustitúyalo por la ruta a la carpeta AWS IoT Greengrass raíz.

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

Licencias

Este componente incluye el siguiente software o licencias de terceros:

- AWS SDK for Python (Boto3)/Apache License 2.0
- botocore/Apache License 2.0
- dateutil/PSF License
- docutils/BSD License, GNU General Public License (GPL), Python Software Foundation License, Public Domain
- jmespath/MIT License
- s3transfer/Apache License 2.0
- urllib3/MIT License

Este componente se publica en virtud del contrato de licencia de software principal de Greengrass.

Registros de cambios

En la siguiente tabla se describen los cambios en cada versión del componente.

Versión	Cambios
2.1.7	Versión actualizada para la versión 2.12.0 de Greengrass Nucleus.
2.1.6	Versión actualizada para el lanzamiento de la versión 2.11.0 de Greengrass nucleus.
2.1.5	Versión actualizada para el lanzamiento de la versión 2.10.0 de Greengrass nucleus.

Versión	Cambios
2.1.4	Versión actualizada para la versión 2.9.0 de Greengrass Nucleus.
2.1.3	Versión actualizada para el lanzamiento de la versión 2.8.0 del núcleo de Greengrass.
2.1.2	Versión actualizada para el lanzamiento de la versión 2.7.0 de Greengrass Nucleus.
2.1.1	Versión actualizada para el lanzamiento de la versión 2.6.0 de Greengrass Nucleus.
2.1.0	Nuevas características • Añade compatibilidad con las configuraciones de proxy de red HTTPS. Para obtener más información, consulte Realizar la conexión en el puerto 443 o a través de un proxy de red y Habilite el dispositivo principal para que confíe en un proxy HTTPS.
2.0.8	Versión actualizada para el lanzamiento de la versión 2.5.0 de Greengrass Nucleus.
2.0.7	Versión actualizada para la versión 2.4.0 de Greengrass Nucleus.
2.0.6	Versión actualizada para el lanzamiento de la versión 2.3.0 de Greengrass nucleus.
2.0.5	Versión actualizada para el lanzamiento de la versión 2.2.0 de Greengrass nucleus.
2.0.4	Versión actualizada para el lanzamiento de la versión 2.1.0 de Greengrass nucleus.
2.0.3	Versión inicial.

Véase también

- ¿Qué es Amazon Data Firehose? en la guía para desarrolladores de Amazon Data Firehose

Lanzador Lambda

El componente Launcher Lambda (`aws.greengrass.LambdaLauncher`) inicia y detiene AWS Lambda las funciones en AWS IoT Greengrass los dispositivos principales. Este componente también configura cualquier contenedorización y ejecuta los procesos según los usuarios que especifique.

> **Note**
>
> Al implementar un componente de función Lambda en un dispositivo principal, la implementación también incluye este componente. Para obtener más información, consulte AWS LambdaFunciones de ejecución.

Temas

Versiones

Este componente tiene las siguientes versiones:

- 2.0.x

Tipo

Este componente es un componente genérico () `aws.greengrass.generic`. El núcleo de Greengrass ejecuta los scripts del ciclo de vida del componente.

Para obtener más información, consulte Tipos de componentes.

Sistema operativo

Este componente solo se puede instalar en los dispositivos principales de Linux.

Requisitos

Este componente tiene los siguientes requisitos:

- El dispositivo principal debe cumplir los requisitos para ejecutar las funciones de Lambda. Si desea que el dispositivo principal ejecute funciones Lambda en contenedores, el dispositivo debe cumplir los requisitos para hacerlo. Para obtener más información, consulte Requisitos de la función de Lambda.
- Se admite la ejecución del componente Lambda Launcher en una VPC.

Dependencias

Al implementar un componente, AWS IoT Greengrass también despliega versiones compatibles de sus dependencias. Esto significa que debe cumplir los requisitos del componente y de todas sus dependencias para poder implementarlo correctamente. En esta sección se enumeran las dependencias de las versiones publicadas de este componente y las restricciones de las versiones semánticas que definen las versiones de los componentes para cada dependencia. También puede ver las dependencias de cada versión del componente en la consola. AWS IoT Greengrass En la página de detalles del componente, busque la lista de dependencias.

2.0.11 – 2.0.13

La siguiente tabla muestra las dependencias de las versiones 2.0.11 a 2.0.13 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Gestor Lambda	>=2.0.0 <2.4.0	Rígido

2.0.9 – 2.0.10

La siguiente tabla muestra las dependencias de las versiones 2.0.9 a 2.0.10 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Gestor Lambda	>=2.0.0 <2.3.0	Rígido

2.0.4 - 2.0.8

La siguiente tabla muestra las dependencias de las versiones 2.0.4 a 2.0.8 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Gestor Lambda	>=2.0.0 <2.2.0	Rígido

2.0.3

La siguiente tabla muestra las dependencias de la versión 2.0.3 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Gestor Lambda	>=2.0.3 <2.1.0	Rígido

Para obtener más información sobre las dependencias de los componentes, consulta la referencia de recetas de componentes.

Configuración

Este componente no tiene ningún parámetro de configuración.

Archivo de registro local

Este componente utiliza el siguiente archivo de registro.

```
/greengrass/v2/logs/lambdaFunctionComponentName.log
```

Para ver los registros de este componente

- Ejecute el siguiente comando en el dispositivo principal para ver el archivo de registro de este componente en tiempo real. Sustituya */greengrass/v2* por la ruta a la carpeta AWS

IoT Greengrass raíz y sustituya *lambdaFunctionComponentNombre* por el nombre del componente de la función Lambda que lanza este componente.

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

Registros de cambios

En la siguiente tabla se describen los cambios en cada versión del componente.

Versión	Cambios
2.0.13	Mejoras y correcciones de errores Corrección de errores y mejoras generales.
2.0.12	Mejoras y correcciones de errores Soluciona un problema por el que el lanzador Lambda podía generar un error si el proceso anterior no se detenía correctamente.
2.0.11	Support para Lambda Manager 2.3.0.
2.0.10	Mejoras y correcciones de errores • Corrección de errores y mejoras generales.
2.0.9	Versión actualizada para la versión 2.5.0 de Greengrass nucleus.
2.0.8	Versión actualizada para la versión 2.4.0 de Greengrass nucleus.
2.0.7	Versión actualizada para la versión 2.3.0 de Greengrass nucleus.
2.0.6	Mejoras de rendimiento generales y correcciones de errores.
2.0.4	Mejoras y correcciones de errores • Soluciona un problema por el que el componente no pasa correctam ente `AddGroupOwner` al contenedor de funciones Lambda.
2.0.3	Versión inicial.

Gestor Lambda

El componente Lambda manager (`aws.greengrass.LambdaManager`) administra los elementos de trabajo y la comunicación entre procesos para AWS Lambda las funciones que se ejecutan en el dispositivo principal de Greengrass.

> **Note**
>
> Al implementar un componente de función Lambda en un dispositivo principal, la implementación también incluye este componente. Para obtener más información, consulte AWS LambdaFunciones de ejecución.

Temas

Versiones

Este componente tiene las siguientes versiones:

- 2.3.x
- 2.2.x
- 2.1.x
- 2.0.x

Sistema operativo

Este componente solo se puede instalar en los dispositivos principales de Linux.

Tipo

Este componente es un componente de complemento (`aws.greengrass.plugin`). El núcleo de Greengrass ejecuta este componente en la misma máquina virtual Java (JVM) que el núcleo. El núcleo se reinicia al cambiar la versión de este componente en el dispositivo principal.

Este componente utiliza el mismo archivo de registro que el núcleo de Greengrass. Para obtener más información, consulte Supervisar AWS IoT Greengrass registros.

Para obtener más información, consulte Tipos de componentes.

Requisitos

Este componente tiene los siguientes requisitos:

- El dispositivo principal debe cumplir los requisitos para ejecutar las funciones de Lambda. Si desea que el dispositivo principal ejecute funciones Lambda en contenedores, el dispositivo debe cumplir los requisitos para hacerlo. Para obtener más información, consulte Requisitos de la función de Lambda.
- Se admite la ejecución del componente Lambda Manager en una VPC.

Dependencias

Al implementar un componente, AWS IoT Greengrass también despliega versiones compatibles de sus dependencias. Esto significa que debe cumplir los requisitos del componente y de todas sus dependencias para poder implementarlo correctamente. En esta sección se enumeran las dependencias de las versiones publicadas de este componente y las restricciones de las versiones semánticas que definen las versiones de los componentes para cada dependencia. También puede ver las dependencias de cada versión del componente en la consola.AWS IoT Greengrass En la página de detalles del componente, busque la lista de dependencias.

2.3.2 and 2.3.3

La siguiente tabla muestra las dependencias de las versiones 2.3.2 y 2.3.3 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.13.0	Flexible

2.2.10 and 2.3.1

La siguiente tabla muestra las dependencias de las versiones 2.2.10 y 2.3.1 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.12.0	Flexible

2.2.8 and 2.2.9

La siguiente tabla muestra las dependencias de las versiones 2.2.8 y 2.2.9 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.11.0	Flexible

2.2.7

La siguiente tabla muestra las dependencias de la versión 2.2.7 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.10.0	Flexible

2.2.6

La siguiente tabla muestra las dependencias de la versión 2.2.6 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.9.0	Flexible

2.2.5

La siguiente tabla muestra las dependencias de la versión 2.2.5 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.8.0	Flexible

2.2.4

La siguiente tabla muestra las dependencias de la versión 2.2.4 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.7.0	Flexible

2.2.1 - 2.2.3

La siguiente tabla muestra las dependencias de las versiones 2.2.1 a 2.2.3 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.6.0	Flexible

2.2.0

La siguiente tabla muestra las dependencias de la versión 2.2.0 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.5.0 <2.6.0	Flexible

2.1.3 and 2.1.4

La siguiente tabla muestra las dependencias de las versiones 2.1.3 y 2.1.4 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.5.0	Flexible

2.1.2

La siguiente tabla muestra las dependencias de la versión 2.1.2 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.4.0	Flexible

2.1.1

La siguiente tabla muestra las dependencias de la versión 2.1.1 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.3.0	Flexible

2.1.0

La siguiente tabla muestra las dependencias de la versión 2.1.0 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.2.0	Flexible

2.0.x

La siguiente tabla muestra las dependencias de la versión 2.0.x de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.3 <2.1.0	Flexible

Para obtener más información sobre las dependencias de los componentes, consulta la referencia de recetas de componentes.

Configuración

Este componente proporciona los siguientes parámetros de configuración que puede personalizar al implementar el componente.

`logHandlerMode`

> **Note**
>
> Solo para las versiones 2.3.0 y posteriores de Lambda Manager

Se utiliza para elegir la implementación del gestor de registros Lambda que se va a utilizar. Establezca el valor en `optimized` para usar menos subprocesos para leer los registros lambda.

`getResultTimeoutInSecond`

(Opcional) El tiempo máximo en segundos que las funciones Lambda pueden ejecutarse antes de que se agote el tiempo de espera.

Valor predeterminado: `60`

Archivo de registro local

Este componente utiliza el mismo archivo de registro que el componente núcleo de Greengrass.

```
/greengrass/v2/logs/greengrass.log
```

Para ver los registros de este componente

- Ejecute el siguiente comando en el dispositivo principal para ver el archivo de registro de este componente en tiempo real. */greengrass/v2*Sustitúyalo por la ruta a la carpeta AWS IoT Greengrass raíz.

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

Registros de cambios

En la siguiente tabla se describen los cambios en cada versión del componente.

Versión	Cambios
2.3.3	Mejoras y correcciones de errores • Corrección de errores y mejoras generales.
2.3.2	Versión actualizada para la versión 2.12.0 de Greengrass nucleus.
2.3.1	Mejoras y correcciones de errores • Ajusta los niveles de registro para detectar determinados errores.
2.3.0	Nuevas características • El controlador de registros se optimizó para reducir la carga de la CPU. Utilice esta función configurando la opción `logHandlerMode` de configuración en. `optimized` Mejoras y correcciones de errores • Ya no se registra todo el stacktrace`WorkQueueFullException` , lo que mejora los registros y el rendimiento. • Establece el tiempo de espera de apagado de Lambda de 15 a 300 segundos para evitar tiempos de espera de apagado. • Soluciona un problema que provocaba que las lambdas bajo demanda no se reiniciaran después de cambiar la configuración.
2.2.11	Mejoras y correcciones de errores • Soluciona un problema por el que la LegacySubscriptionRouter configuración no se actualiza cuando cambia la configuración de Lambda.
2.2.10	Versión actualizada para la versión 2.11.0 de Greengrass nucleus.
2.2.9	Mejoras y correcciones de errores Soluciona un problema por el que el número de puerto estaba dañado debido a un reloj sesgado.
2.2.8	Versión actualizada para la versión 2.10.0 de Greengrass nucleus.
2.2.7	Versión actualizada para la versión 2.9.0 de Greengrass nucleus.

Versión	Cambios
2.2.6	Versión actualizada para la versión 2.8.0 de Greengrass nucleus.
2.2.5	Nuevas características • Añade compatibilidad con los comodines de temas de MQTT en las fuentes de eventos en las que se suscribe a mensajes locales de publicación/suscripción. Esta función requiere la versión 2.6.0 o posterior del componente núcleo de Greengrass. • Versión actualizada para la versión 2.7.0 de Greengrass nucleus.
2.2.4	Versión actualizada para la versión 2.6.0 de Greengrass nucleus.
2.2.3	Mejoras y correcciones de errores • Soluciona un problema por el que varias instancias de una función Lambda comparten un único grupo c. Este componente usa cgroups para administrar el uso de recursos para las funciones de Lambda.
2.2.2	Mejoras y correcciones de errores • Soluciona un problema que provocaba que los componentes de la función Lambda anclados se reiniciaran inesperadamente en determina das situaciones.
2.2.1	Mejoras y correcciones de errores • Cambia las restricciones de la versión de dependencia del núcleo de Greengrass de este componente para solucionar un problema de resolución de dependencias.

Versión	Cambios
2.2.0	Mejoras y correcciones de errores • Soluciona un problema por el que las funciones de Lambda no podían escribir registros tras un reinicio. • Soluciona un problema que provocaba que el router de suscripciones antiguo enviara mensajes duplicados cuando había caracteres comodín en el tema. • Soluciona un problema por el que las funciones Lambda no ancladas no podían utilizar la biblioteca de comunicación entre procesos (IPC) de Greengrass en. SDK para dispositivos con AWS IoT
2.1.4	Mejoras y correcciones de errores • Corrige un problema que provocaba que las funciones de Lambda que utilizan tiempos de ejecución de Nodejs procesaran solo un mensaje. • Versión actualizada para la versión 2.5.0 de Greengrass nucleus.
2.1.3	Versión actualizada para la versión 2.4.0 de Greengrass nucleus.
2.1.2	Versión actualizada para la versión 2.3.0 de Greengrass nucleus.
2.1.1	Versión actualizada para la versión 2.2.0 de Greengrass nucleus.
2.1.0	Versión actualizada para la versión 2.1.0 de Greengrass nucleus.
2.0.3	Versión inicial.

Tiempos de ejecución de Lambda

El componente de tiempos de ejecución de Lambda (`aws.greengrass.LambdaRuntimes`) proporciona los tiempos de ejecución que utilizan los dispositivos principales de Greengrass para ejecutar funciones. AWS Lambda

> **Note**
>
> Al implementar un componente de función Lambda en un dispositivo principal, la implementación también incluye este componente. Para obtener más información, consulte AWS LambdaFunciones de ejecución.

Temas

Versiones

Este componente tiene las siguientes versiones:

- 2.0.x

Tipo

Este componente es un componente genérico () `aws.greengrass.generic`. El núcleo de Greengrass ejecuta los scripts del ciclo de vida del componente.

Para obtener más información, consulte Tipos de componentes.

Sistema operativo

Este componente solo se puede instalar en los dispositivos principales de Linux.

Requisitos

Este componente tiene los siguientes requisitos:

- El dispositivo principal debe cumplir los requisitos para ejecutar las funciones de Lambda. Si desea que el dispositivo principal ejecute funciones Lambda en contenedores, el dispositivo debe cumplir los requisitos para hacerlo. Para obtener más información, consulte Requisitos de la función de Lambda.
- Se admite la ejecución del componente de tiempos de ejecución de Lambda en una VPC.

Dependencias

Este componente no tiene ninguna dependencia.

Configuración

Este componente no tiene ningún parámetro de configuración.

Archivo de registro local

Este componente no genera registros.

Registros de cambios

En la siguiente tabla se describen los cambios en cada versión del componente.

Versión	Cambios
2.0.8	Versión actualizada para el lanzamiento de la versión 2.5.0 de Greengrass nucleus.
2.0.7	Versión actualizada para la versión 2.4.0 de Greengrass Nucleus.
2.0.6	Versión actualizada para el lanzamiento de la versión 2.3.0 de Greengrass nucleus.
2.0.5	Versión actualizada para el lanzamiento de la versión 2.2.0 de Greengrass nucleus.
2.0.4	Versión actualizada para el lanzamiento de la versión 2.1.0 de Greengrass nucleus.
2.0.3	Versión inicial.

Enrutador de suscripción antiguo

El router de suscripciones antiguo (`aws.greengrass.LegacySubscriptionRouter`) gestiona las suscripciones en el dispositivo principal de Greengrass. Las suscripciones son una función de la AWS IoT Greengrass versión 1 que define los temas que las funciones de Lambda pueden utilizar para la mensajería MQTT en un dispositivo principal. Para obtener más información, consulte las suscripciones gestionadas en el flujo de trabajo de mensajería MQTT en la Guía para desarrolladores de la AWS IoT Greengrassversión 1.

Puede usar este componente para habilitar las suscripciones de los componentes de conectores y los componentes de la función Lambda que utilizan el SDK de AWS IoT Greengrass Core.

> **Note**
>
> El componente de router de suscripción antiguo solo es necesario si la función Lambda utiliza la `publish()` función del SDK AWS IoT Greengrass principal. Si actualiza el código de la función Lambda para utilizar la interfaz de comunicación entre procesos (IPC) de la SDK para dispositivos con AWS IoT V2, no necesitará implementar el componente de router de suscripción heredado. Para obtener más información, consulte los siguientes servicios de comunicación entre procesos:
>
> - Publicar/suscribir mensajes locales
> - Publicar/suscribir mensajes MQTT AWS IoT Core

Temas

- Versiones
- Tipo
- Sistema operativo
- Requisitos
- Dependencias
- Configuración
- Archivo de registro local
- Registros de cambios

Versiones

Este componente tiene las siguientes versiones:

- 2.1.x
- 2.0.x

Tipo

Este componente es un componente genérico () `aws.greengrass.generic`. El núcleo de Greengrass ejecuta los scripts del ciclo de vida del componente.

Para obtener más información, consulte Tipos de componentes.

Sistema operativo

Este componente solo se puede instalar en los dispositivos principales de Linux.

Requisitos

Este componente tiene los siguientes requisitos:

- El router de suscripción antiguo es compatible para ejecutarse en una VPC.

Dependencias

Al implementar un componente, AWS IoT Greengrass también despliega versiones compatibles de sus dependencias. Esto significa que debe cumplir los requisitos del componente y de todas sus dependencias para poder implementarlo correctamente. En esta sección se enumeran las dependencias de las versiones publicadas de este componente y las restricciones de las versiones semánticas que definen las versiones de los componentes para cada dependencia. También puede ver las dependencias de cada versión del componente en la consola. AWS IoT Greengrass En la página de detalles del componente, busque la lista de dependencias.

2.1.11

La siguiente tabla muestra las dependencias de la versión 2.1.11 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.13.0	Flexible

2.1.10

La siguiente tabla muestra las dependencias de la versión 2.1.10 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.12.0	Flexible

2.1.9

La siguiente tabla muestra las dependencias de la versión 2.1.9 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.11.0	Flexible

2.1.8

La siguiente tabla muestra las dependencias de la versión 2.1.8 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.10.0	Flexible

2.1.7

La siguiente tabla muestra las dependencias de la versión 2.1.7 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.9.0	Flexible

2.1.6

La siguiente tabla muestra las dependencias de la versión 2.1.6 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.8.0	Flexible

2.1.5

La siguiente tabla muestra las dependencias de la versión 2.1.5 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.7.0	Flexible

2.1.4

La siguiente tabla muestra las dependencias de la versión 2.1.4 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.6.0	Flexible

2.1.3

La siguiente tabla muestra las dependencias de la versión 2.1.3 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.5.0	Flexible

2.1.2

La siguiente tabla muestra las dependencias de la versión 2.1.2 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.4.0	Flexible

2.1.1

La siguiente tabla muestra las dependencias de la versión 2.1.1 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.3.0	Flexible

2.1.0

La siguiente tabla muestra las dependencias de la versión 2.1.0 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.2.0	Flexible

2.0.3

La siguiente tabla muestra las dependencias de la versión 2.0.3 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.3 <2.1.0	Flexible

Para obtener más información sobre las dependencias de los componentes, consulta la referencia de recetas de componentes.

Configuración

Este componente proporciona los siguientes parámetros de configuración que puede personalizar al implementar el componente.

v2.1.x

`subscriptions`

(Opcional) Las suscripciones que se van a habilitar en el dispositivo principal. Se trata de un objeto en el que cada clave es un identificador único y cada valor es un objeto que define la suscripción de ese conector. Debe configurar una suscripción al implementar un componente de conector V1 o una función Lambda que utilice el SDK AWS IoT Greengrass principal.

Cada objeto de suscripción contiene la siguiente información:

`id`

El identificador único de esta suscripción. Este ID debe coincidir con la clave de este objeto de suscripción.

`source`

La función Lambda que usa el SDK AWS IoT Greengrass principal para publicar mensajes MQTT sobre los temas que especifique. `subject` Especifique uno de los siguientes valores:

- El nombre de un componente de la función Lambda en el dispositivo principal. Especifique el nombre del componente con el `component:` prefijo, por ejemplo. **`component:com.example.HelloWorldLambda`**
- El nombre de recurso de Amazon (ARN) de una función Lambda en el dispositivo principal.

> **Important**
>
> Si la versión de la función Lambda cambia, debe configurar la suscripción con la nueva versión de la función. De lo contrario, este componente no enrutará los mensajes hasta que la versión coincida con la suscripción.
> Debe especificar un nombre de recurso de Amazon (ARN) que incluya la versión de la función que se va a importar. No puede utilizar alias de versión como `$LATEST`.

Para implementar una suscripción para un componente de conector V1, especifique el nombre del componente o el ARN de la función Lambda del componente conector.

`subject`

El tema o filtro de temas de MQTT en el que el origen y el destino pueden publicar y recibir mensajes. Este valor admite los caracteres comodín del # tema + y.

`target`

El destino que recibe los mensajes de MQTT sobre los temas que especifique. `subject` La suscripción especifica que la `source` función publique los mensajes MQTT en AWS IoT Core o en una función Lambda del dispositivo principal. Especifique uno de los siguientes valores:

- `cloud`. La `source` función publica mensajes MQTT en. AWS IoT Core
- El nombre de un componente de la función Lambda en el dispositivo principal. Especifique el nombre del componente con el `component:` prefijo, por ejemplo. **`component:com.example.HelloWorldLambda`**
- El nombre de recurso de Amazon (ARN) de una función Lambda en el dispositivo principal.

> **Important**
>
> Si la versión de la función Lambda cambia, debe configurar la suscripción con la nueva versión de la función. De lo contrario, este componente no enrutará los mensajes hasta que la versión coincida con la suscripción.
> Debe especificar un nombre de recurso de Amazon (ARN) que incluya la versión de la función que se va a importar. No puede utilizar alias de versión como `$LATEST`.

Predeterminado: sin suscripciones

Example Ejemplo de actualización de configuración (definición de una suscripción aAWS IoT Core)

El siguiente ejemplo especifica que el componente de la función `com.example.HelloWorldLambda` Lambda publica un mensaje MQTT AWS IoT Core en el tema. `hello/world`

```
{
  "subscriptions": {
```

```
    "Greengrass_HelloWorld_to_cloud": {
      "id": "Greengrass_HelloWorld_to_cloud",
      "source": "component:com.example.HelloWorldLambda",
      "subject": "hello/world",
      "target": "cloud"
    }
  }
}
```

Example Ejemplo de actualización de configuración (definición de una suscripción a otra función de Lambda)

El siguiente ejemplo especifica que el componente de función `com.example.HelloWorldLambda` Lambda publica mensajes MQTT en el componente de función `com.example.MessageRelay` Lambda sobre el tema. `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`

(Opcional) Las suscripciones que se van a habilitar en el dispositivo principal. Se trata de un objeto en el que cada clave es un identificador único y cada valor es un objeto que define la suscripción de ese conector. Debe configurar una suscripción al implementar un componente de conector V1 o una función Lambda que utilice el SDK AWS IoT Greengrass principal.

Cada objeto de suscripción contiene la siguiente información:

`id`

El identificador único de esta suscripción. Este ID debe coincidir con la clave de este objeto de suscripción.

`source`

La función Lambda que usa el SDK AWS IoT Greengrass principal para publicar mensajes MQTT sobre los temas que especifique. `subject` Especifique lo siguiente:

- El nombre de recurso de Amazon (ARN) de una función Lambda en el dispositivo principal.

> **Important**
>
> Si la versión de la función Lambda cambia, debe configurar la suscripción con la nueva versión de la función. De lo contrario, este componente no enrutará los mensajes hasta que la versión coincida con la suscripción.
> Debe especificar un nombre de recurso de Amazon (ARN) que incluya la versión de la función que se va a importar. No puede utilizar alias de versión como `$LATEST`.

Para implementar una suscripción para un componente de conector V1, especifique el ARN de la función Lambda del componente de conector.

`subject`

El tema o filtro de temas de MQTT en el que el origen y el destino pueden publicar y recibir mensajes. Este valor admite los caracteres comodín del # tema + y.

`target`

El destino que recibe los mensajes de MQTT sobre los temas que especifique. `subject` La suscripción especifica que la `source` función publique los mensajes MQTT en AWS IoT Core o en una función Lambda del dispositivo principal. Especifique uno de los siguientes valores:

- `cloud`. La `source` función publica mensajes MQTT en. AWS IoT Core
- El nombre de recurso de Amazon (ARN) de una función Lambda en el dispositivo principal.

> **Important**
>
> Si la versión de la función Lambda cambia, debe configurar la suscripción con la nueva versión de la función. De lo contrario, este componente no enrutará los mensajes hasta que la versión coincida con la suscripción.

Debe especificar un nombre de recurso de Amazon (ARN) que incluya la versión de la función que se va a importar. No puede utilizar alias de versión como `$LATEST`.

Predeterminado: sin suscripciones

Example Ejemplo de actualización de configuración (definición de una suscripción aAWS IoT Core)

El siguiente ejemplo especifica que la `Greengrass_HelloWorld` función publica un mensaje MQTT AWS IoT Core sobre el `hello/world` tema.

```
"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 Ejemplo de actualización de configuración (definición de una suscripción a otra función de Lambda)

El siguiente ejemplo especifica que la `Greengrass_HelloWorld` función publica los mensajes MQTT `Greengrass_MessageRelay` en el `hello/world` tema.

```
"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"
  }
}
```

Archivo de registro local

Este componente no genera registros.

Registros de cambios

En la siguiente tabla se describen los cambios en cada versión del componente.

Versión	Cambios
2.1.11	Versión actualizada para la versión 2.12.0 de Greengrass Nucleus.
2.1.10	Versión actualizada para el lanzamiento de la versión 2.11.0 de Greengrass nucleus.
2.1.9	Versión actualizada para el lanzamiento de la versión 2.10.0 de Greengrass nucleus.
2.1.8	Versión actualizada para la versión 2.9.0 de Greengrass Nucleus.
2.1.7	Versión actualizada para el lanzamiento de la versión 2.8.0 del núcleo de Greengrass.
2.1.6	Versión actualizada para el lanzamiento de la versión 2.7.0 de Greengrass nucleus.
2.1.5	Versión actualizada para el lanzamiento de la versión 2.6.0 de Greengrass Nucleus.
2.1.4	Versión actualizada para el lanzamiento de la versión 2.5.0 de Greengrass Nucleus.
2.1.3	Versión actualizada para la versión 2.4.0 de Greengrass Nucleus.
2.1.2	Versión actualizada para el lanzamiento de la versión 2.3.0 de Greengrass nucleus.
2.1.1	Versión actualizada para el lanzamiento de la versión 2.2.0 de Greengrass nucleus.

Versión	Cambios
2.1.0	Mejoras y correcciones de errores • Añade soporte para especificar nombres de componentes en lugar de ARN para `source` y `target`. Si especifica un nombre de componente para una suscripción, no necesita volver a configurar la suscripción cada vez que cambie la versión de la función Lambda.
2.0.3	Versión inicial.

Consola de depuración local

El componente de la consola de depuración local (`aws.greengrass.LocalDebugConsole`) proporciona un panel local que muestra información sobre los dispositivos AWS IoT Greengrass principales y sus componentes. Puede usar este panel para depurar su dispositivo principal y administrar los componentes locales.

> **Important**
>
> Le recomendamos que utilice este componente únicamente en entornos de desarrollo, no en entornos de producción. Este componente proporciona acceso a información y operaciones que normalmente no necesitará en un entorno de producción. Siga el principio de privilegios mínimos implementando este componente solo en los dispositivos principales donde lo necesite.

Temas

- Registros de cambios

Versiones

Este componente tiene las siguientes versiones:

- 2.3.x
- 2.2.x
- 2.1.x
- 2.0.x

Tipo

Este componente es un componente de complemento ()`aws.greengrass.plugin`. El núcleo de Greengrass ejecuta este componente en la misma máquina virtual Java (JVM) que el núcleo. El núcleo se reinicia al cambiar la versión de este componente en el dispositivo principal.

Este componente utiliza el mismo archivo de registro que el núcleo de Greengrass. Para obtener más información, consulte Supervisar AWS IoT Greengrass registros.

Para obtener más información, consulte Tipos de componentes.

Sistema operativo

Este componente se puede instalar en los dispositivos principales que ejecutan los siguientes sistemas operativos:

- Linux
- Windows

Requisitos

Este componente tiene los siguientes requisitos:

- Utilice un nombre de usuario y una contraseña para iniciar sesión en el panel de control. El nombre de usuario, es `debug` decir, se proporciona para usted. Debe usar la AWS IoT Greengrass CLI para crear una contraseña temporal que lo autentique con el panel de control de un dispositivo principal. Debe poder usar la AWS IoT Greengrass CLI para usar la consola de depuración local.

Para obtener más información, consulte los requisitos de la CLI de Greengrass. Para obtener más información sobre cómo generar la contraseña e iniciar sesión, consulte Uso de los componentes de la consola de depuración local.

- Se admite la ejecución del componente de la consola de depuración local en una VPC.

Dependencias

Al implementar un componente, AWS IoT Greengrass también despliega versiones compatibles de sus dependencias. Esto significa que debe cumplir los requisitos del componente y de todas sus dependencias para poder implementarlo correctamente. En esta sección se enumeran las dependencias de las versiones publicadas de este componente y las restricciones de las versiones semánticas que definen las versiones de los componentes para cada dependencia. También puede ver las dependencias de cada versión del componente en la consola.AWS IoT Greengrass En la página de detalles del componente, busque la lista de dependencias.

2.4.1 – 2.4.2

La siguiente tabla muestra las dependencias de las versiones 2.4.1 a 2.4.2 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.10.0 <2.13.0	Rígido
Greengrass CLI	>=2.10.0 <2.13.0	Rígido

2.4.0

La siguiente tabla muestra las dependencias de la versión 2.4.0 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.10.0 <2.12.0	Rígido
Greengrass CLI	>=2.10.0 <2.12.0	Rígido

2.3.0 and 2.3.1

La siguiente tabla muestra las dependencias de las versiones 2.3.0 y 2.3.1 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.10.0 <2.12.0	Rígido
Greengrass CLI	>=2.10.0 <2.12.0	Rígido

2.2.9

La siguiente tabla muestra las dependencias de la versión 2.2.9 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.1.0 <2.12.0	Rígido
Greengrass CLI	>=2.1.0 <2.12.0	Rígido

2.2.8

La siguiente tabla muestra las dependencias de la versión 2.2.8 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.1.0 <2.11.0	Rígido
Greengrass CLI	>=2.1.0 <2.11.0	Rígido

2.2.7

La siguiente tabla muestra las dependencias de la versión 2.2.7 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.1.0 <2.10.0	Rígido
Greengrass CLI	>=2.1.0 <2.10.0	Rígido

2.2.6

La siguiente tabla muestra las dependencias de la versión 2.2.6 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.1.0 <2.9.0	Rígido
Greengrass CLI	>=2.1.0 <2.9.0	Rígido

2.2.5

La siguiente tabla muestra las dependencias de la versión 2.2.5 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.1.0 <2.8.0	Rígido
Greengrass CLI	>=2.1.0 <2.8.0	Rígido

2.2.4

La siguiente tabla muestra las dependencias de la versión 2.2.4 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.1.0 <2.7.0	Rígido
Greengrass CLI	>=2.1.0 <2.7.0	Rígido

2.2.3

La siguiente tabla muestra las dependencias de la versión 2.2.3 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.1.0 <2.6.0	Rígido

Dependencia	Versiones compatibles	Tipo de dependencia
Greengrass CLI	>=2.1.0 <2.6.0	Rígido

2.2.2

La siguiente tabla muestra las dependencias de la versión 2.2.2 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.1.0 <2.5.0	Rígido
Greengrass CLI	>=2.1.0 <2.5.0	Rígido

2.2.1

La siguiente tabla muestra las dependencias de la versión 2.2.1 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.1.0 <2.4.0	Rígido
Greengrass CLI	>=2.1.0 <2.4.0	Rígido

2.2.0

La siguiente tabla muestra las dependencias de la versión 2.2.0 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.1.0 <2.3.0	Rígido
Greengrass CLI	>=2.1.0 <2.3.0	Rígido

2.1.0

La siguiente tabla muestra las dependencias de la versión 2.1.0 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.1.0 <2.2.0	Rígido
Greengrass CLI	>=2.1.0 <2.2.0	Rígido

2.0.x

La siguiente tabla muestra las dependencias de la versión 2.0.x de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.3 <2.1.0	Flexible
Greengrass CLI	>=2.0.3 <2.1.0	Flexible

Para obtener más información sobre las dependencias de los componentes, consulta la referencia de recetas de componentes.

Configuración

Este componente proporciona los siguientes parámetros de configuración que puede personalizar al implementar el componente.

v2.1.x - v2.4.x

`httpsEnabled`

(Opcional) Puede habilitar la comunicación HTTPS para la consola de depuración local. Si habilita la comunicación HTTPS, la consola de depuración local crea un certificado autofirmado. Los navegadores web muestran advertencias de seguridad para los sitios web que utilizan certificados autofirmados, por lo que debe verificar el certificado manualmente. A continuación, puede omitir la advertencia. Para obtener más información, consulte Uso.

Valor predeterminado: `true`

`port`

(Opcional) El puerto en el que se va a proporcionar la consola de depuración local.

Valor predeterminado: `1441`

`websocketPort`

(Opcional) El puerto websocket que se utilizará en la consola de depuración local.

Valor predeterminado: `1442`

`bindHostname`

(Opcional) El nombre de host que se utilizará en la consola de depuración local.

Si ejecuta el software AWS IoT Greengrass principal en un contenedor de Docker, defina este parámetro en para `0.0.0.0` poder abrir la consola de depuración local fuera del contenedor de Docker.

Valor predeterminado: `localhost`

Example Ejemplo: actualización de combinación de configuraciones

El siguiente ejemplo de configuración especifica abrir la consola de depuración local en puertos no predeterminados e inhabilitar HTTPS.

```
{
  "httpsEnabled": false,
  "port": "10441",
  "websocketPort": "10442"
}
```

v2.0.x

`port`

(Opcional) El puerto en el que se va a proporcionar la consola de depuración local.

Valor predeterminado: `1441`

`websocketPort`

(Opcional) El puerto websocket que se utilizará en la consola de depuración local.

Valor predeterminado: `1442`

`bindHostname`

(Opcional) El nombre de host que se utilizará en la consola de depuración local.

Si ejecuta el software AWS IoT Greengrass principal en un contenedor de Docker, defina este parámetro en para `0.0.0.0` poder abrir la consola de depuración local fuera del contenedor de Docker.

Valor predeterminado: `localhost`

Example Ejemplo: actualización de combinación de configuraciones

El siguiente ejemplo de configuración especifica abrir la consola de depuración local en puertos no predeterminados.

```
{
  "port": "10441",
  "websocketPort": "10442"
}
```

Uso

Para usar la consola de depuración local, cree una sesión desde la CLI de Greengrass. Al crear una sesión, la CLI de Greengrass proporciona un nombre de usuario y una contraseña temporal que puede usar para iniciar sesión en la consola de depuración local.

Siga estas instrucciones para abrir la consola de depuración local en su dispositivo principal o en su ordenador de desarrollo.

v2.1.x - v2.4.x

En las versiones 2.1.0 y posteriores, la consola de depuración local usa HTTPS de forma predeterminada. Cuando HTTPS está habilitado, la consola de depuración local crea un certificado autofirmado para proteger la conexión. Su navegador web muestra una advertencia de seguridad al abrir la consola de depuración local debido a este certificado autofirmado. Al crear una sesión con la CLI de Greengrass, el resultado incluye las huellas digitales del certificado para que pueda comprobar que el certificado es legítimo y que la conexión es segura.

Puede deshabilitar HTTPS. Para obtener más información, consulte Configuración de la consola de depuración local.

Para abrir la consola de depuración local

1. (Opcional) Para ver la consola de depuración local en tu ordenador de desarrollo, puedes reenviar el puerto de la consola a través de SSH. Sin embargo, primero debes habilitar la `AllowTcpForwarding` opción en el archivo de configuración SSH de tu dispositivo principal. Esta opción está habilitada de forma predeterminada. Ejecuta el siguiente comando en tu ordenador de desarrollo para ver el panel de control `localhost:1441` en tu ordenador de desarrollo.

   ```
   ssh -L 1441:localhost:1441 -L 1442:localhost:1442 username@core-device-ip-address
   ```

   > **Note**
   >
   > Puede cambiar los puertos predeterminados de `1441` y`1442`. Para obtener más información, consulte Configuración de la consola de depuración local.

2. Cree una sesión para usar la consola de depuración local. Al crear una sesión, se genera una contraseña que se utiliza para autenticarse. La consola de depuración local requiere una contraseña para aumentar la seguridad, ya que puede utilizar este componente para ver información importante y realizar operaciones en el dispositivo principal. La consola de depuración local también crea un certificado para proteger la conexión si se habilita HTTPS en la configuración del componente. HTTPS está activado de forma predeterminada.

   Utilice la AWS IoT Greengrass CLI para crear la sesión. Este comando genera una contraseña aleatoria de 43 caracteres que caduca después de 8 horas. Sustituya */greengrass/v2* o *C:\greengrass\v2* por la ruta de acceso a la carpeta AWS IoT Greengrass V2 raíz.

   Linux or Unix

   ```
   sudo /greengrass/v2/bin/greengrass-cli get-debug-password
   ```

   Windows

   ```
   C:\greengrass\v2\bin\greengrass-cli get-debug-password
   ```

El resultado del comando es similar al del siguiente ejemplo si ha configurado la consola de depuración local para que utilice HTTPS. Las huellas digitales del certificado se utilizan para comprobar que la conexión es segura al abrir la consola de depuración local.

```
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
```

El componente de vista de depuración crea una sesión que dura 8 horas. Después de eso, debe generar una nueva contraseña para volver a ver la consola de depuración local.

3. Abre e inicia sesión en el panel de control. Vea el panel de control en su dispositivo principal de Greengrass o en su ordenador de desarrollo si reenvía el puerto a través de SSH. Realice una de las acciones siguientes:

   - Si habilitó HTTPS en la consola de depuración local, que es la configuración predeterminada, haga lo siguiente:

     a. Ábrelo `https://localhost:1441` en tu dispositivo principal o en tu ordenador de desarrollo si reenviaste el puerto a través de SSH.

        Es posible que tu navegador muestre una advertencia de seguridad sobre un certificado de seguridad no válido.

     b. Si su navegador muestra una advertencia de seguridad, compruebe que el certificado sea legítimo y omita la advertencia de seguridad. Haga lo siguiente:

        i. Busque la huella digital SHA-256 o SHA-1 del certificado y compruebe que coincide con la huella digital SHA-256 o SHA-1 que el comando imprimió anteriormente. `get-debug-password` Es posible que su navegador proporcione una huella digital o ambas. Consulte la documentación del

navegador para ver el certificado y sus huellas digitales. En algunos navegadores, la huella digital del certificado se denomina huella digital.

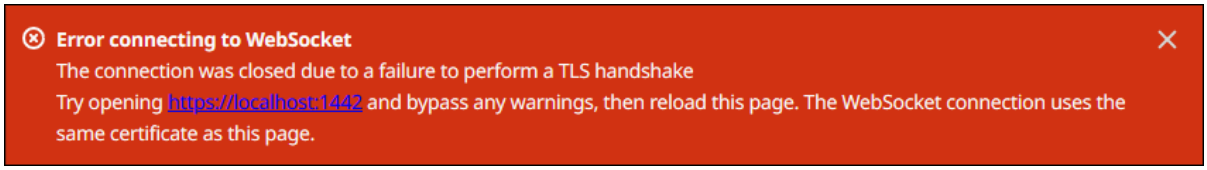
> **Note**
>
> Si la huella digital del certificado no coincide, vaya Step 2 a para crear una nueva sesión. Si la huella digital del certificado sigue sin coincidir, es posible que la conexión no sea segura.

ii. Si la huella digital del certificado coincide, omite la advertencia de seguridad del navegador para abrir la consola de depuración local. Consulte la documentación del navegador para omitir la advertencia de seguridad del navegador.

c. Inicie sesión en el sitio web con el nombre de usuario y la contraseña que el `get-debug-password` comando imprimía anteriormente.

Se abre la consola de depuración local.

d. Si la consola de depuración local muestra un error que indica que no se puede conectar a ella WebSocket debido a un error en el protocolo de enlace TLS, debes omitir la advertencia de seguridad autofirmada de la URL. WebSocket



Haga lo siguiente:

i. Ábrelo `https://localhost:1442` en el mismo navegador en el que abriste la consola de depuración local.

ii. Compruebe el certificado y omita la advertencia de seguridad.

Es posible que tu navegador muestre una página HTTP 404 después de omitir la advertencia.

iii. Abre de `https://localhost:1441` nuevo.

La consola de depuración local muestra información sobre el dispositivo principal.

- Si deshabilitó HTTPS en la consola de depuración local, haga lo siguiente:

a. Ábrelo `http://localhost:1441` en tu dispositivo principal o ábrelo en tu ordenador de desarrollo si has reenviado el puerto a través de SSH.

b. Inicie sesión en el sitio web con el nombre de usuario y la contraseña que el `get-debug-password` comando imprimió anteriormente.

   Se abre la consola de depuración local.

v2.0.x

Para abrir la consola de depuración local

1. (Opcional) Para ver la consola de depuración local en tu ordenador de desarrollo, puedes reenviar el puerto de la consola a través de SSH. Sin embargo, primero debes habilitar la `AllowTcpForwarding` opción en el archivo de configuración SSH de tu dispositivo principal. Esta opción está habilitada de forma predeterminada. Ejecuta el siguiente comando en tu ordenador de desarrollo para ver el panel de control `localhost:1441` en tu ordenador de desarrollo.

   ```
   ssh -L 1441:localhost:1441 -L 1442:localhost:1442 username@core-device-ip-address
   ```

   > **Note**
   >
   > Puede cambiar los puertos predeterminados de `1441` y`1442`. Para obtener más información, consulte Configuración de la consola de depuración local.

2. Cree una sesión para usar la consola de depuración local. Al crear una sesión, se genera una contraseña que se utiliza para autenticarse. La consola de depuración local requiere una contraseña para aumentar la seguridad, ya que puede utilizar este componente para ver información importante y realizar operaciones en el dispositivo principal.

   Utilice la AWS IoT Greengrass CLI para crear la sesión. Este comando genera una contraseña aleatoria de 43 caracteres que caduca después de 8 horas. Sustituya */greengrass/v2* o *C:\greengrass\v2* por la ruta de acceso a la carpeta AWS IoT Greengrass V2 raíz.

Linux or Unix

```
sudo /greengrass/v2/bin/greengrass-cli get-debug-password
```

Windows

```
C:\greengrass\v2\bin\greengrass-cli get-debug-password
```

El resultado del comando es similar al del siguiente ejemplo.

```
Username: debug
Password: bEDp3MOHdj8ou2w5de_sCBI2XAaguy3a8XxREXAMPLE
Password will expire at: 2021-04-01T17:01:43.921999931-07:00
```

El componente de vista de depuración crea una sesión que dura 4 horas y, a continuación, debe generar una nueva contraseña para volver a ver la consola de depuración local.

3. Ábrelo `http://localhost:1441` en tu dispositivo principal o ábrelo en tu ordenador de desarrollo si has reenviado el puerto a través de SSH.
4. Inicie sesión en el sitio web con el nombre de usuario y la contraseña que el `get-debug-password` comando imprimió anteriormente.

   Se abre la consola de depuración local.

Archivo de registro local

Este componente utiliza el mismo archivo de registro que el componente núcleo de Greengrass.

Linux

```
/greengrass/v2/logs/greengrass.log
```

Windows

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

Para ver los registros de este componente

- Ejecute el siguiente comando en el dispositivo principal para ver el archivo de registro de este componente en tiempo real. Sustituya */greengrass/v2* o *C:\greengrass\v2* por la ruta a la carpeta AWS IoT Greengrass raíz.

  Linux

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

  Windows (PowerShell)

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

Registros de cambios

En la siguiente tabla se describen los cambios en cada versión del componente.

Versión	Cambios
2.4.2	Mejoras y correcciones de errores • Corrección de errores y mejoras generales.
2.4.1	Versión actualizada para la versión 2.12.0 de Greengrass nucleus.
2.4.0	Nuevas características • Añade la consola de depuración de Stream Manager.
2.3.1	Versión actualizada para la versión 2.11.0 de Greengrass nucleus.
2.3.0	Versión actualizada para la versión 2.10.0 de Greengrass nucleus. Nuevas características • Incluye un cliente de PubSub depuración AWS IoT Core MQTT.
2.2.7	Versión actualizada para la versión 2.9.0 de Greengrass nucleus.
2.2.6	Versión actualizada para la versión 2.8.0 de Greengrass nucleus.

Versión	Cambios
2.2.5	Versión actualizada para la versión 2.7.0 de Greengrass Nucleus.
2.2.4	Versión actualizada para la versión 2.6.0 de Greengrass nucleus.
2.2.3	Mejoras y correcciones de errores • Corrige un problema que impedía el inicio cuando el componente no podía descifrar el almacén de claves que contiene la clave privada SSL. • Versión actualizada para la versión 2.5.0 de Greengrass nucleus.
2.2.2	Versión actualizada para la versión 2.4.0 de Greengrass Nucleus.
2.2.1	Versión actualizada para la versión 2.3.0 de Greengrass nucleus.
2.2.0	Versión actualizada para el lanzamiento de la versión 2.2.0 de Greengrass nucleus.
2.1.0	Nuevas características • Utiliza HTTPS para proteger la conexión a la consola de depuración local. HTTPS está activado de forma predeterminada. Mejoras y correcciones de errores • Puede descartar los mensajes de la barra flash en el editor de configura ción.
2.0.3	Versión inicial.

Gestor de registros

El componente gestor de registros (`aws.greengrass.LogManager`) carga los registros de los dispositivos AWS IoT Greengrass principales a Amazon CloudWatch Logs. Puede cargar registros desde el núcleo de Greengrass, otros componentes de Greengrass y otras aplicaciones y servicios que no son componentes de Greengrass. Para obtener más información sobre cómo supervisar los CloudWatch registros en Logs y en el sistema de archivos local, consulte. Supervisar AWS IoT Greengrass registros

Cuando se utiliza el componente gestor de registros para escribir en los CloudWatch registros, se tienen en cuenta las siguientes consideraciones:

- Retrasos de registro

> **Note**
>
> Le recomendamos que actualice a la versión 2.3.0 del administrador de registros, ya que reduce las demoras en el registro de los archivos de registro activos y rotados. Cuando actualice a log manager 2.3.0, le recomendamos que también actualice a Greengrass nucleus 2.9.1.

La versión 2.2.8 (y anteriores) del componente gestor de registros procesa y carga los registros únicamente a partir de archivos de registro rotados. De forma predeterminada, el software AWS IoT Greengrass Core rota los archivos de registro cada hora o después de que ocupen 1024 KB. Como resultado, el componente del administrador de registros carga los registros solo después de que el software AWS IoT Greengrass Core o un componente de Greengrass hayan escrito registros con un valor superior a 1024 KB. Puede configurar un límite de tamaño de archivo de registro inferior para que los archivos de registro roten con más frecuencia. Esto hace que el componente del administrador de registros cargue registros en CloudWatch Logs con más frecuencia.

La versión 2.3.0 (y posteriores) del componente administrador de registros procesa y carga todos los registros. Al escribir un registro nuevo, la versión 2.3.0 (y posteriores) del administrador de registros procesa y carga directamente el archivo de registro activo en lugar de esperar a que se rote. Esto significa que puede ver el nuevo registro en 5 minutos o menos.

El componente del administrador de registros carga nuevos registros periódicamente. De forma predeterminada, el componente del administrador de registros carga nuevos registros cada 5 minutos. Puede configurar un intervalo de carga más bajo, de modo que el componente del administrador de registros cargue los registros en los CloudWatch registros con más frecuencia configurando el. `periodicUploadIntervalSec` Para obtener más información sobre cómo configurar este intervalo periódico, consulte Configuración.

Los registros se pueden cargar prácticamente en tiempo real desde el mismo sistema de archivos de Greengrass. Si necesita observar los registros en tiempo real, considere la posibilidad de utilizar los registros del sistema de archivos.

> **Note**
>
> Si utiliza distintos sistemas de archivos para escribir los registros, el administrador de registros vuelve al comportamiento de las versiones 2.2.8 y anteriores de los componentes del administrador de registros. Para obtener información sobre cómo acceder a los registros del sistema de archivos, consulte Acceder a los registros del sistema de archivos.

- Inclinación del reloj

  El componente de gestión de registros utiliza el proceso de firma estándar de la versión 4 de Signature para crear solicitudes de API a CloudWatch los registros. Si la hora del sistema en un dispositivo principal está desincronizada durante más de 15 minutos, CloudWatch Logs rechaza las solicitudes. Para obtener más información, consulte Proceso de firma Signature Version 4 en la Referencia general de AWS.

Para obtener información sobre los grupos de registros y los flujos de registros a los que este componente carga los registros, consulteUso.

Temas

- Versiones
- Tipo
- Sistema operativo
- Requisitos
- Dependencias
- Configuración
- Uso
- Archivo de registro local
- Registros de cambios

Versiones

Este componente tiene las siguientes versiones:

- 2.3.x
- 2.2.x

- 2.1.x
- 2.0.x

Tipo

Este componente es un componente de complemento ()`aws.greengrass.plugin`. El núcleo de Greengrass ejecuta este componente en la misma máquina virtual Java (JVM) que el núcleo. El núcleo se reinicia al cambiar la versión de este componente en el dispositivo principal.

Este componente utiliza el mismo archivo de registro que el núcleo de Greengrass. Para obtener más información, consulte Supervisar AWS IoT Greengrass registros.

Para obtener más información, consulte Tipos de componentes.

Sistema operativo

Este componente se puede instalar en los dispositivos principales que ejecutan los siguientes sistemas operativos:

- Linux
- Windows

Requisitos

Este componente tiene los siguientes requisitos:

- El rol de dispositivo Greengrass debe permitir las `logs:DescribeLogStreams` acciones`logs:CreateLogGroup`,, y `logs:CreateLogStreamlogs:PutLogEvents`, como se muestra en el siguiente ejemplo de política de IAM.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:DescribeLogStreams"
      ],
```

```
      "Effect": "Allow",
      "Resource": "arn:aws:logs:*:*:*"
    }
  ]
}
```

> **Note**
>
> La función de dispositivo Greengrass que se crea al instalar el software AWS IoT Greengrass Core incluye los permisos de esta política de ejemplo de forma predeterminada.

Para obtener más información, consulte Uso de políticas basadas en la identidad (políticas de IAM) para CloudWatch los registros en la Guía del usuario de Amazon CloudWatch Logs.

- Se admite la ejecución del componente de administrador de registros en una VPC. Para implementar este componente en una VPC, se requiere lo siguiente.
  - El componente del administrador de registros debe tener una conectividad con la `logs.region.amazonaws.com` que tenga el punto final de la VPC de. `com.amazonaws.us-east-1.logs`

Puntos finales y puertos

Este componente debe poder realizar solicitudes salientes a los siguientes puntos finales y puertos, además de a los puntos finales y puertos necesarios para el funcionamiento básico. Para obtener más información, consulte Permitir el tráfico del dispositivo a través de un proxy o firewall.

punto de enlace	Puerto	Obligatoria	Descripción
`logs.`*`region`*`.amazonaws.com`	443	No	Es obligatorio si escribes registros en Logs. CloudWatch

Dependencias

Al implementar un componente, AWS IoT Greengrass también despliega versiones compatibles de sus dependencias. Esto significa que debe cumplir los requisitos del componente y de todas sus dependencias para poder implementarlo correctamente. En esta sección se enumeran las dependencias de las versiones publicadas de este componente y las restricciones de las versiones semánticas que definen las versiones de los componentes para cada dependencia. También puede ver las dependencias de cada versión del componente en la consola. AWS IoT Greengrass En la página de detalles del componente, busque la lista de dependencias.

2.3.7

La siguiente tabla muestra las dependencias de la versión 2.3.7 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.1.0 <2.13.0	Flexible

2.3.5 and 2.3.6

La siguiente tabla muestra las dependencias de las versiones 2.3.5 y 2.3.6 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.1.0 <2.12.0	Flexible

2.3.3 – 2.3.4

La siguiente tabla muestra las dependencias de las versiones 2.3.3 a 2.3.4 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.1.0 <2.11.0	Flexible

2.2.8 – 2.3.2

La siguiente tabla muestra las dependencias de las versiones 2.2.8 a 2.3.2 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.1.0 <2.10.0	Flexible

2.2.7

La siguiente tabla muestra las dependencias de la versión 2.2.7 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.1.0 <2.9.0	Flexible

2.2.6

La siguiente tabla muestra las dependencias de la versión 2.2.6 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.1.0 <2.8.0	Flexible

2.2.5

La siguiente tabla muestra las dependencias de la versión 2.2.5 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.1.0 <2.7.0	Flexible

2.2.1 - 2.2.4

La siguiente tabla muestra las dependencias de las versiones 2.2.1 a 2.2.4 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.1.0 <2.6.0	Flexible

2.1.3 and 2.2.0

La siguiente tabla muestra las dependencias de las versiones 2.1.3 y 2.2.0 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.1.0 <2.5.0	Flexible

2.1.2

La siguiente tabla muestra las dependencias de la versión 2.1.2 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.1.0 <2.4.0	Flexible

2.1.1

La siguiente tabla muestra las dependencias de la versión 2.1.1 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.1.0 <2.3.0	Flexible

2.1.0

La siguiente tabla muestra las dependencias de la versión 2.1.0 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.1.0 <2.2.0	Flexible

2.0.x

La siguiente tabla muestra las dependencias de la versión 2.0.x de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.3 <2.1.0	Flexible

Para obtener más información sobre las dependencias de los componentes, consulta la referencia de recetas de componentes.

Configuración

Este componente proporciona los siguientes parámetros de configuración que puede personalizar al implementar el componente.

v2.3.6 – v2.3.7

`logsUploaderConfiguration`

(Opcional) La configuración de los registros que carga el componente del administrador de registros. Este objeto contiene la siguiente información:

`systemLogsConfiguration`

(Opcional) La configuración de los registros del sistema del software AWS IoT Greengrass principal, que incluyen los registros del núcleo y los componentes del complemento de Greengrass. Especifique esta configuración para permitir que el componente gestor de registros gestione los registros del sistema. Este objeto contiene la siguiente información:

`uploadToCloudWatch`

(Opcional) Puede cargar los registros del sistema en CloudWatch Logs.

Valor predeterminado: `false`

`minimumLogLevel`

(Opcional) El nivel mínimo de mensajes de registro que se deben cargar. Este nivel mínimo se aplica solo si configura el componente núcleo de Greengrass para generar registros en formato JSON. Para habilitar los registros en formato JSON, especifique `JSON` el parámetro de formato de registro (`logging.format`).

Elija uno de los siguientes niveles de registro, que se muestran aquí en orden de niveles:

- `DEBUG`

- `INFO`
- `WARN`
- `ERROR`

Valor predeterminado: `INFO`

`diskSpaceLimit`

(Opcional) El tamaño total máximo de los archivos de registro del sistema Greengrass, en la unidad en la que especifique. `diskSpaceLimitUnit` Cuando el tamaño total de los archivos de registro del sistema Greengrass supera este tamaño total máximo, el software AWS IoT Greengrass Core elimina los archivos de registro del sistema Greengrass más antiguos.

Este parámetro equivale al parámetro de límite de tamaño logarítmico (`totalLogsSizeKB`) del componente núcleo de Greengrass. El software AWS IoT Greengrass Core utiliza el mínimo de los dos valores como tamaño máximo total del registro del sistema Greengrass.

`diskSpaceLimitUnit`

(Opcional) La unidad para. `diskSpaceLimit` Puede elegir entre las siguientes opciones:

- `KB`— kilobytes
- `MB`— megabytes
- `GB`— gigabytes

Valor predeterminado: `KB`

`deleteLogFileAfterCloudUpload`

(Opcional) Puede eliminar un archivo de registro después de que el componente del administrador de registros cargue los registros en Logs. CloudWatch

Valor predeterminado: `false`

`componentLogsConfigurationMap`

(Opcional) Un mapa de las configuraciones de registro de los componentes del dispositivo principal. Cada `componentName` objeto de este mapa define la configuración de registro del componente o la aplicación. El componente del administrador de registros carga estos registros de componentes en CloudWatch Logs.

> **Important**
>
> Recomendamos encarecidamente utilizar una única clave de configuración por componente. Al usar el, solo debe dirigirse a un grupo de archivos que solo tengan un archivo de registro en el que se esté escribiendo activamente`logFileRegex`. Si no se sigue esta recomendación, es posible que se carguen registros duplicados en él CloudWatch. Si se dirige a varios archivos de registro activos con una sola expresión regular, le recomendamos que actualice a la versión 2.3.1 o posterior del administrador de registros y que considere la posibilidad de cambiar la configuración mediante la configuración de ejemplo.

> **Note**
>
> Si está actualizando desde una versión del administrador de registros anterior a la v2.2.0, puede seguir utilizando la lista en su lugar. `componentLogsConfiguration` `componentLogsConfigurationMap` Sin embargo, le recomendamos encarecidamente que utilice el formato de mapa para poder utilizar las actualizaciones de combinación y restablecimiento para modificar las configuraciones de componentes específicos. Para obtener información sobre el `componentLogsConfiguration` parámetro, consulte los parámetros de configuración de la versión 2.1.x de este componente.

`componentName`

La configuración de registro del *`componentName`*componente o la aplicación de esta configuración de registro. Puede especificar el nombre de un componente de Greengrass u otro valor para identificar este grupo de registros.

Cada objeto contiene la siguiente información:

`minimumLogLevel`

(Opcional) El nivel mínimo de mensajes de registro que se deben cargar. Este nivel mínimo solo se aplica si los registros de este componente utilizan un formato JSON específico, que se encuentra en el repositorio del módulo de AWS IoT Greengrass registro GitHub.

Elija uno de los siguientes niveles de registro, que se muestran aquí en orden de niveles:

- `DEBUG`
- `INFO`
- `WARN`
- `ERROR`

Valor predeterminado: `INFO`

`diskSpaceLimit`

(Opcional) El tamaño total máximo de todos los archivos de registro de este componente, en la unidad en la que especifique`diskSpaceLimitUnit`. Cuando el tamaño total de los archivos de registro de este componente supere este tamaño total máximo, el software AWS IoT Greengrass Core elimina los archivos de registro más antiguos de este componente.

Este parámetro está relacionado con el parámetro límite de tamaño logarítmico (`totalLogsSizeKB`) del componente núcleo de Greengrass. El software AWS IoT Greengrass Core utiliza el mínimo de los dos valores como tamaño de registro total máximo para este componente.

`diskSpaceLimitUnit`

(Opcional) La unidad del`diskSpaceLimit`. Puede elegir entre las siguientes opciones:

- `KB`— kilobytes
- `MB`— megabytes
- `GB`— gigabytes

Valor predeterminado: `KB`

`logFileDirectoryPath`

(Opcional) La ruta a la carpeta que contiene los archivos de registro de este componente.

No es necesario especificar este parámetro para los componentes de Greengrass que imprimen en salida estándar (stdout) y error estándar (stderr).

Valor predeterminado: */greengrass/v2*/logs.

`logFileRegex`

(Opcional) Expresión regular que especifica el formato de nombre del archivo de registro que utiliza el componente o la aplicación. El componente gestor de registros utiliza esta expresión regular para identificar los archivos de registro de la carpeta situada en`logFileDirectoryPath`.

No es necesario especificar este parámetro para los componentes de Greengrass que imprimen en salida estándar (stdout) y error estándar (stderr).

Si su componente o aplicación rota los archivos de registro, especifique una expresión regular que coincida con los nombres de los archivos de registro rotados. Por ejemplo, puede especificar **`hello_world\\\\w*.log`** cargar los registros de una aplicación de Hello World. El `\\\\w*` patrón coincide con cero o más caracteres de palabra, lo que incluye caracteres alfanuméricos y guiones bajos. Esta expresión regular hace coincidir los archivos de registro con y sin marcas de tiempo en su nombre. En este ejemplo, el administrador de registros carga los siguientes archivos de registro:

- `hello_world.log`— El archivo de registro más reciente de la aplicación Hello World.
- `hello_world_2020_12_15_17_0.log`— Un archivo de registro antiguo para la aplicación Hello World.

Predeterminado:*componentName*`\\\\w*.log`, donde *componentName* es el nombre del componente de esta configuración de registro.

`deleteLogFileAfterCloudUpload`

(Opcional) Puede eliminar un archivo de registro después de que el componente del administrador de registros cargue los registros en Logs. CloudWatch

Valor predeterminado: `false`

`multiLineStartPattern`

(Opcional) Expresión regular que identifica cuándo un mensaje de registro de una línea nueva es un mensaje de registro nuevo. Si la expresión regular no coincide con la nueva línea, el componente del administrador de registros anexa la nueva línea al mensaje de registro de la línea anterior.

De forma predeterminada, el componente del administrador de registros comprueba si la línea comienza con un carácter de espacio en blanco, como una pestaña o un espacio. Si no es así, el administrador de registros trata esa línea como un nuevo mensaje de registro. De lo contrario, añade esa línea al mensaje de registro actual. Este comportamiento garantiza que el componente del administrador de registros no divida los mensajes que ocupan varias líneas, como los seguimientos de pila.

`periodicUploadIntervalSec`

(Opcional) El período en segundos durante el que el componente del administrador de registros comprueba si hay nuevos archivos de registro para cargar.

Predeterminado: `300` (5 minutos)

Mínimo: `0.000001` (1 microsegundo)

`deprecatedVersionSupport`

Indica si el administrador de registros debe utilizar las mejoras de velocidad de registro introducidas en la versión 2.3.5 del administrador de registros. Establezca el valor en `false` para utilizar las mejoras.

Si establece este valor `false` al actualizar desde la versión 2.3.1 del administrador de registros o una versión anterior, es posible que se carguen entradas de registro duplicadas.

El valor predeterminado es `true`.

Example Ejemplo: actualización de combinación de configuraciones

El siguiente ejemplo de configuración especifica cargar los registros del sistema y los registros de los `com.example.HelloWorld` componentes en CloudWatch Logs.

```
{
  "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 Ejemplo: configuración para cargar varios archivos de registro activos mediante el administrador de registros v2.3.1

El siguiente ejemplo de configuración es el ejemplo recomendado si desea dirigirse a varios archivos de registro activos. En este ejemplo de configuración se especifican los archivos de registro activos en los que desea cargarlos CloudWatch. Si utiliza este ejemplo de configuración, la configuración también cargará todos los archivos rotados que coincidan con. `logFileRegex` Este ejemplo de configuración es compatible con la versión 2.3.1 del administrador de registros.

```
{
  "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`

(Opcional) La configuración de los registros que carga el componente del administrador de registros. Este objeto contiene la siguiente información:

`systemLogsConfiguration`

(Opcional) La configuración de los registros del sistema del software AWS IoT Greengrass principal, que incluyen los registros del núcleo y los componentes del complemento de Greengrass. Especifique esta configuración para permitir que el componente gestor de registros gestione los registros del sistema. Este objeto contiene la siguiente información:

`uploadToCloudWatch`

(Opcional) Puede cargar los registros del sistema en CloudWatch Logs.

Valor predeterminado: `false`

`minimumLogLevel`

(Opcional) El nivel mínimo de mensajes de registro que se deben cargar. Este nivel mínimo se aplica solo si configura el componente núcleo de Greengrass para generar registros en formato JSON. Para habilitar los registros en formato JSON, especifique `JSON` el parámetro de formato de registro (`logging.format`).

Elija uno de los siguientes niveles de registro, que se muestran aquí en orden de niveles:

- `DEBUG`
- `INFO`
- `WARN`
- `ERROR`

Valor predeterminado: `INFO`

`diskSpaceLimit`

(Opcional) El tamaño total máximo de los archivos de registro del sistema Greengrass, en la unidad en la que especifique. `diskSpaceLimitUnit` Cuando el tamaño total de los archivos de registro del sistema Greengrass supera este tamaño total máximo, el software AWS IoT Greengrass Core elimina los archivos de registro del sistema Greengrass más antiguos.

Este parámetro equivale al parámetro de límite de tamaño logarítmico (`totalLogsSizeKB`) del componente núcleo de Greengrass. El software AWS IoT Greengrass Core utiliza el mínimo de los dos valores como tamaño máximo total del registro del sistema Greengrass.

`diskSpaceLimitUnit`

(Opcional) La unidad para. `diskSpaceLimit` Puede elegir entre las siguientes opciones:

- `KB`— kilobytes
- `MB`— megabytes
- `GB`— gigabytes

Valor predeterminado: `KB`

`deleteLogFileAfterCloudUpload`

(Opcional) Puede eliminar un archivo de registro después de que el componente del administrador de registros cargue los registros en Logs. CloudWatch

Valor predeterminado: `false`

`componentLogsConfigurationMap`

(Opcional) Un mapa de las configuraciones de registro de los componentes del dispositivo principal. Cada `componentName` objeto de este mapa define la configuración de registro del componente o la aplicación. El componente del administrador de registros carga estos registros de componentes en CloudWatch Logs.

> **Important**
>
> Recomendamos encarecidamente utilizar una única clave de configuración por componente. Al usar el, solo debe dirigirse a un grupo de archivos que solo tengan un archivo de registro en el que se esté escribiendo activamente`logFileRegex`. Si no se sigue esta recomendación, es posible que se carguen registros duplicados en él CloudWatch. Si se dirige a varios archivos de registro activos con una sola expresión regular, le recomendamos que actualice a la versión 2.3.1 del administrador de registros y que considere la posibilidad de cambiar la configuración mediante la configuración de ejemplo.

> **Note**
>
> Si está actualizando desde una versión del administrador de registros anterior a la v2.2.0, puede seguir utilizando la lista en su lugar.

componentLogsConfiguration componentLogsConfigurationMap Sin embargo, le recomendamos encarecidamente que utilice el formato de mapa para poder utilizar las actualizaciones de combinación y restablecimiento para modificar las configuraciones de componentes específicos. Para obtener información sobre el componentLogsConfiguration parámetro, consulte los parámetros de configuración de la versión 2.1.x de este componente.

componentName

La configuración de registro del *componentName*componente o la aplicación de esta configuración de registro. Puede especificar el nombre de un componente de Greengrass u otro valor para identificar este grupo de registros.

Cada objeto contiene la siguiente información:

`minimumLogLevel`

(Opcional) El nivel mínimo de mensajes de registro que se deben cargar. Este nivel mínimo solo se aplica si los registros de este componente utilizan un formato JSON específico, que se encuentra en el repositorio del módulo de AWS IoT Greengrass registro GitHub.

Elija uno de los siguientes niveles de registro, que se muestran aquí en orden de niveles:

- `DEBUG`
- `INFO`
- `WARN`
- `ERROR`

Valor predeterminado: `INFO`

`diskSpaceLimit`

(Opcional) El tamaño total máximo de todos los archivos de registro de este componente, en la unidad en la que especifiquedisk`SpaceLimitUnit`. Cuando el tamaño total de los archivos de registro de este componente supere este tamaño total máximo, el software AWS IoT Greengrass Core elimina los archivos de registro más antiguos de este componente.

Este parámetro está relacionado con el parámetro límite de tamaño logarítmico (`totalLogsSizeKB`) del componente núcleo de Greengrass. El software AWS IoT Greengrass Core utiliza el mínimo de los dos valores como tamaño de registro total máximo para este componente.

`diskSpaceLimitUnit`

(Opcional) La unidad del`diskSpaceLimit`. Puede elegir entre las siguientes opciones:

- `KB`— kilobytes
- `MB`— megabytes
- `GB`— gigabytes

Valor predeterminado: `KB`

`logFileDirectoryPath`

(Opcional) La ruta a la carpeta que contiene los archivos de registro de este componente.

No es necesario especificar este parámetro para los componentes de Greengrass que imprimen en salida estándar (stdout) y error estándar (stderr).

Valor predeterminado: */greengrass/v2*`/logs`.

`logFileRegex`

(Opcional) Expresión regular que especifica el formato de nombre del archivo de registro que utiliza el componente o la aplicación. El componente gestor de registros utiliza esta expresión regular para identificar los archivos de registro de la carpeta situada en`logFileDirectoryPath`.

No es necesario especificar este parámetro para los componentes de Greengrass que imprimen en salida estándar (stdout) y error estándar (stderr).

Si su componente o aplicación rota los archivos de registro, especifique una expresión regular que coincida con los nombres de los archivos de registro rotados. Por ejemplo, puede especificar **`hello_world\\\\w*.log`** cargar los registros de una aplicación de Hello World. El `\\\\w*` patrón coincide con cero o más caracteres de palabra, lo que incluye caracteres alfanuméricos y guiones bajos.

Esta expresión regular hace coincidir los archivos de registro con y sin marcas de tiempo en su nombre. En este ejemplo, el administrador de registros carga los siguientes archivos de registro:

- `hello_world.log`— El archivo de registro más reciente de la aplicación Hello World.
- `hello_world_2020_12_15_17_0.log`— Un archivo de registro antiguo para la aplicación Hello World.

Predeterminado:*componentName*`\\\\w*.log`, donde *componentName* es el nombre del componente de esta configuración de registro.

`deleteLogFileAfterCloudUpload`

(Opcional) Puede eliminar un archivo de registro después de que el componente del administrador de registros cargue los registros en Logs. CloudWatch

Valor predeterminado: `false`

`multiLineStartPattern`

(Opcional) Expresión regular que identifica cuándo un mensaje de registro de una línea nueva es un mensaje de registro nuevo. Si la expresión regular no coincide con la nueva línea, el componente del administrador de registros anexa la nueva línea al mensaje de registro de la línea anterior.

De forma predeterminada, el componente del administrador de registros comprueba si la línea comienza con un carácter de espacio en blanco, como una pestaña o un espacio. Si no es así, el administrador de registros trata esa línea como un nuevo mensaje de registro. De lo contrario, añade esa línea al mensaje de registro actual. Este comportamiento garantiza que el componente del administrador de registros no divida los mensajes que ocupan varias líneas, como los seguimientos de pila.

`periodicUploadIntervalSec`

(Opcional) El período en segundos durante el que el componente del administrador de registros comprueba si hay nuevos archivos de registro para cargar.

Predeterminado: `300` (5 minutos)

Mínimo: `0.000001` (1 microsegundo)

Example Ejemplo: actualización de la combinación de configuraciones

El siguiente ejemplo de configuración especifica cargar los registros del sistema y los registros de los `com.example.HelloWorld` componentes en CloudWatch Logs.

```
{
  "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 Ejemplo: configuración para cargar varios archivos de registro activos mediante el administrador de registros v2.3.1

El siguiente ejemplo de configuración es el ejemplo recomendado si desea dirigirse a varios archivos de registro activos. En este ejemplo de configuración se especifican los archivos de registro activos en los que desea cargarlos CloudWatch. Si utiliza este ejemplo de configuración, la configuración también cargará todos los archivos rotados que coincidan con. `logFileRegex` Este ejemplo de configuración es compatible con la versión 2.3.1 del administrador de registros.

```
{
  "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`

(Opcional) La configuración de los registros que carga el componente del administrador de registros. Este objeto contiene la siguiente información:

`systemLogsConfiguration`

(Opcional) La configuración de los registros del sistema del software AWS IoT Greengrass principal, que incluyen los registros del núcleo y los componentes del complemento de Greengrass. Especifique esta configuración para permitir que el componente gestor de registros gestione los registros del sistema. Este objeto contiene la siguiente información:

`uploadToCloudWatch`

(Opcional) Puede cargar los registros del sistema en CloudWatch Logs.

Valor predeterminado: `false`

`minimumLogLevel`

(Opcional) El nivel mínimo de mensajes de registro que se deben cargar. Este nivel mínimo se aplica solo si configura el componente núcleo de Greengrass para generar registros en formato JSON. Para habilitar los registros en formato JSON, especifique `JSON` el parámetro de formato de registro (`logging.format`).

Elija uno de los siguientes niveles de registro, que se muestran aquí en orden de niveles:

- `DEBUG`
- `INFO`
- `WARN`
- `ERROR`

Valor predeterminado: `INFO`

`diskSpaceLimit`

(Opcional) El tamaño total máximo de los archivos de registro del sistema Greengrass, en la unidad en la que especifique. `diskSpaceLimitUnit` Cuando el tamaño total de los archivos de registro del sistema Greengrass supera este tamaño total máximo, el software AWS IoT Greengrass Core elimina los archivos de registro del sistema Greengrass más antiguos.

Este parámetro equivale al parámetro de límite de tamaño logarítmico (`totalLogsSizeKB`) del componente núcleo de Greengrass. El software AWS IoT Greengrass Core utiliza el mínimo de los dos valores como tamaño máximo total del registro del sistema Greengrass.

`diskSpaceLimitUnit`

(Opcional) La unidad para. `diskSpaceLimit` Puede elegir entre las siguientes opciones:

- `KB`— kilobytes
- `MB`— megabytes
- `GB`— gigabytes

Valor predeterminado: `KB`

`deleteLogFileAfterCloudUpload`

(Opcional) Puede eliminar un archivo de registro después de que el componente del administrador de registros cargue los registros en Logs. CloudWatch

Valor predeterminado: `false`

`componentLogsConfigurationMap`

(Opcional) Un mapa de las configuraciones de registro de los componentes del dispositivo principal. Cada `componentName` objeto de este mapa define la configuración de registro del componente o la aplicación. El componente del administrador de registros carga estos registros de componentes en CloudWatch Logs.

> **Note**
>
> Si está actualizando desde una versión del administrador de registros anterior a la 2.2.0, puede seguir utilizando la `componentLogsConfiguration`

lista en su lugar. `componentLogsConfigurationMap` Sin embargo, le recomendamos encarecidamente que utilice el formato de mapa para poder utilizar las actualizaciones de combinación y restablecimiento para modificar las configuraciones de componentes específicos. Para obtener información sobre el `componentLogsConfiguration` parámetro, consulte los parámetros de configuración de la versión 2.1.x de este componente.

`componentName`

La configuración de registro del *`componentName`*componente o la aplicación de esta configuración de registro. Puede especificar el nombre de un componente de Greengrass u otro valor para identificar este grupo de registros.

Cada objeto contiene la siguiente información:

`minimumLogLevel`

(Opcional) El nivel mínimo de mensajes de registro que se deben cargar. Este nivel mínimo solo se aplica si los registros de este componente utilizan un formato JSON específico, que se encuentra en el repositorio del módulo de AWS IoT Greengrass registro GitHub.

Elija uno de los siguientes niveles de registro, que se muestran aquí en orden de niveles:

- `DEBUG`
- `INFO`
- `WARN`
- `ERROR`

Valor predeterminado: `INFO`

`diskSpaceLimit`

(Opcional) El tamaño total máximo de todos los archivos de registro de este componente, en la unidad en la que especifiqued`iskSpaceLimitUnit`. Cuando el tamaño total de los archivos de registro de este componente supere este tamaño total máximo, el software AWS IoT Greengrass Core elimina los archivos de registro más antiguos de este componente.

Este parámetro está relacionado con el parámetro límite de tamaño logarítmico (`totalLogsSizeKB`) del componente núcleo de Greengrass. El software AWS IoT Greengrass Core utiliza el mínimo de los dos valores como tamaño de registro total máximo para este componente.

`diskSpaceLimitUnit`

(Opcional) La unidad del`diskSpaceLimit`. Puede elegir entre las siguientes opciones:

- `KB`— kilobytes
- `MB`— megabytes
- `GB`— gigabytes

Valor predeterminado: `KB`

`logFileDirectoryPath`

(Opcional) La ruta a la carpeta que contiene los archivos de registro de este componente.

No es necesario especificar este parámetro para los componentes de Greengrass que imprimen en salida estándar (stdout) y error estándar (stderr).

Valor predeterminado: */greengrass/v2*`/logs`.

`logFileRegex`

(Opcional) Expresión regular que especifica el formato de nombre del archivo de registro que utiliza el componente o la aplicación. El componente gestor de registros utiliza esta expresión regular para identificar los archivos de registro de la carpeta situada en`logFileDirectoryPath`.

No es necesario especificar este parámetro para los componentes de Greengrass que imprimen en salida estándar (stdout) y error estándar (stderr).

Si su componente o aplicación rota los archivos de registro, especifique una expresión regular que coincida con los nombres de los archivos de registro rotados. Por ejemplo, puede especificar **`hello_world\\\\w*.log`** cargar los registros de una aplicación de Hello World. El `\\\\w*` patrón coincide con cero o más caracteres de palabra, lo que incluye caracteres alfanuméricos y guiones bajos.

Esta expresión regular hace coincidir los archivos de registro con y sin marcas de tiempo en su nombre. En este ejemplo, el administrador de registros carga los siguientes archivos de registro:

- `hello_world.log`— El archivo de registro más reciente de la aplicación Hello World.
- `hello_world_2020_12_15_17_0.log`— Un archivo de registro antiguo para la aplicación Hello World.

Predeterminado:*componentName*`\\\\w*.log`, donde *componentName* es el nombre del componente de esta configuración de registro.

`deleteLogFileAfterCloudUpload`

(Opcional) Puede eliminar un archivo de registro después de que el componente del administrador de registros cargue los registros en Logs. CloudWatch

Valor predeterminado: `false`

`multiLineStartPattern`

(Opcional) Expresión regular que identifica cuándo un mensaje de registro de una línea nueva es un mensaje de registro nuevo. Si la expresión regular no coincide con la nueva línea, el componente del administrador de registros anexa la nueva línea al mensaje de registro de la línea anterior.

De forma predeterminada, el componente del administrador de registros comprueba si la línea comienza con un carácter de espacio en blanco, como una pestaña o un espacio. Si no es así, el administrador de registros trata esa línea como un nuevo mensaje de registro. De lo contrario, añade esa línea al mensaje de registro actual. Este comportamiento garantiza que el componente del administrador de registros no divida los mensajes que ocupan varias líneas, como los seguimientos de pila.

`periodicUploadIntervalSec`

(Opcional) El período en segundos durante el que el componente del administrador de registros comprueba si hay nuevos archivos de registro para cargar.

Predeterminado: `300` (5 minutos)

Mínimo: `0.000001` (1 microsegundo)

Example Ejemplo: actualización de la combinación de configuraciones

El siguiente ejemplo de configuración especifica cargar los registros del sistema y los registros de los `com.example.HelloWorld` componentes en CloudWatch Logs.

```
{
  "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`

(Opcional) La configuración de los registros que carga el componente del administrador de registros. Este objeto contiene la siguiente información:

`systemLogsConfiguration`

(Opcional) La configuración de los registros del sistema del software AWS IoT Greengrass principal, que incluyen los registros del núcleo y los componentes del complemento de Greengrass. Especifique esta configuración para permitir que el componente gestor de registros gestione los registros del sistema. Este objeto contiene la siguiente información:

`uploadToCloudWatch`

(Opcional) Puede cargar los registros del sistema en CloudWatch Logs.

Valor predeterminado: `false`

`minimumLogLevel`

(Opcional) El nivel mínimo de mensajes de registro que se deben cargar. Este nivel mínimo se aplica solo si configura el componente núcleo de Greengrass para generar registros en formato JSON. Para habilitar los registros en formato JSON, especifique `JSON` el parámetro de formato de registro (`logging.format`).

Elija uno de los siguientes niveles de registro, que se muestran aquí en orden de niveles:

- `DEBUG`
- `INFO`
- `WARN`
- `ERROR`

Valor predeterminado: `INFO`

`diskSpaceLimit`

(Opcional) El tamaño total máximo de los archivos de registro del sistema Greengrass, en la unidad en la que especifique. `diskSpaceLimitUnit` Cuando el tamaño total de los archivos de registro del sistema Greengrass supera este tamaño total máximo, el software AWS IoT Greengrass Core elimina los archivos de registro del sistema Greengrass más antiguos.

Este parámetro equivale al parámetro de límite de tamaño logarítmico (`totalLogsSize`KB) del componente núcleo de Greengrass. El software AWS IoT Greengrass Core utiliza el mínimo de los dos valores como tamaño máximo total del registro del sistema Greengrass.

`diskSpaceLimitUnit`

(Opcional) La unidad para. `diskSpaceLimit` Puede elegir entre las siguientes opciones:

- `KB`— kilobytes
- `MB`— megabytes
- `GB`— gigabytes

Valor predeterminado: `KB`

`deleteLogFileAfterCloudUpload`

(Opcional) Puede eliminar un archivo de registro después de que el componente del administrador de registros cargue los registros en Logs. CloudWatch

Valor predeterminado: `false`

`componentLogsConfiguration`

(Opcional) Una lista de las configuraciones de registro de los componentes del dispositivo principal. Cada configuración de esta lista define la configuración de registro de un componente o aplicación. El componente del administrador de registros carga estos registros de componentes en Logs CloudWatch

Cada objeto contiene la siguiente información:

`componentName`

El nombre del componente o la aplicación de esta configuración de registro. Puede especificar el nombre de un componente de Greengrass u otro valor para identificar este grupo de registros.

`minimumLogLevel`

(Opcional) El nivel mínimo de mensajes de registro que se deben cargar. Este nivel mínimo solo se aplica si los registros de este componente utilizan un formato JSON específico, que se encuentra en el repositorio del módulo de AWS IoT Greengrass registro GitHub.

Elija uno de los siguientes niveles de registro, que se muestran aquí en orden de niveles:

- `DEBUG`
- `INFO`
- `WARN`
- `ERROR`

Valor predeterminado: `INFO`

`diskSpaceLimit`

(Opcional) El tamaño total máximo de todos los archivos de registro de este componente, en la unidad en la que especifique`diskSpaceLimitUnit`. Cuando el

tamaño total de los archivos de registro de este componente supere este tamaño total máximo, el software AWS IoT Greengrass Core elimina los archivos de registro más antiguos de este componente.

Este parámetro está relacionado con el parámetro límite de tamaño logarítmico (`totalLogsSizeKB`) del componente núcleo de Greengrass. El software AWS IoT Greengrass Core utiliza el mínimo de los dos valores como tamaño de registro total máximo para este componente.

`diskSpaceLimitUnit`

(Opcional) La unidad del`diskSpaceLimit`. Puede elegir entre las siguientes opciones:

- `KB`— kilobytes
- `MB`— megabytes
- `GB`— gigabytes

Valor predeterminado: `KB`

`logFileDirectoryPath`

(Opcional) La ruta a la carpeta que contiene los archivos de registro de este componente.

No es necesario especificar este parámetro para los componentes de Greengrass que imprimen en salida estándar (stdout) y error estándar (stderr).

Valor predeterminado: *`/greengrass/v2`*`/logs`.

`logFileRegex`

(Opcional) Expresión regular que especifica el formato de nombre del archivo de registro que utiliza el componente o la aplicación. El componente gestor de registros utiliza esta expresión regular para identificar los archivos de registro de la carpeta situada en`logFileDirectoryPath`.

No es necesario especificar este parámetro para los componentes de Greengrass que imprimen en salida estándar (stdout) y error estándar (stderr).

Si su componente o aplicación rota los archivos de registro, especifique una expresión regular que coincida con los nombres de los archivos de registro rotados. Por ejemplo, puede especificar **`hello_world\\\\w*.log`** cargar los registros de una aplicación de

Hello World. El `\\\\w*` patrón coincide con cero o más caracteres de palabra, lo que incluye caracteres alfanuméricos y guiones bajos. Esta expresión regular hace coincidir los archivos de registro con y sin marcas de tiempo en su nombre. En este ejemplo, el administrador de registros carga los siguientes archivos de registro:

- `hello_world.log`— El archivo de registro más reciente de la aplicación Hello World.
- `hello_world_2020_12_15_17_0.log`— Un archivo de registro antiguo para la aplicación Hello World.

Predeterminado:*componentName*`\\\\w*.log`, donde *componentName* es el nombre del componente de esta configuración de registro.

`deleteLogFileAfterCloudUpload`

(Opcional) Puede eliminar un archivo de registro después de que el componente del administrador de registros cargue los registros en Logs. CloudWatch

Valor predeterminado: `false`

`multiLineStartPattern`

(Opcional) Expresión regular que identifica cuándo un mensaje de registro de una línea nueva es un mensaje de registro nuevo. Si la expresión regular no coincide con la nueva línea, el componente del administrador de registros anexa la nueva línea al mensaje de registro de la línea anterior.

De forma predeterminada, el componente del administrador de registros comprueba si la línea comienza con un carácter de espacio en blanco, como una pestaña o un espacio. Si no es así, el administrador de registros trata esa línea como un nuevo mensaje de registro. De lo contrario, añade esa línea al mensaje de registro actual. Este comportamiento garantiza que el componente del administrador de registros no divida los mensajes que ocupan varias líneas, como los seguimientos de pila.

`periodicUploadIntervalSec`

(Opcional) El período en segundos durante el que el componente del administrador de registros comprueba si hay nuevos archivos de registro para cargar.

Predeterminado: `300` (5 minutos)

Mínimo: `0.000001` (1 microsegundo)

Example Ejemplo: actualización de la combinación de configuraciones

El siguiente ejemplo de configuración especifica cargar los registros del sistema y los registros de los `com.example.HelloWorld` componentes en CloudWatch Logs.

```
{
  "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`

(Opcional) La configuración de los registros que carga el componente del administrador de registros. Este objeto contiene la siguiente información:

`systemLogsConfiguration`

(Opcional) La configuración de los registros del sistema de software AWS IoT Greengrass principal. Especifique esta configuración para permitir que el componente gestor de registros gestione los registros del sistema. Este objeto contiene la siguiente información:

`uploadToCloudWatch`

(Opcional) Puede cargar los registros del sistema en CloudWatch Logs.

Valor predeterminado: `false`

`minimumLogLevel`

(Opcional) El nivel mínimo de mensajes de registro que se deben cargar. Este nivel mínimo se aplica solo si configura el componente núcleo de Greengrass para generar registros en formato JSON. Para habilitar los registros en formato JSON, especifique `JSON` el parámetro de formato de registro (`logging.format`).

Elija uno de los siguientes niveles de registro, que se muestran aquí en orden de niveles:

- `DEBUG`
- `INFO`
- `WARN`
- `ERROR`

Valor predeterminado: `INFO`

`diskSpaceLimit`

(Opcional) El tamaño total máximo de los archivos de registro del sistema Greengrass, en la unidad en la que especifique. `diskSpaceLimitUnit` Cuando el tamaño total de los archivos de registro del sistema Greengrass supera este tamaño total máximo, el software AWS IoT Greengrass Core elimina los archivos de registro del sistema Greengrass más antiguos.

Este parámetro equivale al parámetro de límite de tamaño logarítmico (`totalLogsSize`KB) del componente núcleo de Greengrass. El software AWS IoT Greengrass Core utiliza el mínimo de los dos valores como tamaño máximo total del registro del sistema Greengrass.

`diskSpaceLimitUnit`

(Opcional) La unidad para. `diskSpaceLimit` Puede elegir entre las siguientes opciones:

- `KB`— kilobytes
- `MB`— megabytes
- `GB`— gigabytes

Valor predeterminado: `KB`

`deleteLogFileAfterCloudUpload`

(Opcional) Puede eliminar un archivo de registro después de que el componente del administrador de registros cargue los registros en Logs. CloudWatch

Valor predeterminado: `false`

`componentLogsConfiguration`

(Opcional) Una lista de las configuraciones de registro de los componentes del dispositivo principal. Cada configuración de esta lista define la configuración de registro de un componente o aplicación. El componente del administrador de registros carga estos registros de componentes en Logs CloudWatch

Cada objeto contiene la siguiente información:

`componentName`

El nombre del componente o la aplicación de esta configuración de registro. Puede especificar el nombre de un componente de Greengrass u otro valor para identificar este grupo de registros.

`minimumLogLevel`

(Opcional) El nivel mínimo de mensajes de registro que se deben cargar. Este nivel mínimo solo se aplica si los registros de este componente utilizan un formato JSON específico, que se encuentra en el repositorio del módulo de AWS IoT Greengrass registro GitHub.

Elija uno de los siguientes niveles de registro, que se muestran aquí en orden de niveles:

- `DEBUG`
- `INFO`
- `WARN`
- `ERROR`

Valor predeterminado: `INFO`

`diskSpaceLimit`

(Opcional) El tamaño total máximo de todos los archivos de registro de este componente, en la unidad en la que especifique`diskSpaceLimitUnit`. Cuando el

tamaño total de los archivos de registro de este componente supere este tamaño total máximo, el software AWS IoT Greengrass Core elimina los archivos de registro más antiguos de este componente.

Este parámetro está relacionado con el parámetro límite de tamaño logarítmico (`totalLogsSizeKB`) del componente núcleo de Greengrass. El software AWS IoT Greengrass Core utiliza el mínimo de los dos valores como tamaño de registro total máximo para este componente.

`diskSpaceLimitUnit`

(Opcional) La unidad del`diskSpaceLimit`. Puede elegir entre las siguientes opciones:

- `KB`— kilobytes
- `MB`— megabytes
- `GB`— gigabytes

Valor predeterminado: `KB`

`logFileDirectoryPath`

La ruta a la carpeta que contiene los archivos de registro de este componente.

Para cargar los registros de un componente de Greengrass***/greengrass/v2*****/logs**, especifique y reemplácelos por su carpeta */greengrass/v2* raíz de Greengrass.

`logFileRegex`

Expresión regular que especifica el formato de nombre del archivo de registro que utiliza el componente o la aplicación. El componente gestor de registros utiliza esta expresión regular para identificar los archivos de registro de la carpeta en la que se encuentra`logFileDirectoryPath`.

Para cargar los registros de un componente de Greengrass, especifique una expresión regular que coincida con los nombres de los archivos de registro rotados. Por ejemplo, puede especificar **`com.example.HelloWorld\\w*.log`** cargar los registros de un componente de Hello World. El `\\w*` patrón coincide con cero o más caracteres de palabra, lo que incluye caracteres alfanuméricos y guiones bajos. Esta expresión regular hace coincidir los archivos de registro con y sin marcas de tiempo en su nombre. En este ejemplo, el administrador de registros carga los siguientes archivos de registro:

- `com.example.HelloWorld.log`— El archivo de registro más reciente del componente Hello World.
- `com.example.HelloWorld_2020_12_15_17_0.log`— Un archivo de registro antiguo para el componente Hello World. El núcleo de Greengrass añade una marca de tiempo rotativa a los archivos de registro.

`deleteLogFileAfterCloudUpload`

(Opcional) Puede eliminar un archivo de registro después de que el componente del administrador de registros cargue los registros en Logs. CloudWatch

Valor predeterminado: `false`

`multiLineStartPattern`

(Opcional) Expresión regular que identifica cuándo un mensaje de registro de una línea nueva es un mensaje de registro nuevo. Si la expresión regular no coincide con la nueva línea, el componente del administrador de registros anexa la nueva línea al mensaje de registro de la línea anterior.

De forma predeterminada, el componente del administrador de registros comprueba si la línea comienza con un carácter de espacio en blanco, como una pestaña o un espacio. Si no es así, el administrador de registros trata esa línea como un nuevo mensaje de registro. De lo contrario, añade esa línea al mensaje de registro actual. Este comportamiento garantiza que el componente del administrador de registros no divida los mensajes que ocupan varias líneas, como los seguimientos de pila.

`periodicUploadIntervalSec`

(Opcional) El período en segundos durante el que el componente del administrador de registros comprueba si hay nuevos archivos de registro para cargar.

Predeterminado: `300` (5 minutos)

Mínimo: `0.000001` (1 microsegundo)

Example Ejemplo: actualización de la combinación de configuraciones

El siguiente ejemplo de configuración especifica cargar los registros del sistema y los registros de los `com.example.HelloWorld` componentes en CloudWatch Logs.

```
{
  "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"
}
```

Uso

El componente del administrador de registros se carga en los siguientes grupos de registros y flujos de registros.

2.1.0 and later

Nombre del grupo de registros

```
/aws/greengrass/componentType/region/componentName
```

El nombre del grupo de registros utiliza las siguientes variables:

- `componentType`— El tipo de componente, que puede ser uno de los siguientes:
  - `GreengrassSystemComponent`— Este grupo de registros incluye los registros de los componentes del núcleo y del complemento, que se ejecutan en la misma JVM que el núcleo de Greengrass. El componente forma parte del núcleo de Greengrass.

- `UserComponent`— Este grupo de registros incluye registros de componentes genéricos, componentes de Lambda y otras aplicaciones del dispositivo. El componente no forma parte del núcleo de Greengrass.

Para obtener más información, consulte Tipos de componentes.

- `region`— La AWS región que utiliza el dispositivo principal.
- `componentName`— El nombre del componente. Para los registros del sistema, este valor es`System`.

Nombre del flujo de registro

```
/date/thing/thingName
```

El nombre del flujo de registro utiliza las siguientes variables:

- `date`— La fecha del registro, por ejemplo`2020/12/15`. El componente del administrador de registros usa el `yyyy/MM/dd` formato.
- `thingName`— El nombre del dispositivo principal.

> **Note**
>
> Si el nombre de un elemento contiene dos puntos (`:`), el administrador de registros los sustituye por un signo más (`+`).

2.0.x

Nombre del grupo de registros

```
/aws/greengrass/componentType/region/componentName
```

El nombre del grupo de registros utiliza las siguientes variables:

- `componentType`— El tipo de componente, que puede ser uno de los siguientes:
  - `GreengrassSystemComponent`— El componente forma parte del núcleo de Greengrass.
  - `UserComponent`— El componente no forma parte del núcleo de Greengrass. El administrador de registros usa este tipo para los componentes de Greengrass y otras aplicaciones del dispositivo.

- `region`— La AWS región que utiliza el dispositivo principal.
- `componentName`— El nombre del componente. Para los registros del sistema, este valor es`System`.

Nombre del flujo de registro

```
/date/deploymentTargets/thingName
```

El nombre del flujo de registro utiliza las siguientes variables:

- `date`— La fecha del registro, por ejemplo`2020/12/15`. El componente del administrador de registros usa el `yyyy/MM/dd` formato.
- `deploymentTargets`— Las cosas cuyas implementaciones incluyen el componente. El componente del administrador de registros separa cada objetivo mediante una barra oblicua. Si el componente se ejecuta en el dispositivo principal como resultado de una implementación local, este valor es`LOCAL_DEPLOYMENT`.

  Considere un ejemplo en el que tiene un nombre `MyGreengrassCore` para un dispositivo principal y el dispositivo principal tiene dos implementaciones:

  - Una implementación dirigida al dispositivo principal,`MyGreengrassCore`.
  - Una implementación que se dirige a un grupo de cosas denominado`MyGreengrassCoreGroup`, que contiene el dispositivo principal.

  Los `deploymentTargets` de este dispositivo principal son`thing/MyGreengrassCore/thinggroup/MyGreengrassCoreGroup`.
- `thingName`— El nombre del dispositivo principal.

Formatos de entradas de registro.

El núcleo de Greengrass escribe los archivos de registro en formato de cadena o JSON. En el caso de los registros del sistema, usted controla el formato configurando el `format` campo de la `logging` entrada. Puede encontrar la `logging` entrada en el archivo de configuración del componente núcleo de Greengrass. Para obtener más información, consulte Configuración del núcleo de Greengrass.

El formato de texto es de formato libre y acepta cualquier cadena. El siguiente mensaje del servicio de estado de la flota es un ejemplo de registro con formato de cadena:

```
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}
```

Debe usar el formato JSON si quiere ver los registros con el comando logs de la CLI de Greengrass o interactuar con los registros mediante programación. El siguiente ejemplo describe la forma JSON:

```
{
   "loggerName": <string>,
   "level": <"DEBUG" | "INFO" | "ERROR" | "TRACE" | "WARN">,
   "eventType": <string, optional>,
   "cause": <string, optional>,
   "contexts": {},
   "thread": <string>,
   "message": <string>,
   "timestamp": <epoch time> # Needs to be epoch time
}
```

Para controlar la salida de los registros de su componente, puede usar la opción `minimumLogLevel` de configuración. Para usar esta opción, el componente debe escribir sus entradas de registro en formato JSON. Debe usar el mismo formato que el archivo de registro del sistema.

Archivo de registro local

Este componente utiliza el mismo archivo de registro que el componente núcleo de Greengrass.

Linux

```
/greengrass/v2/logs/greengrass.log
```

Windows

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

Para ver los registros de este componente

- Ejecute el siguiente comando en el dispositivo principal para ver el archivo de registro de este componente en tiempo real. Sustituya */greengrass/v2* o *C:\greengrass\v2* por la ruta a la carpeta AWS IoT Greengrass raíz.

Linux

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

Windows (PowerShell)

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

Registros de cambios

En la siguiente tabla se describen los cambios en cada versión del componente.

Versión	Cambios
2.3.7	Versión actualizada para la versión 2.12.0 de Greengrass nucleus.
2.3.6	Mejoras y correcciones de errores • Ajusta los niveles de registro para detectar determinados errores.
2.3.5	Mejoras Mejora la velocidad de carga de los registros. Versión actualizada para la versión 2.11.0 de Greengrass nucleus.
2.3.4	Mejoras y correcciones de errores • Añade soporte para configurar el `periodicUploadIntervalSec` parámetro en valores fraccionarios. El mínimo es 1 microsegundo. • Soluciona un problema por el que el administrador de registros no respetaba los CloudWatch `putLogEvents` límites.
2.3.3	Versión actualizada para la versión 2.10.0 de Greengrass nucleus.
2.3.2	Mejoras y correcciones de errores • Mejora la gestión del espacio para que los archivos de registro no se eliminen antes de cargarlos.

Versión	Cambios
	• Soluciona problemas con la administración de la memoria caché. • Mejoras y correcciones de errores menores adicionales.
2.3.1	Mejoras y correcciones de errores • Soluciona un problema por el que los grupos de archivos de destino con varios archivos de registro activos cargaban entradas duplicadas en ellas. CloudWatch • Mejoras y correcciones de errores menores adicionales.
2.3.0	**Note** Le recomendamos que actualice a Greengrass nucleus 2.9.1 cuando actualice a log manager 2.3.0. Nuevas características Reduce las demoras en el registro al procesar y cargar directamente los archivos de registro activos en lugar de esperar a que se roten los archivos nuevos. Mejoras y correcciones de errores • Mejora la compatibilidad con la rotación de registros al rotar archivos con un nombre único. • Mejoras y correcciones de errores menores adicionales.
2.2.8	Versión actualizada para la versión 2.9.0 de Greengrass Nucleus.
2.2.7	Versión actualizada para el lanzamiento de la versión 2.8.0 de Greengrass nucleus.
2.2.6	Versión actualizada para el lanzamiento de la versión 2.7.0 de Greengrass nucleus.
2.2.5	Versión actualizada para la versión 2.6.0 de Greengrass Nucleus.

Versión	Cambios
2.2.4	Mejoras y correcciones de errores • Mejora la estabilidad al gestionar configuraciones no válidas. • Mejoras y correcciones menores adicionales.
2.2.3	Mejoras y correcciones de errores • Mejora la estabilidad en ciertos escenarios en los que el componente se reinicia o encuentra errores. • Soluciona problemas por los que los mensajes de registro y los archivos de registro de gran tamaño no se cargaban en determinadas situacion es. • Soluciona problemas relacionados con la forma en que este component e gestiona las actualizaciones de restablecimiento de la configuración. • Soluciona un problema por el que un valor de `null` `diskSpaceLimit` configuración impedía la implementación del componente.
2.2.2	Mejoras y correcciones de errores • Añade compatibilidad con los mensajes de registro de más de 256 kilobytes. El componente del administrador de registros divide estos mensajes de registro de gran tamaño en varios mensajes con la misma marca temporal de eventos de registro.
2.2.1	Versión actualizada para el lanzamiento de la versión 2.5.0 de Greengrass nucleus.
2.2.0	Nueva característica • Agrega el parámetro `componentLogsConfigurationMap` de configuración para admitir un formato de mapa para las configuraciones de registro de componentes. Cada `componentName` objeto del mapa define la configuración de registro de un componente o aplicación.
2.1.3	Versión actualizada para la versión 2.4.0 de Greengrass Nucleus.
2.1.2	Versión actualizada para el lanzamiento de la versión 2.3.0 de Greengrass nucleus.

Versión	Cambios
2.1.1	Mejoras y correcciones de errores • Soluciona un problema por el que la configuración del registro del sistema no se actualizaba en algunos casos.
2.1.0	Mejoras y correcciones de errores • Utilice valores predeterminados para `logFileDirectoryPath` y `logFileRegex` que funcionen para los componentes de Greengrass que imprimen con salida estándar (stdout) y error estándar (stderr). • Dirija correctamente el tráfico a través de un proxy de red configurado al cargar registros en Logs. CloudWatch • Maneje correctamente los dos puntos (`:`) en los nombres de los flujos de registro. CloudWatch Los nombres de los flujos de registro de registros no admiten signos de dos puntos. • Simplifique los nombres de los flujos de registro eliminando los nombres de los grupos de cosas del flujo de registro. • Elimine un mensaje de registro de errores que se imprime con un comportamiento normal.
2.0.x	Versión inicial.

Componentes de aprendizaje automático

AWS IoT Greengrassproporciona los siguientes componentes de aprendizaje automático que puede implementar en dispositivos compatibles para realizar inferencias de aprendizaje automático mediante modelos entrenados en Amazon SageMaker o con sus propios modelos previamente entrenados que se almacenan en Amazon S3.

AWSproporciona las siguientes categorías de componentes de aprendizaje automático:

- Componente de modelo: contiene modelos de aprendizaje automático como artefactos de Greengrass.
- Componente de tiempo de ejecución: contiene el script que instala el marco de aprendizaje automático y sus dependencias en el dispositivo principal de Greengrass.

- Componente de inferencia: contiene el código de inferencia e incluye las dependencias de los componentes para instalar el marco de aprendizaje automático y descargar modelos de aprendizaje automático previamente entrenados.

Puede utilizar el código de inferencia de muestra y los modelos previamente entrenados de los componentes de aprendizaje automático AWS proporcionados para clasificar imágenes y detectar objetos mediante DLR y Lite. TensorFlow Para realizar inferencias de aprendizaje automático personalizadas con sus propios modelos almacenados en Amazon S3, o para utilizar un marco de aprendizaje automático diferente, puede utilizar las recetas de estos componentes públicos como plantillas para crear componentes de aprendizaje automático personalizados. Para obtener más información, consulte Personalice sus componentes de aprendizaje automático.

AWS IoT Greengrasstambién incluye un componente AWS proporcionado para gestionar la instalación y el ciclo de vida del agente SageMaker Edge Manager en los dispositivos principales de Greengrass. Con SageMaker Edge Manager, puede utilizar los modelos SageMaker compilados por Amazon NEO directamente en su dispositivo principal. Para obtener más información, consulte Utilice Amazon SageMaker Edge Manager en los dispositivos principales de Greengrass.

En la siguiente tabla se enumeran los componentes de aprendizaje automático que están disponibles enAWS IoT Greengrass.

> **Note**
>
> Varios AWS de los componentes proporcionados dependen de versiones secundarias específicas del núcleo de Greengrass. Debido a esta dependencia, es necesario actualizar estos componentes al actualizar el núcleo de Greengrass a una nueva versión secundaria. Para obtener información sobre las versiones específicas del núcleo de las que depende cada componente, consulte el tema del componente correspondiente. Para obtener más información sobre la actualización del núcleo, consulteActualice el software AWS IoT Greengrass principal (OTA).

Cuando un componente tiene un tipo de componente genérico y Lambda, la versión actual del componente es del tipo genérico y la versión anterior del componente es del tipo Lambda.

Componente	Descripción	Tipo de componente	Sistema operativo admitido	Código abierto
Lookout for Vision Edge Agent	Implementa el motor de ejecución Amazon Lookout for Vision en el dispositivo principal de Greengrass, para que pueda utilizar la visión artificial para detectar defectos en productos industriales.	Genérico	Linux	No
SageMaker Administrador perimetral	Implementa el agente Amazon SageMaker Edge Manager en el dispositivo principal de Greengrass.	Genérico	Linux, Windows	No
Clasificación de imágenes DLR	Componente de inferencia que utiliza el almacén de modelos de clasificación	Genérico	Linux, Windows	No

Componente	Descripción	Tipo de componente	Sistema operativo admitido	Código abierto
	de imágenes del DLR y el componente de tiempo de ejecución del DLR como dependencias para instalar el DLR, descargar modelos de clasificación de imágenes de muestra y realizar inferencias de clasificación de imágenes en los dispositivos compatibles.			

Componente	Descripción	Tipo de componente	Sistema operativo admitido	Código abierto
Detección de objetos DLR	Componente de inferencia que utiliza el almacén de modelos de detección de objetos del DLR y el componente de tiempo de ejecución del DLR como dependencias para instalar el DLR, descargar modelos de detección de objetos de muestra y realizar inferencias de detección de objetos en los dispositivos compatibles.	Genérico	Linux, Windows	No

Componente	Descripción	Tipo de componente	Sistema operativo admitido	Código abierto
Tienda de modelos de clasificación de imágenes DLR	Componente de modelo que contiene ejemplos de ResNet -50 modelos de clasificación de imágenes como artefactos de Greengrass.	Genérico	Linux, Windows	No
Tienda de modelos de detección de objetos DLR	Componente de modelo que contiene ejemplos de modelos de detección de objetos de YoloV3 como artefactos de Greengrass.	Genérico	Linux, Windows	No

Componente	Descripción	Tipo de componente	Sistema operativo admitido	Código abierto
Tiempo de ejecución de DLR	Componente de tiempo de ejecución que contiene un script de instalación que se utiliza para instalar el DLR y sus dependencias en el dispositivo principal de Greengrass.	Genérico	Linux, Windows	No

Componente	Descripción	Tipo de componente	Sistema operativo admitido	Código abierto
TensorFlow Clasificación de imágenes Lite	Componente de inferencia que utiliza el TensorFlow almacén de modelos de clasificación de imágenes de TensorFlow Lite y el componente de tiempo de ejecución de Lite como dependencias para instalar TensorFlow Lite, descargar modelos de clasificación de imágenes de muestra y realizar inferencias de clasificación de imágenes en dispositivos compatibles.	Genérico	Linux, Windows	No

Componente	Descripción	Tipo de componente	Sistema operativo admitido	Código abierto
TensorFlow Detección de objetos Lite	Componente de inferencia que utiliza el TensorFlow almacén de modelos de detección de objetos de TensorFlow Lite y el componente de tiempo de ejecución de Lite como dependencias para instalar TensorFlow Lite, descargar modelos de detección de objetos de muestra y realizar inferencias de detección de objetos en dispositivos compatibles.	Genérico	Linux, Windows	No

Componente	Descripción	Tipo de componente	Sistema operativo admitido	Código abierto
TensorFlow Tienda de modelos de clasificación de imágenes Lite	Componente de modelo que contiene un modelo MobileNet v1 de muestra como artefacto de Greengrass.	Genérico	Linux, Windows	No
TensorFlow Tienda de modelos de detección de objetos Lite	Componente de modelo que contiene un MobileNet modelo de muestra de detección de disparo único (SSD) como un artefacto de Greengrass.	Genérico	Linux, Windows	No

Componente	Descripción	Tipo de componente	Sistema operativo admitido	Código abierto
TensorFlow Tiempo de ejecución Lite	Componente de tiempo de ejecución que contiene un script de instalación que se utiliza para instalar TensorFlow Lite y sus dependencias en el dispositivo principal de Greengrass.	Genérico	Linux, Windows	No

Lookout for Vision Edge Agent

El componente Lookout for Vision Edge Agent `aws.iot.lookoutvision.EdgeAgent` () instala un servidor de tiempo de ejecución local de Amazon Lookout for Vision, que utiliza la visión artificial para detectar defectos visuales en productos industriales.

Para usar este componente, cree e implemente los componentes del modelo de aprendizaje automático de Lookout for Vision. Estos modelos de aprendizaje automático predicen la presencia de anomalías en las imágenes mediante la búsqueda de patrones en las imágenes que se utilizan para entrenar el modelo. A continuación, puede desarrollar e implementar componentes personalizados de Greengrass, denominados componentes de aplicaciones cliente, que proporcionan imágenes y secuencias de vídeo a este componente de tiempo de ejecución para detectar anomalías mediante los modelos de aprendizaje automático.

Puede utilizar la API de agente de Lookout for Vision Edge para interactuar con este componente desde otros componentes de Greengrass. Esta API se implementa mediante gRPC, que es un protocolo para realizar llamadas a procedimientos remotos. Para obtener más información, consulte

Cómo escribir un componente de aplicación cliente y la referencia sobre la API de agente de Lookout for Vision Edge en la guía para desarrolladores de Amazon Lookout for Vision.

Para obtener más información sobre cómo utilizar este componente, consulte lo siguiente:

- Amazon Lookout Lookout for Vision out out out Greengrass out out out out
- ¿Qué es Amazon Lookout for Vision? en la guía para desarrolladores de Amazon Lookout for Vision
- Creación de un modelo de Lookout for Vision en la guía para desarrolladores de Amazon Lookout for Vision.
- Uso de un modelo de Lookout for Vision en un dispositivo periférico en la guía para desarrolladores de Amazon Lookout for Vision.

> **Note**
>
> El componente Lookout for Vision Edge Agent solo está disponible en las Regiones de AWS siguientes ubicaciones:
>
> - US East (Ohio)
> - Este de EE. UU. (Norte de Virginia)
> - Oeste de EE. UU. (Oregón)
> - Europa (Fráncfort)
> - Europa (Irlanda)
> - Asia-Pacífico (Tokio)
> - Asia-Pacífico (Seúl)

Temas

- Versiones
- Tipo
- Sistema operativo
- Requisitos
- Dependencias
- Configuración

- Archivo de registro local
- Registros de cambios

Versiones

Este componente tiene las siguientes versiones:

- 1.2.x
- 1.1.x
- 1.0.x
- 0.1.x

Tipo

Este componente es un componente genérico () `aws.greengrass.generic`. El núcleo de Greengrass ejecuta los scripts del ciclo de vida del componente.

Para obtener más información, consulte Tipos de componentes.

Sistema operativo

Este componente solo se puede instalar en los dispositivos principales de Linux.

Requisitos

Este componente tiene los siguientes requisitos:

- El dispositivo principal de Greengrass debe utilizar una arquitectura Armv8 (AArch64) o x86_64.
- Si usa la versión 1.0.0 o posterior de este componente, Python 3.8 o Python 3.9, incluidos`pip`, instalados en el dispositivo principal de Greengrass.

  Si utiliza la versión 0.1.x de este componente, Python 3.7, incluida`pip`, está instalada en el dispositivo principal de Greengrass.

  > **Important**
  >
  > El dispositivo debe tener una de estas versiones exactas de Python. Este componente no es compatible con las versiones posteriores de Python.

- Para utilizar la inferencia de la unidad de procesamiento de gráficos (GPU), el dispositivo principal debe cumplir los siguientes requisitos. La inferencia mediante GPU es opcional en la versión 1.1.0 y versiones posteriores de este componente.
  - Una unidad de procesamiento de gráficos (GPU) compatible con CUDA. Para obtener más información, consulte Verificar que tiene una GPU compatible con CUDA en la documentación del kit de herramientas CUDA.
  - CuDNN, CUDA y TensorRT instalados en el dispositivo principal de Greengrass.
    - En los dispositivos NVIDIA Jetson, como Jetson Nano o Jetson Xavier, cuDNN, CUDA y TensorRT vienen instalados con NVIDIA. JetPack No es necesario realizar ningún cambio. Este componente es compatible con las versiones JetPack 4.4, JetPack4.5, JetPack 4.5.1 y JetPack4.6.1.

      > **Important**
      >
      > Debe instalar una de estas versiones JetPack y no otra. El servicio Lookout for Vision recopila modelos de visión artificial para JetPack estas plataformas.

    - En los dispositivos x86 con una GPU que tenga la microarquitectura NVIDIA Ampere (o la capacidad de procesamiento de la GPU sea de 8.0), haga lo siguiente:
      - Instale cuDNN siguiendo las instrucciones de la Guía de instalación de cuDNN de NVIDIA.
      - Instale la versión 11.2 de CUDA siguiendo las instrucciones de la Guía de instalación de NVIDIA CUDA para Linux.
      - Instala TensorRT versión 8.2.0 siguiendo las instrucciones de la documentación de NVIDIA TensorRT.
    - En dispositivos x86 con una GPU que tenga una arquitectura NVIDIA anterior a Ampere (o la capacidad de procesamiento de la GPU sea inferior a 8.0), haga lo siguiente:
      - Instale cuDNN siguiendo las instrucciones de la Guía de instalación de cuDNN de NVIDIA.
      - Instale la versión 10.2 de CUDA siguiendo las instrucciones de la Guía de instalación de NVIDIA CUDA para Linux.
      - Instala TensorRT versión 7.1.3 o posterior, pero anterior a la versión 8.0.0, siguiendo las instrucciones de la documentación de NVIDIA TensorRT.
- El usuario del sistema que ejecute este componente debe ser miembro del grupo de sistemas que tenga acceso a la GPU del dispositivo. El nombre de este grupo varía según el sistema operativo. Consulte la documentación del sistema operativo y de la GPU para determinar el nombre de este grupo de sistemas.

Por ejemplo, en los dispositivos NVIDIA Jetson, el nombre de este grupo es `video` y puede ejecutar el siguiente comando para añadir un usuario del sistema a este grupo. Sustituya *`ggc_user`* por el nombre del usuario que desee añadir.

```
sudo usermod -aG video ggc_user
```

Dependencias

Este componente no tiene ninguna dependencia.

Configuración

Este componente proporciona los siguientes parámetros de configuración que puede personalizar al implementar el componente.

`Socket`

(Opcional) El socket de archivos donde funciona el agente Edge. Los componentes del modelo Lookout for Vision utilizan este socket de archivos para comunicarse con el Edge Agent. Si cambias este parámetro, debes especificar el mismo valor al implementar los componentes del modelo Lookout for Vision.

Valor predeterminado: `unix:///tmp/aws.iot.lookoutvision.EdgeAgent.sock`

Archivo de registro local

Este componente utiliza el siguiente archivo de registro.

```
/greengrass/v2/logs/aws.iot.lookoutvision.EdgeAgent.log
```

Para ver los registros de este componente

- Ejecute el siguiente comando en el dispositivo principal para ver el archivo de registro de este componente en tiempo real. */greengrass/v2*Sustitúyalo por la ruta a la carpeta AWS IoT Greengrass raíz.

  ```
  sudo tail -f /greengrass/v2/logs/aws.iot.lookoutvision.EdgeAgent.log
  ```

Registros de cambios

En la siguiente tabla se describen los cambios en cada versión del componente.

Versión	Cambios
1.2.0	Corrección de errores y mejoras generales.
1.1.9	Corrección de errores y mejoras generales.
1.1.8	Corrección de errores y mejoras generales.
1.1.7	Nuevas características • Instala el `opencv-python-headless` paquete en el entorno virtual de Lookout for Vision Edge Agent. Mejoras y correcciones de errores • Mejora el cálculo de la puntuación de confianza. • Cambia el tamaño de la máscara del modelo del mapa de calor al tamaño del archivo original. • Corrección de errores y mejoras generales.
1.1.6	Nuevas características Se han añadido nuevos valores al `DetectAnomalies` resultado. • `anomaly_score` — El número entre 0.0 y 1.0 que indica qué tan anómala es una imagen. • `anomaly_threshold` — Umbral establecido durante el entrenamiento del modelo que determina el límite entre una imagen anómala y una imagen normal. Corrección de errores y mejoras generales.
1.1.4	Nuevas características Se agregó soporte para OpenCV para cambiar el tamaño de las imágenes cuando esté disponible. El agente Edge usa Pillow cuando OpenCV no está disponible.

Versión	Cambios
	Mejoras y correcciones de errores Corrección de errores y mejoras generales.
1.1.3	Corrección de errores y mejoras generales.
1.1.1	Corrección de errores y mejoras generales.
1.1.0	Nuevas características • Añade compatibilidad con los modelos de segmentación de imágenes, que identifican las anomalías en las imágenes. • Añade compatibilidad con la inferencia de CPU, por lo que puedes usar los modelos Lookout for Vision en dispositivos principales sin GPU. Mejoras y correcciones de errores • Corrección de errores y mejoras generales.
1.0.0	Esta versión del componente Lookout for Vision Edge Agent requiere una versión de Python diferente a la 0.1.x. Si desea actualizar de la v0.1.x a la v1.x, debe actualizar la instalación de Python en el dispositivo principal. Mejoras y correcciones de errores • Corrección de errores y mejoras generales.
0.1.37	Corrección de errores y mejoras generales.
0.1,36	Versión inicial.

SageMaker Administrador perimetral

> **Important**
>
> SageMaker Edge Manager dejará de fabricarse el 26 de abril de 2024. Para obtener más información sobre cómo seguir implementando sus modelos en los dispositivos periféricos, consulte el final del ciclo de vida de SageMaker Edge Manager.

El componente Amazon SageMaker Edge Manager (`aws.greengrass.SageMakerEdgeManager`) instala el binario del agente SageMaker Edge Manager.

SageMaker Edge Manager proporciona una gestión de modelos para los dispositivos periféricos, de forma que pueda optimizar, proteger, supervisar y mantener los modelos de aprendizaje automático en las flotas de dispositivos periféricos. El componente SageMaker Edge Manager instala y administra el ciclo de vida del agente SageMaker Edge Manager en su dispositivo principal. También puede usar SageMaker Edge Manager para empaquetar y usar modelos SageMaker compilados en NEO como componentes de modelos en los dispositivos principales de Greengrass. Para obtener más información sobre el uso SageMaker del agente Edge Manager en su dispositivo principal, consulte. Utilice Amazon SageMaker Edge Manager en los dispositivos principales de Greengrass

SageMaker El componente Edge Manager v1.3.x instala el agente binario de Edge Manager v1.20220822.836f3023. Para obtener más información sobre las versiones binarias del agente Edge Manager, consulte Edge Manager Agent.

> **Note**
>
> El componente SageMaker Edge Manager solo está disponible en los siguientes casos Regiones de AWS:
>
> - US East (Ohio)
> - Este de EE. UU. (Norte de Virginia)
> - Oeste de EE. UU. (Oregón)
> - UE (Fráncfort)
> - UE (Irlanda)
> - Asia-Pacífico (Tokio)

Temas

- Archivo de registro local
- Registros de cambios

Versiones

Este componente tiene las siguientes versiones:

- 1.3.x
- 1.2.x
- 1.1.x
- 1.0.x

Tipo

Este componente es un componente genérico (`aws.greengrass.generic`). El núcleo de Greengrass ejecuta los scripts del ciclo de vida del componente.

Para obtener más información, consulte Tipos de componentes.

Sistema operativo

Este componente se puede instalar en los dispositivos principales que ejecutan los siguientes sistemas operativos:

- Linux
- Windows

Requisitos

Este componente tiene los siguientes requisitos:

- Un dispositivo principal de Greengrass que se ejecuta en Amazon Linux 2, una plataforma Linux basada en Debian (x86_64 o Armv8) o Windows (x86_64). Si no dispone de una, consulte Tutorial: Introducción a AWS IoT Greengrass V2.
- Python 3.6 o posterior, incluida `pip` la versión de Python, instalada en el dispositivo principal.
- El rol del dispositivo Greengrass se configuró con lo siguiente:

- Una relación de confianza que `sagemaker.amazonaws.com` permite `credentials.iot.amazonaws.com` y asume el rol, como se muestra en el siguiente ejemplo de política de IAM.

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

- La política gestionada por AmazonSageMakerEdgeDeviceFleetPolicyIAM.
- La `s3:PutObject` acción, tal como se muestra en el siguiente ejemplo de política de IAM.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "s3:PutObject"
      ],
      "Resource": [
        "*"
      ],
      "Effect": "Allow"
    }
  ]
}
```

- Un bucket de Amazon S3 creado en el mismo dispositivo principal de Greengrass Cuenta de AWS y Región de AWS en el mismo que él. SageMaker Edge Manager necesita un bucket S3 para crear una flota de dispositivos perimetrales y almacenar datos de muestra derivados de la ejecución de inferencias en su dispositivo. Para obtener información sobre la creación de buckets de S3, consulte Introducción a Amazon S3.
- Una flota de dispositivos SageMaker periféricos que usa el mismo alias de AWS IoT rol que su dispositivo principal de Greengrass. Para obtener más información, consulte Cree una flota de dispositivos periféricos.
- Su dispositivo principal Greengrass está registrado como dispositivo perimetral en su flota de dispositivos SageMaker Edge. El nombre del dispositivo perimetral debe coincidir con el AWS IoT nombre del dispositivo principal. Para obtener más información, consulte Registra tu dispositivo principal de Greengrass.

Puntos finales y puertos

Este componente debe poder realizar solicitudes salientes a los siguientes puntos finales y puertos, además de a los puntos finales y puertos necesarios para el funcionamiento básico. Para obtener más información, consulte Permitir el tráfico del dispositivo a través de un proxy o firewall.

punto de enlace	Puerto	Obligatoria	Descripción
`edge.sagemaker.` *`region`*`.amazonaws.com`	443	Sí	Compruebe el estado de registro del dispositivo y envíe las métricas a. SageMaker
`*.s3.amazonaws.com`	443	Sí	Cargue los datos de captura en el depósito de S3 que

punto de enlace	Puerto	Obligatoria	Descripción
			especifique. Puede sustituir los por * el nombre de cada depósito en el que cargue los datos.

Dependencias

Al implementar un componente, AWS IoT Greengrass también despliega versiones compatibles de sus dependencias. Esto significa que debe cumplir los requisitos del componente y de todas sus dependencias para poder implementarlo correctamente. En esta sección se enumeran las dependencias de las versiones publicadas de este componente y las restricciones de las versiones semánticas que definen las versiones de los componentes para cada dependencia. También puede ver las dependencias de cada versión del componente en la consola.AWS IoT Greengrass En la página de detalles del componente, busque la lista de dependencias.

1.3.5 and 1.3.6

La siguiente tabla muestra las dependencias de las versiones 1.3.5 y 1.3.6 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.13.0	Flexible
Servicio de intercambio de fichas	>=0.0.0	Rígido

1.3.4

La siguiente tabla muestra las dependencias de la versión 1.3.4 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.12.0	Flexible
Servicio de intercambio de fichas	>=0.0.0	Rígido

1.3.3

La siguiente tabla muestra las dependencias de la versión 1.3.3 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.11.0	Flexible
Servicio de intercambio de fichas	>=0.0.0	Rígido

1.3.2

La siguiente tabla muestra las dependencias de la versión 1.3.2 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.10.0	Flexible
Servicio de intercambio de fichas	>=0.0.0	Rígido

1.3.1

La siguiente tabla muestra las dependencias de la versión 1.3.1 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.9.0	Flexible
Servicio de intercambio de fichas	>=0.0.0	Rígido

1.1.1 - 1.3.0

La siguiente tabla muestra las dependencias de las versiones 1.1.1 a 1.3.0 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.8.0	Flexible
Servicio de intercambio de fichas	>=0.0.0	Rígido

1.1.0

La siguiente tabla muestra las dependencias de la versión 1.1.0 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.6.0	Flexible
Servicio de intercambio de fichas	>=0.0.0	Rígido

1.0.3

La siguiente tabla muestra las dependencias de la versión 1.0.3 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.5.0	Flexible

Dependencia	Versiones compatibles	Tipo de dependencia
Servicio de intercambio de fichas	>=0.0.0	Rígido

1.0.1 and 1.0.2

La siguiente tabla muestra las dependencias de las versiones 1.0.1 y 1.0.2 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.4.0	Flexible
Servicio de intercambio de fichas	>=0.0.0	Rígido

1.0.0

La siguiente tabla muestra las dependencias de la versión 1.0.0 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.3.0	Flexible
Servicio de intercambio de fichas	>=0.0.0	Rígido

Para obtener más información sobre las dependencias de los componentes, consulta la referencia de recetas de componentes.

Configuración

Este componente proporciona los siguientes parámetros de configuración que puede personalizar al implementar el componente.

> **Note**
>
> En esta sección se describen los parámetros de configuración que se establecen en el componente. Para obtener más información sobre la configuración de SageMaker Edge Manager correspondiente, consulte Edge Manager Agent en la Guía para SageMaker desarrolladores de Amazon.

`DeviceFleetName`

El nombre de la flota de dispositivos SageMaker Edge Manager que contiene su dispositivo principal Greengrass.

Debe especificar un valor para este parámetro en la actualización de la configuración al implementar este componente.

`BucketName`

El nombre del depósito de S3 al que se cargan los datos de inferencia capturados. El nombre del depósito debe contener la cadena`sagemaker`.

Si se establece `CaptureDataDestination` en`Cloud`, o si se establece `CaptureDataPeriodicUpload` en`true`, debe especificar un valor para este parámetro en la actualización de la configuración al implementar este componente.

> **Note**
>
> La captura de datos es una SageMaker función que se utiliza para cargar entradas de inferencia, resultados de inferencias y datos de inferencia adicionales a un bucket de S3 o a un directorio local para futuros análisis. Para obtener más información sobre el uso de datos de captura con SageMaker Edge Manager, consulte Manage Model en la Guía para SageMaker desarrolladores de Amazon.

`CaptureDataBatchSize`

(Opcional) El tamaño de un lote de solicitudes de datos de captura que gestiona el agente. Este valor debe ser inferior al tamaño del búfer que especifique`CaptureDataBufferSize`. Se recomienda no superar la mitad del tamaño del búfer.

El agente gestiona un lote de solicitudes cuando el número de solicitudes del búfer es igual al `CaptureDataBatchSize` número o cuando `CaptureDataPushPeriodSeconds` transcurre el intervalo, lo que ocurra primero.

Valor predeterminado: `10`

`CaptureDataBufferSize`

(Opcional) El número máximo de solicitudes de datos de captura almacenadas en el búfer.

Valor predeterminado: `30`

`CaptureDataDestination`

(Opcional) El destino en el que se almacenan los datos capturados. Este parámetro puede tener los siguientes valores:

- `Cloud`—Carga los datos capturados en el depósito de S3 que especifique. `BucketName`
- `Disk`—Escribe los datos capturados en el directorio de trabajo del componente.

Si lo especifica`Disk`, también puede optar por cargar periódicamente los datos capturados en su bucket de S3 `CaptureDataPeriodicUpload` configurándolo en. `true`

Valor predeterminado: `Cloud`

`CaptureDataPeriodicUpload`

(Opcional) Valor de cadena que especifica si se deben cargar periódicamente los datos capturados. Los valores admitidos son `true` y `false`.

Establezca este parámetro en `true` si lo ha establecido `CaptureDataDestination` y si también desea que el agente cargue periódicamente los datos capturados en su bucket de S3. `Disk`

Valor predeterminado: `false`

`CaptureDataPeriodicUploadPeriodSeconds`

(Opcional) El intervalo en segundos en el que el agente de SageMaker Edge Manager carga los datos capturados en el depósito de S3. Utilice este parámetro si lo ha establecido en`CaptureDataPeriodicUpload`. `true`

Valor predeterminado: 8

`CaptureDataPushPeriodSeconds`

(Opcional) El intervalo en segundos en el que el agente de SageMaker Edge Manager gestiona un lote de solicitudes de captura de datos desde el búfer.

El agente gestiona un lote de solicitudes cuando el número de solicitudes del búfer es igual al `CaptureDataBatchSize` número o cuando `CaptureDataPushPeriodSeconds` transcurre el intervalo, lo que ocurra primero.

Valor predeterminado: `4`

`CaptureDataBase64EmbedLimit`

(Opcional) El tamaño máximo en bytes de los datos capturados que carga el agente de SageMaker Edge Manager.

Valor predeterminado: `3072`

`FolderPrefix`

(Opcional) El nombre de la carpeta en la que el agente escribe los datos capturados. Si se establece `CaptureDataDestination` en`Disk`, el agente crea la carpeta en el directorio especificado por`CaptureDataDiskPath`. Si se establece `CaptureDataDestination` en`Cloud`, o si se establece `CaptureDataPeriodicUpload` en`true`, el agente crea la carpeta en el bucket de S3.

Valor predeterminado: `sme-capture`

`CaptureDataDiskPath`

Esta función está disponible en la versión 1.1.0 y en las versiones posteriores del componente SageMaker Edge Manager.

(Opcional) La ruta a la carpeta en la que el agente crea la carpeta de datos capturados. Si lo establece `CaptureDataDestination`en`Disk`, el agente crea la carpeta de datos capturados en este directorio. Si no especifica este valor, el agente crea la carpeta de datos capturados en el directorio de trabajo del componente. Utilice el `FolderPrefix` parámetro para especificar el nombre de la carpeta de datos capturados.

Valor predeterminado: *`/greengrass/v2`*`/work/aws.greengrass.SageMakerEdgeManager/capture`

`LocalDataRootPath`

Esta función está disponible en la versión 1.2.0 y en las versiones posteriores del componente SageMaker Edge Manager.

(Opcional) La ruta en la que este componente almacena los siguientes datos en el dispositivo principal:

- La base de datos local para los datos de tiempo de ejecución cuando se configura `DbEnable` en`true`.
- SageMaker Modelos neocompilados que este componente descarga automáticamente cuando se configura`DeploymentEnable`. `true`

Valor predeterminado: */greengrass/v2*`/work/aws.greengrass.SageMakerEdgeManager`

`DbEnable`

(Opcional) Puede habilitar este componente para almacenar los datos de tiempo de ejecución en una base de datos local para conservar los datos en caso de que el componente falle o el dispositivo se quede sin alimentación.

Esta base de datos requiere 5 MB de almacenamiento en el sistema de archivos del dispositivo principal.

Valor predeterminado: `false`

`DeploymentEnable`

Esta función está disponible en la versión 1.2.0 y en las versiones posteriores del componente SageMaker Edge Manager.

(Opcional) Puede habilitar este componente para recuperar automáticamente los modelos SageMaker compilados en NEO que cargue en Amazon S3. Después de cargar un modelo nuevo en Amazon S3, utilice SageMaker Studio o la SageMaker API para implementar el nuevo modelo en este dispositivo principal. Al habilitar esta función, puede implementar nuevos modelos en los dispositivos principales sin necesidad de crear una AWS IoT Greengrass implementación.

> **Important**
>
> Para utilizar esta función, debe `DbEnable` configurarla en`true`. Esta función utiliza la base de datos local para rastrear los modelos que recupera de Nube de AWS.

Valor predeterminado: `false`

`DeploymentPollInterval`

Esta función está disponible en la versión 1.2.0 y en las versiones posteriores del componente SageMaker Edge Manager.

(Opcional) El tiempo (en minutos) entre el que este componente comprueba si hay nuevos modelos para descargar. Esta opción se aplica cuando se configura `DeploymentEnable` en`true`.

Predeterminado: `1440` (1 día)

`DLRBackendOptions`

Esta función está disponible en la versión 1.2.0 y en las versiones posteriores del componente SageMaker Edge Manager.

(Opcional) El tiempo de ejecución del DLR marca el tiempo de ejecución del DLR que utiliza este componente. Puede configurar el siguiente indicador:

- `TVM_TENSORRT_CACHE_DIR`— Habilita el almacenamiento en caché del modelo TensorRT. Especifique una ruta absoluta a una carpeta existente que tenga permisos de lectura/escritura.
- `TVM_TENSORRT_CACHE_DISK_SIZE_MB`— Asigna el límite superior de la carpeta de caché del modelo TensorRT. Cuando el tamaño del directorio supera este límite, se eliminan los motores en caché que se utilizan menos. El valor predeterminado es 512 MB.

Por ejemplo, puedes establecer este parámetro en el siguiente valor para habilitar el almacenamiento en caché del modelo TensorRT y limitar el tamaño de la caché a 800 MB.

```
TVM_TENSORRT_CACHE_DIR=/data/secured_folder/trt/cache;
 TVM_TENSORRT_CACHE_DISK_SIZE_MB=800
```

`SagemakerEdgeLogVerbose`

(Opcional) Valor de cadena que especifica si se debe habilitar el registro de depuración. Los valores admitidos son `true` y `false`.

Valor predeterminado: `false`

`UnixSocketName`

(Opcional) La ubicación del descriptor del archivo de socket de SageMaker Edge Manager en el dispositivo principal.

Valor predeterminado: `/tmp/aws.greengrass.SageMakerEdgeManager.sock`

Example Ejemplo: actualización de combinación de configuraciones

El siguiente ejemplo de configuración especifica que el dispositivo principal forma parte del bucket de S3 *MyEdgeDeviceFleet*y que el agente escribe los datos de captura tanto en el dispositivo como en un bucket de S3. Esta configuración también permite el registro de depuración.

```
{
    "DeviceFleetName": "MyEdgeDeviceFleet",
    "BucketName": "DOC-EXAMPLE-BUCKET",
    "CaptureDataDestination": "Disk",
    "CaptureDataPeriodicUpload": "true",
    "SagemakerEdgeLogVerbose": "true"
}
```

Archivo de registro local

Este componente utiliza el siguiente archivo de registro.

Linux

```
/greengrass/v2/logs/aws.greengrass.SageMakerEdgeManager.log
```

Windows

```
C:\greengrass\v2\logs\aws.greengrass.SageMakerEdgeManager.log
```

Para ver los registros de este componente

- Ejecute el siguiente comando en el dispositivo principal para ver el archivo de registro de este componente en tiempo real. Sustituya */greengrass/v2* o *C:\greengrass\v2* por la ruta a la carpeta AWS IoT Greengrass raíz.

  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
```

Registros de cambios

En la siguiente tabla se describen los cambios en cada versión del componente.

Versión	Cambios
1.3.6	Versión actualizada para la versión 2.12.5 de Greengrass nucleus.
1.3.5	Versión actualizada para la versión 2.12.0 de Greengrass nucleus.
1.3.4	Versión actualizada para la versión 2.11.0 de Greengrass nucleus.
1.3.3	Versión actualizada para la versión 2.10.0 de Greengrass nucleus.
1.3.2	Versión actualizada para la versión 2.9.0 de Greengrass nucleus.
1.3.1	Versión actualizada para la versión 2.8.0 de Greengrass nucleus.
1.3.0	Nuevas características • Añade compatibilidad con la administración del tamaño del disco caché de TensorRT. • Agrega el `TVM_TENSORRT_CACHE_DISK_SIZE_MB` indicador opcional al BackendOptions parámetro DLR para establecer el límite de tamaño de los modelos en caché en el disco. Mejoras • Proporciona una simultaneidad de predicciones mejorada. Esto ayuda a aprovechar mejor los motores de aceleración de los dispositivos, como las GPU.
1.2.0	Nuevas características • Añade compatibilidad con este componente para recuperar automátic amente los modelos SageMaker compilados en NEO que cargue

Versión	Cambios
	en Amazon S3. Al habilitar esta función, puede implementar nuevos modelos en los dispositivos principales sin necesidad de crear una AWS IoT Greengrass implementación. • Añade compatibilidad con una base de datos de respaldo que este componente utiliza para conservar los datos de tiempo de ejecución, en caso de que el componente falle o el dispositivo se quede sin alimentac ión. • Añade compatibilidad para configurar los indicadores de tiempo de ejecución del DLR al configurar este componente.
1.1.1	Versión actualizada para la versión 2.7.0 de Greengrass Nucleus.
1.1.0	Nuevas características • Añade compatibilidad con los dispositivos principales de Greengrass que ejecutan Amazon Linux 2. • Añade el nuevo parámetro `CaptureDataDiskPath` de configura ción. Puede usar este parámetro para especificar la ruta de la carpeta de datos capturados en su dispositivo. Mejoras y correcciones de errores • Versión actualizada para la versión 2.5.0 de Greengrass nucleus.
1.0.3	Versión actualizada para la versión 2.4.0 de Greengrass nucleus.
1.0.2	Mejoras y correcciones de errores Actualiza el script de instalación en el ciclo de vida del componente. Sus dispositivos principales ahora deben tener Python 3.6 o posterior, incluida `pip` su versión de Python, instalado en el dispositivo antes de implement ar este componente.
1.0.1	Versión actualizada para la versión 2.3.0 de Greengrass nucleus.
1.0.0	Versión inicial.

Clasificación de imágenes DLR

El componente de clasificación de imágenes del DLR (`aws.greengrass.DLRImageClassification`) contiene un ejemplo de código de inferencia para realizar inferencias de clasificación de imágenes mediante Deep Learning Runtime y los modelos resnet-50. Este componente utiliza la variante Tienda de modelos de clasificación de imágenes DLR y los Tiempo de ejecución de DLR componentes como dependencias para descargar el DLR y los modelos de muestra.

Para utilizar este componente de inferencia con un modelo de DLR personalizado, cree una versión personalizada del componente de tienda de modelos dependiente. Para usar su propio código de inferencia personalizado, puede usar la receta de este componente como plantilla para crear un componente de inferencia personalizado.

Temas

- Versiones
- Tipo
- Sistema operativo
- Requisitos
- Dependencias
- Configuración
- Archivo de registro local
- Registros de cambios

Versiones

Este componente tiene las siguientes versiones:

- 2.1.x
- 2.0.x

Tipo

Este componente es un componente genérico () `aws.greengrass.generic`. El núcleo de Greengrass ejecuta los scripts del ciclo de vida del componente.

Para obtener más información, consulte Tipos de componentes.

Sistema operativo

Este componente se puede instalar en los dispositivos principales que ejecutan los siguientes sistemas operativos:

- Linux
- Windows

Requisitos

Este componente tiene los siguientes requisitos:

- En los dispositivos principales de Greengrass que ejecutan Amazon Linux 2 o Ubuntu 18.04, se instala en el dispositivo la versión 2.27 o posterior de la Biblioteca C GNU (glibc).
- En los dispositivos ARMv7L, como Raspberry Pi, las dependencias para OpenCV-Python están instaladas en el dispositivo. Ejecute el siguiente comando para instalar las dependencias.

  ```
  sudo apt-get install libopenjp2-7 libilmbase23 libopenexr-dev libavcodec-dev
   libavformat-dev libswscale-dev libv4l-dev libgtk-3-0 libwebp-dev
  ```
- Los dispositivos Raspberry Pi que ejecutan el sistema operativo Bullseye de Raspberry Pi deben cumplir los siguientes requisitos:
  - NumPy 1.22.4 o una versión posterior instalada en el dispositivo. Raspberry Pi OS Bullseye incluye una versión anterior de NumPy, por lo que puede ejecutar el siguiente comando para actualizar NumPy el dispositivo.

    ```
    pip3 install --upgrade numpy
    ```
  - La pila de cámaras antigua habilitada en el dispositivo. El Raspberry Pi OS Bullseye incluye una nueva pila de cámaras que está habilitada de forma predeterminada y no es compatible, por lo que debes activar la pila de cámaras antigua.

    Para activar la pila de cámaras antigua

    1. Ejecute el siguiente comando para abrir la herramienta de configuración de Raspberry Pi.

       ```
       sudo raspi-config
       ```

2. Seleccione Opciones de interfaz.
3. Seleccione Cámara antigua para activar la pila de cámaras antiguas.
4. Reinicie el Raspberry Pi.

Dependencias

Al implementar un componente, AWS IoT Greengrass también despliega versiones compatibles de sus dependencias. Esto significa que debe cumplir los requisitos del componente y de todas sus dependencias para poder implementarlo correctamente. En esta sección se enumeran las dependencias de las versiones publicadas de este componente y las restricciones de las versiones semánticas que definen las versiones de los componentes para cada dependencia. También puede ver las dependencias de cada versión del componente en la consola.AWS IoT Greengrass En la página de detalles del componente, busque la lista de dependencias.

2.1.13 and 2.1.14

La siguiente tabla muestra las dependencias de las versiones 2.1.13 y 2.1.14 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.13.0	Flexible
Tienda de modelos de clasificación de imágenes DLR	~2.1.0	Rígido
DLR	~1.6.0	Rígido

2.1.12

La siguiente tabla muestra las dependencias de la versión 2.1.12 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.12.0	Flexible

Dependencia	Versiones compatibles	Tipo de dependencia
Tienda de modelos de clasificación de imágenes DLR	~2.1.0	Rígido
DLR	~1.6.0	Rígido

2.1.11

La siguiente tabla muestra las dependencias de la versión 2.1.11 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.11.0	Flexible
Tienda de modelos de clasificación de imágenes DLR	~2.1.0	Rígido
DLR	~1.6.0	Rígido

2.1.10

La siguiente tabla muestra las dependencias de la versión 2.1.10 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.10.0	Flexible
Tienda de modelos de clasificación de imágenes DLR	~2.1.0	Rígido
DLR	~1.6.0	Rígido

2.1.9

La siguiente tabla muestra las dependencias de la versión 2.1.9 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.9.0	Flexible
Tienda de modelos de clasificación de imágenes DLR	~2.1.0	Rígido
DLR	~1.6.0	Rígido

2.1.8

La siguiente tabla muestra las dependencias de la versión 2.1.8 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.8.0	Flexible
Tienda de modelos de clasificación de imágenes DLR	~2.1.0	Rígido
DLR	~1.6.0	Rígido

2.1.7

La siguiente tabla muestra las dependencias de la versión 2.1.7 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.7.0	Flexible

Dependencia	Versiones compatibles	Tipo de dependencia
Tienda de modelos de clasificación de imágenes DLR	~2.1.0	Rígido
DLR	~1.6.0	Rígido

2.1.6

La siguiente tabla muestra las dependencias de la versión 2.1.6 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.6.0	Flexible
Tienda de modelos de clasificación de imágenes DLR	~2.1.0	Rígido
DLR	~1.6.0	Rígido

2.1.4 - 2.1.5

La siguiente tabla muestra las dependencias de las versiones 2.1.4 a 2.1.5 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.5.0	Flexible
Tienda de modelos de clasificación de imágenes DLR	~2.1.0	Rígido
DLR	~1.6.0	Rígido

2.1.3

La siguiente tabla muestra las dependencias de la versión 2.1.3 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.4.0	Flexible
Tienda de modelos de clasificación de imágenes DLR	~2.1.0	Rígido
DLR	~1.6.0	Rígido

2.1.2

La siguiente tabla muestra las dependencias de la versión 2.1.2 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.3.0	Flexible
Tienda de modelos de clasificación de imágenes DLR	~2.1.0	Rígido
DLR	~1.6.0	Rígido

2.1.1

La siguiente tabla muestra las dependencias de la versión 2.1.1 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.2.0	Flexible

Dependencia	Versiones compatibles	Tipo de dependencia
Tienda de modelos de clasificación de imágenes DLR	~2.1.0	Rígido
DLR	~1.6.0	Rígido

2.0.x

La siguiente tabla muestra las dependencias de la versión 2.0.x de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	~2.0.0	Flexible
Tienda de modelos de clasificación de imágenes DLR	~2.0.0	Rígido
DLR	~1.3.0	Flexible

Configuración

Este componente proporciona los siguientes parámetros de configuración que puede personalizar al implementar el componente.

2.1.x

`accessControl`

(Opcional) El objeto que contiene la política de autorización que permite al componente publicar mensajes en el tema de notificaciones predeterminado.

Predeterminado:

```
{
   "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`

(Opcional) El tema sobre el que desea publicar los resultados de la inferencia. Si modifica este valor, también debe modificar el valor del `resources` `accessControl` parámetro para que coincida con el nombre del tema personalizado.

Valor predeterminado: `ml/dlr/image-classification`

`Accelerator`

El acelerador que quieres usar. Los valores admitidos son `cpu` y `gpu`.

Los modelos de muestra del componente del modelo dependiente solo admiten la aceleración de la CPU. Para usar la aceleración de la GPU con un modelo personalizado diferente, cree un componente de modelo personalizado para anular el componente del modelo público.

Valor predeterminado: `cpu`

`ImageDirectory`

(Opcional) La ruta de la carpeta del dispositivo donde los componentes de inferencia leen las imágenes. Puede modificar este valor en cualquier ubicación del dispositivo a la que tenga acceso de lectura y escritura.

Valor predeterminado: */greengrass/v2*`/packages/artifacts-unarchived/`*component-name*`/image_classification/sample_images/`

> **Note**
>
> Si establece el valor de `true`, `UseCamera` se ignora este parámetro de configuración.

`ImageName`

(Opcional) El nombre de la imagen que el componente de inferencia utiliza como entrada para realizar una predicción. El componente busca la imagen en la carpeta especificada en`ImageDirectory`. De forma predeterminada, el componente usa la imagen de muestra en el directorio de imágenes predeterminado. AWS IoT Greengrass admite los siguientes formatos de imagen: `jpeg`jpg,png, ynpy.

Valor predeterminado: `cat.jpeg`

> **Note**
>
> Si establece el valor de en`true`, se `UseCamera` ignora este parámetro de configuración.

`InferenceInterval`

(Opcional) El tiempo en segundos entre cada predicción realizada por el código de inferencia. El código de inferencia de muestra se ejecuta indefinidamente y repite sus predicciones en el intervalo de tiempo especificado. Por ejemplo, puede cambiarlo por un intervalo más corto si desea utilizar imágenes tomadas por una cámara para realizar predicciones en tiempo real.

Valor predeterminado: `3600`

`ModelResourceKey`

(Opcional) Los modelos que se utilizan en el componente de modelo público dependiente. Modifique este parámetro solo si anula el componente del modelo público por un componente personalizado.

Predeterminado:

```
{
    "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`

(Opcional) Valor de cadena que define si se deben utilizar imágenes de una cámara conectada al dispositivo principal de Greengrass. Los valores admitidos son `true` y `false`.

Si establece este valor en`true`, el código de inferencia de muestra accede a la cámara del dispositivo y ejecuta la inferencia localmente en la imagen capturada. Se ignoran los valores de los `ImageDirectory` parámetros `ImageName` y. Asegúrese de que el usuario que ejecuta este componente tenga acceso de lectura y escritura a la ubicación en la que la cámara almacena las imágenes capturadas.

Valor predeterminado: `false`

> **Note**
>
> Al ver la receta de este componente, el parámetro de `UseCamera` configuración no aparece en la configuración predeterminada. Sin embargo, puede modificar el valor de este parámetro en una actualización de la combinación de configuraciones al implementar el componente.
> Si lo establece `UseCameratrue`, también debe crear un enlace simbólico para permitir que el componente de inferencia acceda a la cámara desde el entorno virtual creado por el componente de tiempo de ejecución. Para obtener más información sobre el uso de una cámara con los componentes de inferencia de muestra, consulte. Actualizar las configuraciones de los componentes

2.0.x

`MLRootPath`

(Opcional) La ruta de la carpeta en los dispositivos principales de Linux donde los componentes de inferencia leen las imágenes y escriben los resultados de la inferencia. Puede modificar este valor en cualquier ubicación del dispositivo a la que el usuario que ejecuta este componente tenga acceso de lectura y escritura.

Valor predeterminado: *`/greengrass/v2`*`/work/variant.DLR/greengrass_ml`

Valor predeterminado: *`/greengrass/v2`*`/work/variant.TensorFlowLite/greengrass_ml`

`Accelerator`

El acelerador que desea utilizar. Los valores admitidos son `cpu` y `gpu`.

Los modelos de muestra del componente del modelo dependiente solo admiten la aceleración de la CPU. Para usar la aceleración de la GPU con un modelo personalizado diferente, cree un componente de modelo personalizado para anular el componente del modelo público.

Valor predeterminado: `cpu`

`ImageName`

(Opcional) El nombre de la imagen que el componente de inferencia utiliza como entrada para realizar una predicción. El componente busca la imagen en la carpeta especificada en`ImageDirectory`. La ubicación por defecto es*`MLRootPath`*`/images`. AWS IoT Greengrass admite los siguientes formatos de imagen: `jpegjpg`,`png`, y`npy`.

Valor predeterminado: `cat.jpeg`

`InferenceInterval`

(Opcional) El tiempo en segundos entre cada predicción realizada por el código de inferencia. El código de inferencia de muestra se ejecuta indefinidamente y repite sus predicciones en el intervalo de tiempo especificado. Por ejemplo, puede cambiarlo por un intervalo más corto si desea utilizar imágenes tomadas por una cámara para realizar predicciones en tiempo real.

Valor predeterminado: `3600`

`ModelResourceKey`

(Opcional) Los modelos que se utilizan en el componente de modelo público dependiente. Modifique este parámetro solo si anula el componente del modelo público por un componente personalizado.

Predeterminado:

```
armv7l: "DLR-resnet50-armv7l-cpu-ImageClassification"
x86_64: "DLR-resnet50-x86_64-cpu-ImageClassification"
```

Archivo de registro local

Este componente utiliza el siguiente archivo de registro.

Linux

```
/greengrass/v2/logs/aws.greengrass.DLRImageClassification.log
```

Windows

```
C:\greengrass\v2\logs\aws.greengrass.DLRImageClassification.log
```

Para ver los registros de este componente

- Ejecute el siguiente comando en el dispositivo principal para ver el archivo de registro de este componente en tiempo real. Sustituya */greengrass/v2* o *C:\greengrass\v2* por la ruta a la carpeta AWS IoT Greengrass raíz.

  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
  ```

Registros de cambios

En la siguiente tabla se describen los cambios en cada versión del componente.

Versión	Cambios
2.1.14	Versión actualizada para la versión 2.12.5 de Greengrass nucleus.
2.1.13	Versión actualizada para la versión 2.12.0 de Greengrass nucleus.
2.1.12	Versión actualizada para la versión 2.11.0 de Greengrass nucleus.
2.1.11	Versión actualizada para la versión 2.10.0 de Greengrass nucleus.
2.1.10	Versión actualizada para la versión 2.9.0 de Greengrass nucleus.

Versión	Cambios
2.1.9	Versión actualizada para la versión 2.8.0 de Greengrass nucleus.
2.1.8	Versión actualizada para la versión 2.7.0 de Greengrass Nucleus.
2.1.7	Versión actualizada para la versión 2.6.0 de Greengrass nucleus.
2.1.6	Versión actualizada para la versión 2.5.0 de Greengrass nucleus.
2.1.5	Componente publicado en su totalidad. Regiones de AWS
2.1.4	Versión actualizada para la versión 2.4.0 de Greengrass Nucleus. Esta versión no está disponible en Europa (Londres) (`eu-west-2`).
2.1.3	Versión actualizada para el lanzamiento de la versión 2.3.0 de Greengrass nucleus.
2.1.2	Versión actualizada para el lanzamiento de la versión 2.2.0 de Greengrass nucleus.

Versión	Cambios
2.1.1	Nuevas características • Utilice Deep Learning Runtime v1.6.0. • Añada compatibilidad con la clasificación de imágenes de muestra en las plataformas Armv8 (AArch64). Esto amplía el soporte de aprendizaje automático para los dispositivos principales de Greengrass que ejecutan NVIDIA Jetson, como el Jetson Nano. • Habilite la integración de la cámara para la inferencia de muestras. Utilice el nuevo parámetro de `UseCamera` configuración para permitir que el código de inferencia de muestra acceda a la cámara del dispositi vo principal de Greengrass y ejecute la inferencia localmente en la imagen capturada. • Añada soporte para publicar los resultados de la inferencia en. Nube de AWS Utilice el nuevo parámetro `PublishResultsOnTopic` de configuración para especificar el tema sobre el que desea publicar los resultados. • Añada el nuevo parámetro de `ImageDirectory` configuración que le permita especificar un directorio personalizado para la imagen en la que desee realizar la inferencia. Mejoras y correcciones de errores • Escriba los resultados de la inferencia en el archivo de registro del componente en lugar de en un archivo de inferencias independiente. • Utilice el módulo de registro del software AWS IoT Greengrass Core para registrar la salida de los componentes. • Utilice el SDK para dispositivos con AWS IoT para leer la configuración del componente y aplicar los cambios de configuración.
2.0.4	Versión inicial.

Detección de objetos DLR

El componente de detección de objetos del DLR (`aws.greengrass.DLRObjectDetection`) contiene un ejemplo de código de inferencia para realizar inferencias de detección de objetos mediante Deep Learning Runtime y ejemplos de modelos previamente entrenados. Este componente

utiliza la variante Tienda de modelos de detección de objetos DLR y los Tiempo de ejecución de DLR componentes como dependencias para descargar el DLR y los modelos de muestra.

Para utilizar este componente de inferencia con un modelo DLR personalizado, cree una versión personalizada del componente de tienda de modelos dependiente. Para usar su propio código de inferencia personalizado, puede usar la receta de este componente como plantilla para crear un componente de inferencia personalizado.

Temas

- Versiones
- Tipo
- Sistema operativo
- Requisitos
- Dependencias
- Configuración
- Archivo de registro local
- Registros de cambios

Versiones

Este componente tiene las siguientes versiones:

- 2.1.x
- 2.0.x

Tipo

Este componente es un componente genérico () `aws.greengrass.generic`. El núcleo de Greengrass ejecuta los scripts del ciclo de vida del componente.

Para obtener más información, consulte Tipos de componentes.

Sistema operativo

Este componente se puede instalar en los dispositivos principales que ejecutan los siguientes sistemas operativos:

- Linux
- Windows

Requisitos

Este componente tiene los siguientes requisitos:

- En los dispositivos principales de Greengrass que ejecutan Amazon Linux 2 o Ubuntu 18.04, se instala en el dispositivo la versión 2.27 o posterior de la Biblioteca C GNU (glibc).
- En los dispositivos ARMv7L, como Raspberry Pi, las dependencias para OpenCV-Python están instaladas en el dispositivo. Ejecute el siguiente comando para instalar las dependencias.

  ```
  sudo apt-get install libopenjp2-7 libilmbase23 libopenexr-dev libavcodec-dev
   libavformat-dev libswscale-dev libv4l-dev libgtk-3-0 libwebp-dev
  ```

- Los dispositivos Raspberry Pi que ejecutan el sistema operativo Bullseye de Raspberry Pi deben cumplir los siguientes requisitos:
  - NumPy 1.22.4 o una versión posterior instalada en el dispositivo. Raspberry Pi OS Bullseye incluye una versión anterior de NumPy, por lo que puede ejecutar el siguiente comando para actualizar NumPy el dispositivo.

    ```
    pip3 install --upgrade numpy
    ```

  - La pila de cámaras antigua habilitada en el dispositivo. El Raspberry Pi OS Bullseye incluye una nueva pila de cámaras que está habilitada de forma predeterminada y no es compatible, por lo que debes activar la pila de cámaras antigua.

    Para activar la pila de cámaras antigua

    1. Ejecute el siguiente comando para abrir la herramienta de configuración de Raspberry Pi.

       ```
       sudo raspi-config
       ```

    2. Seleccione Opciones de interfaz.
    3. Seleccione Cámara antigua para activar la pila de cámaras antiguas.
    4. Reinicie el Raspberry Pi.

Dependencias

Al implementar un componente, AWS IoT Greengrass también despliega versiones compatibles de sus dependencias. Esto significa que debe cumplir los requisitos del componente y de todas sus dependencias para poder implementarlo correctamente. En esta sección se enumeran las dependencias de las versiones publicadas de este componente y las restricciones de las versiones semánticas que definen las versiones de los componentes para cada dependencia. También puede ver las dependencias de cada versión del componente en la consola.AWS IoT Greengrass En la página de detalles del componente, busque la lista de dependencias.

2.1.13 and 2.1.14

La siguiente tabla muestra las dependencias de las versiones 2.1.13 y 2.1.14 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.13.0	Flexible
Tienda de modelos de detección de objetos DLR	~2.1.0	Rígido
DLR	~1.6.0	Rígido

2.1.12

La siguiente tabla muestra las dependencias de la versión 2.1.12 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.12.0	Flexible
Tienda de modelos de detección de objetos DLR	~2.1.0	Rígido
DLR	~1.6.0	Rígido

2.1.11

La siguiente tabla muestra las dependencias de la versión 2.1.11 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.11.0	Flexible
Tienda de modelos de detección de objetos DLR	~2.1.0	Rígido
DLR	~1.6.0	Rígido

2.1.10

La siguiente tabla muestra las dependencias de la versión 2.1.10 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.10.0	Flexible
Tienda de modelos de detección de objetos DLR	~2.1.0	Rígido
DLR	~1.6.0	Rígido

2.1.9

La siguiente tabla muestra las dependencias de la versión 2.1.9 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.9.0	Flexible
Tienda de modelos de detección de objetos DLR	~2.1.0	Rígido
DLR	~1.6.0	Rígido

2.1.8

La siguiente tabla muestra las dependencias de la versión 2.1.8 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.8.0	Flexible
Tienda de modelos de detección de objetos DLR	~2.1.0	Rígido
DLR	~1.6.0	Rígido

2.1.7

La siguiente tabla muestra las dependencias de la versión 2.1.7 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.7.0	Flexible
Tienda de modelos de detección de objetos DLR	~2.1.0	Rígido
DLR	~1.6.0	Rígido

2.1.6

La siguiente tabla muestra las dependencias de la versión 2.1.6 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.6.0	Flexible
Tienda de modelos de detección de objetos DLR	~2.1.0	Rígido
DLR	~1.6.0	Rígido

2.1.4 - 2.1.5

La siguiente tabla muestra las dependencias de las versiones 2.1.4 a 2.1.5 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.5.0	Flexible
Tienda de modelos de detección de objetos DLR	~2.1.0	Rígido
DLR	~1.6.0	Rígido

2.1.3

La siguiente tabla muestra las dependencias de la versión 2.1.3 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.4.0	Flexible
Tienda de modelos de detección de objetos DLR	~2.1.0	Rígido
DLR	~1.6.0	Rígido

2.1.2

La siguiente tabla muestra las dependencias de la versión 2.1.2 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.3.0	Flexible
Tienda de modelos de detección de objetos DLR	~2.1.0	Rígido
DLR	~1.6.0	Rígido

2.1.1

La siguiente tabla muestra las dependencias de la versión 2.1.1 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.2.0	Flexible
Tienda de modelos de detección de objetos DLR	~2.1.0	Rígido
DLR	~1.6.0	Rígido

2.0.x

La siguiente tabla muestra las dependencias de la versión 2.0.x de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	~2.0.0	Flexible
Tienda de modelos de detección de objetos DLR	~2.0.0	Rígido
DLR	~1.3.0	Flexible

Configuración

Este componente proporciona los siguientes parámetros de configuración que puede personalizar al implementar el componente.

2.1.x

`accessControl`

(Opcional) El objeto que contiene la política de autorización que permite al componente publicar mensajes en el tema de notificaciones predeterminado.

Predeterminado:

```
{
   "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`

(Opcional) El tema sobre el que desea publicar los resultados de la inferencia. Si modifica este valor, también debe modificar el valor del `resources` `accessControl` parámetro para que coincida con el nombre del tema personalizado.

Valor predeterminado: `ml/dlr/object-detection`

`Accelerator`

El acelerador que quieres usar. Los valores admitidos son `cpu` y `gpu`.

Los modelos de muestra del componente del modelo dependiente solo admiten la aceleración de la CPU. Para usar la aceleración de la GPU con un modelo personalizado diferente, cree un componente de modelo personalizado para anular el componente del modelo público.

Valor predeterminado: `cpu`

`ImageDirectory`

(Opcional) La ruta de la carpeta del dispositivo donde los componentes de inferencia leen las imágenes. Puede modificar este valor en cualquier ubicación del dispositivo a la que tenga acceso de lectura y escritura.

Valor predeterminado: */greengrass/v2*`/packages/artifacts-unarchived/`*component-name*`/object_detection/sample_images/`

> **Note**
>
> Si establece el valor de en`true`, `UseCamera` se ignora este parámetro de configuración.

`ImageName`

(Opcional) El nombre de la imagen que el componente de inferencia utiliza como entrada para realizar una predicción. El componente busca la imagen en la carpeta especificada en`ImageDirectory`. De forma predeterminada, el componente usa la imagen de muestra en el directorio de imágenes predeterminado. AWS IoT Greengrass admite los siguientes formatos de imagen: `jpeg`jpg,`png`, ynpy.

Valor predeterminado: `objects.jpg`

> **Note**
>
> Si establece el valor de en`true`, se `UseCamera` ignora este parámetro de configuración.

`InferenceInterval`

(Opcional) El tiempo en segundos entre cada predicción realizada por el código de inferencia. El código de inferencia de muestra se ejecuta indefinidamente y repite sus predicciones en el intervalo de tiempo especificado. Por ejemplo, puede cambiarlo por un intervalo más corto si desea utilizar imágenes tomadas por una cámara para realizar predicciones en tiempo real.

Valor predeterminado: `3600`

`ModelResourceKey`

(Opcional) Los modelos que se utilizan en el componente de modelo público dependiente. Modifique este parámetro solo si anula el componente del modelo público por un componente personalizado.

Predeterminado:

```
{
```

```
    "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`

(Opcional) Valor de cadena que define si se deben utilizar imágenes de una cámara conectada al dispositivo principal de Greengrass. Los valores admitidos son `true` y `false`.

Si establece este valor en`true`, el código de inferencia de muestra accede a la cámara del dispositivo y ejecuta la inferencia localmente en la imagen capturada. Se ignoran los valores de los `ImageDirectory` parámetros `ImageName` y. Asegúrese de que el usuario que ejecuta este componente tenga acceso de lectura y escritura a la ubicación en la que la cámara almacena las imágenes capturadas.

Valor predeterminado: `false`

> **Note**
>
> Al ver la receta de este componente, el parámetro de `UseCamera` configuración no aparece en la configuración predeterminada. Sin embargo, puede modificar el valor de este parámetro en una actualización de la combinación de configuraciones al implementar el componente.
> Si lo establece `UseCameratrue`, también debe crear un enlace simbólico para permitir que el componente de inferencia acceda a la cámara desde el entorno virtual creado por el componente de tiempo de ejecución. Para obtener más información sobre el uso de una cámara con los componentes de inferencia de muestra, consulte. Actualizar las configuraciones de los componentes

2.0.x

`MLRootPath`

(Opcional) La ruta de la carpeta en los dispositivos principales de Linux donde los componentes de inferencia leen las imágenes y escriben los resultados de la inferencia. Puede modificar este valor en cualquier ubicación del dispositivo a la que el usuario que ejecuta este componente tenga acceso de lectura y escritura.

Valor predeterminado: */greengrass/v2*/work/variant.DLR/greengrass_ml

Valor predeterminado: */greengrass/v2*/work/variant.TensorFlowLite/greengrass_ml

`Accelerator`

No lo modifique. Actualmente, el único valor admitido para el acelerador es `cpu` porque los modelos de los componentes del modelo dependiente se compilan únicamente para el acelerador de la CPU.

`ImageName`

(Opcional) El nombre de la imagen que el componente de inferencia utiliza como entrada para realizar una predicción. El componente busca la imagen en la carpeta especificada en`ImageDirectory`. La ubicación por defecto es*MLRootPath*/`images`. AWS IoT Greengrass admite los siguientes formatos de imagen: `jpeg`jpg,png, ynpy.

Valor predeterminado: `objects.jpg`

`InferenceInterval`

(Opcional) El tiempo en segundos entre cada predicción realizada por el código de inferencia. El código de inferencia de muestra se ejecuta indefinidamente y repite sus predicciones en el intervalo de tiempo especificado. Por ejemplo, puede cambiarlo por un intervalo más corto si desea utilizar imágenes tomadas por una cámara para realizar predicciones en tiempo real.

Valor predeterminado: `3600`

`ModelResourceKey`

(Opcional) Los modelos que se utilizan en el componente de modelo público dependiente. Modifique este parámetro solo si anula el componente del modelo público por un componente personalizado.

Predeterminado:

```
{
    armv7l: "DLR-yolo3-armv7l-cpu-ObjectDetection",
    x86_64: "DLR-yolo3-x86_64-cpu-ObjectDetection"
}
```

Archivo de registro local

Este componente utiliza el siguiente archivo de registro.

Linux

```
/greengrass/v2/logs/aws.greengrass.DLRObjectDetection.log
```

Windows

```
C:\greengrass\v2\logs\aws.greengrass.DLRObjectDetection.log
```

Para ver los registros de este componente

- Ejecute el siguiente comando en el dispositivo principal para ver el archivo de registro de este componente en tiempo real. Sustituya */greengrass/v2* o *C:\greengrass\v2* por la ruta a la carpeta AWS IoT Greengrass raíz.

  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
  ```

Registros de cambios

En la siguiente tabla se describen los cambios en cada versión del componente.

Versión	Cambios
2.1.14	Versión actualizada para la versión 2.12.5 de Greengrass nucleus.
2.1.13	Versión actualizada para la versión 2.12.0 de Greengrass nucleus.
2.1.12	Versión actualizada para la versión 2.11.0 de Greengrass nucleus.

Versión	Cambios
2.1.11	Versión actualizada para la versión 2.10.0 de Greengrass nucleus.
2.1.10	Versión actualizada para la versión 2.9.0 de Greengrass Nucleus.
2.1.9	Versión actualizada para el lanzamiento de la versión 2.8.0 de Greengrass nucleus.
2.1.8	Versión actualizada para el lanzamiento de la versión 2.7.0 de Greengrass nucleus.
2.1.7	Versión actualizada para el lanzamiento de la versión 2.6.0 de Greengrass nucleus.
2.1.6	Versión actualizada para el lanzamiento de la versión 2.5.0 de Greengrass nucleus.
2.1.5	Componente publicado en su totalidad. Regiones de AWS
2.1.4	Versión actualizada para la versión 2.4.0 de Greengrass Nucleus. Esta versión no está disponible en Europa (Londres) (`eu-west-2`).
2.1.3	Versión actualizada para el lanzamiento de la versión 2.3.0 de Greengrass nucleus.
2.1.2	Mejoras y correcciones de errores • Corrige un problema de escalado de la imagen que provocaba que los recuadros delimitadores no fueran precisos en los resultados de la inferencia de detección de objetos del DLR de muestra.

Versión	Cambios
2.1.1	Nuevas características • Utilice Deep Learning Runtime v1.6.0. • Añada compatibilidad con la detección de objetos de muestra en las plataformas Armv8 (AArch64). Esto amplía el soporte de aprendizaje automático para los dispositivos principales de Greengrass que ejecutan NVIDIA Jetson, como el Jetson Nano. • Habilite la integración de la cámara para la inferencia de muestras. Utilice el nuevo parámetro de `UseCamera` configuración para permitir que el código de inferencia de muestra acceda a la cámara del dispositivo principal de Greengrass y ejecute la inferencia localmente en la imagen capturada. • Agregue soporte para publicar los resultados de la inferencia en. Nube de AWS Utilice el nuevo parámetro `PublishResultsOnTopic` de configuración para especificar el tema sobre el que desea publicar los resultados. • Agregue el nuevo parámetro de `ImageDirectory` configuración que le permita especificar un directorio personalizado para la imagen en la que desee realizar la inferencia. Mejoras y correcciones de errores • Escriba los resultados de la inferencia en el archivo de registro del componente en lugar de en un archivo de inferencias independiente. • Utilice el módulo de registro del software AWS IoT Greengrass Core para registrar la salida de los componentes. • Utilice el SDK para dispositivos con AWS IoT para leer la configuración del componente y aplicar los cambios de configuración.
2.0.4	Versión inicial.

Tienda de modelos de clasificación de imágenes DLR

El almacén de modelos de clasificación de imágenes del DLR es un componente del modelo de aprendizaje automático que contiene ResNet -50 modelos previamente entrenados como artefactos

de Greengrass. Los modelos previamente entrenados que se utilizan en este componente se obtienen del GluonCV Model Zoo y se compilan con Neo Deep Learning Runtime. SageMaker

El componente de inferencia de clasificación de imágenes del DLR utiliza este componente como una dependencia de la fuente del modelo. Para utilizar un modelo DLR personalizado, cree una versión personalizada de este componente del modelo e incluya el modelo personalizado como un artefacto componente. Puede utilizar la receta de este componente como plantilla para crear componentes de modelo personalizados.

> **Note**
>
> El nombre del componente de la tienda de modelos de clasificación de imágenes del DLR varía en función de su versión. El nombre del componente para la versión 2.1.x y las versiones posteriores es. `variant.DLR.ImageClassification.ModelStore` El nombre del componente de la versión 2.0.x es. `variant.ImageClassification.ModelStore`

Temas

- Versiones
- Tipo
- Sistema operativo
- Requisitos
- Dependencias
- Configuración
- Archivo de registro local
- Registros de cambios

Versiones

Este componente tiene las siguientes versiones:

- 2.1.x () `variant.DLR.ImageClassification.ModelStore`
- 2.0.x () `variant.ImageClassification.ModelStore`

Tipo

Este componente es un componente genérico () `aws.greengrass.generic`. El núcleo de Greengrass ejecuta los scripts del ciclo de vida del componente.

Para obtener más información, consulte Tipos de componentes.

Sistema operativo

Este componente se puede instalar en los dispositivos principales que ejecutan los siguientes sistemas operativos:

- Linux
- Windows

Requisitos

Este componente tiene los siguientes requisitos:

- En los dispositivos principales de Greengrass que ejecutan Amazon Linux 2 o Ubuntu 18.04, se instala en el dispositivo la versión 2.27 o posterior de la Biblioteca C GNU (glibc).
- En los dispositivos ARMv7L, como Raspberry Pi, las dependencias para OpenCV-Python están instaladas en el dispositivo. Ejecute el siguiente comando para instalar las dependencias.

  ```
  sudo apt-get install libopenjp2-7 libilmbase23 libopenexr-dev libavcodec-dev
   libavformat-dev libswscale-dev libv4l-dev libgtk-3-0 libwebp-dev
  ```

- Los dispositivos Raspberry Pi que ejecutan el sistema operativo Bullseye de Raspberry Pi deben cumplir los siguientes requisitos:
  - NumPy 1.22.4 o una versión posterior instalada en el dispositivo. Raspberry Pi OS Bullseye incluye una versión anterior de NumPy, por lo que puede ejecutar el siguiente comando para actualizar NumPy el dispositivo.

    ```
    pip3 install --upgrade numpy
    ```

  - La pila de cámaras antigua habilitada en el dispositivo. El Raspberry Pi OS Bullseye incluye una nueva pila de cámaras que está habilitada de forma predeterminada y no es compatible, por lo que debes activar la pila de cámaras antigua.

Para activar la pila de cámaras antigua

1. Ejecute el siguiente comando para abrir la herramienta de configuración de Raspberry Pi.

   ```
   sudo raspi-config
   ```

2. Seleccione Opciones de interfaz.
3. Seleccione Cámara antigua para activar la pila de cámaras antiguas.
4. Reinicie el Raspberry Pi.

Dependencias

Al implementar un componente, AWS IoT Greengrass también despliega versiones compatibles de sus dependencias. Esto significa que debe cumplir los requisitos del componente y de todas sus dependencias para poder implementarlo correctamente. En esta sección se enumeran las dependencias de las versiones publicadas de este componente y las restricciones de las versiones semánticas que definen las versiones de los componentes para cada dependencia. También puede ver las dependencias de cada versión del componente en la consola.AWS IoT Greengrass En la página de detalles del componente, busque la lista de dependencias.

2.1.12 - 2.1.14

La siguiente tabla muestra las dependencias de las versiones 2.1.12 y 2.1.13 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.13.0	Flexible

2.1.11

La siguiente tabla muestra las dependencias de la versión 2.1.11 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.12.0	Flexible

2.1.10

La siguiente tabla muestra las dependencias de la versión 2.1.10 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.11.0	Flexible

2.1.9

La siguiente tabla muestra las dependencias de la versión 2.1.9 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.10.0	Flexible

2.1.8

La siguiente tabla muestra las dependencias de la versión 2.1.8 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.9.0	Flexible

2.1.7

La siguiente tabla muestra las dependencias de la versión 2.1.7 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.8.0	Flexible

2.1.6

La siguiente tabla muestra las dependencias de la versión 2.1.6 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.7.0	Flexible

2.1.5

La siguiente tabla muestra las dependencias de la versión 2.1.5 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.6.0	Flexible

2.1.4

La siguiente tabla muestra las dependencias de la versión 2.1.4 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.5.0	Flexible

2.1.3

La siguiente tabla muestra las dependencias de la versión 2.1.3 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.4.0	Flexible

2.1.2

La siguiente tabla muestra las dependencias de la versión 2.1.2 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.3.0	Flexible

2.1.1

La siguiente tabla muestra las dependencias de la versión 2.1.1 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.2.0	Flexible

2.0.x

La siguiente tabla muestra las dependencias de la versión 2.0.x de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	~2.0.0	Flexible

Configuración

Este componente no tiene ningún parámetro de configuración.

Archivo de registro local

Este componente no genera registros.

Registros de cambios

En la siguiente tabla se describen los cambios en cada versión del componente.

Versión	Cambios
2.1.13	Versión actualizada para la versión 2.12.5 de Greengrass nucleus.
2.1.12	Versión actualizada para el lanzamiento de la versión 2.12.0 de Greengrass nucleus.
2.1.11	Versión actualizada para el lanzamiento de la versión 2.11.0 de Greengrass nucleus.

Versión	Cambios
2.1.10	Versión actualizada para el lanzamiento de la versión 2.10.0 de Greengrass nucleus.
2.1.9	Versión actualizada para la versión 2.9.0 de Greengrass Nucleus.
2.1.8	Versión actualizada para el lanzamiento de la versión 2.8.0 de Greengrass nucleus.
2.1.7	Versión actualizada para el lanzamiento de la versión 2.7.0 de Greengrass nucleus.
2.1.6	Versión actualizada para el lanzamiento de la versión 2.6.0 de Greengrass Nucleus.
2.1.5	Nuevas características • Añade ejemplos de modelos de clasificación de imágenes para los dispositivos principales de Windows. • Versión actualizada para el lanzamiento de la versión 2.5.0 de Greengrass nucleus.
2.1.4	Versión actualizada para la versión 2.4.0 de Greengrass Nucleus.
2.1.3	Versión actualizada para el lanzamiento de la versión 2.3.0 de Greengrass nucleus.
2.1.2	Versión actualizada para el lanzamiento de la versión 2.2.0 de Greengrass nucleus.
2.1.1	Nuevas características • Agregue un ejemplo de modelo de clasificación de imágenes de ResNet -50 para las plataformas Armv8 (AArch64). Esto amplía el soporte de aprendizaje automático para los dispositivos principales de Greengrass que ejecutan NVIDIA Jetson, como el Jetson Nano.
2.0.4	Versión inicial.

Tienda de modelos de detección de objetos DLR

El almacén de modelos de detección de objetos del DLR es un componente del modelo de aprendizaje automático que contiene modelos YOLOv3 previamente entrenados como artefactos de Greengrass. Los modelos de muestra utilizados en este componente se obtienen del GluonCV Model Zoo y se compilan con Neo Deep Learning Runtime. SageMaker

El componente de inferencia de detección de objetos del DLR utiliza este componente como una dependencia de la fuente del modelo. Para utilizar un modelo DLR personalizado, cree una versión personalizada de este componente del modelo e incluya el modelo personalizado como un artefacto componente. Puede utilizar la receta de este componente como plantilla para crear componentes de modelo personalizados.

> **Note**
>
> El nombre del componente de tienda de modelos de detección de objetos del DLR varía en función de su versión. El nombre del componente para la versión 2.1.x y las versiones posteriores es. `variant.DLR.ObjectDetection.ModelStore` El nombre del componente de la versión 2.0.x es. `variant.ObjectDetection.ModelStore`

Temas

- Versiones
- Tipo
- Sistema operativo
- Requisitos
- Dependencias
- Configuración
- Archivo de registro local
- Registros de cambios

Versiones

Este componente tiene las siguientes versiones:

- 2.1.x

- 2.0.x

Tipo

Este componente es un componente genérico () `aws.greengrass.generic`. El núcleo de Greengrass ejecuta los scripts del ciclo de vida del componente.

Para obtener más información, consulte Tipos de componentes.

Sistema operativo

Este componente se puede instalar en los dispositivos principales que ejecutan los siguientes sistemas operativos:

- Linux
- Windows

Requisitos

Este componente tiene los siguientes requisitos:

- En los dispositivos principales de Greengrass que ejecutan Amazon Linux 2 o Ubuntu 18.04, se instala en el dispositivo la versión 2.27 o posterior de la Biblioteca C GNU (glibc).
- En los dispositivos ARMv7L, como Raspberry Pi, las dependencias para OpenCV-Python están instaladas en el dispositivo. Ejecute el siguiente comando para instalar las dependencias.

  ```
  sudo apt-get install libopenjp2-7 libilmbase23 libopenexr-dev libavcodec-dev
   libavformat-dev libswscale-dev libv4l-dev libgtk-3-0 libwebp-dev
  ```

- Los dispositivos Raspberry Pi que ejecutan el sistema operativo Bullseye de Raspberry Pi deben cumplir los siguientes requisitos:
  - NumPy 1.22.4 o una versión posterior instalada en el dispositivo. Raspberry Pi OS Bullseye incluye una versión anterior de NumPy, por lo que puede ejecutar el siguiente comando para actualizar NumPy el dispositivo.

    ```
    pip3 install --upgrade numpy
    ```

- La pila de cámaras antigua habilitada en el dispositivo. El Raspberry Pi OS Bullseye incluye una nueva pila de cámaras que está habilitada de forma predeterminada y no es compatible, por lo que debes activar la pila de cámaras antigua.

  Para activar la pila de cámaras antigua

  1. Ejecute el siguiente comando para abrir la herramienta de configuración de Raspberry Pi.

     ```
     sudo raspi-config
     ```

  2. Seleccione Opciones de interfaz.
  3. Seleccione Cámara antigua para activar la pila de cámaras antiguas.
  4. Reinicie el Raspberry Pi.

Dependencias

Al implementar un componente, AWS IoT Greengrass también despliega versiones compatibles de sus dependencias. Esto significa que debe cumplir los requisitos del componente y de todas sus dependencias para poder implementarlo correctamente. En esta sección se enumeran las dependencias de las versiones publicadas de este componente y las restricciones de las versiones semánticas que definen las versiones de los componentes para cada dependencia. También puede ver las dependencias de cada versión del componente en la consola.AWS IoT Greengrass En la página de detalles del componente, busque la lista de dependencias.

2.1.13 and 2.1.14

La siguiente tabla muestra las dependencias de las versiones 2.1.13 y 2.1.14 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.13.0	Flexible

2.1.12

La siguiente tabla muestra las dependencias de la versión 2.1.12 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.12.0	Flexible

2.1.11

La siguiente tabla muestra las dependencias de la versión 2.1.11 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.11.0	Flexible

2.1.10

La siguiente tabla muestra las dependencias de la versión 2.1.10 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.10.0	Flexible

2.1.9

La siguiente tabla muestra las dependencias de la versión 2.1.9 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.9.0	Flexible

2.1.8

La siguiente tabla muestra las dependencias de la versión 2.1.8 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.8.0	Flexible

2.1.7

La siguiente tabla muestra las dependencias de la versión 2.1.7 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.7.0	Flexible

2.1.5 and 2.1.6

La siguiente tabla muestra las dependencias de las versiones 2.1.5 y 2.1.6 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.6.0	Flexible

2.1.4

La siguiente tabla muestra las dependencias de la versión 2.1.4 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.5.0	Flexible

2.1.3

La siguiente tabla muestra las dependencias de la versión 2.1.3 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.4.0	Flexible

2.1.2

La siguiente tabla muestra las dependencias de la versión 2.1.2 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.3.0	Flexible

2.1.1

La siguiente tabla muestra las dependencias de la versión 2.1.1 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.2.0	Flexible

2.0.x

La siguiente tabla muestra las dependencias de la versión 2.0.x de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	~2.0.0	Flexible

Configuración

Este componente no tiene ningún parámetro de configuración.

Archivo de registro local

Este componente no genera registros.

Registros de cambios

En la siguiente tabla se describen los cambios en cada versión del componente.

Versión	Cambios
2.1.14	Versión actualizada para la versión 2.12.5 de Greengrass nucleus.
2.1.13	Versión actualizada para el lanzamiento de la versión 2.12.0 de Greengrass nucleus.

Versión	Cambios
2.1.12	Versión actualizada para el lanzamiento de la versión 2.11.0 de Greengrass nucleus.
2.1.11	Versión actualizada para el lanzamiento de la versión 2.10.0 de Greengrass nucleus.
2.1.10	Versión actualizada para la versión 2.9.0 de Greengrass Nucleus.
2.1.9	Versión actualizada para el lanzamiento de la versión 2.8.0 del núcleo de Greengrass.
2.1.8	Versión actualizada para el lanzamiento de la versión 2.7.0 de Greengrass nucleus.
2.1.7	Versión actualizada para la versión 2.6.0 de Greengrass Nucleus.
2.1.6	Añade un modelo de CPU para solucionar un problema en los dispositivos Armv8 (AArch64).
2.1.5	Nuevas características • Añade ejemplos de modelos de detección de objetos para los dispositivos principales de Windows. Mejoras y correcciones de errores • Versión actualizada para el lanzamiento de la versión 2.5.0 de Greengrass nucleus.
2.1.4	Versión actualizada para la versión 2.4.0 de Greengrass Nucleus.
2.1.3	Versión actualizada para el lanzamiento de la versión 2.3.0 de Greengrass nucleus.
2.1.2	Versión actualizada para el lanzamiento de la versión 2.2.0 de Greengrass nucleus.

Versión	Cambios
2.1.1	Nuevas características • Añada un ejemplo de modelo de detección de objetos YoloV3 para las plataformas Armv8 (AArch64). Esto amplía el soporte de aprendizaje automático para los dispositivos principales de Greengrass que ejecutan NVIDIA Jetson, como el Jetson Nano.
2.0.4	Versión inicial.

Tiempo de ejecución de DLR

El componente de tiempo de ejecución de DLR (`variant.DLR`) contiene un script que instala Deep Learning Runtime (DLR) y sus dependencias en un entorno virtual del dispositivo. Los Detección de objetos DLR componentes Clasificación de imágenes DLR y utilizan este componente como dependencia para instalar el DLR. La versión del componente 1.6.x instala el DLR v1.6.0 y la versión del componente 1.3.x instala el DLR v1.3.0.

Para usar un entorno de ejecución diferente, puede usar la receta de este componente como plantilla para crear un componente de aprendizaje automático personalizado.

Temas

- Versiones
- Tipo
- Sistema operativo
- Requisitos
- Dependencias
- Configuración
- Uso
- Archivo de registro local
- Registros de cambios

Versiones

Este componente tiene las siguientes versiones:

- 1.6.x
- 1.3.x

Tipo

Este componente es un componente genérico () `aws.greengrass.generic`. El núcleo de Greengrass ejecuta los scripts del ciclo de vida del componente.

Para obtener más información, consulte Tipos de componentes.

Sistema operativo

Este componente se puede instalar en los dispositivos principales que ejecutan los siguientes sistemas operativos:

- Linux
- Windows

Requisitos

Este componente tiene los siguientes requisitos:

- En los dispositivos principales de Greengrass que ejecutan Amazon Linux 2 o Ubuntu 18.04, se instala en el dispositivo la versión 2.27 o posterior de la Biblioteca C GNU (glibc).
- En los dispositivos ARMv7L, como Raspberry Pi, las dependencias para OpenCV-Python están instaladas en el dispositivo. Ejecute el siguiente comando para instalar las dependencias.

  ```
  sudo apt-get install libopenjp2-7 libilmbase23 libopenexr-dev libavcodec-dev
   libavformat-dev libswscale-dev libv4l-dev libgtk-3-0 libwebp-dev
  ```

- Los dispositivos Raspberry Pi que ejecutan el sistema operativo Bullseye de Raspberry Pi deben cumplir los siguientes requisitos:
  - NumPy 1.22.4 o una versión posterior instalada en el dispositivo. Raspberry Pi OS Bullseye incluye una versión anterior de NumPy, por lo que puede ejecutar el siguiente comando para actualizar NumPy el dispositivo.

    ```
    pip3 install --upgrade numpy
    ```

- La pila de cámaras antigua habilitada en el dispositivo. El Raspberry Pi OS Bullseye incluye una nueva pila de cámaras que está habilitada de forma predeterminada y no es compatible, por lo que debes activar la pila de cámaras antigua.

  Para activar la pila de cámaras antigua

  1. Ejecute el siguiente comando para abrir la herramienta de configuración de Raspberry Pi.

     ```
     sudo raspi-config
     ```

  2. Seleccione Opciones de interfaz.
  3. Seleccione Cámara antigua para activar la pila de cámaras antiguas.
  4. Reinicie el Raspberry Pi.

Puntos finales y puertos

De forma predeterminada, este componente utiliza un script de instalación para instalar los paquetes mediante los `pip` comandos `apt` `yum``brew`,, y, según la plataforma que utilice el dispositivo principal. Este componente debe poder realizar solicitudes salientes a varios índices y repositorios de paquetes para ejecutar el script de instalación. Para permitir que el tráfico saliente de este componente pase por un proxy o firewall, debe identificar los puntos finales de los índices y repositorios de paquetes a los que se conecta el dispositivo principal para realizar la instalación.

Tenga en cuenta lo siguiente al identificar los puntos finales necesarios para el script de instalación de este componente:

- Los puntos finales dependen de la plataforma del dispositivo principal. Por ejemplo, un dispositivo central que ejecuta Ubuntu usa en `apt` lugar de `yum` o`brew`. Además, los dispositivos que usan el mismo índice de paquetes pueden tener listas de fuentes diferentes, por lo que pueden recuperar paquetes de diferentes repositorios.
- Los puntos finales pueden diferir entre varios dispositivos que utilizan el mismo índice de paquetes, ya que cada dispositivo tiene sus propias listas de fuentes que definen dónde recuperar los paquetes.
- Los puntos finales pueden cambiar con el tiempo. Cada índice de paquetes proporciona las direcciones URL de los repositorios en los que se descargan los paquetes, y el propietario de un paquete puede cambiar las direcciones URL que proporciona el índice de paquetes.

Para obtener más información sobre las dependencias que instala este componente y sobre cómo deshabilitar el script del instalador, consulte el parámetro de configuración. UseInstaller

Para obtener más información sobre los puntos finales y los puertos necesarios para el funcionamiento básico, consulte. Permitir el tráfico del dispositivo a través de un proxy o firewall

Dependencias

Al implementar un componente, AWS IoT Greengrass también despliega versiones compatibles de sus dependencias. Esto significa que debe cumplir los requisitos del componente y de todas sus dependencias para poder implementarlo correctamente. En esta sección se enumeran las dependencias de las versiones publicadas de este componente y las restricciones de las versiones semánticas que definen las versiones de los componentes para cada dependencia. También puede ver las dependencias de cada versión del componente en la consola.AWS IoT Greengrass En la página de detalles del componente, busque la lista de dependencias.

1.6.11 - 1.6.16

La siguiente tabla muestra las dependencias de las versiones 1.6.11 a 1.6.16 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <3.0.0	Flexible

1.6.10

La siguiente tabla muestra las dependencias de la versión 1.6.10 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.9.0	Flexible

1.6.9

La siguiente tabla muestra las dependencias de la versión 1.6.9 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.8.0	Flexible

1.6.8

La siguiente tabla muestra las dependencias de la versión 1.6.8 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.7.0	Flexible

1.6.6 and 1.6.7

La siguiente tabla muestra las dependencias de las versiones 1.6.6 y 1.6.7 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.6.0	Flexible

1.6.4 and 1.6.5

La siguiente tabla muestra las dependencias de las versiones 1.6.4 y 1.6.5 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.5.0	Flexible

1.6.3

La siguiente tabla muestra las dependencias de la versión 1.6.3 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.4.0	Flexible

1.6.2

La siguiente tabla muestra las dependencias de la versión 1.6.2 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.3.0	Flexible

1.6.1

La siguiente tabla muestra las dependencias de la versión 1.6.1 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.2.0	Flexible

1.3.x

La siguiente tabla muestra las dependencias de la versión 1.3.x de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	~2.0.0	Flexible

Para obtener más información sobre las dependencias de los componentes, consulta la referencia de recetas de componentes.

Configuración

Este componente proporciona los siguientes parámetros de configuración que puede personalizar al implementar el componente.

`MLRootPath`

(Opcional) La ruta de la carpeta en los dispositivos principales de Linux donde los componentes de inferencia leen las imágenes y escriben los resultados de la inferencia. Puede modificar este

valor en cualquier ubicación del dispositivo a la que el usuario que ejecuta este componente tenga acceso de lectura y escritura.

Valor predeterminado: *`/greengrass/v2`*`/work/variant.DLR/greengrass_ml`

`WindowsMLRootPath`

Esta función está disponible en la versión 1.6.6 y versiones posteriores de este componente.

(Opcional) La ruta de la carpeta del dispositivo principal de Windows donde los componentes de inferencia leen las imágenes y escriben los resultados de la inferencia. Puede modificar este valor en cualquier ubicación del dispositivo a la que el usuario que ejecuta este componente tenga acceso de lectura y escritura.

Valor predeterminado: *`C:\greengrass\v2`*`\\work\\variant.DLR\\greengrass_ml`

`UseInstaller`

(Opcional) Valor de cadena que define si se debe utilizar el script de instalación de este componente para instalar el DLR y sus dependencias. Los valores admitidos son `true` y `false`.

Establezca este valor `false` si desea utilizar un script personalizado para la instalación del DLR o si desea incluir las dependencias del tiempo de ejecución en una imagen de Linux prediseñada. Para usar este componente con los componentes AWS de inferencia de DLR proporcionados, instale las siguientes bibliotecas, incluidas las dependencias, y póngalas `ggc_user` a disposición del usuario del sistema, por ejemplo, el que ejecuta los componentes de ML.

- Python 3.7 o posterior, incluida `pip` la versión de Python.
- Deep Learning Runtime v1.6.0
- NumPy.
- OpenCV-Python.
- SDK para dispositivos con AWS IoT v2 para Python.
- AWS Python en tiempo de ejecución común (CRT).
- Picamera (solo para dispositivos Raspberry Pi).
- `awscam`módulo (para AWS DeepLens dispositivos).
- libGL (para dispositivos Linux)

Valor predeterminado: `true`

Uso

Utilice este componente con el parámetro `UseInstaller` de configuración establecido en `true` para instalar el DLR y sus dependencias en el dispositivo. El componente configura un entorno virtual en el dispositivo que incluye el OpenCV NumPy y las bibliotecas necesarias para el DLR.

> **Note**
>
> El script de instalación de este componente también instala las versiones más recientes de las bibliotecas del sistema adicionales necesarias para configurar el entorno virtual en el dispositivo y utilizar el marco de aprendizaje automático instalado. Esto podría actualizar las bibliotecas del sistema existentes en el dispositivo. Consulte la siguiente tabla para ver la lista de bibliotecas que instala este componente para cada sistema operativo compatible. Si desea personalizar este proceso de instalación, defina `false` el parámetro de `UseInstaller` configuración en y desarrolle su propio script de instalación.

Plataforma	Bibliotecas instaladas en el sistema del dispositivo	Bibliotecas instaladas en el entorno virtual
Armv7l	`build-essential` , `cmake`, `ca-certificates` , `git`	`setuptools` , `wheel`
Amazon Linux 2	`mesa-libGL`	Ninguna
Ubuntu	`wget`	Ninguna

Al implementar el componente de inferencia, este componente de tiempo de ejecución comprueba primero si el dispositivo ya tiene instalado el DLR y sus dependencias y, de no ser así, los instala automáticamente.

Archivo de registro local

Este componente utiliza el siguiente archivo de registro.

Linux

```
/greengrass/v2/logs/variant.DLR.log
```

Windows

```
C:\greengrass\v2\logs\variant.DLR.log
```

Para ver los registros de este componente

- Ejecute el siguiente comando en el dispositivo principal para ver el archivo de registro de este componente en tiempo real. Sustituya */greengrass/v2* o *C:\greengrass\v2* por la ruta a la carpeta AWS IoT Greengrass raíz.

  Linux

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

  Windows (PowerShell)

  ```
  Get-Content C:\greengrass\v2\logs\variant.DLR.log -Tail 10 -Wait
  ```

Registros de cambios

En la siguiente tabla se describen los cambios en cada versión del componente.

Versión	Cambios
1.6.16	Versión actualizada para la versión 2.12.5 del núcleo de Greengrass.
1.6.12	Mejoras y correcciones de errores • Corrige el script de instalación para los usuarios del sistema operativo Windows.
1.6.11	Versión actualizada para la versión 2.9.0 de Greengrass Nucleus.
1.6.10	Versión actualizada para el lanzamiento de la versión 2.8.0 del núcleo de Greengrass.
1.6.9	Versión actualizada para el lanzamiento de la versión 2.7.0 de Greengrass nucleus.

Versión	Cambios
1.6.8	Versión actualizada para la versión 2.6.0 de Greengrass Nucleus.
1.6.7	Mejoras y correcciones de errores • Actualiza el script de `UseInstaller` instalación para instalar LibGL, que no está disponible de forma predeterminada en determinadas plataformas Linux. • Actualiza el script de `UseInstaller` instalación para usar siempre Python 3.9 en el entorno virtual de este componente. Este cambio ayuda a garantizar la compatibilidad con otras bibliotecas.
1.6.6	Nuevas características • Añade compatibilidad con los dispositivos principales que ejecutan Windows. • Agrega el nuevo parámetro `WindowsMLRootPath` de configuración que puede usar para configurar la carpeta de resultados de inferencias en los dispositivos principales de Windows.
1.6.5	Nuevas características • Agrega el nuevo parámetro `UseInstaller` de configuración que puede usar para deshabilitar el script de instalación en este component e.
1.6.4	Versión actualizada para la versión 2.4.0 de Greengrass Nucleus.
1.6.3	Versión actualizada para el lanzamiento de la versión 2.3.0 de Greengrass nucleus.
1.6.2	Versión actualizada para el lanzamiento de la versión 2.2.0 de Greengrass nucleus.

Versión	Cambios
1.6.1	Nuevas características • Instale Deep Learning Runtime v1.6.0 y sus dependencias. • Añada soporte para instalar DLR en plataformas Armv8 (AArch64). Esto amplía el soporte de aprendizaje automático para los dispositivos principales de Greengrass que ejecutan NVIDIA Jetson, como el Jetson Nano. Mejoras y correcciones de errores • Instálelo SDK para dispositivos con AWS IoT en el entorno virtual para leer la configuración del componente y aplicar los cambios de configura ción. • Mejoras y correcciones de errores menores adicionales.
1.3.2	Versión inicial. Instala DLR v1.3.0.

TensorFlow Clasificación de imágenes Lite

El componente de clasificación de imágenes de TensorFlow Lite (`aws.greengrass.TensorFlowLiteImageClassification`) contiene un ejemplo de código de inferencia para realizar inferencias de clasificación de imágenes mediante el tiempo de ejecución de TensorFlow Lite y un ejemplo de modelo cuantificado MobileNet 1.0 previamente entrenado. Este componente utiliza la variante TensorFlow Tienda de modelos de clasificación de imágenes Lite y los TensorFlow Tiempo de ejecución Lite componentes como dependencias para descargar el motor de ejecución de TensorFlow Lite y el modelo de muestra.

Para usar este componente de inferencia con un modelo TensorFlow Lite personalizado, cree una versión personalizada del componente de la tienda de modelos dependiente. Para usar su propio código de inferencia personalizado, puede usar la receta de este componente como plantilla para crear un componente de inferencia personalizado.

Temas

- Versiones
- Tipo
- Sistema operativo

- Requisitos
- Dependencias
- Configuración
- Archivo de registro local
- Registros de cambios

Versiones

Este componente tiene las siguientes versiones:

- 2.1.x

Tipo

Este componente es un componente genérico () `aws.greengrass.generic`. El núcleo de Greengrass ejecuta los scripts del ciclo de vida del componente.

Para obtener más información, consulte Tipos de componentes.

Sistema operativo

Este componente se puede instalar en los dispositivos principales que ejecutan los siguientes sistemas operativos:

- Linux
- Windows

Requisitos

Este componente tiene los siguientes requisitos:

- En los dispositivos principales de Greengrass que ejecutan Amazon Linux 2 o Ubuntu 18.04, se instala en el dispositivo la versión 2.27 o posterior de la Biblioteca C GNU (glibc).
- En los dispositivos ARMv7L, como Raspberry Pi, las dependencias para OpenCV-Python están instaladas en el dispositivo. Ejecute el siguiente comando para instalar las dependencias.

```
sudo apt-get install libopenjp2-7 libilmbase23 libopenexr-dev libavcodec-dev
 libavformat-dev libswscale-dev libv4l-dev libgtk-3-0 libwebp-dev
```

- Los dispositivos Raspberry Pi que ejecutan el sistema operativo Bullseye de Raspberry Pi deben cumplir los siguientes requisitos:
  - NumPy 1.22.4 o una versión posterior instalada en el dispositivo. Raspberry Pi OS Bullseye incluye una versión anterior de NumPy, por lo que puede ejecutar el siguiente comando para actualizar NumPy el dispositivo.

    ```
    pip3 install --upgrade numpy
    ```

  - La pila de cámaras antigua habilitada en el dispositivo. El Raspberry Pi OS Bullseye incluye una nueva pila de cámaras que está habilitada de forma predeterminada y no es compatible, por lo que debes activar la pila de cámaras antigua.

    Para activar la pila de cámaras antigua

    1. Ejecute el siguiente comando para abrir la herramienta de configuración de Raspberry Pi.

       ```
       sudo raspi-config
       ```

    2. Seleccione Opciones de interfaz.
    3. Seleccione Cámara antigua para activar la pila de cámaras antiguas.
    4. Reinicie el Raspberry Pi.

Dependencias

Al implementar un componente, AWS IoT Greengrass también despliega versiones compatibles de sus dependencias. Esto significa que debe cumplir los requisitos del componente y de todas sus dependencias para poder implementarlo correctamente. En esta sección se enumeran las dependencias de las versiones publicadas de este componente y las restricciones de las versiones semánticas que definen las versiones de los componentes para cada dependencia. También puede ver las dependencias de cada versión del componente en la consola.AWS IoT Greengrass En la página de detalles del componente, busque la lista de dependencias.

2.1.11 and 2.1.12

La siguiente tabla muestra las dependencias de las versiones 2.1.11 y 2.1.12 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.13.0	Flexible
TensorFlow Tienda de modelos de clasificación de imágenes Lite	>=2.1.0 <2.2.0	Rígido
TensorFlow Lite	>=2.5.0 <2.6.0	Rígido

2.1.10

La siguiente tabla muestra las dependencias de la versión 2.1.10 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.12.0	Flexible
TensorFlow Tienda de modelos de clasificación de imágenes Lite	>=2.1.0 <2.2.0	Rígido
TensorFlow Lite	>=2.5.0 <2.6.0	Rígido

2.1.9

La siguiente tabla muestra las dependencias de la versión 2.1.9 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.11.0	Flexible
TensorFlow Tienda de modelos de clasificación de imágenes Lite	>=2.1.0 <2.2.0	Rígido
TensorFlow Lite	>=2.5.0 <2.6.0	Rígido

2.1.8

La siguiente tabla muestra las dependencias de la versión 2.1.8 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.10.0	Flexible
TensorFlow Tienda de modelos de clasificación de imágenes Lite	>=2.1.0 <2.2.0	Rígido
TensorFlow Lite	>=2.5.0 <2.6.0	Rígido

2.1.7

La siguiente tabla muestra las dependencias de la versión 2.1.7 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.9.0	Flexible
TensorFlow Tienda de modelos de clasificación de imágenes Lite	>=2.1.0 <2.2.0	Rígido
TensorFlow Lite	>=2.5.0 <2.6.0	Rígido

2.1.6

La siguiente tabla muestra las dependencias de la versión 2.1.6 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.8.0	Flexible

Dependencia	Versiones compatibles	Tipo de dependencia
TensorFlow Tienda de modelos de clasificación de imágenes Lite	>=2.1.0 <2.2.0	Rígido
TensorFlow Lite	>=2.5.0 <2.6.0	Rígido

2.1.5

La siguiente tabla muestra las dependencias de la versión 2.1.5 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.7.0	Flexible
TensorFlow Tienda de modelos de clasificación de imágenes Lite	>=2.1.0 <2.2.0	Rígido
TensorFlow Lite	>=2.5.0 <2.6.0	Rígido

2.1.4

La siguiente tabla muestra las dependencias de la versión 2.1.4 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.6.0	Flexible
TensorFlow Tienda de modelos de clasificación de imágenes Lite	>=2.1.0 <2.2.0	Rígido
TensorFlow Lite	>=2.5.0 <2.6.0	Rígido

2.1.3

La siguiente tabla muestra las dependencias de la versión 2.1.3 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.5.0	Flexible
TensorFlow Tienda de modelos de clasificación de imágenes Lite	>=2.1.0 <2.2.0	Rígido
TensorFlow Lite	>=2.5.0 <2.6.0	Rígido

2.1.2

La siguiente tabla muestra las dependencias de la versión 2.1.2 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.4.0	Flexible
TensorFlow Tienda de modelos de clasificación de imágenes Lite	>=2.1.0 <2.2.0	Rígido
TensorFlow Lite	>=2.5.0 <2.6.0	Rígido

2.1.1

La siguiente tabla muestra las dependencias de la versión 2.1.1 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.3.0	Flexible

Dependencia	Versiones compatibles	Tipo de dependencia
TensorFlow Tienda de modelos de clasificación de imágenes Lite	>=2.1.0 <2.2.0	Rígido
TensorFlow Lite	>=2.5.0 <2.6.0	Rígido

2.1.0

La siguiente tabla muestra las dependencias de la versión 2.1.0 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.2.0	Flexible
TensorFlow Tienda de modelos de clasificación de imágenes Lite	>=2.1.0 <2.2.0	Rígido
TensorFlow Lite	>=2.5.0 <2.6.0	Rígido

Configuración

Este componente proporciona los siguientes parámetros de configuración que puede personalizar al implementar el componente.

`accessControl`

(Opcional) El objeto que contiene la política de autorización que permite al componente publicar mensajes en el tema de notificaciones predeterminado.

Predeterminado:

```
{
   "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`

(Opcional) El tema sobre el que desea publicar los resultados de la inferencia. Si modifica este valor, también debe modificar el valor del `resources` `accessControl` parámetro para que coincida con el nombre del tema personalizado.

Valor predeterminado: `ml/tflite/image-classification`

`Accelerator`

El acelerador que quieres usar. Los valores admitidos son `cpu` y `gpu`.

Los modelos de muestra del componente del modelo dependiente solo admiten la aceleración de la CPU. Para usar la aceleración de la GPU con un modelo personalizado diferente, cree un componente de modelo personalizado para anular el componente del modelo público.

Valor predeterminado: `cpu`

`ImageDirectory`

(Opcional) La ruta de la carpeta del dispositivo donde los componentes de inferencia leen las imágenes. Puede modificar este valor en cualquier ubicación del dispositivo a la que tenga acceso de lectura y escritura.

Valor predeterminado: */greengrass/v2*`/packages/artifacts-unarchived/`*component-name*`/image_classification/sample_images/`

> **Note**
>
> Si establece el valor de `true`, `UseCamera` se ignora este parámetro de configuración.

`ImageName`

(Opcional) El nombre de la imagen que el componente de inferencia utiliza como entrada para realizar una predicción. El componente busca la imagen en la carpeta especificada en`ImageDirectory`. De forma predeterminada, el componente usa la imagen de muestra en el directorio de imágenes predeterminado. AWS IoT Greengrass admite los siguientes formatos de imagen: `jpegjpg`,`png`, y`npy`.

Valor predeterminado: `cat.jpeg`

> **Note**
>
> Si establece el valor de en`true`, se `UseCamera` ignora este parámetro de configuración.

`InferenceInterval`

(Opcional) El tiempo en segundos entre cada predicción realizada por el código de inferencia. El código de inferencia de muestra se ejecuta indefinidamente y repite sus predicciones en el intervalo de tiempo especificado. Por ejemplo, puede cambiarlo por un intervalo más corto si desea utilizar imágenes tomadas por una cámara para realizar predicciones en tiempo real.

Valor predeterminado: `3600`

`ModelResourceKey`

(Opcional) Los modelos que se utilizan en el componente de modelo público dependiente. Modifique este parámetro solo si anula el componente del modelo público por un componente personalizado.

Predeterminado:

```
{
    "model": "TensorFlowLite-Mobilenet"
}
```

`UseCamera`

(Opcional) Valor de cadena que define si se deben utilizar imágenes de una cámara conectada al dispositivo principal de Greengrass. Los valores admitidos son `true` y `false`.

Si establece este valor en `true`, el código de inferencia de muestra accede a la cámara del dispositivo y ejecuta la inferencia localmente en la imagen capturada. Se ignoran los valores de los `ImageDirectory` parámetros `ImageName` y. Asegúrese de que el usuario que ejecuta este componente tenga acceso de lectura y escritura a la ubicación en la que la cámara almacena las imágenes capturadas.

Valor predeterminado: `false`

> **Note**
>
> Al ver la receta de este componente, el parámetro de `UseCamera` configuración no aparece en la configuración predeterminada. Sin embargo, puede modificar el valor de este parámetro en una actualización de la combinación de configuraciones al implementar el componente.
> Si lo establece `UseCameratrue`, también debe crear un enlace simbólico para permitir que el componente de inferencia acceda a la cámara desde el entorno virtual creado por el componente de tiempo de ejecución. Para obtener más información sobre el uso de una cámara con los componentes de inferencia de muestra, consulte. Actualizar las configuraciones de los componentes

Archivo de registro local

Este componente utiliza el siguiente archivo de registro.

Linux

```
/greengrass/v2/logs/aws.greengrass.TensorFlowLiteImageClassification.log
```

Windows

```
C:\greengrass\v2\logs\aws.greengrass.TensorFlowLiteImageClassification.log
```

Para ver los registros de este componente

- Ejecute el siguiente comando en el dispositivo principal para ver el archivo de registro de este componente en tiempo real. Sustituya */greengrass/v2* o *C:\greengrass\v2* por la ruta a la carpeta AWS IoT Greengrass raíz.

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
```

Registros de cambios

En la siguiente tabla se describen los cambios en cada versión del componente.

Versión	Cambios
2.1.12	Versión actualizada para la versión 2.12.5 de Greengrass nucleus.
2.1.11	Versión actualizada para la versión 2.12.0 de Greengrass nucleus.
2.1.10	Versión actualizada para la versión 2.11.0 de Greengrass nucleus.
2.1.9	Versión actualizada para la versión 2.10.0 de Greengrass nucleus.
2.1.8	Versión actualizada para la versión 2.9.0 de Greengrass Nucleus.
2.1.7	Versión actualizada para el lanzamiento de la versión 2.8.0 de Greengrass nucleus.
2.1.6	Versión actualizada para el lanzamiento de la versión 2.7.0 de Greengrass nucleus.
2.1.5	Versión actualizada para la versión 2.6.0 de Greengrass Nucleus.
2.1.4	Versión actualizada para el lanzamiento de la versión 2.5.0 de Greengrass nucleus.
2.1.3	Versión actualizada para la versión 2.4.0 de Greengrass Nucleus.

Versión	Cambios
2.1.2	Versión actualizada para el lanzamiento de la versión 2.3.0 de Greengrass nucleus.
2.1.1	Versión actualizada para el lanzamiento de la versión 2.2.0 de Greengrass nucleus.
2.1.0	Versión inicial.

TensorFlow Detección de objetos Lite

El componente de detección de objetos TensorFlow Lite (`aws.greengrass.TensorFlowLiteObjectDetection`) contiene un ejemplo de código de inferencia para realizar inferencias de detección de objetos con TensorFlow Lite y un ejemplo del modelo de detección de un solo disparo (SSD) MobileNet 1.0 previamente entrenado. Este componente utiliza la variante TensorFlow Tienda de modelos de detección de objetos Lite y los TensorFlow Tiempo de ejecución Lite componentes como dependencias para descargar TensorFlow Lite y el modelo de muestra.

Para usar este componente de inferencia con un modelo TensorFlow Lite personalizado, puede crear una versión personalizada del componente de la tienda de modelos dependiente. Para utilizar su propio código de inferencia personalizado, utilice la receta de este componente como plantilla para crear un componente de inferencia personalizado.

Temas

- Versiones
- Tipo
- Sistema operativo
- Requisitos
- Dependencias
- Configuración
- Archivo de registro local
- Registros de cambios

Versiones

Este componente tiene las siguientes versiones:

- 2.1.x

Tipo

Este componente es un componente genérico () `aws.greengrass.generic`. El núcleo de Greengrass ejecuta los scripts del ciclo de vida del componente.

Para obtener más información, consulte Tipos de componentes.

Sistema operativo

Este componente se puede instalar en los dispositivos principales que ejecutan los siguientes sistemas operativos:

- Linux
- Windows

Requisitos

Este componente tiene los siguientes requisitos:

- En los dispositivos principales de Greengrass que ejecutan Amazon Linux 2 o Ubuntu 18.04, se instala en el dispositivo la versión 2.27 o posterior de la Biblioteca C GNU (glibc).
- En los dispositivos ARMv7L, como Raspberry Pi, las dependencias para OpenCV-Python están instaladas en el dispositivo. Ejecute el siguiente comando para instalar las dependencias.

  ```
  sudo apt-get install libopenjp2-7 libilmbase23 libopenexr-dev libavcodec-dev
   libavformat-dev libswscale-dev libv4l-dev libgtk-3-0 libwebp-dev
  ```

- Los dispositivos Raspberry Pi que ejecutan el sistema operativo Bullseye de Raspberry Pi deben cumplir los siguientes requisitos:
  - NumPy 1.22.4 o una versión posterior instalada en el dispositivo. Raspberry Pi OS Bullseye incluye una versión anterior de NumPy, por lo que puede ejecutar el siguiente comando para actualizar NumPy el dispositivo.

```
pip3 install --upgrade numpy
```

- La pila de cámaras antigua habilitada en el dispositivo. El Raspberry Pi OS Bullseye incluye una nueva pila de cámaras que está habilitada de forma predeterminada y no es compatible, por lo que debes activar la pila de cámaras antigua.

  Para activar la pila de cámaras antigua

  1. Ejecute el siguiente comando para abrir la herramienta de configuración de Raspberry Pi.

     ```
     sudo raspi-config
     ```

  2. Seleccione Opciones de interfaz.
  3. Seleccione Cámara antigua para activar la pila de cámaras antiguas.
  4. Reinicie el Raspberry Pi.

Dependencias

Al implementar un componente, AWS IoT Greengrass también despliega versiones compatibles de sus dependencias. Esto significa que debe cumplir los requisitos del componente y de todas sus dependencias para poder implementarlo correctamente. En esta sección se enumeran las dependencias de las versiones publicadas de este componente y las restricciones de las versiones semánticas que definen las versiones de los componentes para cada dependencia. También puede ver las dependencias de cada versión del componente en la consola.AWS IoT Greengrass En la página de detalles del componente, busque la lista de dependencias.

2.1.11 and 2.1.12

La siguiente tabla muestra las dependencias de las versiones 2.1.11 y 2.1.12 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.13.0	Flexible
TensorFlow Tienda de modelos de clasificación de imágenes Lite	>=2.1.0 <2.2.0	Rígido

Dependencia	Versiones compatibles	Tipo de dependencia
TensorFlow Lite	>=2.5.0 <2.6.0	Rígido

2.1.10

La siguiente tabla muestra las dependencias de la versión 2.1.10 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.12.0	Flexible
TensorFlow Tienda de modelos de clasificación de imágenes Lite	>=2.1.0 <2.2.0	Rígido
TensorFlow Lite	>=2.5.0 <2.6.0	Rígido

2.1.9

La siguiente tabla muestra las dependencias de la versión 2.1.9 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.11.0	Flexible
TensorFlow Tienda de modelos de clasificación de imágenes Lite	>=2.1.0 <2.2.0	Rígido
TensorFlow Lite	>=2.5.0 <2.6.0	Rígido

2.1.8

La siguiente tabla muestra las dependencias de la versión 2.1.8 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.10.0	Flexible
TensorFlow Tienda de modelos de clasificación de imágenes Lite	>=2.1.0 <2.2.0	Rígido
TensorFlow Lite	>=2.5.0 <2.6.0	Rígido

2.1.7

La siguiente tabla muestra las dependencias de la versión 2.1.7 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.9.0	Flexible
TensorFlow Tienda de modelos de clasificación de imágenes Lite	>=2.1.0 <2.2.0	Rígido
TensorFlow Lite	>=2.5.0 <2.6.0	Rígido

2.1.6

La siguiente tabla muestra las dependencias de la versión 2.1.6 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.8.0	Flexible
TensorFlow Tienda de modelos de clasificación de imágenes Lite	>=2.1.0 <2.2.0	Rígido
TensorFlow Lite	>=2.5.0 <2.6.0	Rígido

2.1.5

La siguiente tabla muestra las dependencias de la versión 2.1.5 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.7.0	Flexible
TensorFlow Tienda de modelos de clasificación de imágenes Lite	>=2.1.0 <2.2.0	Rígido
TensorFlow Lite	>=2.5.0 <2.6.0	Rígido

2.1.4

La siguiente tabla muestra las dependencias de la versión 2.1.4 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.6.0	Flexible
TensorFlow Tienda de modelos de clasificación de imágenes Lite	>=2.1.0 <2.2.0	Rígido
TensorFlow Lite	>=2.5.0 <2.6.0	Rígido

2.1.3

La siguiente tabla muestra las dependencias de la versión 2.1.3 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.5.0	Flexible

Dependencia	Versiones compatibles	Tipo de dependencia
TensorFlow Tienda de modelos de clasificación de imágenes Lite	>=2.1.0 <2.2.0	Rígido
TensorFlow Lite	>=2.5.0 <2.6.0	Rígido

2.1.2

La siguiente tabla muestra las dependencias de la versión 2.1.2 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.4.0	Flexible
TensorFlow Tienda de modelos de clasificación de imágenes Lite	>=2.1.0 <2.2.0	Rígido
TensorFlow Lite	>=2.5.0 <2.6.0	Rígido

2.1.1

La siguiente tabla muestra las dependencias de la versión 2.1.1 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.3.0	Flexible
TensorFlow Tienda de modelos de clasificación de imágenes Lite	>=2.1.0 <2.2.0	Rígido
TensorFlow Lite	>=2.5.0 <2.6.0	Rígido

2.1.0

La siguiente tabla muestra las dependencias de la versión 2.1.0 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.2.0	Flexible
TensorFlow Tienda de modelos de clasificación de imágenes Lite	>=2.1.0 <2.2.0	Rígido
TensorFlow Lite	>=2.5.0 <2.6.0	Rígido

Configuración

Este componente proporciona los siguientes parámetros de configuración que puede personalizar al implementar el componente.

`accessControl`

(Opcional) El objeto que contiene la política de autorización que permite al componente publicar mensajes en el tema de notificaciones predeterminado.

Predeterminado:

```
{
   "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`

(Opcional) El tema sobre el que desea publicar los resultados de la inferencia. Si modifica este valor, también debe modificar el valor del `resources` `accessControl` parámetro para que coincida con el nombre del tema personalizado.

Valor predeterminado: `ml/tflite/object-detection`

`Accelerator`

El acelerador que quieres usar. Los valores admitidos son `cpu` y `gpu`.

Los modelos de muestra del componente del modelo dependiente solo admiten la aceleración de la CPU. Para usar la aceleración de la GPU con un modelo personalizado diferente, cree un componente de modelo personalizado para anular el componente del modelo público.

Valor predeterminado: `cpu`

`ImageDirectory`

(Opcional) La ruta de la carpeta del dispositivo donde los componentes de inferencia leen las imágenes. Puede modificar este valor en cualquier ubicación del dispositivo a la que tenga acceso de lectura y escritura.

Valor predeterminado: */greengrass/v2*`/packages/artifacts-unarchived/`*component-name*`/object_detection/sample_images/`

> **Note**
>
> Si establece el valor de `true`, `UseCamera` se ignora este parámetro de configuración.

`ImageName`

(Opcional) El nombre de la imagen que el componente de inferencia utiliza como entrada para realizar una predicción. El componente busca la imagen en la carpeta especificada en`ImageDirectory`. De forma predeterminada, el componente usa la imagen de muestra en el directorio de imágenes predeterminado. AWS IoT Greengrass admite los siguientes formatos de imagen: `jpeg`jpg,png, ynpy.

Valor predeterminado: `objects.jpg`

> **Note**
>
> Si establece el valor de `entrue`, se `UseCamera` ignora este parámetro de configuración.

`InferenceInterval`

(Opcional) El tiempo en segundos entre cada predicción realizada por el código de inferencia. El código de inferencia de muestra se ejecuta indefinidamente y repite sus predicciones en el intervalo de tiempo especificado. Por ejemplo, puede cambiarlo por un intervalo más corto si desea utilizar imágenes tomadas por una cámara para realizar predicciones en tiempo real.

Valor predeterminado: `3600`

`ModelResourceKey`

(Opcional) Los modelos que se utilizan en el componente de modelo público dependiente. Modifique este parámetro solo si anula el componente del modelo público por un componente personalizado.

Predeterminado:

```
{
    "model": "TensorFlowLite-SSD"
}
```

`UseCamera`

(Opcional) Valor de cadena que define si se deben utilizar imágenes de una cámara conectada al dispositivo principal de Greengrass. Los valores admitidos son `true` y `false`.

Si establece este valor en`true`, el código de inferencia de muestra accede a la cámara del dispositivo y ejecuta la inferencia localmente en la imagen capturada. Se ignoran los valores de los `ImageDirectory` parámetros `ImageName` y. Asegúrese de que el usuario que ejecuta este componente tenga acceso de lectura y escritura a la ubicación en la que la cámara almacena las imágenes capturadas.

Valor predeterminado: `false`

> **Note**
>
> Al ver la receta de este componente, el parámetro de `UseCamera` configuración no aparece en la configuración predeterminada. Sin embargo, puede modificar el valor de este parámetro en una actualización de la combinación de configuraciones al implementar el componente.
> Si lo establece `UseCameratrue`, también debe crear un enlace simbólico para permitir que el componente de inferencia acceda a la cámara desde el entorno virtual creado por el componente de tiempo de ejecución. Para obtener más información sobre el uso de una cámara con los componentes de inferencia de muestra, consulte. Actualizar las configuraciones de los componentes

> **Note**
>
> Al ver la receta de este componente, el parámetro `UseCamera` de configuración no aparece en la configuración predeterminada. Sin embargo, puede modificar el valor de este parámetro en una actualización de la combinación de configuraciones al implementar el componente.
> Si lo establece `UseCameratrue`, también debe crear un enlace simbólico para permitir que el componente de inferencia acceda a la cámara desde el entorno virtual creado por el componente de tiempo de ejecución. Para obtener más información sobre el uso de una cámara con los componentes de inferencia de muestra, consulte. Actualizar las configuraciones de los componentes

Archivo de registro local

Este componente utiliza el siguiente archivo de registro.

Linux

```
/greengrass/v2/logs/aws.greengrass.TensorFlowLiteObjectDetection.log
```

Windows

```
C:\greengrass\v2\logs\aws.greengrass.TensorFlowLiteObjectDetection.log
```

Para ver los registros de este componente

- Ejecute el siguiente comando en el dispositivo principal para ver el archivo de registro de este componente en tiempo real. Sustituya */greengrass/v2* o *C:\greengrass\v2* por la ruta a la carpeta AWS IoT Greengrass raíz.

  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
  ```

Registros de cambios

En la siguiente tabla se describen los cambios en cada versión del componente.

Versión	Cambios
2.1.12	Versión actualizada para la versión 2.12.5 de Greengrass nucleus.
2.1.11	Versión actualizada para la versión 2.12.0 de Greengrass nucleus.
2.1.10	Versión actualizada para la versión 2.11.0 de Greengrass nucleus.
2.1.9	Versión actualizada para la versión 2.10.0 de Greengrass nucleus.
2.1.8	Versión actualizada para la versión 2.9.0 de Greengrass Nucleus.
2.1.7	Versión actualizada para el lanzamiento de la versión 2.8.0 de Greengrass nucleus.
2.1.6	Versión actualizada para el lanzamiento de la versión 2.7.0 de Greengrass nucleus.

Versión	Cambios
2.1.5	Versión actualizada para el lanzamiento de la versión 2.6.0 de Greengrass Nucleus.
2.1.4	Versión actualizada para el lanzamiento de la versión 2.5.0 de Greengrass nucleus.
2.1.3	Versión actualizada para el lanzamiento de la versión 2.4.0 de Greengrass nucleus.
2.1.2	Versión actualizada para el lanzamiento de la versión 2.3.0 de Greengrass nucleus.
2.1.1	Mejoras y correcciones de errores • Corrige un problema de escalado de la imagen que provocaba que los recuadros delimitadores no fueran precisos en los resultados de la inferencia de detección de objetos de TensorFlow Lite de muestra.
2.1.0	Versión inicial.

TensorFlow Tienda de modelos de clasificación de imágenes Lite

El almacén de modelos de clasificación de imágenes TensorFlow Lite (`variant.TensorFlowLite.ImageClassification.ModelStore`) es un componente del modelo de aprendizaje automático que contiene un modelo MobileNet v1 previamente entrenado como artefacto de Greengrass. El modelo de muestra utilizado en este componente se obtiene del TensorFlowHub y se implementa con Lite. TensorFlow

El componente de TensorFlow Clasificación de imágenes Lite inferencia usa este componente como una dependencia para la fuente del modelo. Para usar un modelo TensorFlow Lite personalizado, cree una versión personalizada de este componente del modelo e incluya su modelo personalizado como un artefacto componente. Puede utilizar la receta de este componente como plantilla para crear componentes de modelo personalizados.

Temas

- Versiones
- Tipo

- Sistema operativo
- Requisitos
- Dependencias
- Configuración
- Archivo de registro local
- Registros de cambios

Versiones

Este componente tiene las siguientes versiones:

- 2.1.x

Tipo

Este componente es un componente genérico () `aws.greengrass.generic`. El núcleo de Greengrass ejecuta los scripts del ciclo de vida del componente.

Para obtener más información, consulte Tipos de componentes.

Sistema operativo

Este componente se puede instalar en los dispositivos principales que ejecutan los siguientes sistemas operativos:

- Linux
- Windows

Requisitos

Este componente tiene los siguientes requisitos:

- En los dispositivos principales de Greengrass que ejecutan Amazon Linux 2 o Ubuntu 18.04, se instala en el dispositivo la versión 2.27 o posterior de la Biblioteca C GNU (glibc).
- En los dispositivos ARMv7L, como Raspberry Pi, las dependencias para OpenCV-Python están instaladas en el dispositivo. Ejecute el siguiente comando para instalar las dependencias.

```
sudo apt-get install libopenjp2-7 libilmbase23 libopenexr-dev libavcodec-dev
 libavformat-dev libswscale-dev libv4l-dev libgtk-3-0 libwebp-dev
```

- Los dispositivos Raspberry Pi que ejecutan el sistema operativo Bullseye de Raspberry Pi deben cumplir los siguientes requisitos:
  - NumPy 1.22.4 o una versión posterior instalada en el dispositivo. Raspberry Pi OS Bullseye incluye una versión anterior de NumPy, por lo que puede ejecutar el siguiente comando para actualizar NumPy el dispositivo.

    ```
    pip3 install --upgrade numpy
    ```

  - La pila de cámaras antigua habilitada en el dispositivo. El Raspberry Pi OS Bullseye incluye una nueva pila de cámaras que está habilitada de forma predeterminada y no es compatible, por lo que debes activar la pila de cámaras antigua.

    Para activar la pila de cámaras antigua

    1. Ejecute el siguiente comando para abrir la herramienta de configuración de Raspberry Pi.

       ```
       sudo raspi-config
       ```

    2. Seleccione Opciones de interfaz.
    3. Seleccione Cámara antigua para activar la pila de cámaras antiguas.
    4. Reinicie el Raspberry Pi.

Dependencias

Al implementar un componente, AWS IoT Greengrass también despliega versiones compatibles de sus dependencias. Esto significa que debe cumplir los requisitos del componente y de todas sus dependencias para poder implementarlo correctamente. En esta sección se enumeran las dependencias de las versiones publicadas de este componente y las restricciones de las versiones semánticas que definen las versiones de los componentes para cada dependencia. También puede ver las dependencias de cada versión del componente en la consola.AWS IoT Greengrass En la página de detalles del componente, busque la lista de dependencias.

2.1.11 and 2.1.12

La siguiente tabla muestra las dependencias de las versiones 2.1.11 y 2.1.12 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.13.0	Flexible

2.1.10

La siguiente tabla muestra las dependencias de la versión 2.1.10 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.12.0	Flexible

2.1.9

La siguiente tabla muestra las dependencias de la versión 2.1.9 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.11.0	Flexible

2.1.8

La siguiente tabla muestra las dependencias de la versión 2.1.8 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.10.0	Flexible

2.1.7

La siguiente tabla muestra las dependencias de la versión 2.1.7 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.9.0	Flexible

2.1.6

La siguiente tabla muestra las dependencias de la versión 2.1.6 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.8.0	Flexible

2.1.5

La siguiente tabla muestra las dependencias de la versión 2.1.5 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.7.0	Flexible

2.1.4

La siguiente tabla muestra las dependencias de la versión 2.1.4 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.6.0	Flexible

2.1.3

La siguiente tabla muestra las dependencias de la versión 2.1.3 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.5.0	Flexible

2.1.2

La siguiente tabla muestra las dependencias de la versión 2.1.2 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.4.0	Flexible

2.1.1

La siguiente tabla muestra las dependencias de la versión 2.1.1 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.3.0	Flexible

2.1.0

La siguiente tabla muestra las dependencias de la versión 2.1.0 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.2.0	Flexible

Configuración

Este componente no tiene ningún parámetro de configuración.

Archivo de registro local

Este componente no genera registros.

Registros de cambios

En la siguiente tabla se describen los cambios en cada versión del componente.

Versión	Cambios
2.1.12	Versión actualizada para la versión 2.12.5 de Greengrass nucleus.
2.1.11	Versión actualizada para la versión 2.12.0 de Greengrass nucleus.
2.1.10	Versión actualizada para la versión 2.11.0 de Greengrass nucleus.
2.1.9	Versión actualizada para la versión 2.10.0 de Greengrass nucleus.
2.1.8	Versión actualizada para la versión 2.9.0 de Greengrass nucleus.
2.1.7	Versión actualizada para el lanzamiento de la versión 2.8.0 de Greengrass nucleus.
2.1.6	Versión actualizada para el lanzamiento de la versión 2.7.0 de Greengrass nucleus.
2.1.5	Versión actualizada para la versión 2.6.0 de Greengrass Nucleus.
2.1.4	Versión actualizada para el lanzamiento de la versión 2.5.0 de Greengrass nucleus.
2.1.3	Versión actualizada para la versión 2.4.0 de Greengrass Nucleus.
2.1.2	Versión actualizada para el lanzamiento de la versión 2.3.0 de Greengrass nucleus.
2.1.1	Versión actualizada para el lanzamiento de la versión 2.2.0 de Greengrass nucleus.
2.1.0	Versión inicial.

TensorFlow Tienda de modelos de detección de objetos Lite

El almacén de modelos de detección de objetos TensorFlow Lite (`variant.TensorFlowLite.ObjectDetection.ModelStore`) es un componente del modelo de aprendizaje automático que contiene un modelo de detección de disparo único (SSD)

previamente entrenado como un MobileNet artefacto de Greengrass. El modelo de muestra utilizado en este componente se obtiene del TensorFlow Hub y se implementa con Lite. TensorFlow

El componente de inferencia de detección de objetos de TensorFlow Lite utiliza este componente como una dependencia para la fuente del modelo. Para usar un modelo TensorFlow Lite personalizado, cree una versión personalizada de este componente del modelo e incluya su modelo personalizado como un artefacto componente. Puede utilizar la receta de este componente como plantilla para crear componentes de modelo personalizados.

Temas

- Versiones
- Tipo
- Sistema operativo
- Requisitos
- Dependencias
- Configuración
- Archivo de registro local
- Registros de cambios

Versiones

Este componente tiene las siguientes versiones:

- 2.1.x

Tipo

Este componente es un componente genérico () `aws.greengrass.generic`. El núcleo de Greengrass ejecuta los scripts del ciclo de vida del componente.

Para obtener más información, consulte Tipos de componentes.

Sistema operativo

Este componente se puede instalar en los dispositivos principales que ejecutan los siguientes sistemas operativos:

- Linux
- Windows

Requisitos

Este componente tiene los siguientes requisitos:

- En los dispositivos principales de Greengrass que ejecutan Amazon Linux 2 o Ubuntu 18.04, se instala en el dispositivo la versión 2.27 o posterior de la Biblioteca C GNU (glibc).
- En los dispositivos ARMv7L, como Raspberry Pi, las dependencias para OpenCV-Python están instaladas en el dispositivo. Ejecute el siguiente comando para instalar las dependencias.

  ```
  sudo apt-get install libopenjp2-7 libilmbase23 libopenexr-dev libavcodec-dev
   libavformat-dev libswscale-dev libv4l-dev libgtk-3-0 libwebp-dev
  ```

- Los dispositivos Raspberry Pi que ejecutan el sistema operativo Bullseye de Raspberry Pi deben cumplir los siguientes requisitos:
  - NumPy 1.22.4 o una versión posterior instalada en el dispositivo. Raspberry Pi OS Bullseye incluye una versión anterior de NumPy, por lo que puede ejecutar el siguiente comando para actualizar NumPy el dispositivo.

    ```
    pip3 install --upgrade numpy
    ```

  - La pila de cámaras antigua habilitada en el dispositivo. El Raspberry Pi OS Bullseye incluye una nueva pila de cámaras que está habilitada de forma predeterminada y no es compatible, por lo que debes activar la pila de cámaras antigua.

    Para activar la pila de cámaras antigua

    1. Ejecute el siguiente comando para abrir la herramienta de configuración de Raspberry Pi.

       ```
       sudo raspi-config
       ```

    2. Seleccione Opciones de interfaz.
    3. Seleccione Cámara antigua para activar la pila de cámaras antiguas.
    4. Reinicie el Raspberry Pi.

Dependencias

Al implementar un componente, AWS IoT Greengrass también despliega versiones compatibles de sus dependencias. Esto significa que debe cumplir los requisitos del componente y de todas sus dependencias para poder implementarlo correctamente. En esta sección se enumeran las dependencias de las versiones publicadas de este componente y las restricciones de las versiones semánticas que definen las versiones de los componentes para cada dependencia. También puede ver las dependencias de cada versión del componente en la consola.AWS IoT Greengrass En la página de detalles del componente, busque la lista de dependencias.

2.1.11 and 2.1.12

La siguiente tabla muestra las dependencias de las versiones 2.1.11 y 2.1.12 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.13.0	Flexible

2.1.10

La siguiente tabla muestra las dependencias de la versión 2.1.10 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.12.0	Flexible

2.1.9

La siguiente tabla muestra las dependencias de la versión 2.1.9 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.11.0	Flexible

2.1.8

La siguiente tabla muestra las dependencias de la versión 2.1.8 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.10.0	Flexible

2.1.7

La siguiente tabla muestra las dependencias de la versión 2.1.7 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.9.0	Flexible

2.1.6

La siguiente tabla muestra las dependencias de la versión 2.1.6 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.8.0	Flexible

2.1.5

La siguiente tabla muestra las dependencias de la versión 2.1.5 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.7.0	Flexible

2.1.4

La siguiente tabla muestra las dependencias de la versión 2.1.4 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.6.0	Flexible

2.1.3

La siguiente tabla muestra las dependencias de la versión 2.1.3 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.5.0	Flexible

2.1.2

La siguiente tabla muestra las dependencias de la versión 2.1.2 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.4.0	Flexible

2.1.1

La siguiente tabla muestra las dependencias de la versión 2.1.1 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.3.0	Flexible

2.1.0

La siguiente tabla muestra las dependencias de la versión 2.1.0 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.2.0	Flexible

Configuración

Este componente no tiene ningún parámetro de configuración.

Archivo de registro local

Este componente no genera registros.

Registros de cambios

En la siguiente tabla se describen los cambios en cada versión del componente.

Versión	Cambios
2.1.12	Versión actualizada para la versión 2.12.5 de Greengrass nucleus.
2.1.11	Versión actualizada para la versión 2.12.0 de Greengrass nucleus.
2.1.10	Versión actualizada para la versión 2.11.0 de Greengrass nucleus.
2.1.9	Versión actualizada para la versión 2.10.0 de Greengrass nucleus.
2.1.8	Versión actualizada para la versión 2.9.0 de Greengrass nucleus.
2.1.7	Versión actualizada para el lanzamiento de la versión 2.8.0 de Greengrass nucleus.
2.1.6	Versión actualizada para el lanzamiento de la versión 2.7.0 de Greengrass nucleus.
2.1.5	Versión actualizada para la versión 2.6.0 de Greengrass Nucleus.
2.1.4	Versión actualizada para el lanzamiento de la versión 2.5.0 de Greengrass nucleus.
2.1.3	Versión actualizada para la versión 2.4.0 de Greengrass Nucleus.
2.1.2	Versión actualizada para el lanzamiento de la versión 2.3.0 de Greengrass nucleus.
2.1.1	Versión actualizada para el lanzamiento de la versión 2.2.0 de Greengrass nucleus.
2.1.0	Versión inicial.

TensorFlow Tiempo de ejecución Lite

El componente de tiempo de ejecución de TensorFlow Lite (`variant.TensorFlowLite`) contiene un script que instala la versión 2.5.0 de TensorFlow Lite y sus dependencias en un entorno virtual del dispositivo. La clasificación de imágenes de TensorFlow TensorFlow Lite y el componente de detección de objetos de Lite utilizan este componente de tiempo de ejecución como una dependencia para instalar Lite. TensorFlow

> **Note**
>
> TensorFlow El componente de tiempo de ejecución de Lite, versión 2.5.6 y versiones posteriores, reinstala las instalaciones existentes del entorno de ejecución de TensorFlow Lite y sus dependencias. Esta reinstalación ayuda a garantizar que el dispositivo principal ejecute versiones compatibles de Lite y sus dependencias. TensorFlow

Para usar un entorno de ejecución diferente, puede usar la receta de este componente como plantilla para crear un componente de aprendizaje automático personalizado.

Temas

Versiones

Este componente tiene las siguientes versiones:

- 2.5.x

Tipo

Este componente es un componente genérico () `aws.greengrass.generic`. El núcleo de Greengrass ejecuta los scripts del ciclo de vida del componente.

Para obtener más información, consulte Tipos de componentes.

Sistema operativo

Este componente se puede instalar en los dispositivos principales que ejecutan los siguientes sistemas operativos:

- Linux
- Windows

Requisitos

Este componente tiene los siguientes requisitos:

- En los dispositivos principales de Greengrass que ejecutan Amazon Linux 2 o Ubuntu 18.04, se instala en el dispositivo la versión 2.27 o posterior de la Biblioteca C GNU (glibc).
- En los dispositivos ARMv7L, como Raspberry Pi, las dependencias para OpenCV-Python están instaladas en el dispositivo. Ejecute el siguiente comando para instalar las dependencias.

  ```
  sudo apt-get install libopenjp2-7 libilmbase23 libopenexr-dev libavcodec-dev
   libavformat-dev libswscale-dev libv4l-dev libgtk-3-0 libwebp-dev
  ```

- Los dispositivos Raspberry Pi que ejecutan el sistema operativo Bullseye de Raspberry Pi deben cumplir los siguientes requisitos:
  - NumPy 1.22.4 o una versión posterior instalada en el dispositivo. Raspberry Pi OS Bullseye incluye una versión anterior de NumPy, por lo que puede ejecutar el siguiente comando para actualizar NumPy el dispositivo.

    ```
    pip3 install --upgrade numpy
    ```

  - La pila de cámaras antigua habilitada en el dispositivo. El Raspberry Pi OS Bullseye incluye una nueva pila de cámaras que está habilitada de forma predeterminada y no es compatible, por lo que debes activar la pila de cámaras antigua.

Para activar la pila de cámaras antigua

1. Ejecute el siguiente comando para abrir la herramienta de configuración de Raspberry Pi.

   ```
   sudo raspi-config
   ```

2. Seleccione Opciones de interfaz.
3. Seleccione Cámara antigua para activar la pila de cámaras antiguas.
4. Reinicie el Raspberry Pi.

Puntos finales y puertos

De forma predeterminada, este componente utiliza un script de instalación para instalar los paquetes mediante los `pip` comandos `apt` `yum``brew`,, y, según la plataforma que utilice el dispositivo principal. Este componente debe poder realizar solicitudes salientes a varios índices y repositorios de paquetes para ejecutar el script de instalación. Para permitir que el tráfico saliente de este componente pase a través de un proxy o firewall, debe identificar los puntos finales de los índices y repositorios de paquetes a los que se conecta el dispositivo principal para realizar la instalación.

Tenga en cuenta lo siguiente al identificar los puntos finales necesarios para el script de instalación de este componente:

- Los puntos finales dependen de la plataforma del dispositivo principal. Por ejemplo, un dispositivo central que ejecuta Ubuntu usa en `apt` lugar de `yum` o`brew`. Además, los dispositivos que usan el mismo índice de paquetes pueden tener listas de fuentes diferentes, por lo que pueden recuperar paquetes de diferentes repositorios.
- Los puntos finales pueden diferir entre varios dispositivos que utilizan el mismo índice de paquetes, ya que cada dispositivo tiene sus propias listas de fuentes que definen dónde recuperar los paquetes.
- Los puntos finales pueden cambiar con el tiempo. Cada índice de paquetes proporciona las direcciones URL de los repositorios en los que se descargan los paquetes, y el propietario de un paquete puede cambiar las direcciones URL que proporciona el índice de paquetes.

Para obtener más información sobre las dependencias que instala este componente y sobre cómo deshabilitar el script del instalador, consulte el parámetro de configuración. UseInstaller

Para obtener más información sobre los puntos finales y los puertos necesarios para el funcionamiento básico, consulte. Permitir el tráfico del dispositivo a través de un proxy o firewall

Dependencias

Al implementar un componente, AWS IoT Greengrass también despliega versiones compatibles de sus dependencias. Esto significa que debe cumplir los requisitos del componente y de todas sus dependencias para poder implementarlo correctamente. En esta sección se enumeran las dependencias de las versiones publicadas de este componente y las restricciones de las versiones semánticas que definen las versiones de los componentes para cada dependencia. También puede ver las dependencias de cada versión del componente en la consola.AWS IoT Greengrass En la página de detalles del componente, busque la lista de dependencias.

2.5.14 and 2.5.15

La siguiente tabla muestra las dependencias de las versiones 2.5.14 y 2.5.15 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.13.0	Flexible

2.5.13

La siguiente tabla muestra las dependencias de la versión 2.5.13 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.12.0	Flexible

2.5.12

La siguiente tabla muestra las dependencias de la versión 2.5.12 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.11.0	Flexible

2.5.11

La siguiente tabla muestra las dependencias de la versión 2.5.11 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.10.0	Flexible

2.5.10

La siguiente tabla muestra las dependencias de la versión 2.5.10 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.9.0	Flexible

2.5.9

La siguiente tabla muestra las dependencias de la versión 2.5.9 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.8.0	Flexible

2.5.8

La siguiente tabla muestra las dependencias de la versión 2.5.8 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.7.0	Flexible

2.5.5 - 2.5.7

En la siguiente tabla se enumeran las dependencias de las versiones 2.5.5 a 2.5.7 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.6.0	Flexible

2.5.3 and 2.5.4

La siguiente tabla muestra las dependencias de las versiones 2.5.3 y 2.5.4 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.5.0	Flexible

2.5.2

La siguiente tabla muestra las dependencias de la versión 2.5.2 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.4.0	Flexible

2.5.1

La siguiente tabla muestra las dependencias de la versión 2.5.1 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.3.0	Flexible

2.5.0

La siguiente tabla muestra las dependencias de la versión 2.5.0 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.2.0	Flexible

Para obtener más información sobre las dependencias de los componentes, consulta la referencia de recetas de componentes.

Configuración

Este componente proporciona los siguientes parámetros de configuración que puede personalizar al implementar el componente.

`MLRootPath`

(Opcional) La ruta de la carpeta en los dispositivos principales de Linux donde los componentes de inferencia leen las imágenes y escriben los resultados de la inferencia. Puede modificar este valor en cualquier ubicación del dispositivo a la que el usuario que ejecuta este componente tenga acceso de lectura y escritura.

Valor predeterminado: *`/greengrass/v2`*`/work/variant.TensorFlowLite/greengrass_ml`

`WindowsMLRootPath`

Esta función está disponible en la versión 1.6.6 y versiones posteriores de este componente.

(Opcional) La ruta de la carpeta del dispositivo principal de Windows donde los componentes de inferencia leen las imágenes y escriben los resultados de la inferencia. Puede modificar este valor en cualquier ubicación del dispositivo a la que el usuario que ejecuta este componente tenga acceso de lectura y escritura.

Valor predeterminado: *`C:\greengrass\v2`*`\\work\\variant.DLR\\greengrass_ml`

`UseInstaller`

(Opcional) Valor de cadena que define si se debe utilizar el script de instalación de este componente para instalar TensorFlow Lite y sus dependencias. Los valores admitidos son `true` y `false`.

Establezca este valor `false` si desea utilizar un script personalizado para la instalación de TensorFlow Lite o si desea incluir las dependencias del tiempo de ejecución en una imagen de Linux prediseñada. Para usar este componente con los componentes AWS de inferencia de TensorFlow Lite proporcionados, instale las siguientes bibliotecas, incluidas las dependencias, y póngalas a disposición del usuario del sistema, por ejemplo, el que ejecuta los componentes de `ggc_user` ML.

- Python 3.8 o posterior, incluso `pip` para su versión de Python
- TensorFlow Lite v2.5.0
- NumPy
- OpenCV-Python
- SDK para dispositivos con AWS IoT v2 para Python
- AWS Python en tiempo de ejecución común (CRT)
- Picamera (para dispositivos Raspberry Pi)
- `awscam`módulo (para AWS DeepLens dispositivos)
- libGL (para dispositivos Linux)

Valor predeterminado: `true`

Uso

Utilice este componente con el parámetro `UseInstaller` de configuración establecido en `true` para instalar TensorFlow Lite y sus dependencias en el dispositivo. El componente configura un entorno virtual en el dispositivo que incluye el OpenCV NumPy y las bibliotecas necesarias para Lite. TensorFlow

> **Note**
>
> El script de instalación de este componente también instala las versiones más recientes de las bibliotecas del sistema adicionales necesarias para configurar el entorno virtual en el dispositivo y utilizar el marco de aprendizaje automático instalado. Esto podría actualizar las bibliotecas del sistema existentes en el dispositivo. Consulte la siguiente tabla para ver la lista de bibliotecas que instala este componente para cada sistema operativo compatible. Si desea personalizar este proceso de instalación, defina `false` el parámetro de `UseInstaller` configuración en y desarrolle su propio script de instalación.

Plataforma	Bibliotecas instaladas en el sistema del dispositivo	Bibliotecas instaladas en el entorno virtual
Armv7l	`build-essential` , `cmake`, `ca-certificates` , `git`	`setuptools` , `wheel`

Plataforma	Bibliotecas instaladas en el sistema del dispositivo	Bibliotecas instaladas en el entorno virtual
Amazon Linux 2	`mesa-libGL`	Ninguna
Ubuntu	`wget`	Ninguna

Al implementar el componente de inferencia, este componente de tiempo de ejecución comprueba primero si el dispositivo ya tiene instalado TensorFlow Lite y sus dependencias. Si no es así, el componente de tiempo de ejecución los instala automáticamente.

Archivo de registro local

Este componente utiliza el siguiente archivo de registro.

Linux

```
/greengrass/v2/logs/variant.TensorFlowLite.log
```

Windows

```
C:\greengrass\v2\logs\variant.TensorFlowLite.log
```

Para ver los registros de este componente

- Ejecute el siguiente comando en el dispositivo principal para ver el archivo de registro de este componente en tiempo real. Sustituya */greengrass/v2* o *C:\greengrass\v2* por la ruta a la carpeta AWS IoT Greengrass raíz.

  Linux

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

  Windows (PowerShell)

  ```
  Get-Content C:\greengrass\v2\logs\variant.TensorFlowLite.log -Tail 10 -Wait
  ```

Registros de cambios

En la siguiente tabla se describen los cambios en cada versión del componente.

Versión	Cambios
2.5.15	Versión actualizada para la versión 2.12.5 de Greengrass nucleus.
2.5.14	Versión actualizada para la versión 2.12.0 de Greengrass nucleus.
2.5.13	Versión actualizada para el lanzamiento de la versión 2.11.0 de Greengrass nucleus.
2.5.12	Versión actualizada para la versión 2.10.0 de Greengrass nucleus.
2.5.11	Versión actualizada para la versión 2.9.0 de Greengrass Nucleus.
2.5.10	Versión actualizada para el lanzamiento de la versión 2.8.0 de Greengrass nucleus.
2.5.9	Versión actualizada para el lanzamiento de la versión 2.7.0 de Greengrass nucleus.
2.5.8	Versión actualizada para la versión 2.6.0 de Greengrass Nucleus.
2.5.7	Mejoras y correcciones de errores • Actualiza el script de `UseInstaller` instalación para instalar LibGL, que no está disponible de forma predeterminada en determinadas plataformas Linux. • Actualiza el script de `UseInstaller` instalación para usar siempre Python 3.9 en el entorno virtual de este componente. Este cambio ayuda a garantizar la compatibilidad con otras bibliotecas.
2.5.6	Mejoras y correcciones de errores • Actualiza este componente para instalar el último parche de TensorFlow Lite 2.5.0 (`tflite-runtime-2.5.0.post1`), de modo que pueda usar este componente con Python 3.9. Si este componente no puede instalar el parche, se instala `tflite-runtime-2.5.0` en su lugar.

Versión	Cambios
	• Actualiza este componente para volver a instalar las instalaciones existentes de TensorFlow Lite y sus dependencias. Este cambio ayuda a garantizar que el dispositivo principal ejecute versiones compatibles de TensorFlow Lite y sus dependencias.
2.5.5	Nuevas características • Añade compatibilidad con los dispositivos principales que ejecutan Windows. • Agrega el nuevo parámetro `WindowsMLRootPath` de configuración que puede usar para configurar la carpeta de resultados de inferencias en los dispositivos principales de Windows.
2.5.4	Nuevas características • Añade el nuevo parámetro `UseInstaller` de configuración que permite deshabilitar el script de instalación en este componente.
2.5.3	Versión actualizada para la versión 2.4.0 de Greengrass Nucleus.
2.5.2	Versión actualizada para el lanzamiento de la versión 2.3.0 de Greengrass nucleus.
2.5.1	Versión actualizada para el lanzamiento de la versión 2.2.0 de Greengrass nucleus.
2.5.0	Versión inicial.

Adaptador de protocolo Modbus-RTU

El componente adaptador de protocolo Modbus-RTU (`aws.greengrass.Modbus`) recopila información de los dispositivos Modbus RTU locales.

Para solicitar información a un dispositivo Modbus RTU local con este componente, publique un mensaje en el tema al que está suscrito este componente. En el mensaje, especifique la solicitud de Modbus RTU que se va a enviar a un dispositivo. A continuación, este componente publica una respuesta que contiene el resultado de la solicitud de Modbus RTU.

> **Note**
>
> Este componente proporciona una funcionalidad similar a la del conector adaptador del protocolo Modbus RTU de la V1. AWS IoT Greengrass Para obtener más información, consulte el conector adaptador del protocolo Modbus RTU en la Guía para desarrolladores de la AWS IoT Greengrass V1.

Temas

Versiones

Este componente tiene las siguientes versiones:

- 2.1.x
- 2.0.x

Tipo

Este componente es un componente Lambda () `aws.greengrass.lambda`. El núcleo de Greengrass ejecuta la función Lambda de este componente mediante el componente Lambda launcher.

Para obtener más información, consulte Tipos de componentes.

Sistema operativo

Este componente solo se puede instalar en los dispositivos principales de Linux.

Requisitos

Este componente tiene los siguientes requisitos:

- El dispositivo principal debe cumplir los requisitos para ejecutar las funciones de Lambda. Si desea que el dispositivo principal ejecute funciones Lambda en contenedores, el dispositivo debe cumplir los requisitos para hacerlo. Para obtener más información, consulte Requisitos de la función de Lambda.
- Versión 3.7 de Python instalada en el dispositivo principal y añadida a la variable de entorno PATH.
- Una conexión física entre el dispositivo AWS IoT Greengrass principal y los dispositivos Modbus. El dispositivo principal debe estar conectado físicamente a la red Modbus RTU a través de un puerto serie, como un puerto USB.
- Para recibir los datos de salida de este componente, debe combinar la siguiente actualización de configuración para el componente del router de suscripción antiguo (`aws.greengrass.LegacySubscriptionRouter`) al implementar este componente. Esta configuración especifica el tema en el que este componente publica las respuestas.

  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"
    }
  }
}
```

- Sustituya la *región* por la Región de AWS que utilice.
- Sustituya la *versión* por la versión de la función Lambda que ejecuta este componente. Para encontrar la versión de la función Lambda, debe ver la receta de la versión de este componente que desee implementar. Abra la página de detalles de este componente en la AWS IoT Greengrass consola y busque el par clave-valor de la función Lambda. Este par clave-valor contiene el nombre y la versión de la función Lambda.

> **Important**
>
> Debe actualizar la versión de la función Lambda en el router de suscripción anterior cada vez que implemente este componente. Esto garantiza que utilice la versión correcta de la función Lambda para la versión del componente que implemente.

Para obtener más información, consulte Crear implementaciones.

- Se admite la ejecución del adaptador de protocolo Modbus-RTU en una VPC.

Dependencias

Al implementar un componente, AWS IoT Greengrass también despliega versiones compatibles de sus dependencias. Esto significa que debe cumplir los requisitos del componente y de todas sus dependencias para poder implementarlo correctamente. En esta sección se enumeran las dependencias de las versiones publicadas de este componente y las restricciones de las versiones semánticas que definen las versiones de los componentes para cada dependencia. También puede ver las dependencias de cada versión del componente en la consola. AWS IoT Greengrass En la página de detalles del componente, busque la lista de dependencias.

2.1.8

La siguiente tabla muestra las dependencias de la versión 2.1.8 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.13.0	Rígido
Lanzador Lambda	^2.0.0	Rígido
Tiempos de ejecución de Lambda	^2.0.0	Flexible
Servicio de intercambio de fichas	^2.0.0	Rígido

2.1.7

La siguiente tabla muestra las dependencias de la versión 2.1.7 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.12.0	Rígido
Lanzador Lambda	^2.0.0	Rígido
Tiempos de ejecución de Lambda	^2.0.0	Flexible
Servicio de intercambio de fichas	^2.0.0	Rígido

2.1.6

La siguiente tabla muestra las dependencias de la versión 2.1.6 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.11.0	Rígido
Lanzador Lambda	^2.0.0	Rígido
Tiempos de ejecución de Lambda	^2.0.0	Flexible
Servicio de intercambio de fichas	^2.0.0	Rígido

2.1.4 and 2.1.5

La siguiente tabla muestra las dependencias de las versiones 2.1.4 y 2.1.5 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.10.0	Rígido
Lanzador Lambda	^2.0.0	Rígido
Tiempos de ejecución de Lambda	^2.0.0	Flexible
Servicio de intercambio de fichas	^2.0.0	Rígido

2.1.3

La siguiente tabla muestra las dependencias de la versión 2.1.3 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.9.0	Rígido
Lanzador Lambda	^2.0.0	Rígido

Dependencia	Versiones compatibles	Tipo de dependencia
Tiempos de ejecución de Lambda	^2.0.0	Flexible
Servicio de intercambio de fichas	^2.0.0	Rígido

2.1.2

La siguiente tabla muestra las dependencias de la versión 2.1.2 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.8.0	Rígido
Lanzador Lambda	^2.0.0	Rígido
Tiempos de ejecución de Lambda	^2.0.0	Flexible
Servicio de intercambio de fichas	^2.0.0	Rígido

2.1.1

La siguiente tabla muestra las dependencias de la versión 2.1.1 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.7.0	Rígido
Lanzador Lambda	^2.0.0	Rígido
Tiempos de ejecución de Lambda	^2.0.0	Flexible

Dependencia	Versiones compatibles	Tipo de dependencia
Servicio de intercambio de fichas	^2.0.0	Rígido

2.0.8 and 2.1.0

La siguiente tabla muestra las dependencias de las versiones 2.0.8 y 2.1.0 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.6.0	Rígido
Lanzador Lambda	^2.0.0	Rígido
Tiempos de ejecución de Lambda	^2.0.0	Flexible
Servicio de intercambio de fichas	^2.0.0	Rígido

2.0.7

La siguiente tabla muestra las dependencias de la versión 2.0.7 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.5.0	Rígido
Lanzador Lambda	^2.0.0	Rígido
Tiempos de ejecución de Lambda	^2.0.0	Flexible
Servicio de intercambio de fichas	^2.0.0	Rígido

2.0.6

La siguiente tabla muestra las dependencias de la versión 2.0.6 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.4.0	Rígido
Lanzador Lambda	^2.0.0	Rígido
Tiempos de ejecución de Lambda	^2.0.0	Flexible
Servicio de intercambio de fichas	^2.0.0	Rígido

2.0.5

La siguiente tabla muestra las dependencias de la versión 2.0.5 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.3.0	Rígido
Lanzador Lambda	^2.0.0	Rígido
Tiempos de ejecución de Lambda	^2.0.0	Flexible
Servicio de intercambio de fichas	^2.0.0	Rígido

2.0.4

La siguiente tabla muestra las dependencias de la versión 2.0.4 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.2.0	Rígido
Lanzador Lambda	^2.0.0	Rígido
Tiempos de ejecución de Lambda	^2.0.0	Flexible
Servicio de intercambio de fichas	^2.0.0	Rígido

2.0.3

La siguiente tabla muestra las dependencias de la versión 2.0.3 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.3 <2.1.0	Rígido
Lanzador Lambda	>=1.0.0	Rígido
Tiempos de ejecución de Lambda	>=1.0.0	Flexible
Servicio de intercambio de fichas	>=1.0.0	Rígido

Para obtener más información sobre las dependencias de los componentes, consulta la referencia de recetas de componentes.

Configuración

Este componente proporciona los siguientes parámetros de configuración que puede personalizar al implementar el componente.

> **Note**
>
> La configuración predeterminada de este componente incluye los parámetros de la función Lambda. Le recomendamos que edite solo los siguientes parámetros para configurar este componente en sus dispositivos.

v2.1.x

`lambdaParams`

Objeto que contiene los parámetros de la función Lambda de este componente. Este objeto contiene la siguiente información:

`EnvironmentVariables`

Objeto que contiene los parámetros de la función Lambda. Este objeto contiene la siguiente información:

`ModbusLocalPort`

La ruta absoluta al puerto serie Modbus físico del dispositivo principal, por ejemplo. `/dev/ttyS2`

Para ejecutar este componente en un contenedor, debe definir esta ruta como un dispositivo del sistema (`encontainerParams.devices`) al que pueda acceder el componente. Este componente se ejecuta en un contenedor de forma predeterminada.

> **Note**
>
> Este componente debe tener acceso de lectura/escritura al dispositivo.

`ModbusBaudRate`

(Opcional) Un valor de cadena que especifica la velocidad en baudios para la comunicación en serie con los dispositivos Modbus TCP locales.

Predeterminado: `9600`

`ModbusByteSize`

(Opcional) Un valor de cadena que especifica el tamaño de un byte en la comunicación en serie con los dispositivos Modbus TCP locales. Elija`5`, `6`, `7`, o `8` bits.

Predeterminado: `8`

`ModbusParity`

(Opcional) El modo de paridad que se utilizará para verificar la integridad de los datos en la comunicación en serie con los dispositivos Modbus TCP locales.

- `E`— Verifique la integridad de los datos con una paridad uniforme.
- `O`— Verificar la integridad de los datos con una paridad impar.
- `N`— No verifique la integridad de los datos.

Predeterminado: `N`

`ModbusStopBits`

(Opcional) Un valor de cadena que especifica el número de bits que indican el final de un byte en la comunicación en serie con los dispositivos Modbus TCP locales.

Predeterminado: `1`

`containerMode`

(Opcional) El modo de contenedorización de este componente. Puede elegir entre las siguientes opciones:

- `GreengrassContainer`— El componente se ejecuta en un entorno de ejecución aislado dentro del AWS IoT Greengrass contenedor.

  Si especifica esta opción, debe especificar un dispositivo del sistema (en`containerParams.devices`) para que el contenedor acceda al dispositivo Modbus.
- `NoContainer`— El componente no se ejecuta en un entorno de ejecución aislado.

Predeterminado: `GreengrassContainer`

`containerParams`

(Opcional) Un objeto que contiene los parámetros del contenedor de este componente. El componente utiliza estos parámetros si se especifica `GreengrassContainer` para`containerMode`.

Este objeto contiene la siguiente información:

`memorySize`

(Opcional) La cantidad de memoria (en kilobytes) que se va a asignar al componente.

El valor predeterminado es 512 MB (525.312 KB).

`devices`

(Opcional) Objeto que especifica los dispositivos del sistema a los que puede acceder el componente en un contenedor.

> **Important**
>
> Para ejecutar este componente en un contenedor, debe especificar el dispositivo del sistema que configura en la variable de `ModbusLocalPort` entorno.

Este objeto contiene la siguiente información:

`0`— Se trata de un índice de matriz en forma de cadena.

Objeto que contiene la siguiente información:

`path`

La ruta al dispositivo del sistema en el dispositivo principal. Debe tener el mismo valor que el valor para el que se configura`ModbusLocalPort`.

`permission`

(Opcional) El permiso para acceder al dispositivo del sistema desde el contenedor. Este valor debe ser `rw` el que especifica que el componente tiene acceso de lectura/escritura al dispositivo del sistema.

Predeterminado: `rw`

`addGroupOwner`

(Opcional) Si se debe agregar o no el grupo de sistemas que ejecuta el componente como propietario del dispositivo del sistema.

Predeterminado: `true`

`pubsubTopics`

(Opcional) Un objeto que contiene los temas a los que el componente se suscribe para recibir mensajes. Puede especificar cada tema y si el componente se suscribe a los temas de MQTT AWS IoT Core o a los temas de publicación/suscripción locales.

Este objeto contiene la siguiente información:

`0`— Se trata de un índice matricial en forma de cadena.

Objeto que contiene la siguiente información:

`type`

(Opcional) El tipo de mensajes de publicación/suscripción que utiliza este componente para suscribirse a los mensajes. Puede elegir entre las siguientes opciones:

- `PUB_SUB` — Suscribirse a mensajes locales de publicación/suscripción. Si elige esta opción, el tema no puede contener caracteres comodín de MQTT. Para obtener más información sobre cómo enviar mensajes desde un componente personalizado al especificar esta opción, consulte. Publicar/suscribir mensajes locales
- `IOT_CORE`— Suscríbase a los mensajes de AWS IoT Core MQTT. Si elige esta opción, el tema puede contener caracteres comodín de MQTT. Para obtener más información sobre cómo enviar mensajes desde componentes personalizados al especificar esta opción, consulte. Publicar/suscribir mensajes MQTT AWS IoT Core

Predeterminado: `PUB_SUB`

`topic`

(Opcional) El tema al que se suscribe el componente para recibir mensajes. Si lo especifica `IotCoretype`, puede usar los comodines MQTT (+y#) en este tema.

Example Ejemplo: actualización de la combinación de configuraciones (modo contenedor)

```
{
  "lambdaExecutionParameters": {
    "EnvironmentVariables": {
      "ModbusLocalPort": "/dev/ttyS2"
    }
  },
  "containerMode": "GreengrassContainer",
```

```
  "containerParams": {
    "devices": {
      "0": {
        "path": "/dev/ttyS2",
        "permission": "rw",
        "addGroupOwner": true
      }
    }
  }
}
```

Example Ejemplo: actualización de la combinación de configuraciones (sin modo contenedor)

```
{
  "lambdaExecutionParameters": {
    "EnvironmentVariables": {
      "ModbusLocalPort": "/dev/ttyS2"
    }
  },
  "containerMode": "NoContainer"
}
```

v2.0.x

`lambdaParams`

Objeto que contiene los parámetros de la función Lambda de este componente. Este objeto contiene la siguiente información:

`EnvironmentVariables`

Objeto que contiene los parámetros de la función Lambda. Este objeto contiene la siguiente información:

`ModbusLocalPort`

La ruta absoluta al puerto serie Modbus físico del dispositivo principal, por ejemplo. `/dev/ttyS2`

Para ejecutar este componente en un contenedor, debe definir esta ruta como un dispositivo del sistema (en`containerParams.devices`) al que pueda acceder el componente. Este componente se ejecuta en un contenedor de forma predeterminada.

> **Note**
>
> Este componente debe tener acceso de lectura/escritura al dispositivo.

`containerMode`

(Opcional) El modo de contenedorización de este componente. Puede elegir entre las siguientes opciones:

- `GreengrassContainer`— El componente se ejecuta en un entorno de ejecución aislado dentro del AWS IoT Greengrass contenedor.

  Si especifica esta opción, debe especificar un dispositivo del sistema (`encontainerParams.devices`) para que el contenedor acceda al dispositivo Modbus.
- `NoContainer`— El componente no se ejecuta en un entorno de ejecución aislado.

Predeterminado: `GreengrassContainer`

`containerParams`

(Opcional) Un objeto que contiene los parámetros del contenedor de este componente. El componente utiliza estos parámetros si se especifica `GreengrassContainer` para`containerMode`.

Este objeto contiene la siguiente información:

`memorySize`

(Opcional) La cantidad de memoria (en kilobytes) que se va a asignar al componente.

El valor predeterminado es 512 MB (525.312 KB).

`devices`

(Opcional) Objeto que especifica los dispositivos del sistema a los que puede acceder el componente en un contenedor.

> **Important**
>
> Para ejecutar este componente en un contenedor, debe especificar el dispositivo del sistema que configura en la variable de `ModbusLocalPort` entorno.

Este objeto contiene la siguiente información:

`0`— Se trata de un índice de matriz en forma de cadena.

Objeto que contiene la siguiente información:

`path`

La ruta al dispositivo del sistema en el dispositivo principal. Debe tener el mismo valor que el valor para el que se configura`ModbusLocalPort`.

`permission`

(Opcional) El permiso para acceder al dispositivo del sistema desde el contenedor. Este valor debe ser `rw` el que especifica que el componente tiene acceso de lectura/escritura al dispositivo del sistema.

Predeterminado: `rw`

`addGroupOwner`

(Opcional) Si se debe agregar o no el grupo de sistemas que ejecuta el componente como propietario del dispositivo del sistema.

Predeterminado: `true`

`pubsubTopics`

(Opcional) Un objeto que contiene los temas a los que el componente se suscribe para recibir mensajes. Puede especificar cada tema y si el componente se suscribe a los temas de MQTT AWS IoT Core o a los temas de publicación/suscripción locales.

Este objeto contiene la siguiente información:

`0`— Se trata de un índice matricial en forma de cadena.

Objeto que contiene la siguiente información:

`type`

(Opcional) El tipo de mensajes de publicación/suscripción que utiliza este componente para suscribirse a los mensajes. Puede elegir entre las siguientes opciones:

- `PUB_SUB` — Suscribirse a mensajes locales de publicación/suscripción. Si elige esta opción, el tema no puede contener caracteres comodín de MQTT. Para obtener más información sobre cómo enviar mensajes desde un componente personalizado al especificar esta opción, consulte. Publicar/suscribir mensajes locales

- `IOT_CORE`— Suscríbase a los mensajes de AWS IoT Core MQTT. Si elige esta opción, el tema puede contener caracteres comodín de MQTT. Para obtener más información sobre cómo enviar mensajes desde componentes personalizados al especificar esta opción, consulte. Publicar/suscribir mensajes MQTT AWS IoT Core

Predeterminado: `PUB_SUB`

`topic`

(Opcional) El tema al que se suscribe el componente para recibir mensajes. Si lo especifica `IotCoretype`, puede usar los comodines MQTT (+y#) en este tema.

Example Ejemplo: actualización de la combinación de configuraciones (modo contenedor)

```
{
  "lambdaExecutionParameters": {
    "EnvironmentVariables": {
      "ModbusLocalPort": "/dev/ttyS2"
    }
  },
  "containerMode": "GreengrassContainer",
  "containerParams": {
    "devices": {
      "0": {
        "path": "/dev/ttyS2",
        "permission": "rw",
        "addGroupOwner": true
      }
    }
  }
}
```

Example Ejemplo: actualización de la combinación de configuraciones (sin modo contenedor)

```
{
  "lambdaExecutionParameters": {
    "EnvironmentVariables": {
      "ModbusLocalPort": "/dev/ttyS2"
    }
  },
  "containerMode": "NoContainer"
}
```

Datos de entrada

Este componente acepta los parámetros de solicitud de Modbus RTU relacionados con el tema siguiente y envía la solicitud de Modbus RTU al dispositivo. De forma predeterminada, este componente se suscribe a los mensajes de publicación/suscripción locales. Para obtener más información sobre cómo publicar mensajes en este componente desde sus componentes personalizados, consulte. Publicar/suscribir mensajes locales

Tema predeterminado (publicación/suscripción local): `modbus/adapter/request`

El mensaje acepta las siguientes propiedades. Los mensajes de entrada deben tener un formato JSON válido.

`request`

Los parámetros de la solicitud de Modbus RTU que se va a enviar.

La forma del mensaje de solicitud depende del tipo de solicitud de Modbus RTU que represente. Las siguientes propiedades son obligatorias para todas las solicitudes.

Tipo: `object` que contiene la siguiente información:

`operation`

El nombre de la operación que se va a ejecutar. Por ejemplo, especifique `ReadCoilsRequest` que se lean las bobinas de un dispositivo Modbus RTU. Para obtener más información sobre las operaciones compatibles, consulte. Solicitudes y respuestas de Modbus RTU

Tipo: `string`

`device`

El dispositivo de destino de la solicitud.

Este valor debe ser un número entero comprendido entre `0` y`247`.

Tipo: `integer`

El resto de los parámetros que se incluirán en la solicitud dependen de la operación. Este componente gestiona la comprobación de redundancia cíclica (CRC) para comprobar las solicitudes de datos por usted.

> **Note**
>
> Si la solicitud incluye una `address` propiedad, debe especificar su valor como un número entero. Por ejemplo, `"address": 1` .

`id`

Un ID arbitrario para la solicitud. Utilice esta propiedad para asignar una solicitud de entrada a una respuesta de salida. Al especificar esta propiedad, el componente establece la `id` propiedad del objeto de respuesta en este valor.

Tipo: `string`

Example Ejemplo de entrada: Solicitud de lectura de salidas digitales (coils)

```
{
  "request": {
    "operation": "ReadCoilsRequest",
    "device": 1,
    "address": 1,
    "count": 1
  },
  "id": "MyRequest"
}
```

Datos de salida

Este componente publica las respuestas como datos de salida sobre el siguiente tema de MQTT de forma predeterminada. Debe especificar este tema como parte de `subject` la configuración del componente antiguo del router de suscripciones. Para obtener más información sobre cómo suscribirse a los mensajes sobre este tema en sus componentes personalizados, consultePublicar/suscribir mensajes MQTT AWS IoT Core.

Tema predeterminado (AWS IoT Core MQTT): `modbus/adapter/response`

La forma del mensaje de respuesta depende de la operación de solicitud y del estado de la respuesta. Para ver ejemplos, consulte Solicitudes y respuestas de ejemplo.

Cada respuesta incluye las siguientes propiedades:

`response`

La respuesta del dispositivo Modbus RTU.

Tipo: `object` que contiene la siguiente información:

`status`

El estado de la solicitud. El estado puede ser uno de los siguientes valores:

- `Success`— La solicitud era válida, el componente la envió a la red Modbus RTU y la red Modbus RTU devolvió una respuesta.
- `Exception`— La solicitud era válida, el componente la envió a la red Modbus RTU y la red Modbus RTU devolvió una excepción. Para obtener más información, consulte Estado de respuesta: excepción.
- `No Response`— La solicitud no era válida y el componente detectó el error antes de enviarla a la red Modbus RTU. Para obtener más información, consulte Estado de respuesta: sin respuesta.

`operation`

La operación que solicitó el componente.

`device`

El dispositivo al que el componente envió la solicitud.

`payload`

La respuesta del dispositivo Modbus RTU. Si `status` es así`No Response`, este objeto contiene solo una `error` propiedad con la descripción del error (por ejemplo,`[Input/Output] No Response received from the remote unit`).

`id`

El identificador de la solicitud, que puede utilizar para identificar qué respuesta corresponde a qué solicitud.

> **Note**
>
> Una respuesta para una operación de escritura es simplemente un eco de la solicitud. Si bien las respuestas escritas no incluyen información significativa, se recomienda comprobar el estado de la respuesta para ver si la solicitud se realiza correctamente o no.

Example Ejemplo de salida: Correcto

```
{
  "response" : {
    "status" : "success",
    "device": 1,
    "operation": "ReadCoilsRequest",
    "payload": {
      "function_code": 1,
      "bits": [1]
    }
  },
  "id" : "MyRequest"
}
```

Example Ejemplo de salida: Error

```
{
  "response" : {
    "status" : "fail",
    "error_message": "Internal Error",
    "error": "Exception",
    "device": 1,
    "operation": "ReadCoilsRequest",
    "payload": {
      "function_code": 129,
      "exception_code": 2
    }
  },
  "id" : "MyRequest"
}
```

Para obtener más ejemplos, consulte Solicitudes y respuestas de ejemplo.

Solicitudes y respuestas de Modbus RTU

Este conector acepta los parámetros de solicitud de Modbus RTU como datos de entrada y publica las respuestas como datos de salida.

Se admiten las siguientes operaciones comunes.

Nombre de la operación en la solicitud	Código de característica en la respuesta
ReadCoilsRequest	01
ReadDiscreteInputsRequest	02
ReadHoldingRegistersRequest	03
ReadInputRegistersRequest	04
WriteSingleCoilRequest	05
WriteSingleRegisterRequest	06
WriteMultipleCoilsRequest	15
WriteMultipleRegistersRequest	16
MaskWriteRegisterRequest	22
ReadWriteMultipleRegistersRequest	23

Solicitudes y respuestas de ejemplo

A continuación, se muestran ejemplos de solicitudes y respuestas de las operaciones compatibles.

Bobinas de lectura

Ejemplo de solicitud:

```
{
  "request": {
    "operation": "ReadCoilsRequest",
    "device": 1,
    "address": 1,
    "count": 1
  },
  "id": "TestRequest"
}
```

Ejemplo de respuesta:

```
{
  "response": {
    "status": "success",
    "device": 1,
    "operation": "ReadCoilsRequest",
    "payload": {
      "function_code": 1,
      "bits": [1]
    }
  },
  "id" : "TestRequest"
}
```

Lea entradas discretas

Ejemplo de solicitud:

```
{
  "request": {
    "operation": "ReadDiscreteInputsRequest",
    "device": 1,
    "address": 1,
    "count": 1
  },
  "id": "TestRequest"
}
```

Ejemplo de respuesta:

```
{
  "response": {
    "status": "success",
    "device": 1,
    "operation": "ReadDiscreteInputsRequest",
    "payload": {
      "function_code": 2,
      "bits": [1]
    }
  },
  "id" : "TestRequest"
}
```

Lea los registros de retención

Ejemplo de solicitud:

```
{
  "request": {
    "operation": "ReadHoldingRegistersRequest",
    "device": 1,
    "address": 1,
    "count": 1
  },
  "id": "TestRequest"
}
```

Ejemplo de respuesta:

```
{
  "response": {
    "status": "success",
    "device": 1,
    "operation": "ReadHoldingRegistersRequest",
    "payload": {
      "function_code": 3,
      "registers": [20,30]
    }
  },
  "id" : "TestRequest"
}
```

Lea los registros de entrada

Ejemplo de solicitud:

```
{
  "request": {
    "operation": "ReadInputRegistersRequest",
    "device": 1,
    "address": 1,
    "count": 1
  },
  "id": "TestRequest"
}
```

Escribe una sola bobina

Ejemplo de solicitud:

```
{
  "request": {
    "operation": "WriteSingleCoilRequest",
    "device": 1,
    "address": 1,
    "value": 1
  },
  "id": "TestRequest"
}
```

Ejemplo de respuesta:

```
{
  "response": {
    "status": "success",
    "device": 1,
    "operation": "WriteSingleCoilRequest",
    "payload": {
      "function_code": 5,
      "address": 1,
      "value": true
    }
  },
  "id" : "TestRequest"
}
```

Escribe un registro único

Ejemplo de solicitud:

```
{
  "request": {
    "operation": "WriteSingleRegisterRequest",
    "device": 1,
    "address": 1,
    "value": 1
  },
  "id": "TestRequest"
```

```
}
```

Escribe varias bobinas

Ejemplo de solicitud:

```
{
  "request": {
    "operation": "WriteMultipleCoilsRequest",
    "device": 1,
    "address": 1,
    "values": [1,0,0,1]
  },
  "id": "TestRequest"
}
```

Ejemplo de respuesta:

```
{
  "response": {
    "status": "success",
    "device": 1,
    "operation": "WriteMultipleCoilsRequest",
    "payload": {
      "function_code": 15,
      "address": 1,
      "count": 4
    }
  },
  "id" : "TestRequest"
}
```

Escribe varios registros

Ejemplo de solicitud:

```
{
  "request": {
    "operation": "WriteMultipleRegistersRequest",
    "device": 1,
    "address": 1,
    "values": [20,30,10]
  },
```

```
  "id": "TestRequest"
}
```

Ejemplo de respuesta:

```
{
  "response": {
    "status": "success",
    "device": 1,
    "operation": "WriteMultipleRegistersRequest",
    "payload": {
      "function_code": 23,
      "address": 1,
      "count": 3
    }
  },
  "id" : "TestRequest"
}
```

Enmascarar, escribir registro

Ejemplo de solicitud:

```
{
  "request": {
    "operation": "MaskWriteRegisterRequest",
    "device": 1,
    "address": 1,
    "and_mask": 175,
    "or_mask": 1
  },
  "id": "TestRequest"
}
```

Ejemplo de respuesta:

```
{
  "response": {
    "status": "success",
    "device": 1,
    "operation": "MaskWriteRegisterRequest",
    "payload": {
      "function_code": 22,
```

```
      "and_mask": 0,
      "or_mask": 8
    }
  },
  "id" : "TestRequest"
}
```

Lectura y escritura de varios registros

Ejemplo de solicitud:

```
{
  "request": {
    "operation": "ReadWriteMultipleRegistersRequest",
    "device": 1,
    "read_address": 1,
    "read_count": 2,
    "write_address": 3,
    "write_registers": [20,30,40]
  },
  "id": "TestRequest"
}
```

Ejemplo de respuesta:

```
{
  "response": {
    "status": "success",
    "device": 1,
    "operation": "ReadWriteMultipleRegistersRequest",
    "payload": {
      "function_code": 23,
      "registers": [10,20,10,20]
    }
  },
  "id" : "TestRequest"
}
```

> **Note**
>
> La respuesta incluye los registros que lee el componente.

Estado de respuesta: excepción

Las excepciones pueden producirse cuando el formato de la solicitud es válido, pero la solicitud no se completó correctamente. En este caso, la respuesta contiene la siguiente información:

- `status` se establece en `Exception`.
- `function_code` equivale al código de la característica de la solicitud + 128.
- `exception_code` contiene el código de excepción. Para obtener más información, consulte los códigos de excepción de Modbus.

Ejemplo:

```
{
  "response": {
    "status": "fail",
    "error_message": "Internal Error",
    "error": "Exception",
    "device": 1,
    "operation": "ReadCoilsRequest",
    "payload": {
      "function_code": 129,
      "exception_code": 2
    }
  },
  "id": "TestRequest"
}
```

Estado de respuesta: sin respuesta

Este conector realiza las comprobaciones de validación de la solicitud de Modbus. Por ejemplo, comprueba los formatos no válidos y los campos que faltan. Si no se supera la validación, el conector no envía la solicitud. En su lugar, devuelve una respuesta que contiene la siguiente información:

- `status` se establece en `No Response`.
- `error` contiene el motivo del error.
- `error_message` contiene el mensaje de error.

Ejemplos:

```
{
  "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"
}
```

Si la solicitud selecciona como destino un dispositivo inexistente o si la red de Modbus RTU no funciona, es posible que aparezca una respuesta `ModbusIOException`, que utiliza el formato de respuesta No.

```
{
  "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"
}
```

Archivo de registro local

Este componente utiliza el siguiente archivo de registro.

```
/greengrass/v2/logs/aws.greengrass.Modbus.log
```

Para ver los registros de este componente

- Ejecute el siguiente comando en el dispositivo principal para ver el archivo de registro de este componente en tiempo real. */greengrass/v2*Sustitúyalo por la ruta a la carpeta AWS IoT Greengrass raíz.

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

Licencias

Este componente incluye el siguiente software o licencias de terceros:

- Licencia pymodbus /BSD
- Licencia pyserie/BSD

Este componente se publica en virtud del contrato de licencia de software principal de Greengrass.

Registros de cambios

En la siguiente tabla se describen los cambios en cada versión del componente.

Versión	Cambios
2.1.8	Versión actualizada para la versión 2.12.0 de Greengrass Nucleus.
2.1.7	Versión actualizada para el lanzamiento de la versión 2.11.0 de Greengrass nucleus.
2.1.6	Versión actualizada para el lanzamiento de la versión 2.10.0 de Greengrass nucleus.
2.1.5	Mejoras y correcciones de errores • Corrige un problema con la `ReadDiscreteInput` operación.
2.1.4	Versión actualizada para la versión 2.9.0 de Greengrass Nucleus.
2.1.3	Versión actualizada para el lanzamiento de la versión 2.8.0 de Greengrass nucleus.

Versión	Cambios
2.1.2	Versión actualizada para el lanzamiento de la versión 2.7.0 de Greengrass nucleus.
2.1.1	Versión actualizada para la versión 2.6.0 de Greengrass Nucleus.
2.1.0	Nuevas características • Añade las `ModbusStopBits` opciones`ModbusBaudRate`, `ModbusByteSize` `ModbusParity`, y que se pueden especificar para configurar la comunicación en serie con los dispositivos Modbus RTU.
2.0.8	Versión actualizada para el lanzamiento de la versión 2.5.0 de Greengrass nucleus.
2.0.7	Versión actualizada para la versión 2.4.0 de Greengrass Nucleus.
2.0.6	Versión actualizada para el lanzamiento de la versión 2.3.0 de Greengrass nucleus.
2.0.5	Versión actualizada para el lanzamiento de la versión 2.2.0 de Greengrass nucleus.
2.0.4	Versión actualizada para el lanzamiento de la versión 2.1.0 de Greengrass nucleus.
2.0.3	Versión inicial.

Puente MQTT

El componente puente MQTT (`aws.greengrass.clientdevices.mqtt.Bridge`) transmite mensajes MQTT entre los dispositivos cliente, la publicación/suscripción local de Greengrass y. AWS IoT Core Puede utilizar este componente para actuar sobre los mensajes MQTT de los dispositivos cliente en componentes personalizados y sincronizar los dispositivos cliente con los. Nube de AWS

> **Note**
>
> Los dispositivos cliente son dispositivos IoT locales que se conectan a un dispositivo central de Greengrass para enviar mensajes MQTT y datos para su procesamiento. Para obtener más información, consulte Interactúa con dispositivos IoT locales.

Puede usar este componente para retransmitir mensajes entre los siguientes agentes de mensajes:

- MQTT local: el intermediario MQTT local gestiona los mensajes entre los dispositivos cliente y un dispositivo principal.
- Publicación/suscripción local: el agente de mensajes local de Greengrass gestiona los mensajes entre los componentes de un dispositivo central. Para obtener más información sobre cómo interactuar con estos mensajes en los componentes de Greengrass, consulte. Publicar/suscribir mensajes locales
- AWS IoT Core — El broker AWS IoT Core MQTT gestiona los mensajes entre dispositivos y Nube de AWS destinos de IoT. Para obtener más información sobre cómo interactuar con estos mensajes en los componentes de Greengrass, consulte. Publicar/suscribir mensajes MQTT AWS IoT Core

  > **Note**
  >
  > El puente MQTT usa QoS 1 para publicar y AWS IoT Core suscribirse, incluso cuando un dispositivo cliente usa QoS 0 para publicar y suscribirse al broker MQTT local. Como resultado, es posible que observe una latencia adicional al retransmitir mensajes MQTT desde los dispositivos cliente del broker MQTT local. AWS IoT Core Para obtener más información sobre la configuración de MQTT en los dispositivos principales, consulte. Configure los tiempos de espera y los ajustes de caché de MQTT

Temas

- Versiones
- Tipo
- Sistema operativo
- Requisitos
- Dependencias

- Configuración
- Archivo de registro local
- Registros de cambios

Versiones

Este componente tiene las siguientes versiones:

- 2.3.x
- 2.2.x
- 2.1.x
- 2.0.x

Tipo

Este componente es un componente de complemento (`aws.greengrass.plugin`). El núcleo de Greengrass ejecuta este componente en la misma máquina virtual Java (JVM) que el núcleo. El núcleo se reinicia al cambiar la versión de este componente en el dispositivo principal.

Este componente utiliza el mismo archivo de registro que el núcleo de Greengrass. Para obtener más información, consulte Supervisar AWS IoT Greengrass registros.

Para obtener más información, consulte Tipos de componentes.

Sistema operativo

Este componente se puede instalar en los dispositivos principales que ejecutan los siguientes sistemas operativos:

- Linux
- Windows

Requisitos

Este componente tiene los siguientes requisitos:

- Si configura el componente intermediario MQTT del dispositivo principal para que utilice un puerto que no sea el puerto 8883 predeterminado, debe usar MQTT bridge v2.1.0 o posterior. Configúrelo para que se conecte al puerto en el que opera el intermediario.
- Se admite la ejecución del componente puente MQTT en una VPC.

Dependencias

Al implementar un componente, AWS IoT Greengrass también despliega versiones compatibles de sus dependencias. Esto significa que debe cumplir los requisitos del componente y de todas sus dependencias para poder implementarlo correctamente. En esta sección se enumeran las dependencias de las versiones publicadas de este componente y las restricciones de las versiones semánticas que definen las versiones de los componentes para cada dependencia. También puede ver las dependencias de cada versión del componente en la consola.AWS IoT Greengrass En la página de detalles del componente, busque la lista de dependencias.

2.3.2

La siguiente tabla muestra las dependencias de la versión 2.3.2 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Autenticación del dispositivo cliente	>=2.2.0 <2.6.0	Rígido

2.3.0 and 2.3.1

La siguiente tabla muestra las dependencias de las versiones 2.3.0 y 2.3.1 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Autenticación del dispositivo cliente	>=2.2.0 <2.5.0	Rígido

2.2.5 and 2.2.6

La siguiente tabla muestra las dependencias de las versiones 2.2.5 y 2.2.6 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Autenticación del dispositivo cliente	>=2.2.0 <2.5.0	Rígido

2.2.3 and 2.2.4

La siguiente tabla muestra las dependencias de las versiones 2.2.3 y 2.2.4 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Autenticación del dispositivo cliente	>=2.2.0 <2.4.0	Rígido

2.2.0 – 2.2.2

La siguiente tabla muestra las dependencias de las versiones 2.2.0 a 2.2.2 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Autenticación del dispositivo cliente	>=2.2.0 <2.3.0	Rígido

2.1.1

La siguiente tabla muestra las dependencias de la versión 2.1.1 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Autenticación del dispositivo cliente	>=2.0.0 <2.2.0	Rígido

2.0.0 to 2.1.0

La siguiente tabla muestra las dependencias de las versiones 2.0.0 a 2.1.0 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Autenticación del dispositivo cliente	>=2.0.0 <2.1.0	Rígido

Para obtener más información sobre las dependencias de los componentes, consulta la referencia de recetas de componentes.

Configuración

Este componente proporciona los siguientes parámetros de configuración que puede personalizar al implementar el componente.

2.3.0 – 2.3.2

`mqttTopicMapping`

Las asignaciones de temas que desea unir. Este componente se suscribe a los mensajes del tema de origen y publica los mensajes que recibe en el tema de destino. Cada mapeo de temas define el tema, el tipo de fuente y el tipo de destino.

Este objeto contiene la siguiente información:

`topicMappingNameKey`

El nombre de este mapeo de temas. Sustituya *`topicMappingNameKey`* por un nombre que le ayude a identificar este mapeo de temas.

Este objeto contiene la siguiente información:

`topic`

El tema o filtro de temas que sirve de puente entre los corredores de origen y de destino.

Puede utilizar los caracteres comodín de temas + y de # MQTT para transmitir mensajes sobre todos los temas que coincidan con un filtro de temas. Para obtener más información, consulte los temas de MQTT en la AWS IoT Core Guía para desarrolladores.

> **Note**
>
> Para usar comodines de temas de MQTT con el agente de `Pubsub` código fuente, debe usar la versión 2.6.0 o posterior del componente núcleo de Greengrass.

`targetTopicPrefix`

El prefijo que se añade al tema de destino cuando este componente transmite el mensaje.

`source`

El intermediario de mensajes de origen. Puede elegir entre las siguientes opciones:

- `LocalMqtt`— El intermediario MQTT local donde se comunican los dispositivos cliente.
- `Pubsub`— El intermediario local de mensajes de publicación/suscripción de Greengrass.
- `IotCore`— El intermediario de mensajes AWS IoT Core MQTT.

> **Note**
>
> El puente MQTT usa QoS 1 para publicar y AWS IoT Core suscribirse, incluso cuando un dispositivo cliente usa QoS 0 para publicar y suscribirse al broker MQTT local. Como resultado, es posible que observe una latencia adicional al retransmitir mensajes MQTT desde los dispositivos cliente del broker MQTT local. AWS IoT Core Para obtener más información sobre la configuración de MQTT en los dispositivos principales, consulte. Configure los tiempos de espera y los ajustes de caché de MQTT

`source`y `target` debe ser diferente.

`target`

El intermediario de mensajes de destino. Puede elegir entre las siguientes opciones:

- `LocalMqtt`— El intermediario MQTT local donde se comunican los dispositivos cliente.

- `Pubsub`— El intermediario local de mensajes de publicación/suscripción de Greengrass.
- `IotCore`— El intermediario de mensajes AWS IoT Core MQTT.

> **Note**
>
> El puente MQTT usa QoS 1 para publicar y AWS IoT Core suscribirse, incluso cuando un dispositivo cliente usa QoS 0 para publicar y suscribirse al broker MQTT local. Como resultado, es posible que observe una latencia adicional al retransmitir mensajes MQTT desde los dispositivos cliente del broker MQTT local. AWS IoT Core Para obtener más información sobre la configuración de MQTT en los dispositivos principales, consulte. Configure los tiempos de espera y los ajustes de caché de MQTT

`sourcey target` debe ser diferente.

mqtt5 RouteOptions

(Opcional) Proporciona opciones para configurar las asignaciones de temas para unir los mensajes del tema de origen al tema de destino.

Este objeto contiene la siguiente información:

`mqtt5 RouteOptionsNameKey`

El nombre de las opciones de ruta para el mapeo de un tema. Sustituya *`mqtt5 RouteOptionsNameKey`* por la *`topicMappingNameclave`* correspondiente definida en el `mqttTopicMapping` campo.

Este objeto contiene la siguiente información:

No es local

(Opcional) Cuando está activado, el puente no reenvía los mensajes sobre un tema que haya publicado el propio puente. Utilice esta opción para evitar los bucles, de la siguiente manera:

```
{
    "mqtt5RouteOptions": {
        "toIoTCore": {
```

```
            "noLocal": true
        }
    },
    "mqttTopicMapping": {
        "toIoTCore": {
            "topic": "device",
            "source": "LocalMqtt",
            "target": "IotCore"
        },
        "toLocal": {
            "topic": "device",
            "source": "IotCore",
            "target": "LocalMqtt"
        }
    }
}
```

`noLocal`solo se admite en las rutas en las que `source` está`LocalMqtt`.

Predeterminado: false

retainAsPublished

(Opcional) Cuando está habilitada, los mensajes reenviados por el puente tienen la misma `retain` marca que los mensajes publicados en el intermediario para esa ruta.

`retainAsPublished`solo se admite en las rutas en las que `source` está`LocalMqtt`.

Predeterminado: false

mqtt

(Opcional) Configuración del protocolo MQTT para comunicarse con el intermediario local.

versión

(Opcional) La versión del protocolo MQTT utilizada por el puente para comunicarse con el intermediario local. Debe ser la misma que la versión de MQTT seleccionada en la configuración del núcleo.

Elija una de las siguientes opciones:

- `mqtt3`
- `mqtt5`

Debe implementar un agente MQTT cuando el `target` campo `source` o del `mqttTopicMapping` objeto esté establecido en. `LocalMqtt` Si elige `mqtt5` esta opción, debe usar laBróker MQTT 5 (EMQX).

Valor predeterminado: `mqtt3`

ackTimeoutSeconds

(Opcional) Intervalo de tiempo para esperar a que lleguen los paquetes PUBACK, SUBACK o UNSUBACK antes de que se produzca un error en la operación.

Predeterminado: 60

connAckTimeoutSra

(Opcional) Intervalo de tiempo para esperar un paquete CONNACK antes de cerrar la conexión.

Predeterminado: 20000 (20 segundos)

pingTimeoutMs

(Opcional) El tiempo en milisegundos que el puente espera a recibir un mensaje PINGACK del intermediario local. Si la espera supera el tiempo de espera, el puente se cierra y vuelve a abrir la conexión MQTT. Este valor debe ser inferior a. `keepAliveTimeoutSeconds`

Predeterminado: 30000 (30 segundos)

keepAliveTimeoutSegundos

(Opcional) El tiempo en segundos que transcurre entre cada mensaje PING que envía el puente para mantener activa la conexión MQTT. Este valor debe ser superior a. `pingTimeoutMs`

Predeterminado: 60

maxReconnectDelaySrta.

(Opcional) El tiempo máximo en segundos que tarda MQTT en volver a conectarse.

Predeterminado: 30000 (30 segundos)

minReconnectDelaySrta.

(Opcional) El tiempo mínimo en segundos para que MQTT se vuelva a conectar.

Reciba el máximo

(Opcional) El número máximo de paquetes QoS1 no confirmados que el puente puede enviar.

Predeterminado: 100

maximumPacketSize

El número máximo de bytes que el cliente aceptará para un paquete MQTT.

Predeterminado: nulo (sin límite)

sessionExpiryInterval

(Opcional) La cantidad de tiempo en segundos que puede solicitar para que dure una sesión entre el puente y el intermediario local.

Predeterminado: 4294967295 (la sesión nunca caduca)

`brokerUri`

(Opcional) El URI del broker MQTT local. Debe especificar este parámetro si configura el broker MQTT para que utilice un puerto diferente al puerto predeterminado 8883. Utilice el siguiente formato y sustituya el puerto por el *`puerto`* en el que opera el corredor MQTT:. `ssl://localhost:`*`port`*

Valor predeterminado: `ssl://localhost:8883`

startupTimeoutSeconds

(Opcional) El tiempo máximo en segundos para que se inicie el componente. El estado del componente cambia a `BROKEN` si supera este tiempo de espera.

Valor predeterminado: `120`

Example Ejemplo: actualización de la combinación de configuraciones

El siguiente ejemplo de actualización de configuración especifica lo siguiente:

- Transmita mensajes desde los dispositivos cliente a AWS IoT Core temas que coincidan con el filtro de `clients/+/hello/world` temas.
- Transmita los mensajes desde los dispositivos cliente a los usuarios locales, publique o suscriba los temas que coincidan con el filtro de temas y añada el `events/input/` prefijo al

`clients/+/detections` tema de destino. El tema de destino resultante coincide con el filtro de temas. `events/input/clients/+/detections`

- Transmita los mensajes de los dispositivos cliente a los AWS IoT Core temas que coincidan con el `clients/+/status` filtro de temas y añada el `$aws/rules/StatusUpdateRule/` prefijo al tema de destino. En este ejemplo, se transmiten estos mensajes directamente a una AWS IoT regla denominada `StatusUpdateRule` reducción de costes mediante 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 Ejemplo: configurar MQTT 5

El siguiente ejemplo de configuración actualiza lo siguiente:

- Permite que el puente utilice el protocolo MQTT 5 con el intermediario local.
- Configura MQTT keep como configuración publicada para la `ClientDeviceHelloWorld` asignación de temas.

```
{
  "mqttTopicMapping": {
```

```
    "ClientDeviceHelloWorld": {
      "topic": "clients/+/hello/world",
      "source": "LocalMqtt",
      "target": "IotCore"
    }
  },
  "mqtt5RouteOptions": {
    "ClientDeviceHelloWorld": {
      "retainAsPublished": true
    }
  },
  "mqtt": {
    "version": "mqtt5"
  }
}
```

2.2.6

`mqttTopicMapping`

Los mapeos de temas que desea unir. Este componente se suscribe a los mensajes del tema de origen y publica los mensajes que recibe en el tema de destino. Cada mapeo de temas define el tema, el tipo de fuente y el tipo de destino.

Este objeto contiene la siguiente información:

`topicMappingNameKey`

El nombre de este mapeo de temas. Sustituya *`topicMappingNameKey`* por un nombre que le ayude a identificar este mapeo de temas.

Este objeto contiene la siguiente información:

`topic`

El tema o filtro de temas que sirve de puente entre los corredores de origen y de destino.

Puede utilizar los caracteres comodín de temas + y de # MQTT para transmitir mensajes sobre todos los temas que coincidan con un filtro de temas. Para obtener más información, consulte los temas de MQTT en la AWS IoT Core Guía para desarrolladores.

> **Note**
>
> Para usar comodines de temas de MQTT con el agente de Pubsub código fuente, debe usar la versión 2.6.0 o posterior del componente núcleo de Greengrass.

`targetTopicPrefix`

El prefijo que se añade al tema de destino cuando este componente transmite el mensaje.

`source`

El intermediario de mensajes de origen. Puede elegir entre las siguientes opciones:

- `LocalMqtt`— El intermediario MQTT local donde se comunican los dispositivos cliente.
- `Pubsub`— El intermediario local de mensajes de publicación/suscripción de Greengrass.
- `IotCore`— El intermediario de mensajes AWS IoT Core MQTT.

> **Note**
>
> El puente MQTT usa QoS 1 para publicar y AWS IoT Core suscribirse, incluso cuando un dispositivo cliente usa QoS 0 para publicar y suscribirse al broker MQTT local. Como resultado, es posible que observe una latencia adicional al retransmitir mensajes MQTT desde los dispositivos cliente del broker MQTT local. AWS IoT Core Para obtener más información sobre la configuración de MQTT en los dispositivos principales, consulte. Configure los tiempos de espera y los ajustes de caché de MQTT

`source`y `target` debe ser diferente.

`target`

El intermediario de mensajes de destino. Puede elegir entre las siguientes opciones:

- `LocalMqtt`— El intermediario MQTT local donde se comunican los dispositivos cliente.

- `Pubsub`— El intermediario local de mensajes de publicación/suscripción de Greengrass.
- `IotCore`— El intermediario de mensajes AWS IoT Core MQTT.

> **Note**
>
> El puente MQTT usa QoS 1 para publicar y AWS IoT Core suscribirse, incluso cuando un dispositivo cliente usa QoS 0 para publicar y suscribirse al broker MQTT local. Como resultado, es posible que observe una latencia adicional al retransmitir mensajes MQTT desde los dispositivos cliente del broker MQTT local. AWS IoT Core Para obtener más información sobre la configuración de MQTT en los dispositivos principales, consulte. Configure los tiempos de espera y los ajustes de caché de MQTT

`source` y `target` debe ser diferente.

`brokerUri`

(Opcional) El URI del broker MQTT local. Debe especificar este parámetro si configura el broker MQTT para que utilice un puerto diferente al puerto predeterminado 8883. Utilice el siguiente formato y sustituya el puerto por el *`puerto`* en el que opera el corredor MQTT:. `ssl://localhost:`*`port`*

Valor predeterminado: `ssl://localhost:8883`

startupTimeoutSeconds

(Opcional) El tiempo máximo en segundos para que se inicie el componente. El estado del componente cambia a `BROKEN` si supera este tiempo de espera.

Valor predeterminado: `120`

Example Ejemplo: actualización de la combinación de configuraciones

El siguiente ejemplo de actualización de configuración especifica lo siguiente:

- Transmita mensajes desde los dispositivos cliente a AWS IoT Core temas que coincidan con el filtro de `clients/+/hello/world` temas.
- Transmita los mensajes desde los dispositivos cliente a los usuarios locales, publique o suscriba los temas que coincidan con el filtro de temas y añada el `events/input/` prefijo al

`clients/+/detections` tema de destino. El tema de destino resultante coincide con el filtro de temas. `events/input/clients/+/detections`

- Transmita los mensajes de los dispositivos cliente a los AWS IoT Core temas que coincidan con el `clients/+/status` filtro de temas y añada el `$aws/rules/StatusUpdateRule/` prefijo al tema de destino. En este ejemplo, se transmiten estos mensajes directamente a una AWS IoT regla denominada `StatusUpdateRule` reducción de costes mediante 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`

Los mapeos de temas que desea unir. Este componente se suscribe a los mensajes del tema de origen y publica los mensajes que recibe en el tema de destino. Cada mapeo de temas define el tema, el tipo de fuente y el tipo de destino.

Este objeto contiene la siguiente información:

`topicMappingNameKey`

El nombre de este mapeo de temas. Sustituya *`topicMappingNameKey`* por un nombre que le ayude a identificar este mapeo de temas.

Este objeto contiene la siguiente información:

`topic`

El tema o filtro de temas que sirve de puente entre los corredores de origen y de destino.

Puede utilizar los caracteres comodín de temas + y de # MQTT para transmitir mensajes sobre todos los temas que coincidan con un filtro de temas. Para obtener más información, consulte los temas de MQTT en la AWS IoT Core Guía para desarrolladores.

> **Note**
>
> Para usar comodines de temas de MQTT con el agente de `Pubsub` código fuente, debe usar la versión 2.6.0 o posterior del componente núcleo de Greengrass.

`targetTopicPrefix`

El prefijo que se añade al tema de destino cuando este componente transmite el mensaje.

`source`

El intermediario de mensajes de origen. Puede elegir entre las siguientes opciones:

- `LocalMqtt`— El intermediario MQTT local donde se comunican los dispositivos cliente.
- `Pubsub`— El intermediario local de mensajes de publicación/suscripción de Greengrass.
- `IotCore`— El intermediario de mensajes AWS IoT Core MQTT.

> **Note**
>
> El puente MQTT usa QoS 1 para publicar y AWS IoT Core suscribirse, incluso cuando un dispositivo cliente usa QoS 0 para publicar y suscribirse al broker MQTT local. Como resultado, es posible que observe una latencia adicional al retransmitir mensajes MQTT desde los dispositivos cliente del broker MQTT local. AWS IoT Core Para obtener más información sobre la configuración de MQTT en los dispositivos principales, consulte. Configure los tiempos de espera y los ajustes de caché de MQTT

`sourcey` `target` debe ser diferente.

`target`

El intermediario de mensajes de destino. Puede elegir entre las siguientes opciones:

- `LocalMqtt`— El intermediario MQTT local donde se comunican los dispositivos cliente.
- `Pubsub`— El intermediario local de mensajes de publicación/suscripción de Greengrass.
- `IotCore`— El intermediario de mensajes AWS IoT Core MQTT.

> **Note**
>
> El puente MQTT usa QoS 1 para publicar y AWS IoT Core suscribirse, incluso cuando un dispositivo cliente usa QoS 0 para publicar y suscribirse al broker MQTT local. Como resultado, es posible que observe una latencia adicional al retransmitir mensajes MQTT desde los dispositivos cliente del broker MQTT local. AWS IoT Core Para obtener más información sobre la configuración de MQTT en los dispositivos principales, consulte. Configure los tiempos de espera y los ajustes de caché de MQTT

`sourcey` `target` debe ser diferente.

`brokerUri`

(Opcional) El URI del broker MQTT local. Debe especificar este parámetro si configura el broker MQTT para que utilice un puerto diferente al puerto predeterminado 8883. Utilice el

siguiente formato y sustituya el puerto por el *puerto* en el que opera el corredor MQTT:. `ssl://localhost:`*port*

Valor predeterminado: `ssl://localhost:8883`

Example Ejemplo: actualización de combinación de configuraciones

El siguiente ejemplo de actualización de configuración especifica lo siguiente:

- Transmita mensajes desde los dispositivos cliente a AWS IoT Core temas que coincidan con el filtro de `clients/+/hello/world` temas.
- Transmita los mensajes desde los dispositivos cliente a los usuarios locales, publique o suscriba los temas que coincidan con el filtro de temas y añada el `events/input/` prefijo al `clients/+/detections` tema de destino. El tema de destino resultante coincide con el filtro de temas. `events/input/clients/+/detections`
- Transmita los mensajes de los dispositivos cliente a los AWS IoT Core temas que coincidan con el `clients/+/status` filtro de temas y añada el `$aws/rules/StatusUpdateRule/` prefijo al tema de destino. En este ejemplo, se transmiten estos mensajes directamente a una AWS IoT regla denominada `StatusUpdateRule` reducción de costes mediante 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`

Los mapeos de temas que desea unir. Este componente se suscribe a los mensajes del tema de origen y publica los mensajes que recibe en el tema de destino. Cada mapeo de temas define el tema, el tipo de fuente y el tipo de destino.

Este objeto contiene la siguiente información:

`topicMappingNameKey`

El nombre de este mapeo de temas. Sustituya *`topicMappingNameKey`* por un nombre que le ayude a identificar este mapeo de temas.

Este objeto contiene la siguiente información:

`topic`

El tema o filtro de temas que sirve de puente entre los corredores de origen y de destino.

Si especifica el intermediario `LocalMqtt` o el intermediario `IotCore` fuente, puede utilizar los caracteres comodín de temas + y # MQTT para retransmitir mensajes sobre todos los temas que coincidan con un filtro de temas. Para obtener más información, consulte los temas de MQTT en la AWS IoT Core Guía para desarrolladores.

`source`

El agente de mensajes de origen. Puede elegir entre las siguientes opciones:

- `LocalMqtt`— El intermediario MQTT local donde se comunican los dispositivos cliente.
- `Pubsub`— El intermediario local de mensajes de publicación/suscripción de Greengrass.
- `IotCore`— El intermediario de mensajes AWS IoT Core MQTT.

> **Note**
>
> El puente MQTT usa QoS 1 para publicar y AWS IoT Core suscribirse, incluso cuando un dispositivo cliente usa QoS 0 para publicar y suscribirse

al broker MQTT local. Como resultado, es posible que observe una latencia adicional al retransmitir mensajes MQTT desde los dispositivos cliente del broker MQTT local. AWS IoT Core Para obtener más información sobre la configuración de MQTT en los dispositivos principales, consulte. Configure los tiempos de espera y los ajustes de caché de MQTT

`source` y `target` debe ser diferente.

`target`

El intermediario de mensajes de destino. Puede elegir entre las siguientes opciones:

- `LocalMqtt`— El intermediario MQTT local donde se comunican los dispositivos cliente.
- `Pubsub`— El intermediario local de mensajes de publicación/suscripción de Greengrass.
- `IotCore`— El intermediario de mensajes AWS IoT Core MQTT.

> **Note**
>
> El puente MQTT usa QoS 1 para publicar y AWS IoT Core suscribirse, incluso cuando un dispositivo cliente usa QoS 0 para publicar y suscribirse al broker MQTT local. Como resultado, es posible que observe una latencia adicional al retransmitir mensajes MQTT desde los dispositivos cliente del broker MQTT local. AWS IoT Core Para obtener más información sobre la configuración de MQTT en los dispositivos principales, consulte. Configure los tiempos de espera y los ajustes de caché de MQTT

`source` y `target` debe ser diferente.

`brokerUri`

(Opcional) El URI del broker MQTT local. Debe especificar este parámetro si configura el broker MQTT para que utilice un puerto diferente al puerto predeterminado 8883. Utilice el siguiente formato y sustituya el puerto por el *`puerto`* en el que opera el corredor MQTT:. `ssl://localhost:`*`port`*

Valor predeterminado: `ssl://localhost:8883`

Example Ejemplo: actualización de combinación de configuraciones

El siguiente ejemplo de actualización de configuración especifica la retransmisión de mensajes desde los dispositivos cliente AWS IoT Core a los `clients/MyClientDevice2/hello/world` temas `clients/MyClientDevice1/hello/world` y.

```
{
  "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`

Las asignaciones de temas que desea unir. Este componente se suscribe a los mensajes del tema de origen y publica los mensajes que recibe en el tema de destino. Cada mapeo de temas define el tema, el tipo de fuente y el tipo de destino.

Este objeto contiene la siguiente información:

`topicMappingNameKey`

El nombre de este mapeo de temas. Sustituya *`topicMappingNameKey`* por un nombre que le ayude a identificar este mapeo de temas.

Este objeto contiene la siguiente información:

`topic`

El tema o filtro de temas que sirve de puente entre los corredores de origen y de destino.

Si especifica el intermediario `LocalMqtt` o el intermediario `IotCore` fuente, puede utilizar los caracteres comodín de temas + y # MQTT para retransmitir mensajes sobre todos los temas que coincidan con un filtro de temas. Para obtener más información, consulte los temas de MQTT en la AWS IoT Core Guía para desarrolladores.

`source`

El agente de mensajes de origen. Puede elegir entre las siguientes opciones:

- `LocalMqtt`— El intermediario MQTT local donde se comunican los dispositivos cliente.
- `Pubsub`— El intermediario local de mensajes de publicación/suscripción de Greengrass.
- `IotCore`— El intermediario de mensajes AWS IoT Core MQTT.

> **Note**
>
> El puente MQTT usa QoS 1 para publicar y AWS IoT Core suscribirse, incluso cuando un dispositivo cliente usa QoS 0 para publicar y suscribirse al broker MQTT local. Como resultado, es posible que observe una latencia adicional al retransmitir mensajes MQTT desde los dispositivos cliente del broker MQTT local. AWS IoT Core Para obtener más información sobre la configuración de MQTT en los dispositivos principales, consulte. Configure los tiempos de espera y los ajustes de caché de MQTT

`source`y `target` debe ser diferente.

`target`

El intermediario de mensajes de destino. Puede elegir entre las siguientes opciones:

- `LocalMqtt`— El intermediario MQTT local donde se comunican los dispositivos cliente.
- `Pubsub`— El intermediario local de mensajes de publicación/suscripción de Greengrass.
- `IotCore`— El intermediario de mensajes AWS IoT Core MQTT.

> **Note**
>
> El puente MQTT usa QoS 1 para publicar y AWS IoT Core suscribirse, incluso cuando un dispositivo cliente usa QoS 0 para publicar y suscribirse al broker MQTT local. Como resultado, es posible que observe una latencia adicional al retransmitir mensajes MQTT desde los dispositivos cliente del broker MQTT local. AWS IoT Core Para obtener más información sobre la configuración de MQTT en los dispositivos principales, consulte. Configure los tiempos de espera y los ajustes de caché de MQTT

`sourcey target` debe ser diferente.

Example Ejemplo: actualización de combinación de configuraciones

El siguiente ejemplo de actualización de configuración especifica la retransmisión de mensajes desde los dispositivos cliente AWS IoT Core a los `clients/MyClientDevice2/hello/world` temas `clients/MyClientDevice1/hello/world` y.

```
{
  "mqttTopicMapping": {
    "ClientDevice1HelloWorld": {
      "topic": "clients/MyClientDevice1/hello/world",
      "source": "LocalMqtt",
      "target": "IotCore"
    },
    "ClientDevice2HelloWorld": {
      "topic": "clients/MyClientDevice2/hello/world",
      "source": "LocalMqtt",
      "target": "IotCore"
    }
  }
}
```

Archivo de registro local

Este componente utiliza el mismo archivo de registro que el componente núcleo de Greengrass.

Linux

```
/greengrass/v2/logs/greengrass.log
```

Windows

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

Para ver los registros de este componente

- Ejecute el siguiente comando en el dispositivo principal para ver el archivo de registro de este componente en tiempo real. Sustituya */greengrass/v2* o *C:\greengrass\v2* por la ruta a la carpeta AWS IoT Greengrass raíz.

  Linux

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

  Windows (PowerShell)

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

Registros de cambios

En la siguiente tabla se describen los cambios en cada versión del componente.

Versión	Cambios
2.3.2	Versión actualizada para la versión 2.5.0 de autenticación de dispositivos cliente.
2.3.1	Mejoras y correcciones de errores Soluciona un problema por el que el cliente MQTT local entra en un bucle de desconexión.

Versión	Cambios
2.3.0	Nuevas características Añade compatibilidad con MQTT5 para conectar fuentes MQTT locales AWS IoT Core y entre ellas.
2.2.6	Nuevas características Añade una nueva opción de configuración. `startupTimeoutSeconds`
2.2.5	Versión actualizada para la versión 2.4.0 de autenticación de dispositivos cliente.
2.2.4	Versión actualizada para la versión 2.3.0 de autenticación de dispositivos cliente de Greengrass.
2.2.3	Esta versión contiene correcciones de errores y mejoras.
2.2.2	Mejoras y correcciones de errores • Ajustes de registro.
2.2.1	Mejoras y correcciones de errores Corrige problemas que podían provocar que MQTT Bridge no pudiera suscribirse a los temas de MQTT.
2.2.0	Nuevas características • Añade compatibilidad con los caracteres comodín (#y+) de los temas MQTT cuando se especifica la publicación o suscripción local como agente de mensajes de origen. Esta función requiere la versión 2.6.0 o posterior del componente núcleo de Greengrass. • Añade la `targetTopicPrefix` opción, que puede especificar, para configurar el puente MQTT de forma que añada un prefijo al tema de destino cuando retransmita un mensaje.

Versión	Cambios
2.1.1	Mejoras y correcciones de errores • Soluciona problemas relacionados con la forma en que este component e gestiona las actualizaciones de restablecimiento de la configuración. • Reduce la frecuencia con la que el cliente MQTT se desconecta cuando los certificados rotan.
2.1.0	Nuevas características • Añade el `brokerUri` parámetro, que le permite utilizar un puerto de intermediario MQTT no predeterminado.
2.0.1	Esta versión incluye correcciones de errores y mejoras.
2.0.0	Versión inicial.

Bróker MQTT 3.1.1 (Moquette)

El componente broker MQTT de Moquette (`aws.greengrass.clientdevices.mqtt.Moquette`) gestiona los mensajes MQTT entre los dispositivos cliente y un dispositivo principal de Greengrass. Este componente proporciona una versión modificada del broker MQTT de Moquette. Implemente este bróker MQTT para ejecutar un bróker MQTT ligero. Para obtener más información sobre cómo elegir un bróker de MQTT, consulte. Elija un bróker MQTT

Este bróker implementa el protocolo MQTT 3.1.1. Incluye compatibilidad con QoS 0, QoS 1, QoS 2, mensajes de última voluntad y sesiones persistentes.

> **Note**
>
> Los dispositivos cliente son dispositivos IoT locales que se conectan a un dispositivo central de Greengrass para enviar mensajes MQTT y datos para su procesamiento. Para obtener más información, consulte Interactúa con dispositivos IoT locales.

Temas

- Versiones
- Tipo

- Sistema operativo
- Requisitos
- Dependencias
- Configuración
- Archivo de registro local
- Registros de cambios

Versiones

Este componente tiene las siguientes versiones:

- 2.3.x
- 2.2.x
- 2.1.x
- 2.0.x

Tipo

Este componente es un componente de complemento ()`aws.greengrass.plugin`. El núcleo de Greengrass ejecuta este componente en la misma máquina virtual Java (JVM) que el núcleo. El núcleo se reinicia al cambiar la versión de este componente en el dispositivo principal.

Este componente utiliza el mismo archivo de registro que el núcleo de Greengrass. Para obtener más información, consulte Supervisar AWS IoT Greengrass registros.

Para obtener más información, consulte Tipos de componentes.

Sistema operativo

Este componente se puede instalar en los dispositivos principales que ejecutan los siguientes sistemas operativos:

- Linux
- Windows

Requisitos

Este componente tiene los siguientes requisitos:

- El dispositivo principal debe poder aceptar conexiones en el puerto en el que opera el broker MQTT. Este componente ejecuta el broker MQTT en el puerto 8883 de forma predeterminada. Puede especificar un puerto diferente al configurar este componente.

  Si especifica un puerto diferente y utiliza el componente de puente MQTT para retransmitir mensajes MQTT a otros intermediarios, debe utilizar MQTT bridge v2.1.0 o posterior. Configúrelo para que utilice el puerto en el que opera el bróker MQTT.

  Si especifica un puerto diferente y utiliza el componente IP detector para gestionar los puntos finales del broker MQTT, debe utilizar IP detector v2.1.0 o una versión posterior. Configúrelo para que indique el puerto en el que opera el broker MQTT.
- Se admite la ejecución del componente broker MQTT de Moquette en una VPC.

Dependencias

Al implementar un componente, AWS IoT Greengrass también despliega versiones compatibles de sus dependencias. Esto significa que debe cumplir los requisitos del componente y de todas sus dependencias para poder implementarlo correctamente. En esta sección se enumeran las dependencias de las versiones publicadas de este componente y las restricciones de las versiones semánticas que definen las versiones de los componentes para cada dependencia. También puede ver las dependencias de cada versión del componente en la consola.AWS IoT Greengrass En la página de detalles del componente, busque la lista de dependencias.

2.3.7

La siguiente tabla muestra las dependencias de la versión 2.3.7 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Autenticación del dispositivo cliente	>=2.2.0 <2.6.0	Rígido

2.3.2 – 2.3.6

La siguiente tabla muestra las dependencias de las versiones 2.3.2 a 2.3.6 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Autenticación del dispositivo cliente	>=2.2.0 <2.5.0	Rígido

2.3.0 and 2.3.1

La siguiente tabla muestra las dependencias de las versiones 2.3.0 y 2.3.1 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Autenticación del dispositivo cliente	>=2.2.0 <2.4.0	Rígido

2.2.0

La siguiente tabla muestra las dependencias de la versión 2.2.0 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Autenticación del dispositivo cliente	>=2.2.0 <2.3.0	Rígido

2.1.0

La siguiente tabla muestra las dependencias de la versión 2.1.0 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Autenticación del dispositivo cliente	>=2.0.0 <2.2.0	Rígido

2.0.0 - 2.0.2

La siguiente tabla muestra las dependencias de las versiones 2.0.0 a 2.0.2 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Autenticación del dispositivo cliente	>=2.0.0 <2.1.0	Rígido

Para obtener más información sobre las dependencias de los componentes, consulta la referencia de recetas de componentes.

Configuración

Este componente proporciona los siguientes parámetros de configuración que puede personalizar al implementar el componente.

`moquette`

(Opcional) La configuración del broker MQTT de Moquette que se va a utilizar. Puede configurar un subconjunto de opciones de configuración de Moquette en este componente. Para obtener más información, consulte los comentarios en línea del archivo de configuración de Moquette.

Este objeto contiene la siguiente información:

`ssl_port`

(Opcional) El puerto en el que opera el bróker MQTT.

> **Note**
>
> Si especifica un puerto diferente y utiliza el componente de puente MQTT para retransmitir mensajes MQTT a otros intermediarios, debe utilizar el puente MQTT v2.1.0 o posterior. Configúrelo para que utilice el puerto en el que opera el bróker MQTT.
> Si especifica un puerto diferente y utiliza el componente IP detector para gestionar los puntos finales del broker MQTT, debe utilizar IP detector v2.1.0 o una versión posterior. Configúrelo para que indique el puerto en el que opera el broker MQTT.

Valor predeterminado: `8883`

`host`

(Opcional) La interfaz a la que se enlaza el bróker MQTT. Por ejemplo, puede cambiar este parámetro para que el corredor MQTT se vincule únicamente a una red local específica.

Predeterminado: `0.0.0.0` (enlaza con todas las interfaces de red)

startupTimeoutSeconds

(Opcional) El tiempo máximo en segundos para que se inicie el componente. El estado del componente cambia a `BROKEN` si supera este tiempo de espera.

Valor predeterminado: `120`

Example Ejemplo: actualización de la combinación de configuraciones

El siguiente ejemplo de configuración especifica el funcionamiento del broker MQTT en el puerto 443.

```
{
  "moquette": {
    "ssl_port": "443"
  }
}
```

Archivo de registro local

Este componente utiliza el mismo archivo de registro que el componente núcleo de Greengrass.

Linux

```
/greengrass/v2/logs/greengrass.log
```

Windows

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

Para ver los registros de este componente

- Ejecute el siguiente comando en el dispositivo principal para ver el archivo de registro de este componente en tiempo real. Sustituya */greengrass/v2* o *C:\greengrass\v2* por la ruta a la carpeta AWS IoT Greengrass raíz.

  Linux

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

  Windows (PowerShell)

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

Registros de cambios

En la siguiente tabla se describen los cambios en cada versión del componente.

Versión	Cambios
2.3.7	Versión actualizada para la versión 2.5.0 de autenticación de dispositivos cliente.
2.3.6	Mejoras y correcciones de errores • Corrección de errores y mejoras generales.
2.3.5	Mejoras y correcciones de errores • Se actualizó Moquette a la versión 0.17.
2.3.4	Mejoras y correcciones de errores • Soluciona un problema por el que los clientes podían experimentar errores de sesión no válidos al enviar o recibir mensajes debido a la duplicación de los ID de cliente. Este problema provocó el cierre de la sesión del cliente.
2.3.3	Nuevas características Añade una nueva opción `startupTimeoutSeconds` de configuración.

Versión	Cambios
2.3.2	Versión actualizada para la versión 2.4.0 de autenticación de dispositivos cliente.
2.3.1	Mejoras y correcciones de errores • Corrige una condición de carrera en la que los clientes podían desconectarse después de intentar volver a conectarse debido a una sesión no válida.
2.3.0	Añade compatibilidad con cadenas de certificados.
2.2.0	Versión actualizada para la versión 2.2.0 de autenticación de dispositivos cliente.
2.1.0	Mejoras y correcciones de errores • Actualiza este componente para que utilice la versión 0.16 de Moquette, que mejora el rendimiento e incluye otras mejoras. • Soluciona un problema por el que el certificado del servidor MQTT local rota con más frecuencia de lo previsto en determinadas situaciones. Para aplicar esta corrección, también debe utilizar la versión 2.1.0 o posterior del componente de autenticación del dispositivo cliente.
2.0.2	Mejoras y correcciones de errores • Aumenta el tamaño máximo de los mensajes MQTT de 8.092 bytes a 128 kilobytes. El límite efectivo de carga útil de los mensajes en MQTT es ligeramente inferior, ya que el límite de tamaño de los mensajes incluye los encabezados de los mensajes. • Añade compatibilidad con valores enteros en el parámetro. `ssl_port`
2.0.1	Versión actualizada para el lanzamiento de la versión 2.4.0 de Greengrass Nucleus.
2.0.0	Versión inicial.

Bróker MQTT 5 (EMQX)

El componente intermediario MQTT de EMQX (`aws.greengrass.clientdevices.mqtt.EMQX`) gestiona los mensajes MQTT entre los dispositivos cliente y un dispositivo principal de Greengrass. Este componente proporciona una versión modificada del broker EMQX MQTT 5.0. Implemente este broker de MQTT para utilizar las funciones de MQTT 5 en la comunicación entre los dispositivos cliente y un dispositivo principal. Para obtener más información sobre cómo elegir un bróker de MQTT, consulte. Elija un bróker MQTT

Este bróker implementa el protocolo MQTT 5.0. Incluye compatibilidad con los intervalos de caducidad de las sesiones y los mensajes, las propiedades de los usuarios, las suscripciones compartidas, los alias de los temas y mucho más. MQTT 5 es compatible con versiones anteriores de MQTT 3.1.1, por lo que si utiliza el bróker MQTT 3.1.1 de Moquette, podrá sustituirlo por el bróker EMQX MQTT 5 y los dispositivos cliente podrán seguir conectándose y funcionando como de costumbre.

> **Note**
>
> Los dispositivos cliente son dispositivos IoT locales que se conectan a un dispositivo central de Greengrass para enviar mensajes MQTT y datos para su procesamiento. Para obtener más información, consulte Interactúa con dispositivos IoT locales.

Temas

Versiones

Este componente tiene las siguientes versiones:

- 2.0.x
- 1.2.x
- 1.1.x
- 1.0.x

Tipo

Este componente es un componente genérico (`aws.greengrass.generic`). El núcleo de Greengrass ejecuta los scripts del ciclo de vida del componente.

Para obtener más información, consulte Tipos de componentes.

Sistema operativo

Este componente se puede instalar en los dispositivos principales que ejecutan los siguientes sistemas operativos:

- Linux
- Windows

Requisitos

Este componente tiene los siguientes requisitos:

- El dispositivo principal debe poder aceptar conexiones en el puerto en el que opera el broker MQTT. Este componente ejecuta el broker MQTT en el puerto 8883 de forma predeterminada. Puede especificar un puerto diferente al configurar este componente.

  Si especifica un puerto diferente y utiliza el componente de puente MQTT para retransmitir mensajes MQTT a otros intermediarios, debe utilizar MQTT bridge v2.1.0 o posterior. Configúrelo para que utilice el puerto en el que opera el broker MQTT.

  Si especifica un puerto diferente y utiliza el componente IP detector para gestionar los puntos finales del broker MQTT, debe utilizar IP detector v2.1.0 o una versión posterior. Configúrelo para que indique el puerto en el que opera el broker MQTT.

- En los dispositivos principales de Linux, Docker se instala y configura en el dispositivo principal:
  - Docker Engine 1.9.1 o posterior instalado en el dispositivo principal de Greengrass. Se ha comprobado que la versión 20.10 es la última versión que funciona con el software Core. AWS IoT Greengrass Debe instalar Docker directamente en el dispositivo principal antes de implementar los componentes que ejecutan contenedores de Docker.
  - El daemon de Docker se inició y se ejecutó en el dispositivo principal antes de implementar este componente.
  - El usuario del sistema que ejecuta este componente debe tener permisos de administrador o root. Como alternativa, puede ejecutar este componente como usuario del sistema en el `docker` grupo y configurar la `requiresPrivileges` opción de este componente `false` para ejecutar el broker MQTT de EQMX sin privilegios.
- Se admite la ejecución del componente broker MQTT de EMQX en una VPC.
- La plataforma no admite el componente de intermediario MQTT de EMQX. `armv7`

Dependencias

Al implementar un componente, AWS IoT Greengrass también despliega versiones compatibles de sus dependencias. Esto significa que debe cumplir los requisitos del componente y de todas sus dependencias para poder implementarlo correctamente. En esta sección se enumeran las dependencias de las versiones publicadas de este componente y las restricciones de las versiones semánticas que definen las versiones de los componentes para cada dependencia. También puede ver las dependencias de cada versión del componente en la consola.AWS IoT Greengrass En la página de detalles del componente, busque la lista de dependencias.

2.0.1

La siguiente tabla muestra las dependencias de la versión 2.0.1 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Autenticación del dispositivo cliente	>=2.2.0 <2.6.0	Rígido

2.0.0

La siguiente tabla muestra las dependencias de la versión 2.0.0 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Autenticación del dispositivo cliente	>=2.2.0 <2.5.0	Rígido

1.2.2 – 1.2.3

La siguiente tabla muestra las dependencias de las versiones 1.2.2 a 1.2.3 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Autenticación del dispositivo cliente	>=2.2.0 <2.5.0	Rígido

1.2.0 and 1.2.1

La siguiente tabla muestra las dependencias de las versiones 1.2.0 y 1.2.1 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Autenticación del dispositivo cliente	>=2.2.0 <2.4.0	Rígido

1.0.0 and 1.1.0

La siguiente tabla muestra las dependencias de las versiones 1.0.0 y 1.1.0 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Autenticación del dispositivo cliente	>=2.2.0 <2.3.0	Rígido

Para obtener más información sobre las dependencias de los componentes, consulta la referencia de recetas de componentes.

Configuración

2.0.0 - 2.0.1

Este componente proporciona los siguientes parámetros de configuración que puede personalizar al implementar el componente.

> **Important**
>
> Si utiliza la versión 2 del componente MQTT 5 broker (EMQX), debe actualizar el archivo de configuración. Los archivos de configuración de la versión 1 no funcionan con la versión 2.

eMqxConfig

(Opcional) La configuración del broker EMQX MQTT que se va a utilizar. Puede configurar las opciones de configuración de EMQX en este componente.

Cuando se utiliza el bróker EMQX, Greengrass utiliza una configuración predeterminada. Esta configuración se utiliza a menos que la modifique mediante este campo.

La modificación de los siguientes ajustes de configuración provoca el reinicio del componente broker EMQX. Se aplican otros cambios de configuración sin reiniciar el componente.

- `emqxConfig/cluster`
- `emqxConfig/node`
- `emqxConfig/rpc`

> **Note**
>
> `aws.greengrass.clientdevices.mqtt.EMQX`le permite configurar opciones sensibles a la seguridad. Estas incluyen la configuración de TLS, la autenticación y los proveedores de autorización. Recomendamos la configuración predeterminada que utiliza la autenticación TLS mutua y el proveedor de autenticación de dispositivos cliente Greengrass.

Example Ejemplo: configuración predeterminada

El siguiente ejemplo muestra los valores predeterminados establecidos para el broker MQTT 5 (EMQX). Puede anular estos parámetros mediante el ajuste de configuración. `emqxConfig`

```
{
  "authorization": {
    "no_match": "deny",
    "sources": []
  },
  "node": {
    "cookie": "<placeholder>"
  },
  "listeners": {
     "ssl": {
       "default": {
         "ssl_options": {
           "keyfile": "{work:path}\\data\\key.pem",
           "certfile": "{work:path}\\data\\cert.pem",
           "cacertfile": null,
           "verify": "verify_peer",
           "versions": ["tlsv1.3", "tlsv1.2"],
           "fail_if_no_peer_cert": true
         }
       }
     },
     "tcp": {
       "default": {
         "enabled": false
       }
     },
     "ws": {
       "default": {
         "enabled": false
       }
     },
     "wss": {
       "default": {
         "enabled": false
       }
     }
  },
  "plugins": {
    "states": [{"name_vsn": "gg-1.0.0", "enable": true}],
```

```
    "install_dir": "plugins"
  }
}
```

AuthMode

(Opcional) Establece el proveedor de autorización del corredor. Puede ser uno de los siguientes valores:

- `enabled`— (Predeterminado) Utilice el proveedor de autenticación y autorización de Greengrass.
- `bypass_on_failure`— Utilice el proveedor de autenticación Greengrass y, a continuación, utilice los demás proveedores de autenticación de la cadena de proveedores de EMQX si Greengrass deniega la autenticación o la autorización.
- `bypass`— El proveedor de Greengrass está deshabilitado. La cadena de proveedores de EMQX gestiona la autenticación y la autorización.

`requiresPrivilege`

(Opcional) En los dispositivos principales de Linux, puede especificar que se ejecute el broker MQTT de EMQX sin privilegios de administrador o root. Si establece esta opción en`false`, el usuario del sistema que ejecute este componente debe ser miembro del grupo. `docker`

Valor predeterminado: `true`

`startupTimeoutSeconds`

(Opcional) El tiempo máximo en segundos para que se inicie el broker MQTT de EMQX. El estado del componente cambia a `BROKEN` si supera este tiempo de espera.

Valor predeterminado: `90`

`ipcTimeoutSeconds`

(Opcional) El tiempo máximo en segundos que tarda el componente en esperar a que el núcleo de Greengrass responda a las solicitudes de comunicación entre procesos (IPC). Aumente este número si este componente informa de errores de tiempo de espera al comprobar si un dispositivo cliente está autorizado.

Valor predeterminado: `5`

`crtLogLevel`

(Opcional) El nivel de registro de la biblioteca AWS Common Runtime (CRT).

El valor predeterminado es el nivel de registro del broker MQTT de EMQX (in). `log.level` `emqx`

`restartIdentifier`

(Opcional) Configure esta opción para reiniciar el broker MQTT de EMQX. Cuando este valor de configuración cambia, este componente reinicia el broker MQTT. Puede utilizar esta opción para forzar la desconexión de los dispositivos cliente.

`dockerOptions`

(Opcional) Configure esta opción solo en los sistemas operativos Linux para añadir parámetros a la línea de comandos de Docker. Por ejemplo, para asignar puertos adicionales, utilice el parámetro de `-p` Docker:

```
"-p 1883:1883"
```

Example Ejemplo: actualizar un archivo de configuración de la versión 1.x a la versión 2.x

El siguiente ejemplo muestra los cambios necesarios para actualizar un archivo de configuración de la versión 1.x a la versión 2.x.

El archivo de configuración de la versión 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"
    }
```

```
}
```

El archivo de configuración equivalente para la versión 2:

```
{
    "emqxConfig": {
        "listeners": {
            "ssl": {
                "default": {
                   "bind": "8883",
                   "max_connections": "1024000",
                   "max_conn_rate": "500",
                   "handshake_timeout": "15s"
                }
            }
        },
        "log": {
            "console": {
              "enable": true,
              "level": "warning"
            }
        }
    },
    "authMode": "enabled"
}
```

No hay ningún equivalente a la entrada `listener.ssl.external.rate_limit` de configuración. Se ha eliminado la opción de `use_greengrass_managed_certificates` configuración.

Example Ejemplo: establecer un puerto nuevo para el bróker

El siguiente ejemplo cambia el puerto en el que opera el broker MQTT del 8883 predeterminado al puerto 1234. Si utiliza Linux, incluya el `dockerOptions` campo.

```
{
  "emqxConfig": {
    "listeners": {
      "ssl": {
        "default": {
          "bind": 1234
        }
```

```
      }
    }
  },
  "dockerOptions": "-p 1234:1234"
}
```

Example Ejemplo: ajuste el nivel de registro del bróker MQTT

En el siguiente ejemplo, se cambia el nivel de registro del bróker de MQTT a. `debug` Puede elegir entre los siguientes niveles de registro:

- `debug`
- `info`
- `notice`
- `warning`
- `error`
- `critical`
- `alert`
- `emergency`

El nivel de registro predeterminado es`warning`.

```
{
  "emqxConfig": {
    "log": {
      "console": {
         "level": "debug"
      }
    }
  }
}
```

Example Ejemplo: habilite el panel de EMQX

El siguiente ejemplo habilita el panel de EMQX para que pueda monitorear y administrar su corredor. Si utiliza Linux, incluya el `dockerOptions` campo.

```
{
```

```
  "emqxConfig": {
    "dashboard": {
      "listeners": {
        "http": {
          "bind": 18083
        }
      }
    }
  },
  "dockerOptions": "-p 18083:18083"
}
```

1.0.0 - 1.2.2

Este componente proporciona los siguientes parámetros de configuración que puede personalizar al implementar el componente.

`emqx`

(Opcional) La configuración del broker MQTT de EMQX que se va a utilizar. Puede configurar un subconjunto de opciones de configuración de EMQX en este componente.

Este objeto contiene la siguiente información:

`listener.ssl.external`

(Opcional) El puerto en el que opera el bróker MQTT.

> **Note**
>
> Si especifica un puerto diferente y utiliza el componente de puente MQTT para retransmitir mensajes MQTT a otros intermediarios, debe utilizar el puente MQTT v2.1.0 o posterior. Configúrelo para que utilice el puerto en el que opera el broker MQTT.
> Si especifica un puerto diferente y utiliza el componente IP detector para gestionar los puntos finales del broker MQTT, debe utilizar IP detector v2.1.0 o una versión posterior. Configúrelo para que indique el puerto en el que opera el broker MQTT.

Valor predeterminado: 8883

`listener.ssl.external.max_connections`

(Opcional) El número máximo de conexiones simultáneas que admite el bróker MQTT.

Valor predeterminado: `1024000`

`listener.ssl.external.max_conn_rate`

(Opcional) El número máximo de conexiones nuevas por segundo que puede recibir el broker de MQTT.

Valor predeterminado: `500`

`listener.ssl.external.rate_limit`

(Opcional) El límite de ancho de banda para todas las conexiones al bróker MQTT. Especifique el ancho de banda y la duración de ese ancho de banda separados por una coma (`,`) en el siguiente formato:. `bandwidth,duration` Por ejemplo, puede especificar `50KB,5s` limitar el broker MQTT a 50 kilobytes (KB) de datos cada 5 segundos.

`listener.ssl.external.handshake_timeout`

(Opcional) El tiempo que espera el bróker de MQTT para terminar de autenticar una nueva conexión.

Valor predeterminado: `15s`

`mqtt.max_packet_size`

(Opcional) El tamaño máximo de un mensaje MQTT.

Predeterminado: `268435455` (256 MB menos 1)

`log.level`

(Opcional) El nivel de registro del bróker MQTT. Puede elegir entre las siguientes opciones:

- `debug`
- `info`
- `notice`
- `warning`
- `error`

- `critical`
- `alert`
- `emergency`

El nivel de registro predeterminado es`warning`.

`requiresPrivilege`

(Opcional) En los dispositivos principales de Linux, puede especificar que se ejecute el broker MQTT de EMQX sin privilegios de administrador o root. Si establece esta opción en`false`, el usuario del sistema que ejecute este componente debe ser miembro del grupo. `docker`

Valor predeterminado: `true`

`startupTimeoutSeconds`

(Opcional) El tiempo máximo en segundos para que se inicie el broker MQTT de EMQX. El estado del componente cambia a `BROKEN` si supera este tiempo de espera.

Valor predeterminado: `90`

`ipcTimeoutSeconds`

(Opcional) El tiempo máximo en segundos que tarda el componente en esperar a que el núcleo de Greengrass responda a las solicitudes de comunicación entre procesos (IPC). Aumente este número si este componente informa de errores de tiempo de espera al comprobar si un dispositivo cliente está autorizado.

Valor predeterminado: `5`

`crtLogLevel`

(Opcional) El nivel de registro de la biblioteca AWS Common Runtime (CRT).

El valor predeterminado es el nivel de registro del broker MQTT de EMQX (in). `log.level` `emqx`

`restartIdentifier`

(Opcional) Configure esta opción para reiniciar el broker MQTT de EMQX. Cuando este valor de configuración cambia, este componente reinicia el broker MQTT. Puede utilizar esta opción para forzar la desconexión de los dispositivos cliente.

`dockerOptions`

(Opcional) Configure esta opción solo en los sistemas operativos Linux para añadir parámetros a la línea de comandos de Docker. Por ejemplo, para asignar puertos adicionales, utilice el parámetro de `-p` Docker:

```
"-p 1883:1883"
```

`mergeConfigurationFiles`

(Opcional) Configure esta opción para añadir o anular los valores predeterminados de los archivos de configuración de EMQX especificados. Para obtener información sobre los archivos de configuración y sus formatos, consulte Configuración en la documentación de EMQX 4.0. Los valores que especifique se adjuntan al archivo de configuración.

En el siguiente ejemplo, se actualiza el `etc/emqx.conf` archivo.

```
"mergeConfigurationFiles": {
    "etc/emqx.conf": "broker.sys_interval=30s\nbroker.sys_heartbeat=10s"
},
```

Además de los archivos de configuración compatibles con EMQX, Greengrass admite un archivo que configura el complemento de autenticación de Greengrass para EMQX llamado. `etc/plugins/aws_greengrass_emqx_auth.conf` Hay dos opciones compatibles: y. `auth_mode` `use_greengrass_managed_certificates` Para usar otro proveedor de autenticación, defina la `auth_mode` opción en una de las siguientes opciones:

- `enabled`— (Predeterminado) Utilice el proveedor de autenticación y autorización de Greengrass.
- `bypass_on_failure`— Utilice el proveedor de autenticación Greengrass y, a continuación, utilice los demás proveedores de autenticación de la cadena de proveedores de EMQX si Greengrass deniega la autenticación o la autorización.
- `bypass`— El proveedor de Greengrass está deshabilitado. Luego, la cadena de proveedores de EMQX gestiona la autenticación y la autorización.

Si `use_greengrass_managed_certificates` es as`true`, esta opción indica que Greengrass administra los certificados TLS del bróker. Si`false`, indica que proporciona los certificados a través de otra fuente.

En el siguiente ejemplo, se actualizan los valores predeterminados del archivo `etc/plugins/aws_greengrass_emqx_auth.conf` de configuración.

```
"mergeConfigurationFiles": {
    "etc/plugins/aws_greengrass_emqx_auth.conf": "auth_mode=enabled\n
 use_greengrass_managed_certificates=true\n"
  },
```

> **Note**
>
> `aws.greengrass.clientdevices.mqtt.EMQX`permite configurar opciones sensibles a la seguridad. Estas incluyen la configuración de TLS, la autenticación y los proveedores de autorización. La configuración recomendada es la configuración predeterminada que utiliza la autenticación TLS mutua y el proveedor de autenticación de dispositivos cliente Greengrass.

`replaceConfigurationFiles`

(Opcional) Configure esta opción para reemplazar los archivos de configuración de EMQX especificados. Los valores que especifique sustituyen a todo el archivo de configuración existente. No puede especificar el `etc/emqx.conf` archivo en esta sección. Debe usarlo `mergeConfigurationFile` para modificar`etc/emqx.conf`.

Example Ejemplo: actualización de la combinación de configuraciones

El siguiente ejemplo de configuración especifica el funcionamiento del broker MQTT en el puerto 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"
}
```

Archivo de registro local

Este componente utiliza el siguiente archivo de registro.

Linux

```
/greengrass/v2/logs/aws.greengrass.clientdevices.mqtt.EMQX.log
```

Windows

```
C:\greengrass\v2\logs\aws.greengrass.clientdevices.mqtt.EMQX.log
```

Para ver los registros de este componente

- Ejecute el siguiente comando en el dispositivo principal para ver el archivo de registro de este componente en tiempo real. Sustituya */greengrass/v2* o *C:\greengrass\v2* por la ruta a la carpeta AWS IoT Greengrass raíz.

  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
  ```

Licencias

En los sistemas operativos Windows, este software incluye código distribuido según los términos de licencia del software de Microsoft: Microsoft Visual Studio Community 2022. Al descargar este software, aceptas los términos de licencia de ese código.

Este componente se publica en virtud del contrato de licencia de software principal de Greengrass.

Registros de cambios

En la siguiente tabla se describen los cambios en cada versión del componente.

v2.x

Versión	Cambios
2.0.1	Versión actualizada para la versión 2.5.0 de autenticación de dispositivos cliente.
2.0.0	Esta versión del broker MQTT 5 (EMQX) espera parámetros de configuración diferentes a los de la versión 1.x. Si utiliza una configuración no predeterminada para la versión 1.x, debe actualizar la configuración del componente para la versión 2.x. Para obtener más información, consulte Configuración. Nuevas características • Actualiza el broker MQTT a EMQX 5.1.1. • Permite realizar cambios en la configuración del broker sin necesidad de reiniciar el componente. Actualizaciones • Añade un nuevo campo `emqxConfig` de configuración que sustituye a los `emqx` campos de `replaceConfigurationFiles` configuración y. `mergeConfigurationFiles`

v1.x

Versión	Cambios
1.2.3	Mejoras y correcciones de errores • Soluciona un problema por el que los clientes no podían interactuar con EMQX después de autenticarse previamente desconectando y volviendo a autenticar el cliente.

Versión	Cambios
1.2.2	Versión actualizada para la autenticación de dispositivos cliente, versión 2.4.0.
1.2.1	Mejoras y correcciones de errores • Soluciona un problema por el que el componente no se iniciaba en Windows si Visual C++ Redistributable aún no estaba presente. • Actualiza EMQX a la versión 4.4.14.
1.2.0	Añade soporte para cadenas de certificados.
1.1.0	Nuevas características • Añade compatibilidad con las configuraciones de EMQX, incluidas las opciones de intermediación y los complementos. Mejoras y correcciones de errores • Actualiza EMQX a la versión 4.4.9.
1.0.1	Corrige un problema durante el protocolo de enlace TLS que provocaba que algunos clientes MQTT no pudieran conectarse.
1.0.0	Versión inicial.

Emisor de telemetría Nucleus

El componente emisor de telemetría nuclear (`aws.greengrass.telemetry.NucleusEmitter`) recopila datos de telemetría de salud del sistema y los publica continuamente sobre un tema local y otro sobre el MQTT. AWS IoT Core Este componente le permite recopilar la telemetría del sistema en tiempo real en sus dispositivos principales de Greengrass. Para obtener información sobre el agente de telemetría de Greengrass que publica los datos de telemetría del sistema en Amazon, consulte. EventBridge Recopile datos de telemetría del estado del sistema de los dispositivos principales AWS IoT Greengrass

De forma predeterminada, el componente emisor de telemetría Nucleus publica los datos de telemetría cada 60 segundos en el siguiente tema local de publicación o suscripción.

```
$local/greengrass/telemetry
```

De forma predeterminada, el componente emisor de telemetría nuclear no publica en un tema de MQTT. AWS IoT Core Puede configurar este componente para que se publique en un tema de AWS IoT Core MQTT al implementarlo. El uso de un tema de MQTT para publicar datos en el Nube de AWS está sujeto a AWS IoT Core los precios.

AWS IoT Greengrassproporciona varios componentes comunitarios para ayudarlo a analizar y visualizar los datos de telemetría localmente en su dispositivo principal mediante InfluxDB y Grafana. Estos componentes utilizan datos de telemetría del componente nucleo-emisor. Para obtener más información, consulte el archivo README del componente de publicación InfluxDB.

Temas

Versiones

Este componente tiene las siguientes versiones:

- 1.0.x

Tipo

Este componente es un componente de complemento (`aws.greengrass.plugin`). El núcleo de Greengrass ejecuta este componente en la misma máquina virtual Java (JVM) que el núcleo. El núcleo se reinicia al cambiar la versión de este componente en el dispositivo principal.

Este componente utiliza el mismo archivo de registro que el núcleo de Greengrass. Para obtener más información, consulte Supervisar AWS IoT Greengrass registros.

Para obtener más información, consulte Tipos de componentes.

Sistema operativo

Este componente se puede instalar en los dispositivos principales que ejecutan los siguientes sistemas operativos:

- Linux
- Windows

Dependencias

Al implementar un componente, AWS IoT Greengrass también despliega versiones compatibles de sus dependencias. Esto significa que debe cumplir los requisitos del componente y de todas sus dependencias para poder implementarlo correctamente. En esta sección se enumeran las dependencias de las versiones publicadas de este componente y las restricciones de las versiones semánticas que definen las versiones de los componentes para cada dependencia. También puede ver las dependencias de cada versión del componente en la consola. AWS IoT Greengrass En la página de detalles del componente, busque la lista de dependencias.

1.0.8

La siguiente tabla muestra las dependencias de la versión 1.0.8 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.4.0 <2.13.0	Rígido

1.0.7

La siguiente tabla muestra las dependencias de la versión 1.0.7 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.4.0 <2.12.0	Rígido

1.0.6

La siguiente tabla muestra las dependencias de la versión 1.0.6 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.4.0 <2.11.0	Rígido

1.0.5

La siguiente tabla muestra las dependencias de la versión 1.0.5 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.4.0 <2.10.0	Rígido

1.0.4

La siguiente tabla muestra las dependencias de la versión 1.0.4 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.4.0 <2.9.0	Rígido

1.0.3

La siguiente tabla muestra las dependencias de la versión 1.0.3 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.4.0 <2.8.0	Rígido

1.0.2

La siguiente tabla muestra las dependencias de la versión 1.0.2 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.4.0 <2.7.0	Rígido

1.0.1

La siguiente tabla muestra las dependencias de la versión 1.0.1 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.4.0 <2.6.0	Rígido

1.0.0

La siguiente tabla muestra las dependencias de la versión 1.0.0 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.4.0 <2.5.0	Rígido

Para obtener más información sobre las dependencias de los componentes, consulta la referencia de recetas de componentes.

Configuración

Este componente proporciona los siguientes parámetros de configuración que puede personalizar al implementar el componente.

`pubSubPublish`

(Opcional) Define si se deben publicar los datos de telemetría en el `$local/greengrass/telemetry` tema. Los valores admitidos son `true` y `false`.

Valor predeterminado: `true`

`mqttTopic`

(Opcional) El tema de AWS IoT Core MQTT en el que este componente publica los datos de telemetría.

Establezca este valor en el tema AWS IoT Core MQTT en el que desee publicar los datos de telemetría. Cuando este valor está vacío, el núcleo emisor no publica los datos de telemetría en el. Nube de AWS

> **Note**
>
> El uso de un tema de MQTT para publicar datos en el Nube de AWS está sujeto a los precios. AWS IoT Core

Valor predeterminado: `""`

`telemetryPublishIntervalMs`

(Opcional) El tiempo (en milisegundos) entre el que el componente publica los datos de telemetría. Si establece este valor por debajo del valor mínimo admitido, el componente utilizará el valor mínimo en su lugar.

> **Note**
>
> Los intervalos de publicación más bajos dan como resultado un mayor uso de la CPU en el dispositivo principal. Te recomendamos empezar con el intervalo de publicación predeterminado y ajustarlo en función del uso de la CPU del dispositivo.

Mínimo: `500`

Valor predeterminado: `60000`

Example Ejemplo: actualización de combinación de configuraciones

El siguiente ejemplo muestra un ejemplo de actualización de combinación de configuraciones que permite publicar datos de telemetría cada 5 segundos en el `$local/greengrass/telemetry` tema y en el tema `greengrass/myTelemetry` AWS IoT Core MQTT.

```
{
  "pubSubPublish": "true",
  "mqttTopic": "greengrass/myTelemetry",
  "telemetryPublishIntervalMs": 5000
}
```

Datos de salida

Este componente publica las métricas de telemetría como una matriz JSON sobre el tema siguiente.

Tema local: `$local/greengrass/telemetry`

Si lo desea, puede optar por publicar también las métricas de telemetría en un tema de AWS IoT Core MQTT. Para obtener más información sobre los temas, consulte los temas de MQTT en la Guía para desarrolladores. AWS IoT Core

Example Datos de ejemplo


```
[
  {
    "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
  }
]
```

La matriz de salida contiene una lista de métricas que tienen las siguientes propiedades:

`A`

El tipo de agregación de la métrica.

Para la `CpuUsage` métrica, esta propiedad se establece en `Average` porque el valor publicado de la métrica es la cantidad media de uso de la CPU desde el último evento de publicación.

Para el resto de las métricas, el núcleo emisor no agrega el valor de la métrica y esta propiedad se establece en. `Count`

`N`

El nombre de la métrica.

`NS`

El espacio de nombres de la métrica.

`TS`

La marca de tiempo del momento en que se recopilaron los datos.

U

La unidad del valor métrico.

V

El valor de la métrica de .

El emisor del núcleo publica las siguientes métricas:

Nombre	Descripción	
System (Sistema)		
`SystemMemUsage`	La cantidad de memoria que utilizan actualmente todas las aplicaciones del dispositivo principal de Greengrass, incluido el sistema operativo.	
`CpuUsage`	La cantidad de CPU que utilizan actualmente todas las aplicaciones del dispositivo principal de Greengrass, incluido el sistema operativo.	
`TotalNumberOfFDs`	El número de descriptores de archivos almacenados por el sistema operativo del dispositivo principal de Greengrass. Un descriptor de archivo identifica exclusivamente un archivo abierto.	
Núcleo de Greengrass		
`NumberOfComponentsRunning`	El número de componentes que se ejecutan en el dispositivo principal de Greengrass.	

Nombre	Descripción
`NumberOfComponentsErrored`	El número de componentes que están en estado de error en el dispositivo principal de Greengrass.
`NumberOfComponentsInstalled`	El número de componentes que están instalados en el dispositivo principal de Greengrass.
`NumberOfComponentsStarting`	El número de componentes que se inician en el dispositivo principal de Greengrass.
`NumberOfComponentsNew`	La cantidad de componentes que son nuevos en el dispositivo principal de Greengrass.
`NumberOfComponentsStopping`	El número de componentes que se detienen en el dispositivo principal de Greengrass.
`NumberOfComponentsFinished`	El número de componentes que están acabados en el dispositivo principal de Greengrass.
`NumberOfComponentsBroken`	La cantidad de componentes que están rotos en el dispositivo principal de Greengrass.
`NumberOfComponentsStateless`	El número de componentes que no tienen estado en el dispositivo principal de Greengrass.

Uso

Para utilizar los datos de telemetría del estado del sistema, puede crear componentes personalizados que se adapten a los temas en los que el núcleo emisor publica los datos de telemetría y que reaccionen a esos datos según sea necesario. Como el componente núcleo emisor ofrece la opción de publicar datos de telemetría en un tema local, puede suscribirse a ese tema y utilizar los datos publicados para actuar de forma local en su dispositivo principal. De este modo, el dispositivo principal puede reaccionar a los datos de telemetría incluso cuando su conectividad a la nube sea limitada.

Por ejemplo, puede configurar un componente que escuche el `$local/greengrass/telemetry` tema en busca de datos de telemetría y envíe los datos al componente del administrador de transmisiones para transmitir sus datos al. Nube de AWS Para obtener más información sobre cómo crear un componente de este tipo, consulte y. Publicar/suscribir mensajes locales Cree componentes personalizados que usen el administrador de transmisiones

Archivo de registro local

Este componente utiliza el mismo archivo de registro que el componente núcleo de Greengrass.

Linux

```
/greengrass/v2/logs/greengrass.log
```

Windows

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

Para ver los registros de este componente

- Ejecute el siguiente comando en el dispositivo principal para ver el archivo de registro de este componente en tiempo real. Sustituya */greengrass/v2* o *C:\greengrass\v2* por la ruta a la carpeta AWS IoT Greengrass raíz.

  Linux

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

Windows (PowerShell)

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

Registros de cambios

En la siguiente tabla se describen los cambios en cada versión del componente.

Versión	Cambios
1.0.8	Versión actualizada para la versión 2.12.0 de Greengrass Nucleus.
1.0.7	Versión actualizada para el lanzamiento de la versión 2.11.0 de Greengrass nucleus.
1.0.6	Versión actualizada para el lanzamiento de la versión 2.10.0 de Greengrass nucleus.
1.0.5	Versión actualizada para la versión 2.9.0 de Greengrass Nucleus.
1.0.4	Versión actualizada para el lanzamiento de la versión 2.8.0 del núcleo de Greengrass.
1.0.3	Versión actualizada para el lanzamiento de la versión 2.7.0 de Greengrass nucleus.
1.0.2	Versión actualizada para la versión 2.6.0 de Greengrass Nucleus.
1.0.1	Versión actualizada para el lanzamiento de la versión 2.5.0 de Greengrass nucleus.
1.0.0	Versión inicial.

Proveedor PKCS #11

El componente de proveedor PKCS #11 (`aws.greengrass.crypto.Pkcs11Provider`) le permite configurar el software AWS IoT Greengrass Core para que utilice un módulo de seguridad de hardware (HSM) a través de la interfaz PKCS #11. Este componente le permite almacenar de forma

segura los archivos de certificados y claves privadas para que no queden expuestos ni duplicados en el software. Para obtener más información, consulte Integración de la seguridad de hardware.

Para aprovisionar un dispositivo principal de Greengrass que almacene su certificado y su clave privada en un HSM, debe especificar este componente como un complemento de aprovisionamiento al instalar el software Core. AWS IoT Greengrass Para obtener más información, consulte Instale el software AWS IoT Greengrass principal con aprovisionamiento manual de recursos.

AWS IoT Greengrassproporciona este componente como un archivo JAR que puede descargar para especificarlo como complemento de aprovisionamiento durante la instalación. Puede descargar la última versión del archivo JAR del componente en la siguiente URL: https://d2s8p88vqu9w66.cloudfront.net/releases/Pkcs11Provider/aws.greengrass.crypto.Pkcs11Provider-latest.jar.

Temas

Versiones

Este componente tiene las siguientes versiones:

- 2.0.x

Tipo

Este componente es un componente de complemento ()`aws.greengrass.plugin`. El núcleo de Greengrass ejecuta este componente en la misma máquina virtual Java (JVM) que el núcleo. El núcleo se reinicia al cambiar la versión de este componente en el dispositivo principal.

Este componente utiliza el mismo archivo de registro que el núcleo de Greengrass. Para obtener más información, consulte Supervisar AWS IoT Greengrass registros.

Para obtener más información, consulte Tipos de componentes.

Sistema operativo

Este componente solo se puede instalar en los dispositivos principales de Linux.

Requisitos

Este componente tiene los siguientes requisitos:

- Módulo de seguridad de hardware que admite el esquema de firmas PKCS #1 v1.5 y claves RSA con un tamaño de clave RSA-2048 (o mayor) o claves ECC.

  > **Note**
  >
  > Para utilizar un módulo de seguridad de hardware con claves ECC, debe utilizar Greengrass nucleus v2.5.6 o posterior.
  > Para usar un módulo de seguridad de hardware y un administrador de secretos, debe usar un módulo de seguridad de hardware con claves RSA.

- Una biblioteca de proveedores PKCS #11 que el software AWS IoT Greengrass principal puede cargar en tiempo de ejecución (mediante libdl) para invocar las funciones PKCS #11. La biblioteca de proveedores PKCS #11 debe implementar las siguientes operaciones de la API PKCS #11:
  - `C_Initialize`
  - `C_Finalize`
  - `C_GetSlotList`
  - `C_GetSlotInfo`
  - `C_GetTokenInfo`
  - `C_OpenSession`
  - `C_GetSessionInfo`
  - `C_CloseSession`
  - `C_Login`
  - `C_Logout`
  - `C_GetAttributeValue`

  - `C_FindObjectsInit`
  - `C_FindObjects`
  - `C_FindObjectsFinal`
  - `C_DecryptInit`
  - `C_Decrypt`
  - `C_DecryptUpdate`
  - `C_DecryptFinal`
  - `C_SignInit`
  - `C_Sign`
  - `C_SignUpdate`
  - `C_SignFinal`
  - `C_GetMechanismList`
  - `C_GetMechanismInfo`
  - `C_GetInfo`
  - `C_GetFunctionList`
- El módulo de hardware debe resolverlo la etiqueta de ranura, tal y como se define en la especificación de PKCS#11.
- Debe almacenar la clave privada y el certificado en el HSM, en la misma ranura, y deben usar la misma etiqueta de objeto e ID de objeto, si el HSM admite los ID de objeto.
- El certificado y la clave privada deben poder resolverse mediante etiquetas de objetos.
- La clave privada debe tener los siguientes permisos:
  - `sign`
  - `decrypt`
- (Opcional) Para usar el componente de administrador de secretos, debe usar la versión 2.1.0 o posterior, y la clave privada debe tener los siguientes permisos:
  - `unwrap`
  - `wrap`
- (Opcional) Si utiliza la biblioteca TPM2 y ejecuta el núcleo de Greengrass como servicio, debe proporcionar una variable de entorno con la ubicación del almacén PKCS #11. El siguiente ejemplo es un archivo de servicio de systemd con la variable de entorno requerida:

```
[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
```

Dependencias

Al implementar un componente, AWS IoT Greengrass también despliega versiones compatibles de sus dependencias. Esto significa que debe cumplir los requisitos del componente y de todas sus dependencias para poder implementarlo correctamente. En esta sección se enumeran las dependencias de las versiones publicadas de este componente y las restricciones de las versiones semánticas que definen las versiones de los componentes para cada dependencia. También puede ver las dependencias de cada versión del componente en la consola. AWS IoT Greengrass En la página de detalles del componente, busque la lista de dependencias.

2.0.7

La siguiente tabla muestra las dependencias de la versión 2.0.7 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.5.3 <2.13.0	Flexible

2.0.6

La siguiente tabla muestra las dependencias de la versión 2.0.6 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.5.3 <2.12.0	Flexible

2.0.5

La siguiente tabla muestra las dependencias de la versión 2.0.5 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.5.3 <2.11.0	Flexible

2.0.4

La siguiente tabla muestra las dependencias de la versión 2.0.4 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.5.3 <2.10.0	Flexible

2.0.3

La siguiente tabla muestra las dependencias de la versión 2.0.3 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.5.3 <2.9.0	Flexible

2.0.2

La siguiente tabla muestra las dependencias de la versión 2.0.2 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.5.3 <2.8.0	Flexible

2.0.1

La siguiente tabla muestra las dependencias de la versión 2.0.1 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.5.3 <2.7.0	Flexible

2.0.0

La siguiente tabla muestra las dependencias de la versión 2.0.0 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.5.3 <2.6.0	Flexible

Para obtener más información sobre las dependencias de los componentes, consulte la referencia de recetas de componentes.

Configuración

Este componente proporciona los siguientes parámetros de configuración que puede personalizar al implementar el componente.

`name`

Nombre para la configuración PKCS #11.

`library`

La ruta absoluta del archivo a la biblioteca de la implementación de PKCS #11 que el software AWS IoT Greengrass Core puede cargar con libdl.

`slot`

El ID de la ranura que contiene la clave privada y el certificado del dispositivo. Este valor es diferente del índice o la etiqueta de la ranura.

`userPin`

El PIN del usuario que se utilizará para acceder a la ranura.

Example Ejemplo: actualización de la combinación de configuraciones

```
{
  "name": "softhsm_pkcs11",
  "library": "/usr/lib/softhsm/libsofthsm2.so",
  "slot": 1,
  "userPin": "1234"
}
```

Archivo de registro local

Este componente utiliza el mismo archivo de registro que el componente núcleo de Greengrass.

Linux

```
/greengrass/v2/logs/greengrass.log
```

Windows

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

Para ver los registros de este componente

- Ejecute el siguiente comando en el dispositivo principal para ver el archivo de registro de este componente en tiempo real. Sustituya */greengrass/v2* o *C:\greengrass\v2* por la ruta a la carpeta AWS IoT Greengrass raíz.

  Linux

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

  Windows (PowerShell)

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

Registros de cambios

En la siguiente tabla se describen los cambios en cada versión del componente.

Versión	Cambios
2.0.7	Versión actualizada para la versión 2.12.0 de Greengrass nucleus.
2.0.6	Versión actualizada para la versión 2.11.0 de Greengrass nucleus.
2.0.5	Versión actualizada para la versión 2.10.0 de Greengrass nucleus.
2.0.4	Versión actualizada para la versión 2.9.0 de Greengrass nucleus.
2.0.3	Versión actualizada para el lanzamiento de la versión 2.8.0 de Greengrass nucleus.
2.0.2	Versión actualizada para la versión 2.7.0 de Greengrass Nucleus.
2.0.1	Versión actualizada para la versión 2.6.0 de Greengrass Nucleus.
2.0.0	Versión inicial.

Gestor secreto

El componente administrador de secretos (`aws.greengrass.SecretManager`) despliega secretos desde los dispositivos AWS Secrets Manager principales de Greengrass. Utilice este componente para utilizar de forma segura las credenciales, como las contraseñas, en los componentes personalizados de sus dispositivos principales de Greengrass. Para obtener más información acerca de Secrets Manager, consulte ¿Qué es AWS Secrets Manager? en la Guía del usuario de AWS Secrets Manager .

Para acceder a los secretos de este componente en sus componentes personalizados de Greengrass, utilice la operación GetSecretValue en. SDK para dispositivos con AWS IoT Para obtener más información, consulte Úselo SDK para dispositivos con AWS IoT para comunicarse con el núcleo de Greengrass, otros componentes y AWS IoT Core y Recuperar valores secretos.

Este componente cifra los secretos del dispositivo principal para mantener sus credenciales y contraseñas seguras hasta que necesite usarlas. Utiliza la clave privada del dispositivo principal para cifrar y descifrar los secretos.

Temas

- Versiones

Versiones

Este componente tiene las siguientes versiones:

- 2.1.x
- 2.0.x

Tipo

Este componente es un componente de complemento ()`aws.greengrass.plugin`. El núcleo de Greengrass ejecuta este componente en la misma máquina virtual Java (JVM) que el núcleo. El núcleo se reinicia al cambiar la versión de este componente en el dispositivo principal.

Este componente utiliza el mismo archivo de registro que el núcleo de Greengrass. Para obtener más información, consulte Supervisar AWS IoT Greengrass registros.

Para obtener más información, consulte Tipos de componentes.

Sistema operativo

Este componente se puede instalar en los dispositivos principales que ejecutan los siguientes sistemas operativos:

- Linux
- Windows

Requisitos

Este componente tiene los siguientes requisitos:

- El rol de dispositivo de Greengrass debe permitir la `secretsmanager:GetSecretValue` acción, como se muestra en el siguiente ejemplo de política de IAM.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "secretsmanager:GetSecretValue"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:secretsmanager:region:123456789012:secret:MySecret"
      ]
    }
  ]
}
```

> **Note**
>
> Si utiliza una AWS Key Management Service clave gestionada por el cliente para cifrar los secretos, el rol del dispositivo también debe permitir la acción. `kms:Decrypt`

  Para obtener más información sobre las políticas de IAM para Secrets Manager, consulte lo siguiente en la Guía del AWS Secrets Manager usuario:

  - Autenticación y control de acceso para AWS Secrets Manager
  - Acciones, recursos y claves de contexto para las que puede utilizar en una política de IAM o en una política secreta AWS Secrets Manager

- Los componentes personalizados deben definir una política de autorización que permita `aws.greengrass#GetSecretValue` obtener los secretos que se almacenan con este componente. En esta política de autorización, puede restringir el acceso de los componentes a secretos específicos. Para obtener más información, consulte la autorización de IPC del administrador secreto.
- (Opcional) Si guarda la clave privada y el certificado del dispositivo principal en un módulo de seguridad de hardware (HSM), el HSM debe admitir las claves RSA, la clave privada debe tener el `unwrap` permiso y la clave pública debe tener el permiso. `wrap`

Puntos finales y puertos

Este componente debe poder realizar solicitudes salientes a los siguientes puntos finales y puertos, además de a los puntos finales y puertos necesarios para el funcionamiento básico. Para obtener más información, consulte Permitir el tráfico del dispositivo a través de un proxy o firewall.

punto de enlace	Puerto	Obligatoria	Descripción
`secretsmanager.` *`region`*`.amazonaws.com`	443	Sí	Descarga los secretos del dispositivo principal.

Dependencias

Al implementar un componente, AWS IoT Greengrass también despliega versiones compatibles de sus dependencias. Esto significa que debe cumplir los requisitos del componente y de todas sus dependencias para poder implementarlo correctamente. En esta sección se enumeran las dependencias de las versiones publicadas de este componente y las restricciones de las versiones semánticas que definen las versiones de los componentes para cada dependencia. También puede ver las dependencias de cada versión del componente en la consola.AWS IoT Greengrass En la página de detalles del componente, busque la lista de dependencias.

2.1.7 – 2.1.8

La siguiente tabla muestra las dependencias de las versiones 2.1.7 y 2.1.8 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.5.0 <2.13.0	Flexible

2.1.6

La siguiente tabla muestra las dependencias de la versión 2.1.6 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.5.0 <2.12.0	Flexible

2.1.5

La siguiente tabla muestra las dependencias de la versión 2.1.5 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.5.0 <2.11.0	Flexible

2.1.4

La siguiente tabla muestra las dependencias de la versión 2.1.4 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.5.0 <2.10.0	Flexible

2.1.3

La siguiente tabla muestra las dependencias de la versión 2.1.3 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.5.0 <2.9.0	Flexible

2.1.2

La siguiente tabla muestra las dependencias de la versión 2.1.2 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.5.0 <2.8.0	Flexible

2.1.1

La siguiente tabla muestra las dependencias de la versión 2.1.1 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.5.0 <2.7.0	Flexible

2.1.0

La siguiente tabla muestra las dependencias de la versión 2.1.0 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.5.0 <2.6.0	Flexible

2.0.9

La siguiente tabla muestra las dependencias de la versión 2.0.9 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.5.0	Flexible

2.0.8

La siguiente tabla muestra las dependencias de la versión 2.0.8 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.4.0	Flexible

2.0.7

La siguiente tabla muestra las dependencias de la versión 2.0.7 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.3.0	Flexible

2.0.6

La siguiente tabla muestra las dependencias de la versión 2.0.6 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.2.0	Flexible

2.0.4 and 2.0.5

La siguiente tabla muestra las dependencias de las versiones 2.0.4 y 2.0.5 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.3 <2.1.0	Flexible

Para obtener más información sobre las dependencias de los componentes, consulta la referencia de recetas de componentes.

Configuración

Este componente proporciona los siguientes parámetros de configuración que puede personalizar al implementar el componente.

`cloudSecrets`

Una lista de los secretos de Secrets Manager para desplegarlos en el dispositivo principal. Puede especificar etiquetas para definir qué versiones de cada secreto se van a implementar. Si no especifica una versión, este componente implementa la versión con la etiqueta `AWSCURRENT` provisional adjunta. Para obtener más información, consulte Etiquetas de puesta en escena en la Guía del AWS Secrets Manager usuario.

El componente de administrador de secretos almacena los secretos en caché de forma local. Si el valor secreto cambia en Secrets Manager, este componente no recupera automáticamente el nuevo valor. Para actualizar la copia local, asigne una nueva etiqueta al secreto y configure este componente para recuperar el secreto identificado por la nueva etiqueta.

Cada objeto contiene la siguiente información:

`arn`

El ARN del secreto a desplegar. El ARN del secreto puede ser un ARN completo o parcial. Le recomendamos que especifique un ARN completo en lugar de uno parcial. Para obtener más información, consulte Búsqueda de un secreto a partir de un ARN parcial. El siguiente es un ejemplo de un ARN completo y un ARN parcial:

- ARN completo: `arn:aws:secretsmanager:us-east-2:111122223333:secret:`*`SecretName`*`-abcdef`
- ARN parcial: `arn:aws:secretsmanager:us-east-2:111122223333:secret:`*`SecretName`*

`labels`

(Opcional) Una lista de etiquetas para identificar las versiones del secreto que se van a implementar en el dispositivo principal.

Cada etiqueta debe ser una cadena.

Example Ejemplo: actualización de combinación de configuraciones

```
{
  "cloudSecrets": [
    {
      "arn": "arn:aws:secretsmanager:us-west-2:123456789012:secret:MyGreengrassSecret-
abcdef"
    }
  ]
}
```

Archivo de registro local

Este componente utiliza el mismo archivo de registro que el componente núcleo de Greengrass.

Linux

```
/greengrass/v2/logs/greengrass.log
```

Windows

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

Para ver los registros de este componente

- Ejecute el siguiente comando en el dispositivo principal para ver el archivo de registro de este componente en tiempo real. Sustituya */greengrass/v2* o *C:\greengrass\v2* por la ruta a la carpeta AWS IoT Greengrass raíz.

  Linux

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

  Windows (PowerShell)

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

Registros de cambios

En la siguiente tabla se describen los cambios en cada versión del componente.

Versión	Cambios
2.1.8	Mejoras y correcciones de errores Soluciona un problema por el que el administrador secreto no aceptaba un arn parcial.
2.1.7	Versión actualizada para la versión 2.12.0 de Greengrass Nucleus.
2.1.6	Versión actualizada para el lanzamiento de la versión 2.11.0 de Greengrass nucleus.

Versión	Cambios
2.1.5	Versión actualizada para el lanzamiento de la versión 2.10.0 de Greengrass nucleus.
2.1.4	Mejoras y correcciones de errores Soluciona un problema que provocaba que los secretos guardados en caché se eliminaran cuando se desplegaba el administrador de secretos y se reiniciaba Greengrass Nucleus. Versión actualizada para la versión 2.9.0 de Greengrass Nucleus.
2.1.3	Versión actualizada para el lanzamiento de la versión 2.8.0 de Greengrass nucleus.
2.1.2	Versión actualizada para el lanzamiento de la versión 2.7.0 de Greengrass nucleus.
2.1.1	Versión actualizada para la versión 2.6.0 de Greengrass Nucleus.
2.1.0	Nuevas características • Añade soporte para la integración de la seguridad del hardware. El componente de administrador de secretos puede cifrar y descifrar secretos mediante una clave privada que se almacena en un módulo de seguridad de hardware (HSM). Para obtener más información, consulte Integración de la seguridad de hardware. Mejoras y correcciones de errores • Versión actualizada para el lanzamiento de la versión 2.5.0 de Greengrass nucleus.
2.0.9	Versión actualizada para la versión 2.4.0 de Greengrass Nucleus.
2.0.8	Versión actualizada para el lanzamiento de la versión 2.3.0 de Greengrass nucleus.
2.0.7	Versión actualizada para el lanzamiento de la versión 2.2.0 de Greengrass nucleus.

Versión	Cambios
2.0.6	Versión actualizada para el lanzamiento de la versión 2.1.0 de Greengrass nucleus.
2.0.5	Mejoras • Añada compatibilidad con las regiones y AWS GovCloud (US) regiones de AWS China.
2.0.4	Versión inicial.

Tunelización segura

Con este `aws.greengrass.SecureTunneling` componente, puede establecer una comunicación bidireccional segura con un dispositivo central de Greengrass ubicado detrás de firewalls restringidos.

Por ejemplo, imagine que tiene un dispositivo central de Greengrass detrás de un firewall que prohíbe todas las conexiones entrantes. La tunelización segura utiliza MQTT para transferir un token de acceso al dispositivo y, a continuación, lo utiliza WebSockets para establecer una conexión SSH con el dispositivo a través del firewall. Con este túnel AWS IoT gestionado, puedes abrir la conexión SSH necesaria para tu dispositivo. Para obtener más información sobre el uso de la tunelización AWS IoT segura para conectarse a dispositivos remotos, consulta la tunelización AWS IoT segura en la Guía para desarrolladores.AWS IoT

Este componente se suscribe al agente de mensajes AWS IoT Core MQTT sobre el `$aws/things/greengrass-core-device/tunnels/notify` tema para recibir notificaciones de tunelización segura.

Temas

- Versiones
- Tipo
- Sistema operativo
- Requisitos
- Dependencias
- Configuración

- Archivo de registro local
- Licencias
- Uso
- Véase también
- Registros de cambios

Versiones

Este componente tiene las siguientes versiones:

- 1.0.x

Tipo

Este componente es un componente genérico (`aws.greengrass.generic`). El núcleo de Greengrass ejecuta los scripts del ciclo de vida del componente.

Para obtener más información, consulte Tipos de componentes.

Sistema operativo

Este componente solo se puede instalar en los dispositivos principales de Linux.

Arquitecturas:

- Armv71
- Armv8 (AArch64)
- x86_64

Requisitos

Este componente tiene los siguientes requisitos:

- Hay un mínimo de 32 MB de espacio en disco disponible para el componente de tunelización segura. Este requisito no incluye el software principal de Greengrass ni otros componentes que se ejecuten en el mismo dispositivo.
- Hay un mínimo de 16 MB de RAM disponibles para el componente de tunelización segura. Este requisito no incluye el software principal de Greengrass ni otros componentes que se ejecuten en

el mismo dispositivo. Para obtener más información, consulte Controle la asignación de memoria con las opciones de JVM.

- Se requiere la versión 2.25 o superior de la Biblioteca C GNU (glibc) con un núcleo de Linux 3.2 o superior para el componente de tunelización segura de la versión 1.0.12 o superior. No se admiten las versiones del sistema operativo y las bibliotecas que hayan superado su fecha de fin de vida útil. Debe utilizar un sistema operativo y bibliotecas con soporte a largo plazo.
- Tanto el sistema operativo como el motor de ejecución de Java deben estar instalados en 64 bits.
- Python 3.5 o posterior instalado en el dispositivo principal de Greengrass y agregado a la variable de entorno PATH.
- `libcrypto.so.1.1`instalado en el dispositivo principal de Greengrass y agregado a la variable de entorno PATH.
- Abra el tráfico saliente en el puerto 443 del dispositivo principal de Greengrass.
- Active la compatibilidad con el servicio de comunicación que desee utilizar para comunicarse con el dispositivo principal de Greengrass. Por ejemplo, para abrir una conexión SSH con el dispositivo, debe activar SSH en ese dispositivo.

Puntos finales y puertos

Este componente debe poder realizar solicitudes salientes a los siguientes puntos finales y puertos, además de a los puntos finales y puertos necesarios para el funcionamiento básico. Para obtener más información, consulte Permitir el tráfico del dispositivo a través de un proxy o firewall.

punto de enlace	Puerto	Obligatoria	Descripción
`data.tunneling.iot.`*`region`*`.amazonaws.com`	443	Sí	Establezca túneles seguros.

Dependencias

Al implementar un componente, AWS IoT Greengrass también despliega versiones compatibles de sus dependencias. Esto significa que debe cumplir los requisitos del componente y de todas sus dependencias para poder implementarlo correctamente. En esta sección se enumeran las dependencias de las versiones publicadas de este componente y las restricciones de las versiones

semánticas que definen las versiones de los componentes para cada dependencia. También puede ver las dependencias de cada versión del componente en la consola.AWS IoT Greengrass En la página de detalles del componente, busque la lista de dependencias.

1.0.19

En la siguiente tabla se enumeran las dependencias de la versión 1.0.19 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <3.0.0	Flexible

1.0.18

La siguiente tabla muestra las dependencias de la versión 1.0.18 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.13.0	Flexible

1.0.16 – 1.0.17

En la siguiente tabla se enumeran las dependencias de las versiones 1.0.16 a 1.0.17 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.12.0	Flexible

1.0.14 – 1.0.15

En la siguiente tabla se enumeran las dependencias de las versiones 1.0.14 a 1.0.15 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.11.0	Flexible

1.0.11 – 1.0.13

En la siguiente tabla se enumeran las dependencias de las versiones 1.0.11 — 1.0.13 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.10.0	Flexible

1.0.10

En la siguiente tabla se enumeran las dependencias de la versión 1.0.10 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.9.0	Flexible

1.0.9

La siguiente tabla muestra las dependencias de la versión 1.0.9 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.8.0	Flexible

1.0.8

La siguiente tabla muestra las dependencias de la versión 1.0.8 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.7.0	Flexible

1.0.5 - 1.0.7

La siguiente tabla muestra las dependencias de las versiones 1.0.5 a 1.0.7 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.6.0	Flexible

1.0.4

La siguiente tabla muestra las dependencias de la versión 1.0.4 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.5.0	Flexible

1.0.3

La siguiente tabla muestra las dependencias de la versión 1.0.3 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.4.0	Flexible

1.0.2

La siguiente tabla muestra las dependencias de la versión 1.0.2 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.3.0	Flexible

1.0.1

La siguiente tabla muestra las dependencias de la versión 1.0.1 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.2.0	Flexible

1.0.0

La siguiente tabla muestra las dependencias de la versión 1.0.0 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.3 <2.1.0	Flexible

Para obtener más información sobre las dependencias de los componentes, consulta la referencia de recetas de componentes.

Configuración

Este componente proporciona los siguientes parámetros de configuración que puede personalizar al implementar el componente.

`OS_DIST_INFO`

(Opcional) El sistema operativo del dispositivo principal. De forma predeterminada, el componente intenta identificar automáticamente el sistema operativo que se ejecuta en el dispositivo principal. Si el componente no se inicia con el valor predeterminado, utilice este valor para especificar el sistema operativo. Para obtener una lista de los sistemas operativos compatibles con este componente, consulteRequisitos de los dispositivos.

Este valor puede ser uno de los siguientes:`auto`,`ubuntu`,`amzn2`,`raspberrypi`.

Valor predeterminado: `auto`

`accessControl`

(Opcional) El objeto que contiene la política de autorización que permite al componente suscribirse al tema de las notificaciones de túneles seguros.

> **Note**
>
> No modifique este parámetro de configuración si la implementación se dirige a un grupo de cosas. Si su implementación se dirige a un dispositivo principal individual y desea restringir su suscripción al tema del dispositivo, especifique el nombre del dispositivo principal. En el `resources` valor de la política de autorización del dispositivo, sustituya el comodín del tema MQTT por el nombre del dispositivo.

```
{
  "aws.greengrass.ipc.mqttproxy": {
    "aws.iot.SecureTunneling:mqttproxy:1": {
      "policyDescription": "Access to tunnel notification pubsub topic",
      "operations": [
        "aws.greengrass#SubscribeToIoTCore"
      ],
      "resources": [
        "$aws/things/+/tunnels/notify"
      ]
    }
  }
}
```

Example Ejemplo: actualización de combinación de configuraciones

El siguiente ejemplo de configuración especifica que se debe permitir que este componente abra túneles seguros en un dispositivo principal denominado **`MyGreengrassCore`** que ejecuta 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"
        ]
      }
    }
  }
}
```

Archivo de registro local

Este componente utiliza el siguiente archivo de registro.

```
/greengrass/v2/logs/aws.greengrass.SecureTunneling.log
```

Para ver los registros de este componente

- Ejecute el siguiente comando en el dispositivo principal para ver el archivo de registro de este componente en tiempo real. */greengrass/v2*Sustitúyalo por la ruta a la carpeta AWS IoT Greengrass raíz.

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

Licencias

Este componente incluye el siguiente software o licencias de terceros:

- AWS IoT Licencia 2.0 de cliente o Apache para dispositivos
- AWS IoT Device SDK para Java/Apache License 2.0
- Licencia gson/Apache 2.0
- Licencia log4j /Apache 2.0
- licencia slf4j /Apache 2.0

Uso

Para usar el componente de tunelización segura en su dispositivo, haga lo siguiente:

1. Implemente el componente de tunelización segura en su dispositivo.
2. Abra la consola de AWS IoT. En el menú de la izquierda, selecciona Acciones remotas y, a continuación, selecciona Proteger túneles.
3. Crea un túnel hacia tu dispositivo Greengrass.
4. Descargue el token de acceso a la fuente.
5. Usa el proxy local con el token de acceso a la fuente para conectarte a tu destino. Para obtener más información, consulta Cómo usar el proxy local en la Guía para AWS IoT desarrolladores.

Véase también

- AWS IoT tunelización segura en la AWS IoT Guía para desarrolladores
- ¿Cómo usar el proxy local en la AWS IoT Guía para desarrolladores

Registros de cambios

En la siguiente tabla se describen los cambios en cada versión del componente.

Versión	Cambios
1.0.19	Mejoras y correcciones de errores • Actualiza el cliente de AWS IoT dispositivo subyacente invocado por el componente de la versión 1.8.0 a la versión 1.9.0. • Aumenta el límite de túneles simultáneos a 20 túneles a nivel de componente. • Aumenta el tiempo de espera predeterminado del AWS IoT Greengrass Core IPC de 3 a 10 segundos. **Warning** Si utiliza el proxy local de tunelización segura como cliente de origen del túnel, no actualice el componente a esta versión hasta que también haya actualizado el proxy local a la versión 3.1.1 o posterior.
1.0.18	Versión actualizada para la versión 2.12.0 de Greengrass nucleus.
1.0.17	Mejoras y correcciones de errores • Corrige el problema de limpieza de subprocesos que impedía a los usuarios crear túneles. Este componente ahora limpiará un hilo una vez que reciba la CloseTunnel señal o si el túnel ha caducado después de 12 horas.
1.0.16	Versión actualizada para la versión 2.11.0 de Greengrass nucleus.
1.0.15	Mejoras y correcciones de errores • Soluciona un problema de inicio para los usuarios que no tienen un directorio principal en el dispositivo. El componente de tunelización segura ahora se inicia sin crear un directorio para documentos paralelos .

Versión	Cambios
1.0.14	Versión actualizada para la versión 2.10.0 de Greengrass nucleus.
1.0.13	Mejoras y correcciones de errores • Soluciona un problema por el que un proceso cliente huérfano impedía que más de un túnel apuntara al dispositivo.
1.0.12	Mejoras y correcciones de errores • Añade compatibilidad con x86_64 (AMD64) y ARMv8 (Aarch64) cuando se ejecuta en el sistema operativo Raspberry Pi.
1.0.11	Versión actualizada para la versión 2.9.0 de Greengrass nucleus.
1.0.10	Versión actualizada para el lanzamiento de la versión 2.8.0 de Greengrass nucleus.
1.0.9	Versión actualizada para la versión 2.7.0 de Greengrass Nucleus.
1.0.8	Versión actualizada para la versión 2.6.0 de Greengrass nucleus.
1.0.7	Mejoras y correcciones de errores • Corrige un problema que provocaba que el componente se desconect ara al transferir archivos grandes a través de SCP.
1.0.6	Esta versión contiene correcciones de errores.
1.0.5	Versión actualizada para la versión 2.5.0 de Greengrass nucleus.
1.0.4	Versión actualizada para la versión 2.4.0 de Greengrass nucleus.
1.0.3	Versión actualizada para la versión 2.3.0 de Greengrass nucleus.
1.0.2	Versión actualizada para el lanzamiento de la versión 2.2.0 de Greengrass nucleus.
1.0.1	Versión actualizada para el lanzamiento de la versión 2.1.0 de Greengrass nucleus.
1.0.0	Versión inicial.

Gestor en la sombra

El componente administrador de sombras (`aws.greengrass.ShadowManager`) habilita el servicio oculto local en su dispositivo principal. El servicio de sombra local permite a los componentes utilizar la comunicación entre procesos para interactuar con las sombras locales. El componente de administrador de sombras gestiona el almacenamiento de los documentos ocultos locales y también gestiona la sincronización de los estados ocultos locales con el servicio AWS IoT Device Shadow.

Para obtener más información sobre cómo los dispositivos principales de Greengrass pueden interactuar con las sombras, consulte. Interactúa con las sombras de los dispositivos

Temas

Versiones

Este componente tiene las siguientes versiones:

- 2.3.x
- 2.2.x
- 2.1.x
- 2.0.x

Tipo

Este componente es un componente de complemento (`aws.greengrass.plugin`). El núcleo de Greengrass ejecuta este componente en la misma máquina virtual Java (JVM) que el núcleo. El núcleo se reinicia al cambiar la versión de este componente en el dispositivo principal.

Este componente utiliza el mismo archivo de registro que el núcleo de Greengrass. Para obtener más información, consulte Supervisar AWS IoT Greengrass registros.

Para obtener más información, consulte Tipos de componentes.

Sistema operativo

Este componente se puede instalar en los dispositivos principales que ejecutan los siguientes sistemas operativos:

- Linux
- Windows

Requisitos

Este componente tiene los siguientes requisitos:

- (Opcional) Para sincronizar las sombras con el servicio AWS IoT Device Shadow, la AWS IoT política del dispositivo principal de Greengrass debe permitir las siguientes acciones de política AWS IoT Core clandestina:
  - `iot:GetThingShadow`
  - `iot:UpdateThingShadow`
  - `iot:DeleteThingShadow`

  Para obtener más información sobre estas AWS IoT Core políticas, consulte las acciones AWS IoT Core políticas en la Guía para AWS IoT desarrolladores.

  Para obtener más información sobre la AWS IoT política mínima, consulte AWS IoTPolítica mínima para los dispositivos AWS IoT Greengrass V2 principales
- Se admite la ejecución del componente shadow manager en una VPC.

Dependencias

Al implementar un componente, AWS IoT Greengrass también despliega versiones compatibles de sus dependencias. Esto significa que debe cumplir los requisitos del componente y de todas sus dependencias para poder implementarlo correctamente. En esta sección se enumeran las dependencias de las versiones publicadas de este componente y las restricciones de las versiones

semánticas que definen las versiones de los componentes para cada dependencia. También puede ver las dependencias de cada versión del componente en la consola.AWS IoT Greengrass En la página de detalles del componente, busque la lista de dependencias.

2.3.5 – 2.3.8

La siguiente tabla muestra las dependencias de las versiones 2.3.5 a 2.3.8 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.5.0 <2.13.0	Flexible

2.3.3 and 2.3.4

La siguiente tabla muestra las dependencias de las versiones 2.3.3 y 2.3.4 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.5.0 <2.12.0	Flexible

2.3.2

La siguiente tabla muestra las dependencias de la versión 2.3.2 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.5.0 <2.11.0	Flexible

2.3.0 and 2.3.1

La siguiente tabla muestra las dependencias de las versiones 2.3.0 y 2.3.1 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.5.0 <2.10.0	Flexible

2.2.3 and 2.2.4

La siguiente tabla muestra las dependencias de las versiones 2.2.3 y 2.2.4 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.2.0 <3.0.0	Flexible

2.2.2

La siguiente tabla muestra las dependencias de la versión 2.2.2 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.2.0 <2.9.0	Flexible

2.2.1

La siguiente tabla muestra las dependencias de la versión 2.2.1 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.2.0 <2.8.0	Flexible

2.1.1 and 2.2.0

La siguiente tabla muestra las dependencias de las versiones 2.1.1 y 2.2.0 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.2.0 <2.7.0	Flexible

2.0.5 - 2.1.0

La siguiente tabla muestra las dependencias de las versiones 2.0.5 a 2.1.0 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.2.0 <2.6.0	Flexible

2.0.3 and 2.0.4

La siguiente tabla muestra las dependencias de las versiones 2.0.3 y 2.0.4 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.2.0 <2.5.0	Flexible

2.0.1 and 2.0.2

La siguiente tabla muestra las dependencias de las versiones 2.0.1 y 2.0.2 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.2.0 <2.4.0	Flexible

2.0.0

La siguiente tabla muestra las dependencias de la versión 2.0.0 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.2.0 <2.3.0	Flexible

Para obtener más información sobre las dependencias de los componentes, consulta la referencia de recetas de componentes.

Configuración

Este componente proporciona los siguientes parámetros de configuración que puede personalizar al implementar el componente.

2.3.x

`strategy`

(Opcional) La estrategia que utiliza este componente para sincronizar las sombras entre AWS IoT Core y el dispositivo principal.

Este objeto contiene la siguiente información.

`type`

(Opcional) El tipo de estrategia que utiliza este componente para sincronizar las sombras entre el dispositivo principal AWS IoT Core y el dispositivo principal. Puede elegir entre las siguientes opciones:

- `realTime`— Sincronice las sombras AWS IoT Core cada vez que se produzca una actualización de sombras.
- `periodic`— Sincronice las sombras AWS IoT Core en un intervalo regular que especifique con el parámetro `delay` de configuración.

Valor predeterminado: `realTime`

`delay`

(Opcional) El intervalo en segundos con AWS IoT Core el que este componente sincroniza las sombras al especificar la estrategia de `periodic` sincronización.

> **Note**
>
> Este parámetro es obligatorio si se especifica la estrategia de `periodic` sincronización.

`synchronize`

(Opcional) Los ajustes de sincronización que determinan cómo se sincronizan las sombras con el Nube de AWS.

> **Note**
>
> Debe crear una actualización de configuración con esta propiedad para sincronizar las sombras con. Nube de AWS

Este objeto contiene la siguiente información.

`coreThing`

(Opcional) Las sombras del dispositivo principal se sincronizan. Este objeto contiene la siguiente información.

`classic`

(Opcional) De forma predeterminada, el administrador de sombras sincroniza el estado local de la sombra clásica del dispositivo principal con el Nube de AWS. Si no quieres sincronizar la sombra clásica del dispositivo, configúrala en. `false`

Valor predeterminado: `true`

`namedShadows`

(Opcional) La lista de sombras de dispositivos principales con nombre que se van a sincronizar. Debe especificar los nombres exactos de las sombras.

> **Warning**
>
> El AWS IoT Greengrass servicio usa el `AWSManagedGreengrassV2Deployment` nombre shadow para administrar las implementaciones que se dirigen a dispositivos principales individuales. Esta sombra denominada está reservada para que la utilice el AWS IoT Greengrass servicio. No actualice ni elimine la sombra con nombre asignado.

`shadowDocumentsMap`

(Opcional) El dispositivo adicional sombrea para sincronizarlo. El uso de este parámetro de configuración facilita la especificación de documentos ocultos. Se recomienda utilizar este parámetro en lugar del `shadowDocuments` objeto.

> **Note**
>
> Si especifica un `shadowDocumentsMap` objeto, no debe `shadowDocuments` especificarlo.

Cada objeto contiene la siguiente información:

thingName

La configuración de sombra de *ThingName* para esta configuración de sombra.

`classic`

(Opcional) Si no desea sincronizar la sombra de dispositivo clásica con el `thingName` dispositivo, configúrela en. `false`

`namedShadows`

La lista de sombras con nombre que quieres sincronizar. Debe especificar los nombres exactos de las sombras.

`shadowDocuments`

(Opcional) La lista de sombras de dispositivos adicionales que se van a sincronizar. Le recomendamos que utilice el `shadowDocumentsMap` parámetro en su lugar.

> **Note**
>
> Si especifica un `shadowDocuments` objeto, no debe `shadowDocumentsMap` especificarlo.

Cada objeto de esta lista contiene la siguiente información.

`thingName`

Nombre del dispositivo con el que se van a sincronizar las sombras.

`classic`

(Opcional) Si no quieres sincronizar la sombra clásica del dispositivo con el `thingName` dispositivo, configúrala en`false`.

Valor predeterminado: `true`

`namedShadows`

(Opcional) La lista de sombras de dispositivos con nombre que deseas sincronizar. Debe especificar los nombres exactos de las sombras.

`direction`

(Opcional) La dirección para sincronizar las sombras entre el servicio de sombras local y el Nube de AWS. Puede configurar esta opción para reducir el ancho de banda y las conexiones al Nube de AWS. Puede elegir entre las siguientes opciones:

- `betweenDeviceAndCloud`— Sincronice las sombras entre el servicio oculto local y el Nube de AWS.
- `deviceToCloud`— Envía actualizaciones de sombras desde el servicio de sombras local al Nube de AWS e ignora las actualizaciones de sombras del Nube de AWS.
- `cloudToDevice`— Recibe actualizaciones ocultas del Nube de AWS servicio paralelo local y no envíes actualizaciones ocultas del servicio paralelo local al Nube de AWS.

Valor predeterminado: `BETWEEN_DEVICE_AND_CLOUD`

`rateLimits`

(Opcional) La configuración que determina los límites de velocidad para las solicitudes de servicios paralelos.

Este objeto contiene la siguiente información.

`maxOutboundSyncUpdatesPerSecond`

(Opcional) El número máximo de solicitudes de sincronización por segundo que transmite el dispositivo.

Predeterminado: 100 solicitudes por segundo

`maxTotalLocalRequestsRate`

(Opcional) El número máximo de solicitudes de IPC locales por segundo que se envían al dispositivo principal.

Predeterminado: 200 solicitudes por segundo

`maxLocalRequestsPerSecondPerThing`

(Opcional) El número máximo de solicitudes de IPC locales por segundo que se envían para cada elemento de IoT conectado.

Predeterminado: 20 solicitudes por segundo para cada cosa

> **Note**
>
> Estos parámetros de límites de velocidad definen el número máximo de solicitudes por segundo para el servicio paralelo local. El número máximo de solicitudes por segundo para el servicio AWS IoT Device Shadow depende de usted Región de AWS. Para obtener más información, consulte los límites de la API de AWS IoT Device Shadow Service en Referencia general de Amazon Web Services.

`shadowDocumentSizeLimitBytes`

(Opcional) El tamaño máximo permitido de cada documento de estado JSON para las sombras locales.

Si aumentas este valor, también debes aumentar el límite de recursos del documento de estado JSON para las sombras de nubes. Para obtener más información, consulte los límites de la API de AWS IoT Device Shadow Service en Referencia general de Amazon Web Services.

Predeterminado: 8192 bytes

Máximo: 30720 bytes

Example Ejemplo: actualización de la combinación de configuraciones

El siguiente ejemplo muestra un ejemplo de actualización de la combinación de configuraciones con todos los parámetros de configuración disponibles para el componente shadow manager.

```
{
   "strategy":{
      "type":"periodic",
      "delay":300
   },
   "synchronize":{
      "shadowDocumentsMap":{
         "MyDevice1":{
            "classic":false,
            "namedShadows":[
               "MyShadowA",
               "MyShadowB"
```

```
            ]
         },
         "MyDevice2":{
            "classic":true,
            "namedShadows":[]
         }
      },
      "direction":"betweenDeviceAndCloud"
   },
   "rateLimits":{
      "maxOutboundSyncUpdatesPerSecond":100,
      "maxTotalLocalRequestsRate":200,
      "maxLocalRequestsPerSecondPerThing":20
   },
   "shadowDocumentSizeLimitBytes":8192
}
```

2.2.x

`strategy`

(Opcional) La estrategia que utiliza este componente para sincronizar las sombras entre el dispositivo principal AWS IoT Core y el dispositivo principal.

Este objeto contiene la siguiente información.

`type`

(Opcional) El tipo de estrategia que utiliza este componente para sincronizar las sombras entre el dispositivo principal AWS IoT Core y el dispositivo principal. Puede elegir entre las siguientes opciones:

- `realTime`— Sincronice las sombras AWS IoT Core cada vez que se produzca una actualización de sombras.
- `periodic`— Sincronice las sombras AWS IoT Core en un intervalo regular que especifique con el parámetro `delay` de configuración.

Valor predeterminado: `realTime`

`delay`

(Opcional) El intervalo en segundos con AWS IoT Core el que este componente sincroniza las sombras al especificar la estrategia de `periodic` sincronización.

> **Note**
>
> Este parámetro es obligatorio si se especifica la estrategia de `periodic` sincronización.

`synchronize`

(Opcional) Los ajustes de sincronización que determinan cómo se sincronizan las sombras con el Nube de AWS.

> **Note**
>
> Debe crear una actualización de configuración con esta propiedad para sincronizar las sombras con. Nube de AWS

Este objeto contiene la siguiente información.

`coreThing`

(Opcional) Las sombras del dispositivo principal se sincronizan. Este objeto contiene la siguiente información.

`classic`

(Opcional) De forma predeterminada, el administrador de sombras sincroniza el estado local de la sombra clásica del dispositivo principal con el Nube de AWS. Si no quieres sincronizar la sombra clásica del dispositivo, configúrala en. `false`

Valor predeterminado: `true`

`namedShadows`

(Opcional) La lista de sombras de dispositivos principales con nombre que se van a sincronizar. Debe especificar los nombres exactos de las sombras.

> **Warning**
>
> El AWS IoT Greengrass servicio usa el `AWSManagedGreengrassV2Deployment` nombre shadow para administrar las implementaciones que se dirigen a dispositivos principales individuales.

Esta sombra denominada está reservada para que la utilice el AWS IoT Greengrass servicio. No actualice ni elimine la sombra con nombre asignado.

`shadowDocumentsMap`

(Opcional) El dispositivo adicional sombrea para sincronizarlo. El uso de este parámetro de configuración facilita la especificación de documentos ocultos. Se recomienda utilizar este parámetro en lugar del `shadowDocuments` objeto.

> **Note**
>
> Si especifica un `shadowDocumentsMap` objeto, no debe `shadowDocuments` especificarlo.

Cada objeto contiene la siguiente información:

`thingName`

La configuración de sombra de *`ThingName`* para esta configuración de sombra.

`classic`

(Opcional) Si no desea sincronizar la sombra de dispositivo clásica con el `thingName` dispositivo, configúrela en. `false`

`namedShadows`

La lista de sombras con nombre que quieres sincronizar. Debe especificar los nombres exactos de las sombras.

`shadowDocuments`

(Opcional) La lista de sombras de dispositivos adicionales que se van a sincronizar. Le recomendamos que utilice el `shadowDocumentsMap` parámetro en su lugar.

> **Note**
>
> Si especifica un `shadowDocuments` objeto, no debe `shadowDocumentsMap` especificarlo.

Cada objeto de esta lista contiene la siguiente información.

`thingName`

Nombre del dispositivo con el que se van a sincronizar las sombras.

`classic`

(Opcional) Si no quieres sincronizar la sombra clásica del dispositivo con el `thingName` dispositivo, configúrala en`false`.

Valor predeterminado: `true`

`namedShadows`

(Opcional) La lista de sombras de dispositivos con nombre que deseas sincronizar. Debe especificar los nombres exactos de las sombras.

`direction`

(Opcional) La dirección para sincronizar las sombras entre el servicio de sombras local y el Nube de AWS. Puede configurar esta opción para reducir el ancho de banda y las conexiones al Nube de AWS. Puede elegir entre las siguientes opciones:

- `betweenDeviceAndCloud`— Sincronice las sombras entre el servicio oculto local y el Nube de AWS.
- `deviceToCloud`— Envía actualizaciones de sombras desde el servicio de sombras local al Nube de AWS e ignora las actualizaciones de sombras del Nube de AWS.
- `cloudToDevice`— Recibe actualizaciones ocultas del Nube de AWS servicio paralelo local y no envíes actualizaciones ocultas del servicio paralelo local al Nube de AWS.

Valor predeterminado: `BETWEEN_DEVICE_AND_CLOUD`

`rateLimits`

(Opcional) La configuración que determina los límites de velocidad para las solicitudes de servicios paralelos.

Este objeto contiene la siguiente información.

`maxOutboundSyncUpdatesPerSecond`

(Opcional) El número máximo de solicitudes de sincronización por segundo que transmite el dispositivo.

Predeterminado: 100 solicitudes por segundo

`maxTotalLocalRequestsRate`

(Opcional) El número máximo de solicitudes de IPC locales por segundo que se envían al dispositivo principal.

Predeterminado: 200 solicitudes por segundo

`maxLocalRequestsPerSecondPerThing`

(Opcional) El número máximo de solicitudes de IPC locales por segundo que se envían para cada elemento de IoT conectado.

Predeterminado: 20 solicitudes por segundo para cada cosa

> **Note**
>
> Estos parámetros de límites de velocidad definen el número máximo de solicitudes por segundo para el servicio paralelo local. El número máximo de solicitudes por segundo para el servicio AWS IoT Device Shadow depende de usted Región de AWS. Para obtener más información, consulte los límites de la API de AWS IoT Device Shadow Service en Referencia general de Amazon Web Services.

`shadowDocumentSizeLimitBytes`

(Opcional) El tamaño máximo permitido de cada documento de estado JSON para las sombras locales.

Si aumentas este valor, también debes aumentar el límite de recursos del documento de estado JSON para las sombras de nubes. Para obtener más información, consulte los límites de la API de AWS IoT Device Shadow Service en Referencia general de Amazon Web Services.

Predeterminado: 8192 bytes

Máximo: 30720 bytes

Example Ejemplo: actualización de la combinación de configuraciones

El siguiente ejemplo muestra un ejemplo de actualización de la combinación de configuraciones con todos los parámetros de configuración disponibles para el componente shadow manager.

```
{
   "strategy":{
      "type":"periodic",
      "delay":300
   },
   "synchronize":{
      "shadowDocumentsMap":{
         "MyDevice1":{
            "classic":false,
            "namedShadows":[
               "MyShadowA",
               "MyShadowB"
            ]
         },
         "MyDevice2":{
            "classic":true,
            "namedShadows":[]
         }
      },
      "direction":"betweenDeviceAndCloud"
   },
   "rateLimits":{
      "maxOutboundSyncUpdatesPerSecond":100,
      "maxTotalLocalRequestsRate":200,
      "maxLocalRequestsPerSecondPerThing":20
   },
   "shadowDocumentSizeLimitBytes":8192
}
```

2.1.x

`strategy`

(Opcional) La estrategia que utiliza este componente para sincronizar las sombras entre el dispositivo principal AWS IoT Core y el dispositivo principal.

Este objeto contiene la siguiente información.

`type`

(Opcional) El tipo de estrategia que utiliza este componente para sincronizar las sombras entre el dispositivo principal AWS IoT Core y el dispositivo principal. Puede elegir entre las siguientes opciones:

- `realTime`— Sincronice las sombras AWS IoT Core cada vez que se produzca una actualización de sombras.
- `periodic`— Sincronice las sombras AWS IoT Core en un intervalo regular que especifique con el parámetro `delay` de configuración.

Valor predeterminado: `realTime`

`delay`

(Opcional) El intervalo en segundos con AWS IoT Core el que este componente sincroniza las sombras al especificar la estrategia de `periodic` sincronización.

> **Note**
>
> Este parámetro es obligatorio si se especifica la estrategia de `periodic` sincronización.

`synchronize`

(Opcional) Los ajustes de sincronización que determinan cómo se sincronizan las sombras con el Nube de AWS.

> **Note**
>
> Debe crear una actualización de configuración con esta propiedad para sincronizar las sombras con. Nube de AWS

Este objeto contiene la siguiente información.

`coreThing`

(Opcional) Las sombras del dispositivo principal se sincronizan. Este objeto contiene la siguiente información.

`classic`

(Opcional) De forma predeterminada, el administrador de sombras sincroniza el estado local de la sombra clásica del dispositivo principal con el Nube de AWS. Si no quieres sincronizar la sombra clásica del dispositivo, configúrala en. `false`

Valor predeterminado: `true`

`namedShadows`

(Opcional) La lista de sombras de dispositivos principales con nombre que se van a sincronizar. Debe especificar los nombres exactos de las sombras.

> **Warning**
>
> El AWS IoT Greengrass servicio usa el `AWSManagedGreengrassV2Deployment` nombre shadow para administrar las implementaciones que se dirigen a dispositivos principales individuales. Esta sombra denominada está reservada para que la utilice el AWS IoT Greengrass servicio. No actualice ni elimine la sombra con nombre asignado.

`shadowDocumentsMap`

(Opcional) El dispositivo adicional sombrea para sincronizarlo. El uso de este parámetro de configuración facilita la especificación de documentos ocultos. Se recomienda utilizar este parámetro en lugar del `shadowDocuments` objeto.

> **Note**
>
> Si especifica un `shadowDocumentsMap` objeto, no debe `shadowDocuments` especificarlo.

Cada objeto contiene la siguiente información:

`thingName`

La configuración de sombra de *`ThingName`* para esta configuración de sombra.

`classic`

(Opcional) Si no desea sincronizar la sombra de dispositivo clásica con el `thingName` dispositivo, configúrela en. `false`

`namedShadows`

La lista de sombras con nombre que quieres sincronizar. Debe especificar los nombres exactos de las sombras.

`shadowDocuments`

(Opcional) La lista de sombras de dispositivos adicionales que se van a sincronizar. Le recomendamos que utilice el `shadowDocumentsMap` parámetro en su lugar.

> **Note**
>
> Si especifica un `shadowDocuments` objeto, no debe `shadowDocumentsMap` especificarlo.

Cada objeto de esta lista contiene la siguiente información.

`thingName`

Nombre del dispositivo con el que se van a sincronizar las sombras.

`classic`

(Opcional) Si no quieres sincronizar la sombra clásica del dispositivo con el `thingName` dispositivo, configúrala en`false`.

Valor predeterminado: `true`

`namedShadows`

(Opcional) La lista de sombras de dispositivos con nombre que deseas sincronizar. Debe especificar los nombres exactos de las sombras.

`rateLimits`

(Opcional) La configuración que determina los límites de velocidad para las solicitudes de servicios paralelos.

Este objeto contiene la siguiente información.

`maxOutboundSyncUpdatesPerSecond`

(Opcional) El número máximo de solicitudes de sincronización por segundo que transmite el dispositivo.

Predeterminado: 100 solicitudes por segundo

`maxTotalLocalRequestsRate`

(Opcional) El número máximo de solicitudes de IPC locales por segundo que se envían al dispositivo principal.

Predeterminado: 200 solicitudes por segundo

`maxLocalRequestsPerSecondPerThing`

(Opcional) El número máximo de solicitudes de IPC locales por segundo que se envían para cada elemento de IoT conectado.

Predeterminado: 20 solicitudes por segundo para cada cosa

> **Note**
>
> Estos parámetros de límites de velocidad definen el número máximo de solicitudes por segundo para el servicio paralelo local. El número máximo de solicitudes por segundo para el servicio AWS IoT Device Shadow depende de usted Región de AWS. Para obtener más información, consulte los límites de la API de AWS IoT Device Shadow Service en Referencia general de Amazon Web Services.

`shadowDocumentSizeLimitBytes`

(Opcional) El tamaño máximo permitido de cada documento de estado JSON para las sombras locales.

Si aumentas este valor, también debes aumentar el límite de recursos del documento de estado JSON para las sombras de nubes. Para obtener más información, consulte los límites de la API de AWS IoT Device Shadow Service en Referencia general de Amazon Web Services.

Predeterminado: 8192 bytes

Máximo: 30720 bytes

Example Ejemplo: actualización de la combinación de configuraciones

El siguiente ejemplo muestra un ejemplo de actualización de la combinación de configuraciones con todos los parámetros de configuración disponibles para el componente shadow manager.

```
{
   "strategy":{
      "type":"periodic",
      "delay":300
   },
   "synchronize":{
      "shadowDocumentsMap":{
         "MyDevice1":{
            "classic":false,
            "namedShadows":[
               "MyShadowA",
               "MyShadowB"
            ]
         },
         "MyDevice2":{
            "classic":true,
            "namedShadows":[]
         }
      },
      "direction":"betweenDeviceAndCloud"
   },
   "rateLimits":{
      "maxOutboundSyncUpdatesPerSecond":100,
      "maxTotalLocalRequestsRate":200,
      "maxLocalRequestsPerSecondPerThing":20
   },
   "shadowDocumentSizeLimitBytes":8192
}
```

2.0.x

`synchronize`

(Opcional) La configuración de sincronización que determina cómo se sincronizan las sombras con el Nube de AWS.

> **Note**
>
> Debe crear una actualización de configuración con esta propiedad para sincronizar las sombras con. Nube de AWS

Este objeto contiene la siguiente información.

`coreThing`

(Opcional) Las sombras del dispositivo principal se sincronizan. Este objeto contiene la siguiente información.

`classic`

(Opcional) De forma predeterminada, el administrador de sombras sincroniza el estado local de la sombra clásica del dispositivo principal con el Nube de AWS. Si no quieres sincronizar la sombra clásica del dispositivo, configúrala en. `false`

Valor predeterminado: `true`

`namedShadows`

(Opcional) La lista de sombras de dispositivos principales con nombre que se van a sincronizar. Debe especificar los nombres exactos de las sombras.

> **Warning**
>
> El AWS IoT Greengrass servicio usa el `AWSManagedGreengrassV2Deployment` nombre shadow para administrar las implementaciones que se dirigen a dispositivos principales individuales. Esta sombra denominada está reservada para que la utilice el AWS IoT Greengrass servicio. No actualice ni elimine la sombra con nombre asignado.

`shadowDocumentsMap`

(Opcional) El dispositivo adicional sombrea para sincronizarlo. El uso de este parámetro de configuración facilita la especificación de documentos ocultos. Se recomienda utilizar este parámetro en lugar del `shadowDocuments` objeto.

> **Note**
>
> Si especifica un `shadowDocumentsMap` objeto, no debe `shadowDocuments` especificarlo.

Cada objeto contiene la siguiente información:

thingName

La configuración de sombra de *ThingName* para esta configuración de sombra.

`classic`

(Opcional) Si no desea sincronizar la sombra de dispositivo clásica con el `thingName` dispositivo, configúrela en. `false`

`namedShadows`

La lista de sombras con nombre que quieres sincronizar. Debe especificar los nombres exactos de las sombras.

`shadowDocuments`

(Opcional) La lista de sombras de dispositivos adicionales que se van a sincronizar. Le recomendamos que utilice el `shadowDocumentsMap` parámetro en su lugar.

> **Note**
>
> Si especifica un `shadowDocuments` objeto, no debe `shadowDocumentsMap` especificarlo.

Cada objeto de esta lista contiene la siguiente información.

`thingName`

Nombre del dispositivo con el que se van a sincronizar las sombras.

`classic`

(Opcional) Si no quieres sincronizar la sombra clásica del dispositivo con el `thingName` dispositivo, configúrala en`false`.

Valor predeterminado: `true`

`namedShadows`

(Opcional) La lista de sombras de dispositivos con nombre que deseas sincronizar. Debe especificar los nombres exactos de las sombras.

`rateLimits`

(Opcional) La configuración que determina los límites de velocidad para las solicitudes de servicios paralelos.

Este objeto contiene la siguiente información.

`maxOutboundSyncUpdatesPerSecond`

(Opcional) El número máximo de solicitudes de sincronización por segundo que transmite el dispositivo.

Predeterminado: 100 solicitudes por segundo

`maxTotalLocalRequestsRate`

(Opcional) El número máximo de solicitudes de IPC locales por segundo que se envían al dispositivo principal.

Predeterminado: 200 solicitudes por segundo

`maxLocalRequestsPerSecondPerThing`

(Opcional) El número máximo de solicitudes de IPC locales por segundo que se envían para cada elemento de IoT conectado.

Predeterminado: 20 solicitudes por segundo para cada cosa

> **Note**
>
> Estos parámetros de límites de velocidad definen el número máximo de solicitudes por segundo para el servicio paralelo local. El número máximo de solicitudes por segundo para el servicio AWS IoT Device Shadow depende de usted Región de AWS. Para obtener más información, consulte los límites de la API de AWS IoT Device Shadow Service en Referencia general de Amazon Web Services.

`shadowDocumentSizeLimitBytes`

(Opcional) El tamaño máximo permitido de cada documento de estado JSON para las sombras locales.

Si aumentas este valor, también debes aumentar el límite de recursos del documento de estado JSON para las sombras de nubes. Para obtener más información, consulte los límites de la API de AWS IoT Device Shadow Service en Referencia general de Amazon Web Services.

Predeterminado: 8192 bytes

Máximo: 30720 bytes

Example Ejemplo: actualización de la combinación de configuraciones

El siguiente ejemplo muestra un ejemplo de actualización de la combinación de configuraciones con todos los parámetros de configuración disponibles para el componente shadow manager.

```
{
  "synchronize": {
    "coreThing": {
      "classic": true,
      "namedShadows": [
        "MyCoreShadowA",
        "MyCoreShadowB"
      ]
    },
    "shadowDocuments": [
      {
        "thingName": "MyDevice1",
        "classic": false,
        "namedShadows": [
          "MyShadowA",
          "MyShadowB"
        ]
      },
      {
        "thingName": "MyDevice2",
        "classic": true,
        "namedShadows": []
      }
    ]
  },
  "rateLimits": {
    "maxOutboundSyncUpdatesPerSecond": 100,
    "maxTotalLocalRequestsRate": 200,
    "maxLocalRequestsPerSecondPerThing": 20
  },
  "shadowDocumentSizeLimitBytes": 8192
}
```

Archivo de registro local

Este componente utiliza el mismo archivo de registro que el componente núcleo de Greengrass.

Linux

```
/greengrass/v2/logs/greengrass.log
```

Windows

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

Para ver los registros de este componente

- Ejecute el siguiente comando en el dispositivo principal para ver el archivo de registro de este componente en tiempo real. Sustituya */greengrass/v2* o *C:\greengrass\v2* por la ruta a la carpeta AWS IoT Greengrass raíz.

  Linux

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

  Windows (PowerShell)

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

Registros de cambios

En la siguiente tabla se describen los cambios en cada versión del componente.

Versión	Cambios
2.3.8	Mejoras y correcciones de errores • Soluciona un problema por el que el administrador oculto crea una situación de punto muerto durante la conexión del cliente MQTT.

Versión	Cambios
2.3.7	Mejoras y correcciones de errores • Soluciona un problema por el que el administrador oculto registrab a periódicamente un `NullPointerException` error durante la sincronización del administrador oculto.
2.3.6	Mejoras y correcciones de errores • Soluciona un problema que provocaba que las propiedades ocultas que se Nube de AWS eliminaran mediante las actualizaciones mientras el dispositivo estaba desconectado siguieran existiendo en la sombra local tras recuperar la conectividad.
2.3.5	Versión actualizada para la versión 2.12.0 de Greengrass Nucleus.
2.3.4	Mejoras y correcciones de errores • Añade compatibilidad con documentos de estado oculto nulos y vacíos.
2.3.3	Versión actualizada para el lanzamiento de la versión 2.11.0 de Greengrass nucleus.
2.3.2	Mejoras y correcciones de errores • Soluciona un problema por el que el administrador oculto entra en el `BROKEN` estado cuando la base de datos oculta local está dañada. • Versión actualizada para el lanzamiento de la versión 2.10.0 de Greengrass nucleus.
2.3.1	Mejoras y correcciones de errores • Corrige una condición que podía impedir la sincronización de las actualizaciones de Cloud Shadow. • Soluciona un problema por el que los cambios en la configuración de sincronización de sombras con nombre se aplicaban solo a una sombra con nombre.

Versión	Cambios
2.3.0	Mejoras y correcciones de errores • Corrige un problema que podía impedir que las sombras se sincroniz aran cuando la clave privada del dispositivo Greengrass estaba almacenada en un módulo de seguridad de hardware.
2.2.4	Mejoras y correcciones de errores • Soluciona un problema por el que la validación del tamaño de la sombra no era coherente con el de la nube al actualizar el documento de sombra local. • Soluciona un problema por el que el administrador oculto deja de escuchar las actualizaciones de la configuración si una implementación se realiza RESET en los nodos de configuración.
2.2.3	Versión actualizada para la versión 2.9.0 de Greengrass Nucleus.
2.2.2	Versión actualizada para el lanzamiento de la versión 2.8.0 de Greengrass nucleus.
2.2.1	Versión actualizada para el lanzamiento de la versión 2.7.0 de Greengrass nucleus.

Versión	Cambios
2.2.0	Nuevas características • Añade compatibilidad con el servicio paralelo local a través de la interfaz local de publicación/suscripción. Ahora puede comunicarse con el agente local de mensajes de publicación/suscripción sobre temas de MQTT ocultos para obtener, actualizar y eliminar las sombras en el dispositivo principal. Esta función le permite conectar los dispositivos cliente al servicio paralelo local mediante el puente MQTT para retransmitir mensajes sobre temas ocultos entre los dispositivos cliente y la interfaz local de publicación/suscripción. Esta función requiere la versión 2.6.0 o posterior del componente núcleo de Greengrass. Para conectar los dispositivos cliente al servicio paralelo local, también debe utilizar la versión 2.2.0 o posterior del componente MQTT bridge. • Añade la `direction` opción que puede configurar para personalizar la dirección de sincronización de las sombras entre el servicio oculto local y el. Nube de AWS Puede configurar esta opción para reducir el ancho de banda y las conexiones al Nube de AWS.
2.1.1	Mejoras y correcciones de errores • Soluciona un problema por el que la profundidad máxima en las `reported` secciones `desired` y del documento de estado oculto del dispositivo JSON era de 4 niveles en lugar de 5 niveles. • Versión actualizada para la versión 2.6.0 de Greengrass Nucleus.
2.1.0	Nuevas características • Añade compatibilidad con intervalos periódicos de sincronización paralela, de modo que puede configurar el dispositivo principal para reducir el uso de ancho de banda y los cargos.
2.0.6	Esta versión contiene mejoras y correcciones de errores.
2.0.5	Versión actualizada para el lanzamiento de la versión 2.5.0 de Greengrass nucleus.

Versión	Cambios
2.0.4	Mejoras y correcciones de errores • Corrige un problema que provocaba que el administrador de sombras eliminara las versiones recién creadas de cualquier sombra que se hubiera eliminado anteriormente. • Actualiza la operación de `DeleteThingShadow` IPC para incrementar la versión oculta cuando se invoca.
2.0.3	Versión actualizada para la versión 2.4.0 de Greengrass Nucleus.
2.0.2	Mejoras y correcciones de errores • Se ha corregido un error que provocaba que el administrador de sombras no reconociera la `delta` propiedad al sincronizar los estados de sombra desde AWS IoT Core ella. • Se ha corregido un problema que a veces provocaba que las solicitudes de sincronización de una sombra se fusionaran de forma incorrecta.
2.0.1	Versión actualizada para el lanzamiento de la versión 2.3.0 de Greengrass nucleus.
2.0.0	Versión inicial.

Amazon SNS

El componente Amazon SNS (`aws.greengrass.SNS`) publica mensajes en un tema del Amazon Simple Notification Service (Amazon SNS). Puede usar este componente para enviar eventos desde los dispositivos principales de Greengrass a servidores web, direcciones de correo electrónico y otros suscriptores de mensajes. Para obtener más información, consulte ¿Qué es Amazon SNS? en la Guía para desarrolladores de Amazon Simple Notification Service.

Para publicar en un tema de Amazon SNS con este componente, publique un mensaje en el tema al que se suscribe este componente. De forma predeterminada, este componente se suscribe al tema `sns/message` local de publicación/suscripción. Puede especificar otros temas, incluidos los temas de AWS IoT Core MQTT, al implementar este componente.

En su componente personalizado, puede que desee implementar una lógica de filtrado o formato para procesar los mensajes de otras fuentes antes de publicarlos en este componente. Esto le permite centralizar la lógica de procesamiento de mensajes en un único componente.

> **Note**
>
> Este componente proporciona una funcionalidad similar a la del conector Amazon SNS de la versión 1. AWS IoT Greengrass Para obtener más información, consulte el conector Amazon SNS en la Guía para desarrolladores de la AWS IoT Greengrass V1.

Temas

Versiones

Este componente tiene las siguientes versiones:

- 2.1.x
- 2.0.x

Tipo

Este componente es un componente Lambda () `aws.greengrass.lambda`. El núcleo de Greengrass ejecuta la función Lambda de este componente mediante el componente Lambda launcher.

Para obtener más información, consulte Tipos de componentes.

Sistema operativo

Este componente solo se puede instalar en los dispositivos principales de Linux.

Requisitos

Este componente tiene los siguientes requisitos:

- El dispositivo principal debe cumplir los requisitos para ejecutar las funciones de Lambda. Si desea que el dispositivo principal ejecute funciones Lambda en contenedores, el dispositivo debe cumplir los requisitos para hacerlo. Para obtener más información, consulte Requisitos de la función de Lambda.
- Versión 3.7 de Python instalada en el dispositivo principal y añadida a la variable de entorno PATH.
- Un tema de Amazon SNS. Para obtener instrucciones, consulte el tema Creación de un tema de Amazon SNS en la Guía para desarrolladores de Amazon Simple Notification Service.
- El rol de dispositivo de Greengrass debe permitir la `sns:Publish` acción, como se muestra en el siguiente ejemplo de política de IAM.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "sns:Publish"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:sns:region:account-id:topic-name"
      ]
    }
  ]
```

```
}
```

Puede anular dinámicamente el tema predeterminado en la carga útil del mensaje de entrada para este componente. Si la aplicación utiliza esta función, la política de IAM debe incluir todos los temas de destino como recursos. Puede conceder acceso granular o condicional a recursos (por ejemplo, utilizando un esquema de nomenclatura con comodín *)

- Para recibir los datos de salida de este componente, debe combinar la siguiente actualización de configuración para el componente antiguo del router de suscripciones (`aws.greengrass.LegacySubscriptionRouter`) al implementar este componente. Esta configuración especifica el tema en el que este componente publica las respuestas.

  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"
      }
    }
  }
  ```

  - Sustituya la *región* por la Región de AWS que utilice.
  - Sustituya la *versión* por la versión de la función Lambda que ejecuta este componente. Para encontrar la versión de la función Lambda, debe ver la receta de la versión de este

componente que desee implementar. Abra la página de detalles de este componente en la AWS IoT Greengrass consola y busque el par clave-valor de la función Lambda. Este par clave-valor contiene el nombre y la versión de la función Lambda.

> **Important**
>
> Debe actualizar la versión de la función Lambda en el router de suscripción anterior cada vez que implemente este componente. Esto garantiza que utilice la versión correcta de la función Lambda para la versión del componente que implemente.

Para obtener más información, consulte Crear implementaciones.

- Se admite la ejecución del componente Amazon SNS en una VPC. Para implementar este componente en una VPC, se requiere lo siguiente.
  - El componente Amazon SNS debe tener una conectividad con la que tenga el punto de `sns.region.amazonaws.com` enlace de la VPC de. `com.amazonaws.us-east-1.sns`

Puntos de conexión y puertos

Este componente debe poder realizar solicitudes salientes a los siguientes puntos finales y puertos, además de a los puntos finales y puertos necesarios para el funcionamiento básico. Para obtener más información, consulte Permitir el tráfico del dispositivo a través de un proxy o firewall.

punto de enlace	Puerto	Obligatoria	Descripción
`sns.region.amazonaws.com`	443	Sí	Publicar mensajes en Amazon SNS.

Dependencias

Al implementar un componente, AWS IoT Greengrass también implementa versiones compatibles de sus dependencias. Esto significa que debe cumplir los requisitos del componente y de todas sus dependencias para poder implementarlo correctamente. En esta sección se enumeran las

dependencias de las versiones publicadas de este componente y las restricciones de las versiones semánticas que definen las versiones de los componentes para cada dependencia. También puede ver las dependencias de cada versión del componente en la consola. AWS IoT Greengrass En la página de detalles del componente, busque la lista de dependencias.

2.1.7

La siguiente tabla muestra las dependencias de la versión 2.1.7 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.13.0	Rígido
Lanzador Lambda	^2.0.0	Rígido
Tiempos de ejecución de Lambda	^2.0.0	Flexible
Servicio de intercambio de fichas	^2.0.0	Rígido

2.1.6

La siguiente tabla muestra las dependencias de la versión 2.1.6 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.12.0	Rígido
Lanzador Lambda	^2.0.0	Rígido
Tiempos de ejecución de Lambda	^2.0.0	Flexible
Servicio de intercambio de fichas	^2.0.0	Rígido

2.1.5

La siguiente tabla muestra las dependencias de la versión 2.1.5 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.11.0	Rígido
Lanzador Lambda	^2.0.0	Rígido
Tiempos de ejecución de Lambda	^2.0.0	Flexible
Servicio de intercambio de fichas	^2.0.0	Rígido

2.1.4

La siguiente tabla muestra las dependencias de la versión 2.1.4 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.10.0	Rígido
Lanzador Lambda	^2.0.0	Rígido
Tiempos de ejecución de Lambda	^2.0.0	Flexible
Servicio de intercambio de fichas	^2.0.0	Rígido

2.1.3

La siguiente tabla muestra las dependencias de la versión 2.1.3 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.9.0	Rígido
Lanzador Lambda	^2.0.0	Rígido
Tiempos de ejecución de Lambda	^2.0.0	Flexible
Servicio de intercambio de fichas	^2.0.0	Rígido

2.1.2

La siguiente tabla muestra las dependencias de la versión 2.1.2 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.8.0	Rígido
Lanzador Lambda	^2.0.0	Rígido
Tiempos de ejecución de Lambda	^2.0.0	Flexible
Servicio de intercambio de fichas	^2.0.0	Rígido

2.1.1

La siguiente tabla muestra las dependencias de la versión 2.1.1 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.7.0	Rígido
Lanzador Lambda	^2.0.0	Rígido

Dependencia	Versiones compatibles	Tipo de dependencia
Tiempos de ejecución de Lambda	^2.0.0	Flexible
Servicio de intercambio de fichas	^2.0.0	Rígido

2.0.8 - 2.1.0

La siguiente tabla muestra las dependencias de las versiones 2.0.8 y 2.1.0 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.6.0	Rígido
Lanzador Lambda	^2.0.0	Rígido
Tiempos de ejecución de Lambda	^2.0.0	Flexible
Servicio de intercambio de fichas	^2.0.0	Rígido

2.0.7

La siguiente tabla muestra las dependencias de la versión 2.0.7 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.5.0	Rígido
Lanzador Lambda	^2.0.0	Rígido
Tiempos de ejecución de Lambda	^2.0.0	Flexible

Dependencia	Versiones compatibles	Tipo de dependencia
Servicio de intercambio de fichas	^2.0.0	Rígido

2.0.6

La siguiente tabla muestra las dependencias de la versión 2.0.6 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.4.0	Rígido
Lanzador Lambda	^2.0.0	Rígido
Tiempos de ejecución de Lambda	^2.0.0	Flexible
Servicio de intercambio de fichas	^2.0.0	Rígido

2.0.5

La siguiente tabla muestra las dependencias de la versión 2.0.5 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.3.0	Rígido
Lanzador Lambda	^2.0.0	Rígido
Tiempos de ejecución de Lambda	^2.0.0	Flexible
Servicio de intercambio de fichas	^2.0.0	Rígido

2.0.4

La siguiente tabla muestra las dependencias de la versión 2.0.4 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.2.0	Rígido
Lanzador Lambda	^2.0.0	Rígido
Tiempos de ejecución de Lambda	^2.0.0	Flexible
Servicio de intercambio de fichas	^2.0.0	Rígido

2.0.3

La siguiente tabla muestra las dependencias de la versión 2.0.3 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.3 <2.1.0	Rígido
Lanzador Lambda	>=1.0.0	Rígido
Tiempos de ejecución de Lambda	>=1.0.0	Flexible
Servicio de intercambio de fichas	>=1.0.0	Rígido

Para obtener más información sobre las dependencias de los componentes, consulta la referencia de recetas de componentes.

Configuración

Este componente proporciona los siguientes parámetros de configuración que puede personalizar al implementar el componente.

> **Note**
>
> La configuración predeterminada de este componente incluye los parámetros de la función Lambda. Le recomendamos que edite solo los siguientes parámetros para configurar este componente en sus dispositivos.

`lambdaParams`

Objeto que contiene los parámetros de la función Lambda de este componente. Este objeto contiene la siguiente información:

`EnvironmentVariables`

Objeto que contiene los parámetros de la función Lambda. Este objeto contiene la siguiente información:

`DEFAULT_SNS_ARN`

El ARN del tema predeterminado de Amazon SNS en el que este componente publica los mensajes. Puede anular el tema de destino con la `sns_topic_arn` propiedad en la carga útil del mensaje de entrada.

`containerMode`

(Opcional) El modo de contenedorización de este componente. Puede elegir entre las siguientes opciones:

- `NoContainer`— El componente no se ejecuta en un entorno de ejecución aislado.
- `GreengrassContainer`— El componente se ejecuta en un entorno de ejecución aislado dentro del AWS IoT Greengrass contenedor.

Predeterminado: `GreengrassContainer`

`containerParams`

(Opcional) Un objeto que contiene los parámetros del contenedor de este componente. El componente utiliza estos parámetros si se especifica `GreengrassContainer` para`containerMode`.

Este objeto contiene la siguiente información:

`memorySize`

(Opcional) La cantidad de memoria (en kilobytes) que se va a asignar al componente.

El valor predeterminado es 512 MB (525.312 KB).

`pubsubTopics`

(Opcional) Objeto que contiene los temas a los que el componente se suscribe para recibir mensajes. Puede especificar cada tema y si el componente se suscribe a los temas de MQTT AWS IoT Core o a los temas de publicación/suscripción locales.

Este objeto contiene la siguiente información:

`0`— Se trata de un índice matricial en forma de cadena.

Objeto que contiene la siguiente información:

`type`

(Opcional) El tipo de mensajes de publicación/suscripción que utiliza este componente para suscribirse a los mensajes. Puede elegir entre las siguientes opciones:

- `PUB_SUB` — Suscribirse a mensajes locales de publicación/suscripción. Si elige esta opción, el tema no puede contener caracteres comodín de MQTT. Para obtener más información sobre cómo enviar mensajes desde un componente personalizado al especificar esta opción, consulte. Publicar/suscribir mensajes locales
- `IOT_CORE`— Suscríbase a los mensajes de AWS IoT Core MQTT. Si elige esta opción, el tema puede contener caracteres comodín de MQTT. Para obtener más información sobre cómo enviar mensajes desde componentes personalizados al especificar esta opción, consulte. Publicar/suscribir mensajes MQTT AWS IoT Core

Predeterminado: `PUB_SUB`

`topic`

(Opcional) El tema al que se suscribe el componente para recibir mensajes. Si lo especifica `IotCoretype`, puede usar los comodines MQTT (+y#) en este tema.

Example Ejemplo: actualización de la combinación de configuraciones (modo contenedor)

```
{
  "lambdaExecutionParameters": {
    "EnvironmentVariables": {
      "DEFAULT_SNS_ARN": "arn:aws:sns:us-west-2:123456789012:mytopic"
    }
  },
```

```
  "containerMode": "GreengrassContainer"
}
```

Example Ejemplo: actualización de la combinación de configuraciones (sin modo contenedor)

```
{
  "lambdaExecutionParameters": {
    "EnvironmentVariables": {
      "DEFAULT_SNS_ARN": "arn:aws:sns:us-west-2:123456789012:mytopic"
    }
  },
  "containerMode": "NoContainer"
}
```

Datos de entrada

Este componente acepta mensajes sobre el siguiente tema y publica el mensaje tal cual en el tema Amazon SNS de destino. De forma predeterminada, este componente se suscribe a los mensajes locales de publicación/suscripción. Para obtener más información sobre cómo publicar mensajes en este componente desde sus componentes personalizados, consulte. Publicar/suscribir mensajes locales

Tema predeterminado (publicación/suscripción local): `sns/message`

El mensaje acepta las siguientes propiedades. Los mensajes de entrada deben tener un formato JSON válido.

`request`

La información sobre el mensaje que se va a enviar al tema Amazon SNS.

Tipo: `object` que contiene la siguiente información:

`message`

El contenido del mensaje en forma de cadena.

Para enviar un objeto JSON, serialícelo como una cadena y especifique `json` la `message_structure` propiedad.

Tipo: `string`

`subject`

(Opcional) El asunto del mensaje.

Tipo: `string`

El asunto puede ser texto ASCII y tener hasta 100 caracteres. Debe empezar por una letra, un número o un signo de puntuación. No puede incluir saltos de línea ni caracteres de control.

`sns_topic_arn`

(Opcional) El ARN del tema de Amazon SNS en el que este componente publica el mensaje. Especifique esta propiedad para anular el tema predeterminado de Amazon SNS.

Tipo: `string`

`message_structure`

(Opcional) La estructura del mensaje. Especifique `json` que se envíe un mensaje JSON que se serialice como una cadena en la `content` propiedad.

Tipo: `string`

Valores válidos: `json`

`id`

Un ID arbitrario para la solicitud. Utilice esta propiedad para asignar una solicitud de entrada a una respuesta de salida. Al especificar esta propiedad, el componente establece la `id` propiedad del objeto de respuesta en este valor.

Tipo: `string`

> **Note**
>
> El tamaño del mensaje puede ser de 256 KB como máximo.

Example Ejemplo de entrada: mensaje en cadena

```
{
  "request": {
```

```
    "subject": "Message subject",
    "message": "Message data",
    "sns_topic_arn": "arn:aws:sns:region:account-id:topic2-name"
  },
  "id": "request123"
}
```

Example Ejemplo de entrada: mensaje JSON

```
{
  "request": {
    "subject": "Message subject",
    "message": "{ \"default\": \"Message data\" }",
    "message_structure": "json"
  },
  "id": "request123"
}
```

Datos de salida

Este componente publica las respuestas como datos de salida sobre el siguiente tema de MQTT de forma predeterminada. Debe especificar este tema como parte de `subject` la configuración del componente antiguo del router de suscripciones. Para obtener más información sobre cómo suscribirse a los mensajes sobre este tema en sus componentes personalizados, consultePublicar/suscribir mensajes MQTT AWS IoT Core.

Tema predeterminado (AWS IoT Core MQTT): `sns/message/status`

Example Ejemplo de salida: Correcto

```
{
  "response": {
    "sns_message_id": "f80a81bc-f44c-56f2-a0f0-d5af6a727c8a",
    "status": "success"
  },
  "id": "request123"
}
```

Example Ejemplo de salida: Error

```
{
```

```
  "response" : {
    "error": "InvalidInputException",
    "error_message": "SNS Topic Arn is invalid",
    "status": "fail"
  },
  "id": "request123"
}
```

Archivo de registro local

Este componente utiliza el siguiente archivo de registro.

```
/greengrass/v2/logs/aws.greengrass.SNS.log
```

Para ver los registros de este componente

- Ejecute el siguiente comando en el dispositivo principal para ver el archivo de registro de este componente en tiempo real. */greengrass/v2*Sustitúyalo por la ruta a la carpeta AWS IoT Greengrass raíz.

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

Licencias

Este componente incluye el siguiente software o licencias de terceros:

- AWS SDK for Python (Boto3)/Apache License 2.0
- botocore/Apache License 2.0
- dateutil/PSF License
- docutils/BSD License, GNU General Public License (GPL), Python Software Foundation License, Public Domain
- jmespath/MIT License
- s3transfer/Apache License 2.0
- urllib3/MIT License

Este componente se publica en virtud del contrato de licencia de software principal de Greengrass.

Registros de cambios

En la siguiente tabla se describen los cambios en cada versión del componente.

Versión	Cambios
2.1.7	Versión actualizada para la versión 2.12.0 de Greengrass nucleus.
2.1.6	Versión actualizada para la versión 2.11.0 de Greengrass nucleus.
2.1.5	Versión actualizada para la versión 2.10.0 de Greengrass nucleus.
2.1.4	Versión actualizada para la versión 2.9.0 de Greengrass Nucleus.
2.1.3	Versión actualizada para el lanzamiento de la versión 2.8.0 de Greengrass nucleus.
2.1.2	Versión actualizada para la versión 2.7.0 de Greengrass Nucleus.
2.1.1	Versión actualizada para la versión 2.6.0 de Greengrass Nucleus.
2.1.0	Nuevas características • Añade compatibilidad con las configuraciones de proxy de red HTTPS. Para obtener más información, consulte Realizar la conexión en el puerto 443 o a través de un proxy de red y Habilite el dispositivo principal para que confíe en un proxy HTTPS.
2.0.8	Versión actualizada para el lanzamiento de la versión 2.5.0 de Greengrass nucleus.
2.0.7	Versión actualizada para la versión 2.4.0 de Greengrass Nucleus.
2.0.6	Versión actualizada para el lanzamiento de la versión 2.3.0 de Greengrass nucleus.
2.0.5	Versión actualizada para el lanzamiento de la versión 2.2.0 de Greengrass nucleus.
2.0.4	Versión actualizada para el lanzamiento de la versión 2.1.0 de Greengrass nucleus.

Versión	Cambios
2.0.3	Versión inicial.

Administrador de transmisiones

El componente administrador de flujos (`aws.greengrass.StreamManager`) le permite procesar flujos de datos para transferirlos Nube de AWS desde los dispositivos principales de Greengrass.

Para obtener más información sobre cómo configurar y usar el administrador de transmisiones en componentes personalizados, consulteGestione los flujos de datos en los dispositivos principales de Greengrass.

Temas

- Versiones
- Tipo
- Sistema operativo
- Requisitos
- Dependencias
- Configuración
- Archivo de registro local
- Registros de cambios

Versiones

Este componente tiene las siguientes versiones:

- 2.1.x
- 2.0.x

> **Note**
>
> Si utilizas el administrador de transmisiones para exportar datos a la nube, no puedes actualizar la versión 2.0.7 del componente de administrador de transmisiones a una versión entre la v2.0.8 y la v2.0.11. Si va a implementar Stream Manager por primera vez, le

recomendamos encarecidamente que implemente la última versión del componente Stream Manager.

Tipo

Este componente es un componente genérico (`aws.greengrass.generic`). El núcleo de Greengrass ejecuta los scripts del ciclo de vida del componente.

Para obtener más información, consulte Tipos de componentes.

Sistema operativo

Este componente se puede instalar en los dispositivos principales que ejecutan los siguientes sistemas operativos:

- Linux
- Windows

Requisitos

Este componente tiene los siguientes requisitos:

- La función de intercambio de fichas debe permitir el acceso a los Nube de AWS destinos que utilices con Stream Manager. Para obtener más información, consulte:
  - the section called “Canales de AWS IoT Analytics”
  - the section called “Amazon Kinesis Data Streams”
  - the section called “AWS IoT SiteWise propiedades de activos”
  - the section called “Objetos de Amazon S3”
- Se admite la ejecución del componente administrador de flujos en una VPC. Para implementar este componente en una VPC, se requiere lo siguiente.
  - El componente del administrador de transmisiones debe tener conectividad con el AWS servicio en el que se publican los datos.
    - Amazon S3: `com.amazonaws.region.s3`
    - Amazon Kinesis Data Streams: `com.amazonaws.region.kinesis-streams`
    - AWS IoT SiteWise: `com.amazonaws.region.iotsitewise.data`

- Si publica datos en Amazon S3 en la `us-east-1` región, este componente intentará utilizar el punto de enlace global de S3 de forma predeterminada; sin embargo, este punto de enlace no está disponible a través del punto de enlace de la interfaz de VPC de Amazon S3. Para obtener más información, consulte Restricciones y limitaciones AWS PrivateLink de Amazon S3. Para resolver este problema, puede elegir entre las siguientes opciones.
  - Configure el componente del administrador de transmisiones para que utilice el punto final S3 regional de la `us-east-1` región, proporcionando la variable de `AWS_S3_US_EAST_1_REGIONAL_ENDPOINT=regional` entorno.
  - Cree un punto de enlace de VPC de puerta de enlace de Amazon S3 en lugar de un punto de enlace de VPC de interfaz de Amazon S3. Los puntos de enlace de la puerta de enlace S3 permiten el acceso al punto de enlace global de S3. Para obtener más información, consulte Crear un punto final de puerta de enlace.

Puntos finales y puertos

Este componente debe poder realizar solicitudes salientes a los siguientes puntos finales y puertos, además de a los puntos finales y puertos necesarios para el funcionamiento básico. Para obtener más información, consulte Permitir el tráfico del dispositivo a través de un proxy o firewall.

punto de enlace	Puerto	Obligatoria	Descripción
`iotanalytics.` *`region`*`.amazonaws.com`	443	No	Necesario si publica datos en. AWS IoT Analytics
`kinesis.`*`region`*`.amazonaws.com`	443	No	Obligatorio si publicas datos en Firehose.
`data.iotsitewise.` *`region`*`.amazonaws.com`	443	No	Obligatorio si publica datos en.

punto de enlace	Puerto	Obligatoria	Descripción
			AWS IoT SiteWise
`*.s3.amazonaws.com`	443	No	Es obligatorio si publicas datos en buckets de S3. Puede sustituir los por * el nombre de cada depósito en el que publique los datos.

Dependencias

Al implementar un componente, AWS IoT Greengrass también despliega versiones compatibles de sus dependencias. Esto significa que debe cumplir los requisitos del componente y de todas sus dependencias para poder implementarlo correctamente. En esta sección se enumeran las dependencias de las versiones publicadas de este componente y las restricciones de las versiones semánticas que definen las versiones de los componentes para cada dependencia. También puede ver las dependencias de cada versión del componente en la consola. AWS IoT Greengrass En la página de detalles del componente, busque la lista de dependencias.

2.1.11

La siguiente tabla muestra las dependencias de las versiones 2.1.11 a 2.1.10 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.13.0	Flexible
Servicio de intercambio de fichas	>=0.0.0	Rígido

2.1.9 – 2.1.10

La siguiente tabla muestra las dependencias de las versiones 2.1.9 a 2.1.10 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.12.0	Flexible
Servicio de intercambio de fichas	>=0.0.0	Rígido

2.1.5 – 2.1.8

La siguiente tabla muestra las dependencias de las versiones 2.1.5 a 2.1.8 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.11.0	Flexible
Servicio de intercambio de fichas	>=0.0.0	Rígido

2.1.2 – 2.1.4

La siguiente tabla muestra las dependencias de las versiones 2.1.2 a 2.1.4 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.10.0	Flexible

Dependencia	Versiones compatibles	Tipo de dependencia
Servicio de intercambio de fichas	>=0.0.0	Rígido

2.1.1

La siguiente tabla muestra las dependencias de la versión 2.1.1 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.9.0	Flexible
Servicio de intercambio de fichas	>=0.0.0	Rígido

2.1.0

La siguiente tabla muestra las dependencias de la versión 2.1.0 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.8.0	Flexible
Servicio de intercambio de fichas	>=0.0.0	Rígido

2.0.15

La siguiente tabla muestra las dependencias de la versión 2.0.15 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.7.0	Flexible
Servicio de intercambio de fichas	>=0.0.0	Rígido

2.0.13 and 2.0.14

La siguiente tabla muestra las dependencias de las versiones 2.0.13 y 2.0.14 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.6.0	Flexible
Servicio de intercambio de fichas	>=0.0.0	Rígido

2.0.11 and 2.0.12

La siguiente tabla muestra las dependencias de las versiones 2.0.11 y 2.0.12 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.5.0	Flexible
Servicio de intercambio de fichas	>=0.0.0	Rígido

2.0.10

La siguiente tabla muestra las dependencias de la versión 2.0.10 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.4.0	Flexible
Servicio de intercambio de fichas	>=0.0.0	Rígido

2.0.9

La siguiente tabla muestra las dependencias de la versión 2.0.9 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.3.0	Flexible
Servicio de intercambio de fichas	>=0.0.0	Rígido

2.0.8

La siguiente tabla muestra las dependencias de la versión 2.0.8 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.0 <2.2.0	Flexible
Servicio de intercambio de fichas	>=0.0.0	Rígido

2.0.7

La siguiente tabla muestra las dependencias de la versión 2.0.7 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.0.3 <2.1.0	Flexible
Servicio de intercambio de fichas	>=0.0.0	Rígido

Para obtener más información sobre las dependencias de los componentes, consulta la referencia de recetas de componentes.

Configuración

Este componente proporciona los siguientes parámetros de configuración que puede personalizar al implementar el componente.

`STREAM_MANAGER_STORE_ROOT_DIR`

(Opcional) La ruta absoluta del directorio local utilizado para almacenar las transmisiones. Este valor debe comenzar con una barra inclinada (por ejemplo, `/data`).

Debe especificar una carpeta existente y el usuario del sistema que ejecuta el componente del administrador de transmisiones debe tener permisos para leer y escribir en esta carpeta. Por ejemplo, puede ejecutar los siguientes comandos para crear y configurar una carpeta`/var/greengrass/streams`, que especifique como carpeta raíz del administrador de flujos. Estos comandos permiten al usuario predeterminado del sistema`ggc_user`,, leer y escribir en esta carpeta.

```
sudo mkdir /var/greengrass/streams
sudo chown ggc_user /var/greengrass/streams
sudo chmod 700 /var/greengrass/streams
```

Predeterminado: *`/greengrass/v2`*`/work/aws.greengrass.StreamManager`

`STREAM_MANAGER_SERVER_PORT`

(Opcional) El número de puerto local que se utilizará para comunicarse con el administrador de transmisiones.

Puede especificar el uso `0` de un puerto disponible de forma aleatoria.

Predeterminado: `8088`

`STREAM_MANAGER_AUTHENTICATE_CLIENT`

(Opcional) Puedes hacer que sea obligatorio que los clientes se autentiquen antes de poder interactuar con Stream Manager. El SDK de Stream Manager controla la interacción entre los clientes y el administrador de transmisiones. Este parámetro determina qué clientes pueden llamar al SDK de Stream Manager para trabajar con las transmisiones. Para obtener más información, consulta la autenticación de clientes de Stream Manager.

Si lo especificas`true`, el SDK de Stream Manager solo permite como clientes los componentes de Greengrass.

Si lo especificas`false`, el SDK de Stream Manager permite que todos los procesos del dispositivo principal sean clientes.

Predeterminado: `true`

`STREAM_MANAGER_EXPORTER_MAX_BANDWIDTH`

(Opcional) El ancho de banda máximo promedio (en kilobits por segundo) que Stream Manager puede usar para exportar datos.

Valor predeterminado: sin límite

`STREAM_MANAGER_EXPORTER_THREAD_POOL_SIZE`

(Opcional) El número máximo de subprocesos activos que Stream Manager puede usar para exportar datos.

El tamaño óptimo depende del hardware, el volumen de secuencias y la cantidad planificada de secuencias de exportación. Si la velocidad de exportación es lenta, puede ajustar esta configuración para encontrar el tamaño óptimo para su hardware y su caso de negocio. La CPU y la memoria del hardware del dispositivo principal son factores limitantes. Para comenzar, puede intentar establecer este valor igual a la cantidad de núcleos de procesador en el dispositivo.

Tenga cuidado de no establecer un tamaño superior al que admite el hardware. Cada transmisión consume recursos de hardware, así que intente limitar la cantidad de transmisiones de exportación en dispositivos restringidos.

Predeterminado: 5 hilos

`STREAM_MANAGER_EXPORTER_S3_DESTINATION_MULTIPART_UPLOAD_MIN_PART_SIZE_BYTES`

(Opcional) El tamaño mínimo (en bytes) de una parte en una carga multiparte a Amazon S3. El administrador de flujos utiliza esta configuración y el tamaño del archivo de entrada para determinar cómo agrupar los datos en una solicitud PUT de varias partes.

> **Note**
>
> El administrador de transmisiones usa la `sizeThresholdForMultipartUploadBytes` propiedad streams para determinar si se debe exportar a Amazon S3 como una carga de una o varias partes. AWS IoT Greengrass los componentes pueden establecer este umbral cuando crean una transmisión que exporta a Amazon S3.

Predeterminado: `5242880` (5 MB). También es el valor mínimo.

`LOG_LEVEL`

(Opcional) El nivel de registro del componente. Elija uno de los siguientes niveles de registro, que se muestran aquí en orden de niveles:

- `TRACE`
- `DEBUG`
- `INFO`
- `WARN`
- `ERROR`

Predeterminado: `INFO`

`JVM_ARGS`

(Opcional) Los argumentos personalizados de la máquina virtual Java para pasarlos al administrador de transmisiones al inicio. Separe los argumentos múltiples mediante espacios.

Utilice este parámetro sólo cuando deba anular la configuración predeterminada utilizada por la JVM. Por ejemplo, puede que necesite aumentar el tamaño predeterminado del montón si planea exportar un gran número de secuencias.

Example Ejemplo: actualización de la combinación de configuraciones

El siguiente ejemplo de configuración especifica el uso de un puerto no predeterminado.

```
{
  "STREAM_MANAGER_SERVER_PORT": "18088"
}
```

Archivo de registro local

Este componente utiliza el siguiente archivo de registro.

Linux

```
/greengrass/v2/logs/aws.greengrass.StreamManager.log
```

Windows

```
C:\greengrass\v2\logs\aws.greengrass.StreamManager.log
```

Para ver los registros de este componente

- Ejecute el siguiente comando en el dispositivo principal para ver el archivo de registro de este componente en tiempo real. Sustituya */greengrass/v2* o *C:\greengrass\v2* por la ruta a la carpeta AWS IoT Greengrass raíz.

  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
  ```

Registros de cambios

En la siguiente tabla se describen los cambios en cada versión del componente.

Versión	Cambios
2.1.12	Mejoras y correcciones de errores Actualiza el orden en que se utilizan las credenciales, de modo que las credenciales de Greengrass son las preferidas para las solicitudes de AWS servicio.
2.1.11	Versión actualizada para la versión 2.12.0 de Greengrass nucleus.
2.1.10	Mejoras y correcciones de errores Soluciona un problema por el que la configuración del proxy HTTPS no confiaba en la cadena de certificados de la autoridad de certificación (CA) de Greengrass.
2.1.9	Versión actualizada para la versión 2.11.0 de Greengrass nucleus.

Versión	Cambios
2.1.8	Mejoras y correcciones de errores Soluciona un problema que provocaba que el administrador de transmisiones volviera a intentar SiteWise exportar de forma infinita. `InvalidRequestException`
2.1.7	Correcciones de errores y mejoras Soluciona un problema por el que el administrador de transmisiones no podía leer correctamente la configuración del proxy.
2.1.6	Correcciones de errores y mejoras Corrige un problema que podía provocar un bloqueo al arrancar algunos procesadores ARMv8, incluido el Jetson Nano.
2.1.5	Versión actualizada para la versión 2.10.0 de Greengrass nucleus.
2.1.4	Mejoras y correcciones de errores • Soluciona un problema por el que las entradas del mismo activo inmobiliario con la misma marca de tiempo en un solo lote se devolvían `ConflictingOperationException` de la SiteWise API, lo que provocaba que el administrador de transmisiones volviera a intentarlo continuamente. • Actualiza el tiempo de espera de conexión predeterminado de 3 segundos a 1 minuto.
2.1.3	Correcciones de errores y mejoras Corrige un problema de inicio en el sistema operativo Windows cuando se ejecuta como usuario del SISTEMA.

Versión	Cambios
2.1.2	Correcciones de errores y mejoras • Soluciona un problema en los sistemas operativos Windows que utilizan un idioma distinto del inglés. • Versión actualizada para el lanzamiento de la versión 2.9.0 del núcleo de Greengrass.
2.1.1	Versión actualizada para el lanzamiento de la versión 2.8.0 de Greengrass nucleus.
2.1.0	Nuevas características • Actualiza este componente para enviar automáticamente las métricas de telemetría a Amazon. EventBridge Para obtener más información, consulte Recopile datos de telemetría del estado del sistema de los dispositivos principales AWS IoT Greengrass. Esta función requiere la versión 2.7.0 o posterior del componente núcleo de Greengrass. • Versión actualizada para el lanzamiento de la versión 2.7.0 de Greengrass Nucleus.
2.0.15	Versión actualizada para el lanzamiento de la versión 2.6.0 de Greengrass Nucleus.
2.0.14	Esta versión contiene correcciones de errores y mejoras.
2.0.13	Versión actualizada para el lanzamiento de la versión 2.5.0 de Greengrass Nucleus.
2.0.12	Mejoras y correcciones de errores Corrige un problema que impedía actualizar Stream Manager v2.0.7 a una versión entre la v2.0.8 y la v2.0.11. Si utilizas Stream Manager para exportar datos a la nube, ahora puedes actualizar a la versión 2.0.12.
2.0.11	Versión actualizada para el lanzamiento de la versión 2.4.0 de Greengrass Nucleus.

Versión	Cambios
2.0.10	Versión actualizada para el lanzamiento de la versión 2.3.0 de Greengrass nucleus.
2.0.9	Versión actualizada para el lanzamiento de la versión 2.2.0 de Greengrass nucleus.
2.0.8	Versión actualizada para el lanzamiento de la versión 2.1.0 de Greengrass nucleus.
2.0.7	Versión inicial.

Agente de Systems Manager

El componente AWS Systems Manager Agent (`aws.greengrass.SystemsManagerAgent`) instala el agente de Systems Manager para que pueda administrar los dispositivos principales con Systems Manager. Systems Manager es un AWS servicio que puede utilizar para ver y controlar su infraestructura AWS, incluidas las instancias de Amazon EC2, los servidores y máquinas virtuales (VM) locales y los dispositivos periféricos. Systems Manager le permite ver los datos operativos, automatizar las tareas operativas y mantener la seguridad y el cumplimiento. Para obtener más información, consulte ¿Qué es AWS Systems Manager? y Acerca de Systems Manager Agent en la Guía AWS Systems Manager del usuario.

Las herramientas y funciones de Systems Manager se denominan capacidades. Los dispositivos principales de Greengrass admiten todas las funciones de Systems Manager. Para obtener más información sobre estas capacidades y sobre cómo usar Systems Manager para administrar los dispositivos principales, consulte Capacidades de Systems Manager en la Guía del AWS Systems Manager usuario.

Temas

- Versiones
- Tipo
- Sistema operativo
- Requisitos
- Dependencias

- Configuración
- Archivo de registro local
- Véase también
- Registros de cambios

Versiones

Este componente tiene las siguientes versiones:

- 1.1.x
- 1.0.x

Tipo

Este componente es un componente genérico (`aws.greengrass.generic`). El núcleo de Greengrass ejecuta los scripts del ciclo de vida del componente.

Para obtener más información, consulte Tipos de componentes.

Sistema operativo

Este componente solo se puede instalar en los dispositivos principales de Linux.

Requisitos

Este componente tiene los siguientes requisitos:

- Un dispositivo principal de Greengrass que se ejecuta en una plataforma Linux de 64 bits: Armv8 (AArch64) o x86_64.
- Debe tener una función de servicio AWS Identity and Access Management (IAM) que pueda asumir Systems Manager. Esta función debe incluir la política ManagedInstanceCore administrada de AmazonSSM o una política personalizada que defina permisos equivalentes. Para obtener más información, consulte Crear una función de servicio de IAM para dispositivos perimetrales en la Guía del AWS Systems Manager usuario.

  Al implementar este componente, debe especificar el nombre de este rol para el parámetro de `SSMRegistrationRole` configuración.

- La función de dispositivo Greengrass debe permitir las acciones `ssm:AddTagsToResource` y`ssm:RegisterManagedInstance`. La función de dispositivo también debe permitir la `iam:PassRole` acción de la función de servicio de IAM que cumpla con el requisito anterior. El siguiente ejemplo de política de IAM concede estos permisos.

```
{
  "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": "*"
    }
  ]
}
```

Puntos finales y puertos

Este componente debe poder realizar solicitudes salientes a los siguientes puntos finales y puertos, además de a los puntos finales y puertos necesarios para el funcionamiento básico. Para obtener más información, consulte Permitir el tráfico del dispositivo a través de un proxy o firewall.

punto de enlace	Puerto	Obligatoria	Descripción
`ec2messages. region.amazonaws.com`	443	Sí	Comuníquese con el servicio

punto de enlace	Puerto	Obligatoria	Descripción
			Systems Manager en el Nube de AWS.
`ssm.region.amazonaws.com`	443	Sí	Registre el dispositivo principal como nodo gestionado por Systems Manager.
`ssmmessages. region.amazonaws.com`	443	Sí	Comuníquese con el Administrador de sesiones, una función de Systems Manager, en el Nube de AWS.

Para obtener más información, consulte Referencia: ec2messages, ssmmessages y otras llamadas a la API en la Guía del usuario.AWS Systems Manager

Dependencias

Al implementar un componente, AWS IoT Greengrass también despliega versiones compatibles de sus dependencias. Esto significa que debe cumplir los requisitos del componente y de todas sus dependencias para poder implementarlo correctamente. En esta sección se enumeran las

dependencias de las versiones publicadas de este componente y las restricciones de las versiones semánticas que definen las versiones de los componentes para cada dependencia. También puede ver las dependencias de cada versión del componente en la consola.AWS IoT Greengrass En la página de detalles del componente, busque la lista de dependencias.

La siguiente tabla muestra las dependencias de las versiones 1.0.0 a 1.2.4 de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Servicio de intercambio de fichas	^2.0.0	Flexible

Para obtener más información sobre las dependencias de los componentes, consulte la referencia de recetas de componentes.

Configuración

Este componente proporciona los siguientes parámetros de configuración que puede personalizar al implementar el componente.

`SSMRegistrationRole`

La función de servicio de IAM que puede asumir Systems Manager y que incluye la política ManagedInstanceCore administrada de AmazonSSM o una política personalizada que defina permisos equivalentes. Para obtener más información, consulte Crear una función de servicio de IAM para dispositivos periféricos en la Guía del usuario.AWS Systems Manager

`SSMOverrideExistingRegistration`

(Opcional) Si el dispositivo principal ya ejecuta el Agente de Systems Manager registrado con una activación híbrida, puede anular el registro del Agente de Systems Manager existente del dispositivo. Defina esta opción `true` para registrar el dispositivo principal como nodo gestionado mediante el Agente de Systems Manager que proporciona este componente.

> **Note**
>
> Esta opción solo se aplica a los dispositivos que están registrados con una activación híbrida. Si el dispositivo principal se ejecuta en una instancia de Amazon EC2 con el agente de Systems Manager instalado y un rol de perfil de instancia configurado, el ID de nodo gestionado existente de la instancia de Amazon EC2 comienza por. `i-` Al instalar

el componente Systems Manager Agent, el agente Systems Manager registra un nuevo nodo gestionado cuyo identificador empieza por `mi-` en lugar de`i-`. A continuación, puede utilizar el nodo gestionado cuyo ID comienza con `mi-` para gestionar el dispositivo principal con Systems Manager.

Valor predeterminado: `false`

`SSMResourceTags`

(Opcional) Las etiquetas que se van a añadir al nodo gestionado por Systems Manager que este componente crea para el dispositivo principal. Puede usar estas etiquetas para administrar grupos de dispositivos principales con Systems Manager. Por ejemplo, puede ejecutar un comando en todos los dispositivos que tengan una etiqueta que especifique.

Especifique una lista en la que cada etiqueta sea un objeto con una `Key` y una`Value`. Por ejemplo, el siguiente valor de `SSMResourceTags` indica a este componente que establezca la **`Owner`** etiqueta **`richard-roe`** en el nodo gestionado del dispositivo principal.

```
[
  {
    "Key": "Owner",
    "Value": "richard-roe"
  }
]
```

Este componente ignora estas etiquetas si el nodo gestionado ya existe y `SSMOverrideExistingRegistration` existe. `false`

Example Ejemplo: actualización de combinación de configuraciones

El siguiente ejemplo de configuración especifica el uso de un rol de servicio denominado `SSMServiceRole` para permitir que el dispositivo principal se registre y se comunique con Systems Manager.

```
{
  "SSMRegistrationRole": "SSMServiceRole",
  "SSMOverrideExistingRegistration": false,
  "SSMResourceTags": [
    {
      "Key": "Owner",
```

```
      "Value": "richard-roe"
    },
    {
      "Key": "Team",
      "Value": "solar"
    }
  ]
}
```

Archivo de registro local

El software Systems Manager Agent escribe los registros en una carpeta fuera de la carpeta raíz de Greengrass. Para obtener más información, consulte Visualización de los registros del agente de Systems Manager en la Guía del AWS Systems Manager usuario.

El componente Systems Manager Agent utiliza scripts de shell para instalar, iniciar y detener el agente Systems Manager. Puede encontrar el resultado de estos scripts en el siguiente archivo de registro.

```
/greengrass/v2/logs/aws.greengrass.SystemsManagerAgent.log
```

Para ver los registros de este componente

- Ejecute el siguiente comando en el dispositivo principal para ver el archivo de registro de este componente en tiempo real. */greengrass/v2*Sustitúyalo por la ruta a la carpeta AWS IoT Greengrass raíz.

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

Véase también

- Gestione los dispositivos principales de Greengrass con AWS Systems Manager
- ¿Qué es AWS Systems Manager? en la Guía del usuario de AWS Systems Manager
- Acerca de Systems Manager Agent en la Guía AWS Systems Manager del usuario

Registros de cambios

En la siguiente tabla se describen los cambios en cada versión del componente.

Versión	Cambios
1.2.4	Mejoras y correcciones de errores Actualiza este componente para obtener la versión 3.2.2303.0 del Agente.
1.2.3	Mejoras y correcciones de errores • Añade reintentos para la instalación del componente Agent con Snap en Greengrass. • Actualiza la configuración del componente Agent para usar solo la identidad Onprem en Greengrass. • Actualiza este componente para actualizar el agente solo cuando la versión del agente instalada no coincide con la versión del componente SSM Agent de Greengrass.
1.1.0	Esta versión contiene correcciones de errores y mejoras.
1.0.0	Versión inicial.

Servicio de intercambio de fichas

El componente del servicio de intercambio de tokens (`aws.greengrass.TokenExchangeService`) proporciona AWS credenciales que puede usar para interactuar con AWS los servicios de sus componentes personalizados.

El servicio de intercambio de tokens ejecuta una instancia de contenedor de Amazon Elastic Container Service (Amazon ECS) como servidor local. Este servidor local se conecta al proveedor de AWS IoT credenciales mediante el alias de AWS IoT rol que usted configura en el componente core core de Greengrass. El componente proporciona dos variables de entorno, `AWS_CONTAINER_CREDENTIALS_FULL_URI` y`AWS_CONTAINER_AUTHORIZATION_TOKEN`. `AWS_CONTAINER_CREDENTIALS_FULL_URI`define el URI de este servidor local. Cuando un componente crea un cliente AWS SDK, el cliente reconoce esta variable de entorno URI y utiliza el token que contiene `AWS_CONTAINER_AUTHORIZATION_TOKEN` para conectarse al servicio de intercambio de tokens y recuperar AWS las credenciales. Esto permite que los dispositivos principales de Greengrass llamen a las operaciones de AWS servicio. Para obtener más información sobre cómo usar este componente en componentes personalizados, consulteInteractúa con AWS los servicios.

> **Important**
>
> El 13 de julio de 2016 se agregó a los AWS SDK el soporte para adquirir AWS credenciales de esta manera. Su componente debe usar una versión AWS del SDK que se haya creado en esa fecha o después. Para obtener más información, consulte Uso de un AWS SDK compatible en la Guía para desarrolladores de Amazon Elastic Container Service.

Temas

Versiones

Este componente tiene las siguientes versiones:

- 2.0.x

Tipo

Este componente es un componente genérico () `aws.greengrass.generic`. El núcleo de Greengrass ejecuta los scripts del ciclo de vida del componente.

Para obtener más información, consulte Tipos de componentes.

Sistema operativo

Este componente se puede instalar en los dispositivos principales que ejecutan los siguientes sistemas operativos:

- Linux

- Windows

Dependencias

Este componente no tiene ninguna dependencia.

Configuración

Este componente no tiene ningún parámetro de configuración.

Archivo de registro local

Este componente utiliza el mismo archivo de registro que el componente núcleo de Greengrass.

Linux

```
/greengrass/v2/logs/greengrass.log
```

Windows

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

Para ver los registros de este componente

- Ejecute el siguiente comando en el dispositivo principal para ver el archivo de registro de este componente en tiempo real. Sustituya */greengrass/v2* o *C:\greengrass\v2* por la ruta a la carpeta AWS IoT Greengrass raíz.

  Linux

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

  Windows (PowerShell)

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

Registros de cambios

En la siguiente tabla se describen los cambios en cada versión del componente.

Versión	Cambios
2.0.3	Versión inicial.

Colector IoT SiteWise OPC-UA

El componente recopilador SiteWise OPC-UA de IoT (`aws.iot.SiteWiseEdgeCollectorOpcua`) permite que AWS IoT SiteWise las pasarelas recopilen datos de los servidores OPC-UA locales.

Con este componente, las AWS IoT SiteWise pasarelas se pueden conectar a varios servidores OPC-UA. Para obtener más información sobre AWS IoT SiteWise las puertas de enlace, consulte Uso AWS IoT SiteWise en el borde en la Guía del usuario.AWS IoT SiteWise

Temas

Versiones

Este componente tiene las siguientes versiones:

- 2.4.x

- 2.3.x
- 2.2.x
- 2.1.x
- 2.0.x

Tipo

Este componente es un componente genérico () `aws.greengrass.generic`. El núcleo de Greengrass ejecuta los scripts del ciclo de vida del componente.

Para obtener más información, consulte Tipos de componentes.

Sistema operativo

Este componente se puede instalar en los dispositivos principales que ejecutan los siguientes sistemas operativos:

- Linux
- Windows

Requisitos

Este componente tiene los siguientes requisitos:

- El dispositivo principal de Greengrass debe ejecutarse en una de las siguientes plataformas:
  - OS: Ubuntu 18.04 o posterior

    Arquitectura: x86_64 (AMD64) o ARMv8 (Aarch64)
  - OS: Red Hat Enterprise Linux (RHEL) 8

    Arquitectura: x86_64 (AMD64) o ARMv8 (Aarch64)
  - OS: Amazon Linux 2

    Arquitectura: x86_64 (AMD64) o ARMv8 (Aarch64)
  - OS: Debian 11

    Arquitectura: x86_64 (AMD64) o ARMv8 (Aarch64)

- Sistema operativo: Windows Server 2019 o posterior

  Arquitectura: x86_64 (AMD64)
- El dispositivo principal de Greengrass debe permitir la conectividad de red saliente a los servidores OPC-UA.

Dependencias

Al implementar un componente, AWS IoT Greengrass también despliega versiones compatibles de sus dependencias. Esto significa que debe cumplir los requisitos del componente y de todas sus dependencias para poder implementarlo correctamente. En esta sección se enumeran las dependencias de las versiones publicadas de este componente y las restricciones de las versiones semánticas que definen las versiones de los componentes para cada dependencia. También puede ver las dependencias de cada versión del componente en la consola.AWS IoT Greengrass En la página de detalles del componente, busque la lista de dependencias.

La siguiente tabla muestra las dependencias de todas las versiones de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.3.0 <3.0.0	Rígido
Administrador de transmisiones	>=2.3.0 2.0.10<3.0.0	Rígido
Gestor secreto	>2.0.10 =2.0.8 <3.0.0	Rígido

Para obtener más información sobre las dependencias de los componentes, consulta la referencia de recetas de componentes.

Configuración

Este componente no tiene ningún parámetro de configuración.

Puede usar la AWS IoT SiteWise consola o la API para configurar el componente recopilador SiteWise OPC-UA de IoT. Para obtener más información, consulte el paso 4: Añadir fuentes de datos (opcional) en la Guía del AWS IoT SiteWise usuario.

Datos de entrada

Este componente solo acepta datos en los siguientes formatos; todos los demás se ignorarán y descartarán. La siguiente tabla asigna los tipos de datos OPC UA a sus SiteWise equivalentes.

SiteWise tipo de datos	tipo de datos OPC UA	Descripción

`STRING`	`String`
	`Guid`
	`XmlElement`
		Cadena con una longitud máxima de 1024 bytes.

`INTEGER`	`SByte`
	`Byte`
	`Int16`
	`UInt16`
	`Int32`
	`UInt32*`
	`Int64*`
		Un entero de 32 bits con signo con un intervalo de`-2,147,483,648 to 2,147,483,647` .

`DOUBLE`	`UInt32*`
	`Int64*`
	`Float`
	`Double`
		Un número de coma flotante con un rango desde `-10^100 to 10^100` y una precisión `IEEE 754` doble.

`BOOLEAN`	`Boolean`	`true` o bien `false`.

* Para los tipos de datos OPC UA `UInt32` y`Int64`, su tipo de SiteWise datos será `INTEGER` si SiteWise es capaz de representar su valor; de lo contrario, lo será`DOUBLE`.

Datos de salida

Este componente escribe `BatchPutAssetPropertyValue` mensajes en el administrador de AWS IoT Greengrass transmisiones. Para obtener más información, consulte BatchPutAssetPropertyValue en la Referencia de la API de AWS IoT SiteWise .

Archivo de registro local

Este componente utiliza el siguiente archivo de registro.

Linux

```
/greengrass/v2/logs/aws.iot.SiteWiseEdgeCollectorOpcua.log
```

Windows

```
C:\greengrass\v2\logs\aws.iot.SiteWiseEdgeCollectorOpcua.log
```

Para ver los registros de este componente

- Ejecute el siguiente comando en el dispositivo principal para ver el archivo de registro de este componente en tiempo real. Sustituya */greengrass/v2* o *C:\greengrass\v2* por la ruta a la carpeta AWS IoT Greengrass raíz.

  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
  ```

Solución de problemas y depuración

Este componente incluye un nuevo registro de eventos para ayudar a los clientes a identificar y solucionar problemas. El archivo de registro es independiente del archivo de registro local y se

encuentra en la siguiente ubicación. Sustituya */greengrass/v2* o *C:\greengrass\v2* por la ruta de acceso a la carpeta AWS IoT Greengrass raíz.

Linux

```
/greengrass/v2/work/aws.iot.SiteWiseEdgeCollectorOpcua/logs/
IotSiteWiseOpcUaCollectorEvents.log
```

Windows

```
C:\greengrass\v2\work\aws.iot.SiteWiseEdgeCollectorOpcua\logs
\IotSiteWiseOpcUaCollectorEvents.log
```

Este registro incluye información detallada e instrucciones de solución de problemas. La información sobre la solución de problemas se proporciona junto con los diagnósticos, con una descripción de cómo solucionar el problema y, a veces, con enlaces a más información. La información de diagnóstico incluye lo siguiente:

- Nivel de gravedad
- Timestamp
- Información adicional específica del evento

Example Registro de ejemplo

```
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"'
```

Licencias

Este componente se publica en virtud del contrato de licencia de software principal de Greengrass.

Registros de cambios

La siguiente tabla describe los cambios en cada versión del componente.

Versión	Cambios
2.4.2	Mejoras y correcciones de errores • Soluciona problemas durante la detección del servidor OPC UA en los que se podía descubrir un nodo varias veces. • Corrige la función de instantáneas para garantizar que la marca de tiempo sea nueva para cada punto de datos de la instantánea.
2.4.1	Correcciones de errores y mejoras • Soluciona problemas relacionados con la compatibilidad con el proxy. • Soluciona el problema por el que la limpieza de subprocesos fallaba y provocaba un bloqueo de datos.
2.4.0	Nuevas características • Añade un registro de eventos para facilitar la identificación y la solución de los problemas. Mejoras y correcciones de errores • Corrige un problema con el cliente OPC-UA que provocaba errores de certificado al conectarse a un servidor OPC-UA que utilizaba la versión 1.05 de la especificación OPC-UA.

Versión	Cambios
2.3.0	Nuevas características • Añade compatibilidad con la configuración del proxy HTTP Greengrass nucleus en Linux. Mejoras y correcciones de errores • Corrige un problema de seguridad (CVE-2019-19135).

2.2.0	Nuevas características
	• Añade compatibilidad con la instalación del paquete de recopilación de datos en la arquitectura ARMv8 de Linux.
	• Requisitos mínimos para Linux ARMv8:
	  • Memoria: 4 GB
	  • CPU: ARM Cortex-A72 o especificación equivalente
	Mejoras y correcciones de errores
	• Mejora el registro de las métricas en el proceso de descubrimiento de nodos.
	• Mejora el manejo de los tipos de datos no compatibles.
	• Mejora el registro de los errores del flujo de datos.

2.1.3	Nuevas características • Añade compatibilidad con Windows Server 2019 o superior. Mejoras y correcciones de errores • Mejora los mensajes de error al implementar este componente en dispositivos no compatibles.

Versión	Cambios

2.1.1	Nuevas características
	• Añade soporte para configurar las siguientes propiedades de suscripci ón:
	  • DataChangeTrigger- Puede definir la condición que inicia una alerta de cambio de datos.
	  • QueueSize- La profundidad de la cola en un servidor OPC-UA para una métrica concreta en la que se ponen en cola las notificaciones de los elementos monitorizados.
	  • PublishingIntervalMilliseconds- El intervalo (en milisegundos) de un ciclo de publicación especificado al crear una suscripción.
	  • SnapshotFrequencyMilliseconds - Puede configurar el tiempo de espera de la frecuencia de las instantáneas para garantizar que AWS IoT SiteWise Edge ingiera un flujo constante de datos.
	• Esta versión admite la ingesta de datos de BAD calidad y filtra los datos en función de las siguientes calidades de datos:
	  • UNCERTAIN datos de calidad
	  • BADdatos de calidad
	Mejoras y correcciones de errores
	• Mejoras en las métricas de los clientes.
	• Corrige la codificación de seguridad que a veces causaba problemas al conectarse a servidores con el cifrado activado.
	• Soluciona un problema por el que el grupo de propiedades no se podía actualizar.

2.0.3	Mejoras y correcciones de errores.
2.0.2	Correcciones de errores y mejoras en la sincronización de prioridad de activos con Edge.
2.0.1	Versión inicial.

Véase también

- ¿Qué es? AWS IoT SiteWise en la Guía AWS IoT SiteWise del usuario.

Simulador de fuente de datos IoT SiteWise OPC-UA

El componente simulador de fuente de datos SiteWise OPC-UA de IoT (`aws.iot.SiteWiseEdgeOpcuaDataSourceSimulator`) inicia un servidor OPC-UA local que genera datos de muestra. Utilice este servidor OPC-UA para simular una fuente de datos leída por el componente recopilador SiteWise OPC-UA de IoT en una puerta de enlace. AWS IoT SiteWise A continuación, puede explorar las AWS IoT SiteWise funciones con estos datos de ejemplo. Para obtener más información sobre AWS IoT SiteWise las puertas de enlace, consulte Uso AWS IoT SiteWise en el borde en la Guía del AWS IoT SiteWise usuario.

Temas

Versiones

Este componente tiene las siguientes versiones:

- 1.0.x

Tipo

Este componente es un componente genérico (`aws.greengrass.generic`). El núcleo de Greengrass ejecuta los scripts del ciclo de vida del componente.

Para obtener más información, consulte Tipos de componentes.

Sistema operativo

Este componente se puede instalar en los dispositivos principales que ejecutan los siguientes sistemas operativos:

- Linux
- Windows

Requisitos

Este componente tiene los siguientes requisitos:

- El dispositivo principal de Greengrass debe poder utilizar el puerto 4840 del host local. El servidor OPC-UA local de este componente se ejecuta en este puerto.

Dependencias

Al implementar un componente, AWS IoT Greengrass también despliega versiones compatibles de sus dependencias. Esto significa que debe cumplir los requisitos del componente y de todas sus dependencias para poder implementarlo correctamente. En esta sección se enumeran las dependencias de las versiones publicadas de este componente y las restricciones de las versiones semánticas que definen las versiones de los componentes para cada dependencia. También puede ver las dependencias de cada versión del componente en la consola. AWS IoT Greengrass En la página de detalles del componente, busque la lista de dependencias.

La siguiente tabla muestra las dependencias de todas las versiones de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.3.0 <3.0.0	Rígido

Para obtener más información sobre las dependencias de los componentes, consulta la referencia de recetas de componentes.

Configuración

Este componente no tiene ningún parámetro de configuración.

Archivo de registro local

Este componente utiliza el siguiente archivo de registro.

Linux

```
/greengrass/v2/logs/aws.iot.SiteWiseEdgeOpcuaDataSourceSimulator.log
```

Windows

```
C:\greengrass\v2\logs\aws.iot.SiteWiseEdgeOpcuaDataSourceSimulator.log
```

Para ver los registros de este componente

- Ejecute el siguiente comando en el dispositivo principal para ver el archivo de registro de este componente en tiempo real. Sustituya */greengrass/v2* o *C:\greengrass\v2* por la ruta a la carpeta AWS IoT Greengrass raíz.

  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
  ```

Registros de cambios

En la siguiente tabla se describen los cambios en cada versión del componente.

Versión	Cambios
1.0.0	Versión inicial.

Versión	Cambios
	Añade compatibilidad con Windows Server 2016 o superior.

Véase también

- ¿Qué esAWS IoT SiteWise? en la Guía AWS IoT SiteWise del usuario.

SiteWise Publicador de IoT

El componente SiteWise publicador de IoT (`aws.iot.SiteWiseEdgePublisher`) permite a AWS IoT SiteWise las pasarelas exportar datos del borde al Nube de AWS.

Para obtener más información sobre AWS IoT SiteWise las puertas de enlace, consulte Uso AWS IoT SiteWise en el borde en la Guía del AWS IoT SiteWise usuario.

Temas

Versiones

Este componente tiene las siguientes versiones:

- 3.1.x

- 3.0.x
- 2.4.x
- 2.3.x
- 2.2.x
- 2.1.x
- 2.0.x

Tipo

Este componente es un componente genérico () `aws.greengrass.generic`. El núcleo de Greengrass ejecuta los scripts del ciclo de vida del componente.

Para obtener más información, consulte Tipos de componentes.

Sistema operativo

Este componente se puede instalar en los dispositivos principales que ejecutan los siguientes sistemas operativos:

- Linux
- Windows

Requisitos

Este componente tiene los siguientes requisitos:

- El dispositivo principal de Greengrass debe ejecutarse en una de las siguientes plataformas:
  - OS: Ubuntu 18.04 o posterior

    Arquitectura: x86_64 (AMD64) o ARMv8 (Aarch64)
  - OS: Red Hat Enterprise Linux (RHEL) 8

    Arquitectura: x86_64 (AMD64) o ARMv8 (Aarch64)
  - OS: Amazon Linux 2

    Arquitectura: x86_64 (AMD64) o ARMv8 (Aarch64)
  - OS: Debian 11

Arquitectura: x86_64 (AMD64) o ARMv8 (Aarch64)

- Sistema operativo: Windows Server 2019 o posterior

  Arquitectura: x86_64 (AMD64)

- El dispositivo principal de Greengrass debe estar conectado a Internet.
- El dispositivo principal de Greengrass debe estar autorizado para realizar la `iotsitewise:BatchPutAssetPropertyValue` acción. Para obtener más información, consulte Autorizar los dispositivos principales para que interactúen con AWS los servicios.

  Example política de permisos

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

Puntos finales y puertos

Este componente debe poder realizar solicitudes salientes a los siguientes puntos finales y puertos, además de a los puntos finales y puertos necesarios para el funcionamiento básico. Para obtener más información, consulte Permitir el tráfico del dispositivo a través de un proxy o firewall.

punto de enlace	Puerto	Obligatoria	Descripción
`data.iotsitewise.` *`region`*`.amazonaws.com`	443	Sí	Publique datos en. AWS IoT SiteWise

Dependencias

Al implementar un componente, AWS IoT Greengrass también despliega versiones compatibles de sus dependencias. Esto significa que debe cumplir los requisitos del componente y de todas sus dependencias para poder implementarlo correctamente. En esta sección se enumeran las dependencias de las versiones publicadas de este componente y las restricciones de las versiones semánticas que definen las versiones de los componentes para cada dependencia. También puede ver las dependencias de cada versión del componente en la consola.AWS IoT Greengrass En la página de detalles del componente, busque la lista de dependencias.

La siguiente tabla muestra las dependencias de las versiones 2.0.x a 2.2.x de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Núcleo de Greengrass	>=2.3.0<3.0.0	Rígido
Administrador de transmisiones	>=2.3.0 =2.0.10<3.0.0	Rígido

Para obtener más información sobre las dependencias de los componentes, consulta la referencia de recetas de componentes.

Configuración

Este componente no tiene ningún parámetro de configuración.

Puede usar la AWS IoT SiteWise consola o la API para configurar el componente de SiteWise editor de IoT. Para obtener más información, consulte el paso 3: Configurar el editor (opcional) en la Guía del AWS IoT SiteWise usuario.

Datos de entrada

Este componente lee `PutAssetPropertyValueEntry` los mensajes del administrador de AWS IoT Greengrass transmisiones. Para obtener más información, consulte PutAssetPropertyValueEntryla referencia de la AWS IoT SiteWise API.

Archivo de registro local

Este componente utiliza el siguiente archivo de registro.

Linux

```
/greengrass/v2/logs/aws.iot.SiteWiseEdgePublisher.log
```

Windows

```
C:\greengrass\v2\logs\aws.iot.SiteWiseEdgePublisher.log
```

Para ver los registros de este componente

- Ejecute el siguiente comando en el dispositivo principal para ver el archivo de registro de este componente en tiempo real. Sustituya */greengrass/v2* o *C:\greengrass\v2* por la ruta a la carpeta AWS IoT Greengrass raíz.

  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
  ```

Solución de problemas y depuración

Este componente incluye un nuevo registro de eventos para ayudar a los clientes a identificar y solucionar problemas. El archivo de registro es independiente del archivo de registro local y se encuentra en la siguiente ubicación. Sustituya */greengrass/v2* o *C:\greengrass\v2* por la ruta de acceso a la carpeta AWS IoT Greengrass raíz.

Linux

```
/greengrass/v2/work/aws.iot.SiteWiseEdgePublisher/logs/
IotSiteWisePublisherEvents.log
```

Windows

```
C:\greengrass\v2\work\aws.iot.SiteWiseEdgePublisher\logs
\IotSiteWisePublisherEvents.log
```

Este registro incluye información detallada e instrucciones de solución de problemas. La información sobre la solución de problemas se proporciona junto con los diagnósticos, con una descripción de cómo solucionar el problema y, a veces, con enlaces a más información. La información de diagnóstico incluye lo siguiente:

- Nivel de gravedad
- Timestamp
- Información adicional específica del evento

Example Registro de ejemplo

```
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
```

Licencias

Este componente se publica en virtud del contrato de licencia de software principal de Greengrass.

Registros de cambios

En la siguiente tabla se describen los cambios en cada versión del componente.

Versión	Cambios
3.1.3	Mejoras y correcciones de errores • Se resolvió un problema por el que se `/greengrass/v2/work/aws.iot.SiteWiseEdgePublisher/logs/IoTSiteWisePublisherEvents.log` creaba el archivo de registro de eventos ubicado en pero no se registraba ningún evento. • Se agregaron las siguientes CloudWatch métricas para monitorear la conexión con el broker MQTT: • `IoTSiteWisePublisher.IsConnectedToMqttBroker` • `IoTSiteWisePublisher.NumberOfSubscriptionsToMqttBroker` • `IoTSiteWisePublisher.NumberOfUniqueMqttTopicsReceived` • `IoTSiteWisePublisher.MqttMessageReceivedSuccessCount` • `IoTSiteWisePublisher.MqttReceivedSuccessBytes` Para obtener más información sobre estas métricas, consulte las métricas de la AWS IoT Greengrass Version 2 puerta de enlace. • Se resolvió un problema por el que se seguía llamando a la `BatchCreateJob` API aunque se produjera un error al cargar un archivo de parquet a S3.

Versión	Cambios
3.1.2	Mejoras y correcciones de errores • Se ha corregido el problema del elevado uso de la CPU introducido en la versión 3.1.1.
3.1.1	Mejoras y correcciones de errores • Añade un registro adicional que identifica los alias de datos afectados cuando se produce un error. • Añade la aplicación local de los límites de la AWS IoT SiteWise API en cuanto a la antigüedad de los datos ingeridos. • Soluciona el problema por el que Publisher confunde los puntos de control de las StreamManager transmisiones cuando hay varios destinos de Amazon S3. • Corrige un obstáculo en el rendimiento relacionado con la forma en que el editor lee las transmisiones. StreamManager
3.1.0	Nuevas características • Añade soporte para publicar datos como archivos de parquet en Amazon S3. • Añade compatibilidad con la ingesta en AWS IoT SiteWise búfer.
3.0.0	Mejoras y correcciones de errores • Soluciona problemas relacionados con la compatibilidad con el proxy. Nuevas características • Permite la ingesta de datos desde un bróker de MQTT.
2.4.1	Mejoras y correcciones de errores • Habilite el componente para que funcione con las versiones 11.0.20.8 .1 y posteriores de Java Corretto 11. Las versiones de los component es 2.4.0 y 2.3.3 muestran el mensaje de `"Could not find or load main class"` error cuando se utilizan con Java Corretto versión 11.0.20.8.1.

Versión	Cambios
2.4.0	Nuevas características • Añade un nuevo registro de eventos para facilitar la identificación y la solución de problemas. Mejoras y correcciones de errores • Mejora la recuperación de los puntos de control de Publisher.
2.3.3	Mejoras y correcciones de errores • Mejora la capacidad de soportar un alto rendimiento.
2.3.2	Mejoras y correcciones de errores • Corrige la compatibilidad con el proxy HTTP al descargar la configura ción de Publisher.
2.3.1	Nuevas características • Añade compatibilidad con la instalación del paquete de recopilación de datos en la arquitectura ARMv8 de Linux. • Requisitos mínimos para Linux ARMv8: • Memoria: 4 GB • CPU: ARM Cortex-A72 o especificación equivalente
2.2.3	Mejoras y correcciones de errores • Elimina el reintento de una excepción genérica que no estaba en la lista de excepciones recuperables.
2.2.2	Mejoras y correcciones de errores • Reintroduce la compatibilidad con la carga de datos a AWS IoT SiteWise través de un servidor proxy HTTP.

Versión	Cambios
2.2.1	**Note** Esta versión no admite la configuración de proxy HTTP. La versión 2.2.2 y las posteriores vuelven a introducir la compatibilidad con esta función. Nuevas características • Añade compatibilidad con este componente para activar o desactivar la compresión al cargar datos en él. AWS IoT SiteWise

Versión	Cambios

2.2.0	**Note**
	Esta versión no admite la configuración de proxy HTTP. La versión 2.2.2 y las posteriores vuelven a introducir la compatibilidad con esta función.
	
	Nuevas características
	• Actualiza este componente para comprimir los datos antes de enviarlos al AWS IoT SiteWise servicio.
	  • En la mayoría de los casos, este cambio reduce el uso del ancho de banda en un 75 por ciento en comparación con las versiones anteriores de este componente.
	  • En la mayoría de los casos, este cambio aumenta el uso de la CPU hasta un 5 por ciento. En las puertas de enlace que procesan grandes cantidades de datos, este cambio puede aumentar el uso de la CPU hasta en un 15 por ciento.
	  • Este cambio no afecta a los cargos por AWS IoT SiteWise servicio ni al uso de la cuota de servicio.
	  • Añade compatibilidad con Windows Server 2019 o superior.
	Mejoras y correcciones de errores
	• Corrige un problema que impedía que este componente se iniciara cuando el archivo de puntos de control estaba dañado.

2.1.4	Mejoras y correcciones de errores • Corrige la compatibilidad con la versión 8 de Java.

Versión	Cambios
2.1.3	**Warning** Esta versión ya no está disponible, excepto en las regiones EE.UU. Este (Ohio), Canadá (Centro) y AWS GovCloud (EE.UU. Este). Esta versión de componente requiere la versión 11 o superior de Java para ejecutarse. Las mejoras de esta versión están disponibles en versiones posteriores de este componente. Mejoras y correcciones de errores • Mejora los mensajes de error al implementar este componente en dispositivos no compatibles. • Actualiza los errores de registro cuando se produce un error al cargar los datos.
2.1.2	Mejoras y correcciones de errores • Se actualiza para activar la función de exportación de datos caducados tan pronto como los datos caduquen.
2.1.1	Mejoras y correcciones de errores.
2.1.0	Nuevas características • Añade compatibilidad para publicar primero los datos más recientes en la nube. • Añade compatibilidad para no publicar datos caducados en la nube. • Añade compatibilidad con el almacenamiento local de datos caducados. Mejoras y correcciones de errores • Reduce las E/S del disco y la latencia correspondiente.
2.0.2	Mejoras y correcciones de errores.
2.0.1	Versión inicial.

Véase también

- ¿Qué es AWS IoT SiteWise? en la Guía AWS IoT SiteWise del usuario.

SiteWise Procesador IoT

El componente de SiteWise procesador de IoT (`aws.iot.SiteWiseEdgeProcessor`) permite que AWS IoT SiteWise las pasarelas procesen los datos en la periferia.

Con este componente, AWS IoT SiteWise las pasarelas pueden utilizar modelos de activos y activos para procesar datos en los dispositivos de pasarela. Para obtener más información sobre AWS IoT SiteWise las puertas de enlace, consulte Uso AWS IoT SiteWise en el borde en la Guía del AWS IoT SiteWise usuario.

Temas

Versiones

Este componente tiene las siguientes versiones:

- 3.2.x
- 3.1.x
- 3.0.x
- 2.2.x

- 2.1.x
- 2.0.x

Tipo

Este componente es un componente genérico () `aws.greengrass.generic`. El núcleo de Greengrass ejecuta los scripts del ciclo de vida del componente.

Para obtener más información, consulte Tipos de componentes.

Sistema operativo

Este componente se puede instalar en los dispositivos principales que ejecutan los siguientes sistemas operativos:

- Linux
- Windows

Requisitos

Este componente tiene los siguientes requisitos:

- El dispositivo principal de Greengrass debe ejecutarse en una de las siguientes plataformas:
  - Sistema operativo: Ubuntu 20.04 o 18.04

    Arquitectura: x86_64 (AMD64)
  - OS: Red Hat Enterprise Linux (RHEL) 8

    Arquitectura: x86_64 (AMD64)
  - OS: Amazon Linux 2

    Arquitectura: x86_64 (AMD64)
  - Sistema operativo: Windows Server 2019 o posterior

    Arquitectura: x86_64 (AMD64)
- El dispositivo principal de Greengrass debe permitir el tráfico entrante en el puerto 443.
- El dispositivo principal de Greengrass debe permitir el tráfico saliente en los puertos 443 y 8883.

- Los siguientes puertos están reservados para su uso por AWS IoT SiteWise: 80, 443, 3001, 4569, 4572, 8000, 8081, 8082, 8084, 8085, 8086, 8445, 9000, 9500, 11080 y 50010. El uso de un puerto reservado para el tráfico puede causar la terminación de la conexión.

  > **Note**
  >
  > El puerto 8087 solo es necesario para la versión 2.0.15 y posteriores de este componente.

- El rol de dispositivo de Greengrass debe tener permisos que le permitan usar AWS IoT SiteWise puertas de enlace en sus dispositivos. AWS IoT Greengrass V2 Para obtener más información, consulte los requisitos en la Guía del AWS IoT SiteWise usuario.

Puntos finales y puertos

Este componente debe poder realizar solicitudes salientes a los siguientes puntos finales y puertos, además de a los puntos finales y puertos necesarios para el funcionamiento básico. Para obtener más información, consulte Permitir el tráfico del dispositivo a través de un proxy o firewall.

punto de enlace	Puerto	Obligatoria	Descripción
`model.iotsitewise.region.amazonaws.com`	443	Sí	Obtenga información sobre sus AWS IoT SiteWise activos y modelos de activos.
`edge.iotsitewise.region.amazonaws.com`	443	Sí	Obtenga información sobre la configuración de la AWS IoT SiteWise

punto de enlace	Puerto	Obligatoria	Descripción
			puerta de enlace del dispositivo principal.
ecr.*region*.amazonaws.com	443	Sí	Descargue las imágenes de Docker de AWS IoT SiteWise Edge Gateway desde Amazon Elastic Container Registry.
iot.*region*.amazonaws.com	443	Sí	Obtenga puntos de conexión de dispositivos para su. Cuenta de AWS
sts.*region*.amazonaws.com	443	Sí	Obtenga la identificación de su. Cuenta de AWS

punto de enlace	Puerto	Obligatoria	Descripción
`monitor.iotsitewise. region.amazonaws.com`	443	No	Obligatorio si accedes a los AWS IoT SiteWise Monitor portales desde el dispositivo principal.

Dependencias

Al implementar un componente, AWS IoT Greengrass también despliega versiones compatibles de sus dependencias. Esto significa que debe cumplir los requisitos del componente y de todas sus dependencias para poder implementarlo correctamente. En esta sección se enumeran las dependencias de las versiones publicadas de este componente y las restricciones de las versiones semánticas que definen las versiones de los componentes para cada dependencia. También puede ver las dependencias de cada versión del componente en la consola.AWS IoT Greengrass En la página de detalles del componente, busque la lista de dependencias.

La siguiente tabla muestra las dependencias de las versiones 2.0.x a 2.1.x de este componente.

Dependencia	Versiones compatibles	Tipo de dependencia
Servicio de intercambio de fichas	>=2.0.3 <3.0.0	Rígido
Administrador de transmisiones	>=2.0.3 =2.0.10 <3.0.0	Rígido
Greengrass CLI	>=2.3.0 <3.0.0	Rígido

Para obtener más información sobre las dependencias de los componentes, consulta la referencia de recetas de componentes.

Configuración

Este componente no tiene ningún parámetro de configuración.

Archivo de registro local

Este componente utiliza el siguiente archivo de registro.

Linux

```
/greengrass/v2/logs/aws.iot.SiteWiseEdgeProcessor.log
```

Windows

```
C:\greengrass\v2\logs\aws.iot.SiteWiseEdgeProcessor.log
```

Para ver los registros de este componente

- Ejecute el siguiente comando en el dispositivo principal para ver el archivo de registro de este componente en tiempo real. Sustituya */greengrass/v2* o *C:\greengrass\v2* por la ruta a la carpeta AWS IoT Greengrass raíz.

  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
  ```

Licencias

Este componente incluye el siguiente software o licencias de terceros:

- Apache-2.0

- MIT
- Cláusula BSD-2
- Cláusula BSD-3
- CDDL-1.0
- CDDL-1.1
- DISC
- Zlib
- GPL-3.0 con excepción de la GCC
- Dominio público
- Python-2.0
- Unicode-DFS-2015
- Cláusula BSD-1
- OpenSSL
- EPL-1.0
- EPL-2.0
- GPL-2.0 con excepción de ruta de clase
- MPL-2.0
- CC0-1.0
- JSON

Este componente se publica en virtud del contrato de licencia de software principal de Greengrass.

Registros de cambios

En la siguiente tabla se describen los cambios en cada versión del componente.

Versión	Cambios
3.2.1	Mejoras y correcciones de errores • Se solucionó el problema por el que las llamadas a la AWS IoT SiteWise API no se paginaban de forma sincrónica con Edge. SiteWise • Se solucionó el problema por el que ya no se publicaba la métrica`MessageRemaining.SiteWise_Edge_Stream` .

Versión	Cambios

	• Se agregaron las siguientes CloudWatch métricas para monitorear la conexión con el broker MQTT.
	  • `IoTSiteWiseProcessor.IsConnectedToMqttBroker`
	  • `IoTSiteWiseProcessor.NumberOfSubscriptionsToMqttBroker`
	  • `IoTSiteWiseProcessor.NumberOfUniqueMqttTopicsReceived`
	  • `IoTSiteWiseProcessor.MqttMessageReceivedSuccessCount`
	  • `IoTSiteWiseProcessor.MqttReceivedSuccessBytes`
	  Para obtener más información sobre estas métricas, consulte las métricas de la AWS IoT Greengrass Version 2 puerta de enlace.

Versión	Cambios
3.2.0	Mejoras de rendimiento • Optimice los servicios de API para que ocupen menos memoria y requieran menos espacio en disco para su instalación • Esto proporciona una reducción de 2 GB en el uso de memoria inicial (ahora utiliza 7,5 GB de memoria al inicio, aunque se recomiendan 16 GB) y una reducción de 500 MB en el tamaño de la descarga (ahora requiere una descarga de 1,4 GB) para todo el componente. Nuevas características • `GetAssetPropertyValueAggregates` La API ahora admite ventanas de agregación periféricas de 15 minutos. • Los puertos 8081 y 8082 ya no necesitan estar disponibles para que este componente se ejecute correctamente. **Note** El punto final local para las API del plano de AWS IoT SiteWise datos, por ejemplo`get-asset-property-value` , se está cambiando de a`http://localhost:8081` . `http://localhost:11080/data` El punto final local para las API del plano de AWS IoT SiteWise control, por ejemplo`list-asset-models`, se está cambiando de `http://localhost:11080` a`http://localhost:11080/control` . AWS siempre recomienda utilizar los puntos de enlace HTTPS de la puerta de enlace SiteWise Edge. Esos puntos finales no han cambiado. Mejoras y correcciones de errores • La sincronización desde ahora AWS IoT SiteWise hará que los recursos pasen a un estado válido si se interrumpió la sincronización anterior. Esto solucionará los problemas relacionados con la corrupción de algunos recursos tras un reinicio forzado. • Corrige un problema poco frecuente en el que un recurso puede dañarse en el extremo si se modifica durante la sincronización. La

Versión	Cambios
	sincronización ahora fallará si se detecta esta condición y el recurso se volverá a intentar en la siguiente sincronización. • Soluciona un problema que podría haber permitido llamar externamente al punto final HTTP de las API. Ahora solo se puede usar HTTPS para llamar a las API fuera de la dirección de bucle invertido local. • `ListAssets` La API ahora muestra las jerarquías de activos de los activos almacenados en la periferia. • Soluciona un problema que provocaba que el paquete de procesami ento de datos no se pudiera reiniciar, actualizar o degradar en Windows. • Corrige un error en el paquete de procesamiento de datos para el sistema operativo Windows que impedía a los clientes utilizar las credenciales para conectarse con un agente de MQTT.
3.1.3	Mejoras y correcciones de errores • Se solucionó el problema por el que el paquete de procesamiento de datos informaba incorrectamente de una sincronización correcta cuando algunos de los recursos fallaban. • Permita que varios activos tengan el mismo nombre siempre que no tengan el mismo elemento principal.
3.1.1	Mejoras y correcciones de errores • Se solucionó el problema por el que la solicitud de SiGv4 fallaba debido a una falta de coincidencia de zona horaria. • Se solucionó el problema por el que las propiedades de transform ación y métrica dejaban de calcularse cuando se basaban en atributos después de reiniciarse. • Habilite la compatibilidad con la configuración de puertos personalizada de Stream Manager. • Soluciona un problema por el que las propiedades que están sincroniz adas con el borde podrían dejar de actualizarse.

Versión	Cambios
3.1.0	Mejoras y correcciones de errores • Se ha solucionado el problema por el que la `ListAssetModels` API no podía generar el siguiente token.
3.0.0	Nuevas características • Permite la ingesta de datos desde un intermediario de MQTT.
2.2.1	Mejoras y correcciones de errores • Ajuste el proceso de sincronización para que el almacenamiento de datos del plano de control sea más coherente con el funcionamiento de la nube. Esto afecta levemente a la actualización. **Note** Los datos del plano de control sincronizados en la versión 2.2.1 o superior no serán compatibles con las versiones anteriores. Para cambiar a versiones anteriores, tendrás que realizar una instalación nueva. Esto no afecta a las actualizaciones, ya que los datos sincronizados en las versiones anteriores funcionarán con la versión 2.2.1. • Modificaciones adicionales en la cadena de AWS credenciales para AWS IoT Greengrass V2 priorizar las credenciales.
2.1.37	Mejoras y correcciones de errores • Elimine el dependency-routing-service proceso e incorpore su funcional idad al property-state-service proceso para reducir el uso de recursos de los procesos que se comunican. • Aumente el límite máximo de resultados de la `get-asset-property-value-history` API a 20 000 para que coincida con el límite utilizado por AWS IoT SiteWise. • Se solucionó un problema por el que el siguiente token no aparecía en los resultados paginados de la `get-asset-property-value-history` API cuando no se especificaba un límite máximo de resultados.

Versión	Cambios
2.1.35	Mejoras y correcciones de errores • Modifica la cadena de AWS credenciales para priorizar AWS IoT Greengrass las credenciales. • Soluciona un problema relacionado con la detección de cuentas al implementarlas como parte de un grupo AWS IoT Thing.
2.1.34	Mejoras y correcciones de errores • Ajusta los cálculos métricos/de transformación para utilizar subproces os múltiples en Linux. Windows sigue ejecutando cálculos de un solo subproceso para garantizar la compatibilidad. • Corrige un problema por el que faltaban cálculos métricos en algunas ventanas de cálculo.
2.1.33	Mejoras y correcciones de errores • Corrige un problema con los informes de estado de error a la consola de Greengrass.
2.1.32	Mejoras y correcciones de errores • Añade compatibilidad con nombres de usuario y grupos personalizados.
2.1.31	Mejoras y correcciones de errores • Añade soporte para calcular el promedio ponderado en el tiempo y la desviación estándar ponderada en el tiempo para los datos modelados. AWS IoT SiteWise
2.1.29	Mejoras y correcciones de errores • Añade compatibilidad con la funcionalidad de filtrado de activos en la periferia.
2.1.28	Mejoras y correcciones de errores • Optimiza la sincronización de recursos para permitir que una gran cantidad de activos se sincronicen desde el punto de Nube de AWS vista periférico.

Versión	Cambios
2.1.24	Mejoras y correcciones de errores • Corrige un problema que provocaba que el panel de control desaparec iera al sincronizar un recurso por segunda vez.
2.1.23	Mejoras y correcciones de errores • Se agregó un tiempo de espera para el proceso de `aws.iot.S iteWiseEdgeProcessor` instalación a fin de evitar errores en la instalación si la conectividad a Internet es lenta. • Sincronización de recursos optimizada para mejorar la eficiencia de la sincronización entre la nube y la periferia.
2.1.21	**Warning** La actualización de la versión 2.0.x a la 2.1.x provocará la pérdida de datos locales. Nuevas características • Añade compatibilidad con Windows Server 2019 o superior. • Elimina el docker para los sistemas operativos basados en Linux.
2.0.16	Esta versión contiene correcciones de errores y mejoras.
2.0.15	Mejoras y correcciones de errores • Cambia el puerto que utiliza este componente para las operaciones de la API de sincronización de recursos del 8085 al 8087. Como resultado , este componente ahora requiere que el puerto 8087 esté disponible. Este componente aún requiere que el puerto 8085 esté disponible. • Actualiza la AWS OpsHub autenticación para denegar a los usuarios no autorizados durante el inicio de sesión, en lugar de cuando un usuario intenta llamar a las operaciones de la API.
2.0.14	Esta versión contiene correcciones de errores y mejoras.

Versión	Cambios
2.0.13	Mejoras y correcciones de errores • Soluciona un problema por el que, cuando este componente reporta datos a CloudWatch las métricas de Amazon, ahora indica correctam ente qué datos no están modelados.
2.0.9	Mejoras y correcciones de errores • Mejora la confiabilidad a la hora de crear y actualizar AWS IoT SiteWise recursos en el dispositivo principal. • Agrega operaciones de API locales adicionales que puede usar para monitorear qué componentes están instalados en el dispositivo principal , la versión de cada componente y el estado de cada componente. Puede ver esta información en la pestaña Configuración de la AWS IoT SiteWise aplicación AWS OpsHub correspondiente al dispositivo principal. • Añade un estado de salud a los contenedores de Docker en los que se ejecuta este componente. Puede ejecutar el `docker ps` comando para ver el estado de salud de los contenedores.
2.0.7	Mejoras y correcciones de errores • Corrige la compatibilidad con la visualización de AWS IoT SiteWise Monitor portales en el dispositivo principal.
2.0.6	Correcciones de errores y mejoras • Corrige las `latest()` funciones AWS IoT SiteWise `statetime () earliest()` , y que este componente calcula en el dispositivo principal.
2.0.5	Mejoras y correcciones de errores • Añade compatibilidad con la AWS IoT SiteWise `pretrigger()` función en las transformaciones que este componente calcula en el dispositivo principal. • Cambia la ruta en la que este componente almacena la configuración del Protocolo ligero de acceso a directorios (LDAP) para la autentica ción.

Versión	Cambios
2.0.2	Versión inicial.

Véase también

- ¿Qué es? AWS IoT SiteWise en la Guía AWS IoT SiteWise del usuario.

Componentes compatibles con Publisher

Los componentes compatibles con Publisher se encuentran en una versión preliminar y están sujetos a cambios. AWS IoT Greengrass Estos componentes no son compatibles con. AWS Debe ponerse en contacto con el editor si tiene algún problema con cada uno de los componentes.

Los componentes compatibles con Greengrass Publisher son desarrollados, ofrecidos y mantenidos por proveedores de componentes externos. Los proveedores de componentes de terceros provienen de AWS Partner Device Catalog, AWS Heroes o vendedores comunitarios. Para comprar los componentes de este catálogo, póngase en contacto directamente con el proveedor de componentes externo.

Los componentes compatibles con Greengrass Publisher incluyen los siguientes:

Temas

- AiShield.edge
- EdgeLabs Sensor de IA
- Inversor Greengrass S3

AiShield.edge

Este componente fue desarrollado y es compatible con AiShield, con tecnología Bosch. Aumente la seguridad de su IA con AIShield.edge. Este componente está diseñado para implementar sin problemas defensas personalizadas e informadas sobre las amenazas en los dispositivos periféricos, lo que protege sus dispositivos contra los ataques de la IA.

Este componente ofrece las siguientes ventajas:

- Pase sin problemas del análisis de vulnerabilidades con AiShield AI Security a las defensas periféricas reforzadas internas AWS
- Implemente defensas personalizadas en varios dispositivos periféricos con facilidad
- Amplia protección adaptada a diversas configuraciones de IA que admite varios tipos de modelos y marcos
- Manténgase actualizado con una integración perfecta en los flujos de Amazon SageMaker trabajo de Greengrass
- Obtenga información inmediata sobre las posibles amenazas, con los datos transmitidos directamente a AWS IoT Core
- AiShield AI Security on the Marketplace ofrece una vía de seguridad de IA cohesiva para el despliegue de la defensa en la periferia AWS

Este componente debe ejecutarse en la siguiente plataforma:

- Sistema operativo: Linux

Si está interesado en adquirir este componente, póngase en contacto con Bosch Software and Digital Solutions: `<AIShield.Contact@bosch.com>`.

EdgeLabs Sensor de IA

Este componente fue desarrollado y es compatible con AI EdgeLabs. AI EdgeLabs Sensor es una aplicación basada en contenedores que contiene capacidades de detección y prevención de amenazas basadas en la IA. El sensor de IA está integrado en un componente de Greengrass y se implementa como un contenedor independiente en el dispositivo principal junto con otros componentes de Greengrass.

Este componente actual es un agente basado en contenedores que verifica continuamente la comunicación de la red y busca patrones de amenazas en el software que se ejecuta en el Edge Host o en la puerta de enlace de IoT. Este componente utiliza eBPF, la verificación del comportamiento del ancho de banda de los procesos y la configuración basada en el host. La funcionalidad principal de este componente se basa en las funciones NDR/IPS y EDR.

Este componente ofrece las siguientes ventajas:

- Detección de amenazas basada en IA contra ataques de red y malware (EDR/NDR)
- Respuesta a incidentes (IPS) automatizada basada en la IA

- Inteligencia de amenazas local en el servidor con una transferencia de datos externa mínima
- Despliegue ligero con Docker y Greengrass

Este componente debe ejecutarse en una de las siguientes plataformas:

- Sistema operativo: Linux

Si está interesado en comprar este componente, póngase en contacto con AI EdgeLabs: `<contact@edgelabs.ai>`.

Inversor Greengrass S3

Este componente fue desarrollado y cuenta con el apoyo de Nathan Glover. El componente Greengrass S3 Ingestor está diseñado para usarse con el componente administrador de flujos. Este componente toma un flujo de mensajes JSON delimitado por líneas del administrador de flujos y los agrupa en un archivo GZIP. Este componente permite la ingesta eficiente de datos en Amazon S3 para su posterior procesamiento o almacenamiento. Este componente no admite el envío de datos a Internet Nube de AWS en tiempo real.

Este componente debe ejecutarse en una de las siguientes plataformas:

- Sistema operativo: Linux
- Sistema operativo: Windows

Si está interesado en adquirir este componente, póngase en contacto con Nathan Glover: `<nathan@glovers.id.au.>`

Componentes de la comunidad

El catálogo de software de Greengrass es un índice de los componentes de Greengrass desarrollados por la comunidad de Greengrass. Desde este catálogo, puede descargar, modificar e implementar componentes para crear sus aplicaciones de Greengrass. Puede ver el catálogo en el siguiente enlace: https://github.com/aws-greengrass/ aws-greengrass-software-catalog.

Cada componente tiene un GitHub repositorio público que puede explorar. Consulte el catálogo de software de Greengrass GitHub para encontrar la lista completa de componentes de la comunidad. Por ejemplo, este catálogo incluye los siguientes componentes:

- Amazon Kinesis Video Streams

  Este componente ingiere transmisiones de audio y vídeo de cámaras locales que utilizan el Protocolo de transmisión en tiempo real (RTSP). A continuación, el componente carga las transmisiones de audio y vídeo a Amazon Kinesis Video Streams.

- Puerta de enlace Bluetooth IoT

  Este componente utiliza la BluePybiblioteca que permite la comunicación con dispositivos Bluetooth de bajo consumo (LE) para crear interfaces de cliente Bluetooth LE.

- Rotador de certificados

  Este componente proporciona una forma de transferir el certificado y la clave privada del dispositivo AWS IoT Greengrass principal a toda la flota y a gran escala.

- Túneles seguros en contenedores

  Este componente proporciona un contenedor Docker para una tunelización segura con todas las dependencias y bibliotecas coincidentes en una fórmula reutilizable que no depende de un sistema operativo host específico.

- Grafana

  Este componente le permite alojar un servidor Grafana en un dispositivo central de Greengrass. Puede usar los paneles de Grafana para visualizar y administrar los datos en el dispositivo principal.

- GStreamer para Amazon Lookout for Vision

  Este componente proporciona un complemento de GStreamer para que puedas realizar la detección de anomalías de Lookout for Vision en tus canalizaciones de GStreamer personalizadas.

- Asistente doméstico

  Este componente permite al cliente utilizar Home Assistant para controlar localmente los dispositivos domésticos inteligentes. Proporciona integración con AWS servicios periféricos y en la nube para ofrecer soluciones de automatización del hogar que amplían Home Assistant.

- Panel de control de InfluxDBGrafana

  Este componente proporciona una experiencia de un solo clic para configurar los componentes InfluxDB y Grafana. Conecta InfluxDB a Grafana y automatiza la configuración de un panel local de Grafana que representa la telemetría en tiempo real. AWS IoT Greengrass

- InfluxDB

Este componente proporciona una base de datos de series temporales de InfluxDB en un dispositivo central de Greengrass. Puede usar este componente para procesar los datos de los sensores de IoT, analizar los datos en tiempo real y monitorear las operaciones en la periferia.

- Publicador InfluxDB

  Este componente transmite la telemetría AWS IoT Greengrass del estado del sistema desde el complemento emisor Nucleus a InfluxDB. Este componente también puede reenviar telemetría personalizada a InfluxDB.

- Marco pubsub para IoT

  Este marco proporciona una arquitectura de aplicación, un código de plantilla y ejemplos desplegables que ayudan a mejorar la calidad del código para las aplicaciones pubsub de IoT distribuidas y basadas en eventos que AWS IoT Greengrass utilizan componentes personalizados de la versión 2. Para obtener más información, consulte Crear AWS IoT Greengrass componentes.

- Jupyter Labs

  Este componente se despliega en un JupyterLab AWS IoT Greengrass dispositivo central. El entorno Jupyter tiene acceso a los recursos variables de proceso y entorno establecidos por él AWS IoT Greengrass, lo que simplifica el proceso de prueba y desarrollo de componentes escritos en Python.

- Servidor web local

  Este componente le permite crear una interfaz de usuario web local en un dispositivo principal de Greengrass. Puede crear una interfaz de usuario web local que le permita configurar los ajustes del dispositivo y la aplicación o monitorizar el dispositivo, por ejemplo.

- LoRaWaAdaptador de protocolo N

  Este componente ingiere datos de dispositivos inalámbricos locales que utilizan el protocolo LoRaWa N, que es un protocolo de red de área amplia (LPWAN) de bajo consumo. El componente le permite analizar los datos y actuar sobre ellos de forma local sin comunicarse con la nube.

- Modbus TCP

  Este componente recopila datos de dispositivos locales mediante el protocolo ModbusTCP y los publica en los flujos de datos seleccionados.

- Node-RED

Este componente instala Node-RED en un AWS IoT Greengrass dispositivo principal mediante NPM. El componente depende del componente de autenticación de Node-RED, que debe implementarse y configurarse de forma explícita. Puede usar la CLI de Node-RED para Greengrass a fin de implementar flujos de Node-RED en los dispositivos. AWS IoT Greengrass

- Docker de Node-RED

  Este componente instala Node-RED en el dispositivo AWS IoT Greengrass principal mediante el contenedor Docker oficial de Node-RED. El componente depende del componente de autenticación de Node-RED, que debe implementarse y configurarse de forma explícita. Puede usar la CLI de Node-RED para Greengrass a fin de implementar flujos de Node-RED en los dispositivos. AWS IoT Greengrass

- Autenticación Node-RED

  Este componente configura un nombre de usuario y una contraseña para proteger la instancia de Node-RED que se ejecuta en un dispositivo principal. AWS IoT Greengrass

- OpenThreadBorder Router

  Este componente implementa el contenedor Docker de OpenThread Border Router. El componente ayuda a crear un dispositivo Matter que incluye un router Thread Border.

- Conector de transmisión de datos OSI Pi

  Este componente permite transmitir la ingesta de datos en tiempo real desde el archivo de datos OSI Pi a una arquitectura de datos moderna. AWS Se integra en OSI Pi Asset Framework, que se gestiona de forma centralizada a través de la AWS IoT PubSub mensajería.

- Proveedor de Parsec

  Este componente permite a AWS IoT Greengrass los dispositivos integrar soluciones de seguridad de hardware mediante el proyecto Parsec de código abierto de Cloud Native Computing Foundation (CNCF).

- Base de datos PostgreSQL

  Este componente proporciona soporte para la base de datos relacional PostgreSQL en la periferia. Los clientes pueden usar este componente para aprovisionar y administrar una instancia local de PostgreSQL dentro de un contenedor docker.

- Cargador de archivos S3

Este componente supervisa un directorio en busca de archivos nuevos, los carga en Amazon Simple Storage Service (Amazon S3) y, a continuación, los elimina una vez que la carga se ha realizado correctamente.

- Cliente Secrets Manager

  Este componente proporciona una herramienta CLI que pueden utilizar otros componentes que necesiten recuperar secretos del componente Secrets Manager en un script de ciclo de vida de recetas.

- Enrutamiento de TES al contenedor

  Este componente configura nftables o iptables en un AWS IoT Greengrass dispositivo para que pueda usar el Servicio de intercambio de fichas componente con contenedores.

- WebRTC

  Este componente ingiere las transmisiones de audio y vídeo de las cámaras RTSP conectadas al dispositivo principal. AWS IoT Greengrass A continuación, el componente convierte las transmisiones de audio y vídeo en peer-to-peer comunicación o retransmisión a través de Amazon Kinesis Video Streams.

Para solicitar una función o informar de un error, abra un GitHub problema en el repositorio de ese componente. AWS no proporciona soporte para los componentes de la comunidad. Para obtener más información, consulte el CONTRIBUTING.mdarchivo del repositorio de cada componente.

Varios AWS de los componentes proporcionados también son de código abierto. Para obtener más información, consulte Software AWS IoT Greengrass Core de código abierto.

AWS IoT Greengrassherramientas de desarrollo

Utilice las herramientas de AWS IoT Greengrass desarrollo para crear, probar, compilar, publicar e implementar componentes personalizados de Greengrass.

- Kit de desarrollo de Greengrass (CLI)

  Utilice la interfaz de línea de comandos del kit de AWS IoT Greengrass desarrollo (CLI de GDK) en su entorno de desarrollo local para crear componentes a partir de plantillas y componentes de la comunidad en el catálogo de software de Greengrass. Puede usar la CLI de GDK para crear

el componente y publicarlo en el AWS IoT Greengrass servicio como un componente privado en suCuenta de AWS.

- Interfaz de línea de comandos Greengrass

  Utilice la interfaz de línea de comandos de Greengrass (CLI de Greengrass) en los dispositivos principales de Greengrass para implementar y depurar los componentes de Greengrass. La CLI de Greengrass es un componente que puede implementar en sus dispositivos principales para crear despliegues locales, ver detalles sobre los componentes instalados y explorar los archivos de registro.

- Consola de depuración local

  Utilice la consola de depuración local de los dispositivos principales de Greengrass para implementar y depurar los componentes de Greengrass mediante la interfaz web de un panel de control local. La consola de depuración local es un componente que puede implementar en sus dispositivos principales para crear implementaciones locales y ver detalles sobre los componentes instalados.

AWS IoT Greengrasstambién proporciona los siguientes SDK que puede usar en los componentes personalizados de Greengrass:

- TheSDK para dispositivos con AWS IoT, que contiene la biblioteca de comunicación entre procesos (IPC). Para obtener más información, consulte Úselo SDK para dispositivos con AWS IoT para comunicarse con el núcleo de Greengrass, otros componentes y AWS IoT Core.
- El SDK de Stream Manager, que puede utilizar para transferir flujos de datos al. Nube de AWS Para obtener más información, consulte Gestione los flujos de datos en los dispositivos principales de Greengrass.

Temas

- AWS IoT GreengrassInterfaz de línea de comandos del kit de desarrollo
- Interfaz de línea de comandos Greengrass
- Utilice el marco AWS IoT Greengrass de pruebas

AWS IoT GreengrassInterfaz de línea de comandos del kit de desarrollo

La interfaz de línea de comandos (CLI de GDK) del kit de AWS IoT Greengrass desarrollo proporciona funciones que le ayudan a desarrollar componentes personalizados de Greengrass. Puede usar la CLI de GDK para crear, compilar y publicar componentes personalizados. Al crear un repositorio de componentes con la CLI de GDK, puede partir de una plantilla o un componente comunitario del catálogo de software de Greengrass. A continuación, puede elegir un sistema de compilación que empaquete los archivos como archivos ZIP, utilice un script de compilación de Maven o Gradle o ejecute un comando de compilación personalizado. Después de crear un componente, puede usar la CLI de GDK para publicarlo en el AWS IoT Greengrass servicio, de modo que puede usar la AWS IoT Greengrass consola o la API para implementar el componente en sus dispositivos principales de Greengrass.

Al desarrollar componentes de Greengrass sin la CLI de GDK, debe actualizar las URIs de versión y artefacto del archivo de recetas del componente cada vez que cree una nueva versión del componente. Al utilizar la CLI de GDK, esta puede actualizar automáticamente las URIs de versión y artefacto cada vez que publique una nueva versión del componente.

La CLI de GDK es de código abierto y está disponible en GitHub. Puede personalizar y ampliar la CLI de GDK para satisfacer sus necesidades de desarrollo de componentes. Lo invitamos a abrir las ediciones y a introducir solicitudes en el GitHub repositorio. Puede encontrar la fuente de la CLI de GDK en el siguiente enlace: https://github.com/aws-greengrass/ aws-greengrass-gdk-cli.

Requisitos previos

Para instalar y usar la CLI del kit de desarrollo de Greengrass, necesita lo siguiente:

- Una Cuenta de AWS. Si no dispone de una, consulte Configure un Cuenta de AWS.
- Un ordenador de desarrollo similar a Windows, macOS o Unix con conexión a Internet.
- Para la versión 1.1.0 o posterior de GDK CLI, Python 3.6 o posterior está instalado en el equipo de desarrollo.

  Para la versión 1.0.0 de GDK CLI, Python 3.8 o posterior instalada en su equipo de desarrollo.
- Git instalado en tu ordenador de desarrollo.
- AWS Command Line Interface(AWS CLI) instalado y configurado con credenciales en tu ordenador de desarrollo. Para obtener más información, consulte Instalación, actualización y desinstalación AWS CLI y configuración del AWS CLI en la Guía del AWS Command Line Interface usuario.

> **Note**
>
> Si utiliza una Raspberry Pi u otro dispositivo ARM de 32 bits, instale AWS CLI la V1. AWS CLI La versión 2 no está disponible para dispositivos ARM de 32 bits. Para obtener más información, consulte Instalación, actualización y desinstalación de la AWS CLI versión 1.

- Para usar la CLI de GDK para publicar componentes en el AWS IoT Greengrass servicio, debe tener los siguientes permisos:
  - `s3:CreateBucket`
  - `s3:GetBucketLocation`
  - `s3:PutObject`
  - `greengrass:CreateComponentVersion`
  - `greengrass:ListComponentVersions`
- Para usar la CLI de GDK para crear un componente cuyos artefactos existan en un bucket de S3 y no en el sistema de archivos local, debe tener los siguientes permisos:
  - `s3:ListBucket`

  Esta función está disponible para GDK CLI v1.1.0 y versiones posteriores.

Registros de cambios

En la siguiente tabla se describen los cambios en cada versión de la CLI de GDK. Para obtener más información, consulte la página de versiones de la CLI de GDK en GitHub.

Versión	Cambios
1.6.2	Mejoras y correcciones de errores • Soluciona un problema por el que el archivo gradlew.bat de Windows no funcionaba debido a la ruta relativa. • Mejoras menores en el registro, las pruebas y el empaquetado.
1.6.1	Mejoras y correcciones de errores • Añade una corrección de seguridad para el análisis de argumentos de la CLI.

Versión	Cambios
	• Permite al GDK obtener el nombre de la versión más reciente de Greengrass Testing Framework (GTF) como versión GTF predeterm inada. • Permite a GDK recomendar a los clientes que utilicen una versión anterior del GTF que la actualicen a la última versión.
1.6.0	Nuevas características • Añade una comprobación de validación de recetas con respecto al esquema de recetas de Greengrass durante los comandos `component build` y`component publish`. Esta actualización ayuda a los desarrolladores a identificar problemas procesables en sus recetas de componentes en una fase temprana del proceso de creación de componentes. • Añade un conjunto de pruebas de confianza a la plantilla que se puede desplegar con un `test-e2e init` comando. Este conjunto de pruebas de confianza incluye ocho pruebas genéricas que se pueden utilizar y ampliar para adaptarse a las necesidades básicas de las pruebas de componentes. Mejoras y correcciones de errores • Actualiza la versión predeterminada de Greengrass Testing Framework (GTF) utilizada por el `test-e2e` comando a la versión 1.2.0.
1.5.0	Mejoras y correcciones de errores Actualiza los patrones reconocidos por la opción de `excludes` construcc ión cuando `build_system` está`zip`. Esta versión ahora reconocerá los patrones globales que coincidan con los nombres de las rutas en función de sus caracteres comodín. Esto permite especificar de forma personali zada los directorios de los que se debe excluir.

Versión	Cambios
1.4.0	Nuevas características • Añade un nuevo `config` comando que inicia un mensaje interactivo para modificar los campos de un archivo de configuración de GDK existente. • Modifica los `gdk component publish` comandos `gdk component build` y para comprobar que el tamaño de la receta cumple con los requisitos de Greengrass (<=16000 bytes) antes de continuar. Mejoras y correcciones de errores • Añade un registro adicional en el resultado del `gdk component build` comando cuando un error de sintaxis de la receta impide que se complete la compilación para detectarlo. • Cambia el nombre de `otf-options` y `otf-version` a `gtf-options` y, `gtf-version` respectivamente, debido al cambio de nombre de Open Test Framework a Greengrass Testing Framework.
1.3.0	Nuevas características • Añade un nuevo `test-e2e` comando para permitir las end-to-end pruebas de componentes mediante Open Test Framework. • Añade una nueva opción de configuración,`zip_name`, para admitir nombres de archivos zip configurables con el sistema de compilación zip. • Hace que la `region` propiedad del archivo de configuración del GDK sea opcional. Mejoras y correcciones de errores • Soluciona un problema por el que se crea un nuevo directorio incluso cuando la plantilla o el repositorio especificados no existen al inicializar un proyecto de GDK con el `--name` argumento.

Versión	Cambios
1.2.3	Mejoras y correcciones de errores • Soluciona un problema por el que la creación del bucket fallaba debido a una gestión incorrecta de los errores. • Soluciona un problema por el que se eliminaban las estructuras de listas de la receta del componente.
1.2.2	Correcciones de errores y mejoras • Las claves de recetas ya no distinguen entre mayúsculas y minúsculas. • Añade una comprobación para determinar si un depósito existe en un depósito Región de AWS y si el usuario puede acceder a él antes de crear uno nuevo. Requiere que el usuario tenga el `GetBucketLocation` permiso. • Soluciona un problema con la `excludes` palabra clave del archivo de configuración CLI de GDK.
1.2.1	Mejoras y correcciones de errores • Acepta la entrada de configuración de Canadá (Central`ca-central-1`) () Región de AWS en la región del `gdk-config.json` archivo. • Soluciona problemas con el argumento CLI de `--region` GDK del `publish` comando.

Versión	Cambios
1.2.0	Nuevas características • Añade la `options` entrada a la `build` configuración en el archivo de configuración CLI de GDK. Permite `excludes` `options` excluir ciertos archivos del artefacto zip cuando se utiliza el sistema de `zip` compilación. • Agrega el sistema de `gradlew` compilación para usar Gradle Wrapper para compilar componentes. • Añade compatibilidad con los archivos de compilación DSL de Kotlin para la opción de compilación. `gradle` • Añade una `options` entrada a la `publish` configuración en el archivo de configuración CLI de GDK. Admite la `file_upload_args` opción under `options` para proporcionar argumentos adicionales al cargar archivos a Amazon S3. Mejoras y correcciones de errores • Corrige un problema por el que las compilaciones de Gradle no se limpiaban antes de ejecutar un comando de compilación. • Soluciona un problema por el que la compilación no se cerraba cuando se producía un error en el comando de compilación. • Mejora el formato de salida del `gdk component list` comando.

Versión	Cambios
1.1.0	Nuevas características • Añade compatibilidad con el sistema de compilación de Gradle. • Añade compatibilidad con el sistema de compilación Maven en dispositivos Windows. • Añade el `--bucket` argumento al comando de publicación del componente. Puede usar este argumento para especificar el depósito exacto en el que la CLI de GDK carga los artefactos de los componentes. • Añade el `--name` argumento al comando component init. Puede usar esta opción para especificar la carpeta en la que la CLI de GDK inicializa el componente. • Añade compatibilidad con los artefactos de los componentes que existen en un bucket de S3 pero no en la carpeta de creación del componente local. Puede utilizar esta función para reducir los costes de ancho de banda de los artefactos de componentes de gran tamaño, como los modelos de aprendizaje automático. Mejoras y correcciones de errores • Actualiza el comando de publicación del componente para comprobar si el componente está creado antes de publicarlo. Si el componente no está creado, este comando ahora lo crea automáticamente. • Soluciona un problema por el que el sistema de compilación zip no se puede compilar en dispositivos Windows cuando el nombre del archivo ZIP contiene letras mayúsculas. • Mejora el formato de los mensajes de registro y cambia el nivel de registro predeterminado a `INFO` en los dispositivos que ejecutan versiones de Python anteriores a la 3.8. • Cambia el requisito mínimo de la versión de Python a Python 3.6.
1.0.0	Versión inicial.

Instalar o actualizar la interfaz de línea AWS IoT Greengrass de comandos del kit de desarrollo

La interfaz de línea de comandos del kit de AWS IoT Greengrass desarrollo (GDK CLI) se basa en Python, por lo que puede `pip` utilizarla para instalarla en su computadora de desarrollo.

> **Tip**
>
> También puede instalar la CLI de GDK en un entorno virtual de Python, como venv. Para obtener más información, consulte Entornos y paquetes virtuales en la documentación de Python 3.

Para instalar o actualizar la CLI de GDK

1. Ejecute el siguiente comando para instalar la última versión de la CLI de GDK desde su GitHubrepositorio.

   ```
   python3 -m pip install -U git+https://github.com/aws-greengrass/aws-greengrass-gdk-cli.git@v1.6.2
   ```

   > **Note**
   >
   > Para instalar una versión específica de la CLI de GDK, sustituya *`versionTag por la etiqueta`* de versión que desee instalar. Puede ver las etiquetas de versión de la CLI de GDK en su GitHubrepositorio.
   >
   > ```
   > python3 -m pip install -U git+https://github.com/aws-greengrass/aws-greengrass-gdk-cli.git@versionTag
   > ```

2. Ejecute el siguiente comando para comprobar que la CLI de GDK se instaló correctamente.

   ```
   gdk --help
   ```

   Si no encuentra el `gdk` comando, añada su carpeta a PATH.

   - En dispositivos Linux, `/home/MyUser/.local/bin` agréguelo a PATH y *`MyUser`*sustitúyalo por el nombre de su usuario.

- En dispositivos Windows, añada *PythonPath*`\\Scripts` a PATH y *PythonPath*sustitúyala por la ruta a la carpeta Python de su dispositivo.

Ahora puede usar la CLI de GDK para crear, compilar y publicar componentes de Greengrass. Para obtener más información sobre cómo utilizar la CLI de GDK, consulteAWS IoT GreengrassComandos de la interfaz de línea de comandos del kit de desarrollo.

AWS IoT GreengrassComandos de la interfaz de línea de comandos del kit de desarrollo

La interfaz de línea de comandos del kit de AWS IoT Greengrass desarrollo (GDK CLI) proporciona una interfaz de línea de comandos que puede usar para crear, compilar y publicar componentes de Greengrass en su equipo de desarrollo. Los comandos CLI de GDK utilizan el siguiente formato.

```
gdk <command> <subcommand> [arguments]
```

Al instalar la CLI de GDK, el instalador añade información `gdk` a la PATH para que pueda ejecutar la CLI de GDK desde la línea de comandos.

Puede usar los siguientes argumentos con cualquier comando:

- Utilice `-h` o `--help` para obtener información sobre un comando CLI de GDK.
- Utilice `-v` o `--version` para ver qué versión de la CLI de GDK está instalada.
- Use `-d` o `--debug` para generar registros detallados que pueda usar para depurar la CLI de GDK.

En esta sección se describen los comandos CLI de GDK y se proporcionan ejemplos para cada comando. La sinopsis de cada comando muestra sus argumentos y su uso. Los argumentos opcionales se muestran entre corchetes.

Comandos disponibles

- componente
- config
- prueba-e2e

componente

Utilice el `component` comando de la interfaz de línea de comandos (CLI de GDK) del kit de AWS IoT Greengrass desarrollo para crear, compilar y publicar componentes personalizados de Greengrass.

Subcomandos

- init
- build
- publish
- list

init

Inicialice una carpeta de componentes de Greengrass desde una plantilla de componentes o un componente de la comunidad.

La CLI de GDK recupera los componentes de la comunidad del catálogo de software de Greengrass y las plantillas de componentes del repositorio de plantillas de AWS IoT Greengrasscomponentes en adelante. GitHub

> **Note**
>
> Si usa GDK CLI v1.0.0, debe ejecutar este comando en una carpeta vacía. La CLI de GDK descarga la plantilla o el componente de la comunidad en la carpeta actual.
> Si usa la versión 1.1.0 de la CLI de GDK o una versión posterior, puede especificar el `--name` argumento para especificar la carpeta en la que la CLI de GDK descarga la plantilla o el componente de la comunidad. Si usa este argumento, especifique una carpeta que no exista. La CLI de GDK crea la carpeta por usted. Si no especifica este argumento, la CLI de GDK utilizará la carpeta actual, que debe estar vacía.
> Si el componente usa el sistema de compilación zip, la CLI de GDK comprime ciertos archivos de la carpeta del componente en un archivo zip con el mismo nombre que la carpeta del componente. Por ejemplo, si el nombre de la carpeta del componente es`HelloWorld`, la CLI de GDK crea un archivo zip denominado`HelloWorld.zip`. En la receta del componente, el nombre del artefacto zip debe coincidir con el nombre de la carpeta del componente. Si utiliza la versión 1.0.0 de la CLI de GDK en un dispositivo Windows, los nombres de las carpetas y los archivos zip de los componentes deben contener solo letras minúsculas.

Si inicializa una plantilla o un componente de comunidad que utiliza el sistema de compilación zip en una carpeta con un nombre diferente al de la plantilla o el componente, debe cambiar el nombre del artefacto zip en la receta del componente. Actualice las `Lifecycle` definiciones `Artifacts` y de forma que el nombre del archivo zip coincida con el nombre de la carpeta del componente. En el siguiente ejemplo, se resalta el nombre del fichero zip en las `Lifecycle` definiciones `Artifacts` y.

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]
```

Argumentos (se inicializan desde la plantilla del componente)

- `-l`, `--language` — El lenguaje de programación que se utilizará para la plantilla que especifique.

  Debe especificar una de las dos `--repository` opciones `--language` y`--template`.

- `-t`, `--template` — La plantilla de componentes que se utilizará en un proyecto de componentes locales. Para ver las plantillas disponibles, utilice el comando list.

  Debe especificar una de las dos `--repository` opciones `--language` y`--template`.

- `-n`, `--name` — (opcional) El nombre de la carpeta local en la que la CLI de GDK inicializa el componente. Especifique una carpeta que no exista. La CLI de GDK crea la carpeta por usted.

  Esta función está disponible para GDK CLI v1.1.0 y versiones posteriores.

Argumentos (se inicializan desde el componente de la comunidad)

- `-r`, `--repository` — El componente de la comunidad que se va a incluir en la carpeta local. Para ver los componentes de la comunidad disponibles, utilice el comando list.

  Debe especificar una de las dos `--repository` opciones `--language` y`--template`.

- `-n`, `--name` — (opcional) El nombre de la carpeta local en la que la CLI de GDK inicializa el componente. Especifique una carpeta que no exista. La CLI de GDK crea la carpeta por usted.

  Esta función está disponible para GDK CLI v1.1.0 y versiones posteriores.

Salida

El siguiente ejemplo muestra el resultado que se produce al ejecutar este comando para inicializar una carpeta de componentes desde la plantilla Hello World de Python.

```
$ 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.
```

El siguiente ejemplo muestra el resultado obtenido al ejecutar este comando para inicializar una carpeta de componentes desde un componente de la comunidad.

```
$ 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

Convierta el código fuente de un componente en una receta y artefactos que pueda publicar en el AWS IoT Greengrass servicio. La CLI de GDK ejecuta el sistema de compilación que especifique en el archivo de configuración de la CLI de GDK,. `gdk-config.json` Debe ejecutar este comando en la misma carpeta en la que se encuentra el `gdk-config.json` archivo.

Al ejecutar este comando, la CLI de GDK crea una receta y artefactos en la `greengrass-build` carpeta de la carpeta del componente. La CLI de GDK guarda la receta en la `greengrass-build/recipes` carpeta y guarda los artefactos en la `greengrass-build/artifacts/`*`componentName`*`/`*`componentVersion`* carpeta.

Si usa GDK CLI v1.1.0 o una versión posterior, la receta del componente puede especificar los artefactos que existen en un bucket de S3 pero no en la carpeta de compilación del componente local. Puede utilizar esta función para reducir el uso del ancho de banda cuando desarrolle componentes con artefactos de gran tamaño, como los modelos de aprendizaje automático.

Después de crear un componente, puede realizar una de las siguientes acciones para probarlo en un dispositivo principal de Greengrass:

- Si desarrolla en un dispositivo diferente al que ejecuta el software AWS IoT Greengrass principal, debe publicar el componente para desplegarlo en un dispositivo principal de Greengrass. Publique el componente en el AWS IoT Greengrass servicio e impleméntelo en el dispositivo principal de Greengrass. Para obtener más información, consulte el comando publish yCrear implementaciones.
- Si desarrolla el software en el mismo dispositivo en el que ejecuta el software AWS IoT Greengrass principal, puede publicar el componente en el AWS IoT Greengrass servicio que desea implementar o puede crear una implementación local para instalar y ejecutar el componente. Para crear una implementación local, utilice la CLI de Greengrass. Para obtener más información, consulte Interfaz de línea de comandos Greengrass y Pruebe AWS IoT Greengrass los componentes con despliegues locales. Al crear la implementación local, especifíquela `greengrass-build/recipes` como carpeta de recetas y `greengrass-build/artifacts` como carpeta de artefactos.

Sinopsis

```
$ gdk component build
```

Arguments

Ninguna

Salida

El siguiente ejemplo muestra el resultado que se produce al ejecutar este comando.

```
$ gdk component build
[2021-11-29 13:18:49] INFO - Getting project configuration from gdk-config.json
[2021-11-29 13:18:49] INFO - Found component recipe file 'recipe.yaml' in the
 project directory.
[2021-11-29 13:18:49] INFO - Building the component 'com.example.PythonHelloWorld'
 with the given project configuration.
[2021-11-29 13:18:49] INFO - Using 'zip' build system to build the component.
[2021-11-29 13:18:49] WARNING - This component is identified as using 'zip' build
 system. If this is incorrect, please exit and specify custom build command in the
 'gdk-config.json'.
[2021-11-29 13:18:49] INFO - Zipping source code files of the component.
[2021-11-29 13:18:49] INFO - Copying over the build artifacts to the greengrass
 component artifacts build folder.
[2021-11-29 13:18:49] INFO - Updating artifact URIs in the recipe.
```

```
[2021-11-29 13:18:49] INFO - Creating component recipe in 'C:\Users\MyUser\Documents
\greengrass-components\python\HelloWorld\greengrass-build\recipes'.
```

publish

Publique este componente en el AWS IoT Greengrass servicio. Este comando carga los artefactos de construcción en un bucket de S3, actualiza el URI del artefacto en la receta y crea una nueva versión del componente a partir de la receta. La CLI de GDK utiliza el bucket y la AWS región de S3 que se especifican en el archivo de configuración de la CLI de GDK,. `gdk-config.json` Debe ejecutar este comando en la misma carpeta en la que se encuentra el `gdk-config.json` archivo.

Si utiliza la versión 1.1.0 de la CLI de GDK o una versión posterior, puede especificar el `--bucket` argumento para especificar el depósito de S3 en el que la CLI de GDK carga los artefactos del componente. Si no especificas este argumento, la CLI de GDK se carga en el bucket de S3 cuyo nombre es*`bucket`*`-`*`region`*`-`*`accountId`*, donde *`bucket`* y *`region`* son los valores que especificas y *`AccountID`* es tu ID. `gdk-config.json` Cuenta de AWS La CLI de GDK crea el bucket si no existe.

Si utiliza la versión 1.2.0 de la CLI de GDK o una versión posterior, puede anular lo Región de AWS especificado en el archivo de configuración de la CLI de GDK mediante el parámetro. `--region` También puede especificar opciones adicionales mediante el parámetro. `--options` Para obtener una lista de las opciones disponibles, consulteArchivo de configuración CLI del kit de desarrollo Greengrass.

Al ejecutar este comando, la CLI de GDK publica el componente con la versión que especifique en la receta. Si lo especifica`NEXT_PATCH`, la CLI de GDK utilizará la siguiente versión de parche que aún no exista. Las versiones semánticas utilizan una versión principal. menor. sistema de numeración de parches. Para obtener más información, consulte la especificación de la versión semántica.

> **Note**
>
> Si utiliza la versión 1.1.0 de la CLI de GDK o una versión posterior, al ejecutar este comando, la CLI de GDK comprueba si el componente está creado. Si el componente no está compilado, la CLI de GDK lo compila antes de publicarlo.

Sinopsis

```
$ gdk component publish
```

```
    [--bucket] [--region] [--options]
```

Arguments

- `-b`, `--bucket` — (Opcional) Especifique el nombre del depósito de S3 en el que la CLI de GDK publica los artefactos de los componentes.

  Si no especificas este argumento, la CLI de GDK se carga en el bucket de S3 cuyo nombre es*`bucket`*`-`*`region`*`-`*`accountId`*, donde *`bucket`* y *`region`* son los valores que especificas y *`AccountID`* es tu ID. `gdk-config.json` Cuenta de AWS La CLI de GDK crea el bucket si no existe.

  La CLI de GDK crea el bucket si no existe.

  Esta función está disponible para GDK CLI v1.1.0 y versiones posteriores.
- `-r`, `--region` — (Opcional) Especifique el nombre del Región de AWS to al crear el componente. Este argumento anula el nombre de la región en la configuración de la CLI de GDK.

  Esta función está disponible para GDK CLI v1.2.0 y versiones posteriores.
- `-o`, `--options` (opcional) Especifique una lista de opciones para publicar un componente. El argumento debe ser una cadena JSON válida o una ruta de archivo a un archivo JSON que contenga las opciones de publicación. Este argumento anula las opciones de la configuración de la CLI de GDK.

  Esta función está disponible para GDK CLI v1.2.0 y versiones posteriores.

Salida

El siguiente ejemplo muestra el resultado que se produce al ejecutar este comando.

```
$ gdk component publish
[2021-11-29 13:45:29] INFO - Getting project configuration from gdk-config.json
[2021-11-29 13:45:29] INFO - Found component recipe file 'recipe.yaml' in the
 project directory.
[2021-11-29 13:45:29] INFO - Found credentials in shared credentials file: ~/.aws/
credentials
[2021-11-29 13:45:30] INFO - Publishing the component 'com.example.PythonHelloWorld'
 with the given project configuration.
[2021-11-29 13:45:30] INFO - No private version of the component
 'com.example.PythonHelloWorld' exist in the account. Using '1.0.0' as the next
 version to create.
```

```
[2021-11-29 13:45:30] INFO - Uploading the component built artifacts to s3 bucket.
[2021-11-29 13:45:30] INFO - Uploading component artifacts to S3 bucket: {bucket}.
 If this is your first time using this bucket, add the 's3:GetObject' permission
 to each core device's token exchange role to allow it to download the component
 artifacts. For more information, see https://docs.aws.amazon.com/greengrass/v2/
developerguide/device-service-role.html.
[2021-11-29 13:45:30] INFO - Not creating an artifacts bucket as it already exists.
[2021-11-29 13:45:30] INFO - Updating the component recipe
 com.example.PythonHelloWorld-1.0.0.
[2021-11-29 13:45:30] INFO - Creating a new greengrass component
 com.example.PythonHelloWorld-1.0.0
[2021-11-29 13:45:30] INFO - Created private version '1.0.0' of the component in the
 account.'com.example.PythonHelloWorld'.
```

list

Recupere la lista de plantillas de componentes y componentes de comunidad disponibles.

La CLI de GDK recupera los componentes de la comunidad del catálogo de software de Greengrass y las plantillas de componentes del repositorio de plantillas de AWS IoT Greengrasscomponentes en adelante. GitHub

Puede pasar el resultado de este comando al comando init para inicializar los repositorios de componentes a partir de plantillas y componentes de la comunidad.

Sinopsis

```
$ gdk component list
    [--template]
    [--repository]
```

Arguments

- `-t`, `--template` — (Opcional) Especifique este argumento para enumerar las plantillas de componentes disponibles. Este comando muestra el nombre y el idioma de cada plantilla en el formato*`name`*`-`*`language`*. Por ejemplo, en`HelloWorld-python`, el nombre de la plantilla es `HelloWorld` y el idioma es`python`.
- `-r`, `--repository` — (Opcional) Especifique este argumento para enumerar los repositorios de componentes de la comunidad disponibles.

Salida

El siguiente ejemplo muestra el resultado que se produce al ejecutar este comando.

```
$ gdk component list --template
[2021-11-29 12:29:04] INFO - Listing all the available component templates from
 Greengrass Software Catalog.
[2021-11-29 12:29:04] INFO - Found '2' component templates to display.
1. HelloWorld-python
2. HelloWorld-java
```

config

Utilice el `config` comando de la interfaz de línea de comandos (CLI de GDK) del kit de AWS IoT Greengrass desarrollo para modificar la configuración del GDK en el archivo de configuración,. `gdk-config.json`

Subcomandos

- update

update

Inicie un mensaje interactivo para modificar los campos de un archivo de configuración de GDK existente.

Sinopsis

```
$ gdk config update
    [--component]
```

Arguments (Argumentos)

- `-c`, `--component` — Para actualizar los campos relacionados con los componentes del archivo. `gdk-config.json` Este argumento es obligatorio porque es la única opción.

Salida

El siguiente ejemplo muestra el resultado obtenido al ejecutar este comando para configurar un componente.

```
$ gdk config update --component
```

```
Current value of the REQUIRED component_name is (default:
 com.example.PythonHelloWorld):
Current value of the REQUIRED author is (default: author):
Current value of the REQUIRED version is (default: NEXT_PATCH):
Do you want to change the build configurations? (y/n)
Do you want to change the publish configurations? (y/n)
[2023-09-26 10:19:48] INFO - Config file has been updated. Exiting...
```

prueba-e2e

Utilice el `test-e2e` comando de la interfaz de línea de comandos (CLI de GDK) del kit de AWS IoT Greengrass desarrollo para inicializar, compilar y end-to-end ejecutar módulos de prueba en el proyecto de GDK.

Subcomandos

- init
- build
- run

init

Inicialice un proyecto CLI de GDK existente con un módulo de pruebas que utilice Greengrass Testing Framework (GTF).

De forma predeterminada, la CLI de GDK recupera la plantilla del módulo maven del repositorio de plantillas de AWS IoT Greengrass componentes en. GitHub Este módulo Maven viene con una dependencia del archivo JAR. `aws-greengrass-testing-standalone`

Este comando crea un nuevo directorio llamado `gg-e2e-tests` dentro del proyecto GDK. Si el directorio del módulo de pruebas ya existe y no está vacío, el comando se cierra sin hacer nada. Esta `gg-e2e-tests` carpeta contiene las definiciones de funciones y pasos de Cucumber estructuradas en un proyecto de Maven.

Por defecto, este comando intentará usar la última versión de GTF.

Sinopsis

```
$ gdk test-e2e init
```

```
    [--gtf-version]
```

Arguments

- `-ov`, `--gtf-version` — (Opcional) La versión del GTF que se utilizará con el módulo de end-to-end pruebas del proyecto GDK. Este valor debe ser una de las versiones del GTF de las versiones. Este argumento anula el de la `gtf_version` configuración CLI de GDK.

Salida

El siguiente ejemplo muestra el resultado obtenido al ejecutar este comando para inicializar el proyecto GDK con el módulo de pruebas.

```
$ 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**
>
> Debe compilar el componente ejecutándolo gdk component build antes de compilar el módulo de end-to-end prueba.

Cree el módulo end-to-end de pruebas. La CLI de GDK crea el módulo de pruebas mediante el sistema de compilación que se especifica en el archivo de configuración de la CLI de GDKgdk-config.json, en la `test-e2e` propiedad. Debe ejecutar este comando en la misma carpeta en la que se encuentra el `gdk-config.json` archivo.

De forma predeterminada, la CLI de GDK usa el sistema de compilación maven para crear el módulo de pruebas. Se requiere Maven para ejecutar el comando. `gdk test-e2e build`

Debe compilar el componente ejecutándolo gdk-component-build antes de compilar el módulo de prueba, si los archivos de funciones de prueba tienen variables como `GDK_COMPONENT_NAME` y `GDK_COMPONENT_RECIPE_FILE` para interpolar.

Al ejecutar este comando, la CLI de GDK interpola todas las variables de la configuración del proyecto de GDK y crea el `gg-e2e-tests` módulo para generar el archivo JAR de prueba final.

Sinopsis

```
$ gdk test-e2e build
```

Arguments

Ninguna

Salida

El siguiente ejemplo muestra el resultado que se produce al ejecutar este comando.

```
$ gdk test-e2e build
[2023-07-20 15:36:48] INFO - Updating feature file: file:///path/to//
HelloWorld/greengrass-build/gg-e2e-tests/src/main/resources/greengrass/features/
component.feature
[2023-07-20 15:36:48] INFO - Creating the E2E testing recipe file:///path/to/
HelloWorld/greengrass-build/recipes/e2e_test_recipe.yaml
[2023-07-20 15:36:48] INFO - Building the E2E testing module
[2023-07-20 15:36:48] INFO - Running the build command 'mvn package'
.........
```

run

Ejecute el módulo de pruebas con las opciones de prueba del archivo de configuración del GDK.

> **Note**
>
> Debe compilar el módulo de pruebas ejecutándolo gdk test-e2e build antes de ejecutar las end-to-end pruebas.

Sinopsis

```
$ gdk test-e2e run
    [--gtf-options]
```

Arguments

- `-oo`, `--gtf-options` — (Opcional) Especifique una lista de opciones para ejecutar las end-to-end pruebas. El argumento debe ser una cadena JSON válida o una ruta de archivo a un

archivo JSON que contenga las opciones de GTF. Las opciones proporcionadas en el archivo de configuración se combinan con las que se proporcionan en los argumentos del comando. Si una opción está presente en ambos lugares, la que está en el argumento tiene prioridad sobre la del archivo de configuración.

Si la `tags` opción no se especifica en este comando, GDK la utiliza para `Sample` las etiquetas. Si no `ggc-archive` se especifica, GDK descarga la última versión del archivo del núcleo de Greengrass.

Salida

El siguiente ejemplo muestra el resultado que se produce al ejecutar este comando.

```
$ gdk test-e2e run
[2023-07-20 16:35:53] INFO - Downloading latest nucleus archive from url https://
d2s8p88vqu9w66.cloudfront.net/releases/greengrass-latest.zip
[2023-07-20 16:35:57] INFO - Running test jar with command java -jar /path/to/
greengrass-build/gg-e2e-tests/target/uat-features-1.0.0.jar –ggc-archive=/path/to/
aws-greengrass-gdk-cli/HelloWorld/greengrass-build/greengrass-nucleus-latest.zip –
tags=Sample

16:35:59.693 [] [] [] [INFO]
 com.aws.greengrass.testing.modules.GreengrassContextModule - Extracting /path/
to/workplace/aws-greengrass-gdk-cli/HelloWorld/greengrass-build/greengrass-
nucleus-latest.zip into /var/folders/7g/ltzcb_3s77nbtmkzfb6brwv40000gr/T/gg-
testing-7718418114158172636/greengrass
16:36:00.534 [gtf-1.1.0-SNAPSHOT] [] [] [INFO]
 com.aws.greengrass.testing.features.LoggerSteps - GTF Version is gtf-1.1.0-SNAPSHOT
.......
```

Archivo de configuración CLI del kit de desarrollo Greengrass

La interfaz de línea de comandos del kit de AWS IoT Greengrass desarrollo (CLI de GDK) lee un archivo de configuración `gdk-config.json` denominado para crear y publicar componentes. Este archivo de configuración debe estar en la raíz del repositorio de componentes. Puede utilizar el comando CLI init de GDK para inicializar los repositorios de componentes con este archivo de configuración.

Temas

- Formato de archivo de configuración GDK CLI

- Ejemplos de archivos de configuración CLI de GDK

Formato de archivo de configuración GDK CLI

Al definir un archivo de configuración CLI de GDK para un componente, se especifica la siguiente información en formato JSON.

`gdk_version`

La versión mínima de la CLI de GDK que es compatible con este componente. Este valor debe ser una de las versiones CLI de GDK de las versiones.

`component`

La configuración de este componente.

`componentName`

`author`

El autor o editor del componente.

`version`

Esta es la versión del componente. Especifique uno de los siguientes valores:

- `NEXT_PATCH`— Al elegir esta opción, la CLI de GDK establece la versión al publicar el componente. La CLI de GDK consulta el AWS IoT Greengrass servicio para identificar la última versión publicada del componente. A continuación, establece la versión en la siguiente versión del parche posterior a esa versión. Si no ha publicado el componente antes, la CLI de GDK usa la versión`1.0.0`.

  Si elige esta opción, no podrá usar la CLI de Greengrass para implementar y probar localmente el componente en su computadora de desarrollo local que ejecuta el software AWS IoT Greengrass Core. Para habilitar las implementaciones locales, debe especificar una versión semántica en su lugar.
- Una versión semántica, como. **`1.0.0`** Las versiones semánticas utilizan una principal. menor. sistema de numeración de parches. Para obtener más información, consulte la especificación de la versión semántica.

  Si desarrolla componentes en un dispositivo principal de Greengrass en el que desee implementar y probar el componente, elija esta opción. Debe compilar el componente con una versión específica para crear despliegues locales con la CLI de Greengrass.

`build`

La configuración que se utilizará para convertir la fuente de este componente en artefactos. Este objeto contiene la siguiente información:

`build_system`

El sistema de compilación que se va a utilizar. Puede elegir entre las siguientes opciones:

- `zip`— Empaqueta la carpeta del componente en un archivo ZIP para definirla como el único artefacto del componente. Seleccione esta opción para los siguientes tipos de componentes:
  - Componentes que utilizan lenguajes de programación interpretados, como Python o JavaScript.
  - Componentes que empaquetan archivos distintos del código, como modelos de aprendizaje automático u otros recursos.

  La CLI de GDK comprime la carpeta del componente en un archivo zip con el mismo nombre que la carpeta del componente. Por ejemplo, si el nombre de la carpeta del componente es`HelloWorld`, la CLI de GDK crea un archivo zip denominado`HelloWorld.zip`.

  > **Note**
  >
  > Si utiliza la versión 1.0.0 de la CLI de GDK en un dispositivo Windows, los nombres de las carpetas y los archivos zip de los componentes deben contener solo letras minúsculas.

  Cuando la CLI de GDK comprime la carpeta del componente en un archivo zip, omite los siguientes archivos:

  - El archivo `gdk-config.json`
  - El archivo de recetas (o) `recipe.json` `recipe.yaml`
  - Cree carpetas, como `greengrass-build`
- `maven`— Ejecuta el `mvn clean package` comando para convertir la fuente del componente en artefactos. Elija esta opción para los componentes que utilizan Maven, como los componentes de Java.

En los dispositivos Windows, esta función está disponible para GDK CLI v1.1.0 y versiones posteriores.

- `gradle`— Ejecuta el `gradle build` comando para convertir la fuente del componente en artefactos. Elige esta opción para los componentes que usan Gradle. Esta función está disponible para GDK CLI v1.1.0 y versiones posteriores.

  El sistema de `gradle` compilación admite Kotlin DSL como archivo de compilación. Esta función está disponible para GDK CLI v1.2.0 y versiones posteriores.
- `gradlew`— Ejecuta el `gradlew` comando para convertir la fuente del componente en artefactos. Elija esta opción para los componentes que utilizan el Gradle Wrapper.

  Esta función está disponible para GDK CLI v1.2.0 y versiones posteriores.
- `custom`— Ejecuta un comando personalizado para convertir la fuente del componente en una receta y artefactos. Especifique el comando personalizado en el `custom_build_command` parámetro.

`custom_build_command`

(Opcional) El comando de compilación personalizado que se ejecutará en un sistema de compilación personalizado. Debe especificar este parámetro si lo especifica `custom` para`build_system`.

> **Important**
>
> Este comando debe crear una receta y artefactos en las siguientes carpetas de la carpeta del componente. La CLI de GDK crea estas carpetas automáticamente al ejecutar el comando component build.
>
> - Carpeta de recetas: `greengrass-build/recipes`
> - Carpeta de artefactos: `greengrass-build/artifacts/componentName/componentVersion`
>
>   Sustituya *componentName* por el nombre del componente y sustituya *componentVersion por la versión* del componente o. `NEXT_PATCH`

Puede especificar una sola cadena o una lista de cadenas, donde cada cadena es una palabra del comando. Por ejemplo, para ejecutar un comando de compilación

personalizado para un componente de C++, puede especificar **`cmake --build build --config Release`** o **`["cmake", "--build", "build", "--config", "Release"]`**.

Para ver un ejemplo de un sistema de compilación personalizado, consulta aws.greengrass.labs.LocalWebServer community componentla siguiente GitHub.

`options`

(Opcional) Se utilizan opciones de configuración adicionales durante el proceso de creación del componente.

Esta función está disponible para GDK CLI v1.2.0 y versiones posteriores.

`excludes`

Una lista de patrones globales que definen qué archivos se deben excluir del directorio de componentes al crear el archivo zip. Solo es válido cuando `build_system` es`zip`.

> **Note**
>
> En las versiones 1.4.0 y anteriores de GDK CLI, cualquier archivo que coincida con una entrada de la lista de exclusiones se excluye de todos los subdirectorios del componente. Para lograr el mismo comportamiento en las versiones 1.5.0 y posteriores de la CLI de GDK, anteponga las `**/` entradas existentes en la lista de exclusiones. Por ejemplo, `*.txt` excluirá los archivos de texto solo del directorio; `**/*.txt` excluirá los archivos de texto de todos los directorios y subdirectorios.
> En las versiones 1.5.0 y posteriores de la CLI de GDK, es posible que vea una advertencia durante la compilación del componente cuando `excludes` esté definido en el archivo de configuración de GDK.
> Para deshabilitar esta advertencia, defina la variable de entorno en. `GDK_EXCLUDES_WARN_IGNORE` `true`

La CLI de GDK siempre excluye los siguientes archivos del archivo zip:

- El archivo `gdk-config.json`
- El archivo de recetas (o) `recipe.json` `recipe.yaml`

- Cree carpetas, como `greengrass-build`

De forma predeterminada, se excluyen los siguientes archivos. Sin embargo, puede controlar cuáles de estos archivos se excluyen con la `excludes` opción.

- Cualquier carpeta que comience con el prefijo «test» () `test*`
- Todos los archivos ocultos
- La carpeta `node_modules`

Si especifica la `excludes` opción, la CLI de GDK excluirá solo los archivos que haya configurado con la `excludes` opción. Si no especifica la `excludes` opción, la CLI de GDK excluye los archivos y carpetas predeterminados indicados anteriormente.

`zip_name`

El nombre del archivo zip que se utilizará al crear un artefacto zip durante el proceso de creación. Solo es válido cuando `build_system` está`zip`. Si `build_system` está vacío, el nombre del componente se utiliza como nombre del archivo zip.

`publish`

La configuración que se utilizará para publicar este componente en el AWS IoT Greengrass servicio.

Si utiliza la versión 1.1.0 de la CLI de GDK o una versión posterior, puede especificar el `--bucket` argumento para especificar el depósito de S3 en el que la CLI de GDK carga los artefactos del componente. Si no especificas este argumento, la CLI de GDK se carga en el bucket de S3 cuyo nombre es*`bucket`*`-`*`region`*`-`*`accountId`*, donde *`bucket`* y *`region`* son los valores que especificas y *`AccountID`* es tu ID. `gdk-config.json` Cuenta de AWS La CLI de GDK crea el bucket si no existe.

Este objeto contiene la siguiente información:

`bucket`

El nombre del bucket de S3 que se utilizará para alojar los artefactos de los componentes.

`region`

El Región de AWS lugar donde la CLI de GDK publica este componente.

Esta propiedad es opcional si utiliza GDK CLI v1.3.0 o una versión posterior.

`options`

(Opcional) Se utilizan opciones de configuración adicionales durante la creación de la versión del componente.

Esta función está disponible para GDK CLI v1.2.0 y versiones posteriores.

`file_upload_args`

Estructura JSON que contiene argumentos que se envían a Amazon S3 al cargar archivos a un bucket, como metadatos y mecanismos de cifrado. Para obtener una lista de los argumentos permitidos, consulte la `S3Transfer`clase en la documentación de Boto3. .

`test-e2e`

(Opcional) La configuración que se utilizará durante end-to-end las pruebas del componente. Esta función está disponible para GDK CLI v1.3.0 y versiones posteriores.

`build`

`build_system`— El sistema de compilación que se va a utilizar. La opción por defecto es`maven`. Puede elegir entre las siguientes opciones:

- `maven`— Ejecuta el `mvn package` comando para construir el módulo de pruebas. Elija esta opción para crear el módulo de pruebas que usa Maven.
- `gradle`— Ejecuta el `gradle build` comando para construir el módulo de pruebas. Elige esta opción para el módulo de pruebas que usa Gradle.

`gtf_version`

(Opcional) La versión del Greengrass Testing Framework (GTF) que se utilizará como dependencia del módulo de end-to-end pruebas al inicializar el proyecto GDK con GTF. Este valor debe ser una de las versiones del GTF de las versiones. El valor predeterminado es la versión 1.1.0 de GTF.

`gtf_options`

(Opcional) Se utilizaron opciones de configuración adicionales durante las end-to-end pruebas del componente.

La siguiente lista incluye las opciones que puede utilizar con la versión 1.1.0 de GTF.

- `additional-plugins`— Plugins de Cucumber adicionales (opcionales)
- `aws-region`— Se dirige a puntos finales regionales específicos para los AWS servicios. El valor predeterminado es lo que descubre el AWS SDK.
- `credentials-path`— Ruta AWS de credenciales de perfil opcional. El valor predeterminado son las credenciales descubiertas en el entorno anfitrión.
- `credentials-path-rotation`— Duración de rotación opcional para AWS las credenciales. El valor predeterminado es 15 minutos o`PT15M`.
- `csr-path`— La ruta de la CSR mediante la cual se generará el certificado del dispositivo.
- `device-mode`— El dispositivo objetivo que se está probando. El valor predeterminado es el dispositivo local.
- `env-stage`— Se dirige al entorno de despliegue de Greengrass. El valor predeterminado es de producción.
- `existing-device-cert-arn`— El ARN de un certificado existente que desee utilizar como certificado de dispositivo para Greengrass.
- `feature-path`— Archivo o directorio que contiene archivos de funciones adicionales. El valor predeterminado es que no se utilizan archivos de funciones adicionales.
- `gg-cli-version`— Anula la versión de la CLI de Greengrass. El valor predeterminado es el que se encuentra en. `ggc.version`
- `gg-component-bucket`— El nombre de un bucket de Amazon S3 existente que aloja los componentes de Greengrass.
- `gg-component-overrides`— Una lista de anulaciones de componentes de Greengrass.
- `gg-persist`— Una lista de los elementos de prueba que se conservan tras una ejecución de la prueba. El comportamiento predeterminado es no conservar nada. Los valores aceptados son: `aws.resourcesinstalled.software`, y`generated.files`.
- `gg-runtime`— Una lista de valores para influir en la forma en que la prueba interactúa con los recursos de la prueba. Estos valores sustituyen al `gg.persist` parámetro. Si el valor predeterminado está vacío, se asume que todos los recursos de prueba se administran por caso de prueba, incluido el entorno de ejecución de Greengrass instalado. Los valores aceptados son: `aws.resourcesinstalled.software`, y. `generated.files`
- `ggc-archive`— El camino hacia el componente del núcleo archivado de Greengrass.
- `ggc-install-root`— Directorio para instalar el componente núcleo de Greengrass. Los valores predeterminados son test.temp.path y test run folder.

- `ggc-log-level`— Establezca el nivel de registro del núcleo de Greengrass para la ejecución de la prueba. El valor predeterminado es «INFO».
- `ggc-tes-rolename`— La función de IAM que asumirá AWS IoT Greengrass Core para acceder a AWS los servicios. Si no existe un rol con un nombre de pila, se creará uno con una política de acceso predeterminada.
- `ggc-trusted-plugins`— La lista separada por comas de las rutas (en el host) de los complementos de confianza que deben añadirse a Greengrass. Para indicar la ruta en el propio DUT, añada a la ruta el prefijo «dut:»
- `ggc-user-name`— El valor user:group posixUser para el núcleo de Greengrass. El valor predeterminado es el nombre de usuario actual con el que se ha iniciado sesión.
- `ggc-version`— Anula la versión del componente núcleo de Greengrass en ejecución. El valor predeterminado es el que se encuentra en ggc.archive.
- `log-level`— Nivel de registro de la ejecución de la prueba. El valor predeterminado es «INFO».
- `parallel-config`— Conjunto del índice de lotes y el número de lotes como cadena JSON. El valor predeterminado del índice de lotes es 0 y el número de lotes es 1.
- `proxy-url`— Configure todas las pruebas para enrutar el tráfico a través de esta URL.
- `tags`— Ejecute únicamente etiquetas de características. Se puede intersecar con '&'
- `test-id-prefix`— Un prefijo común que se aplica a todos los recursos específicos de la prueba, incluidos los nombres y las etiquetas AWS de los recursos. El prefijo predeterminado es «gg».
- `test-log-path`— Directorio que contendrá los resultados de toda la ejecución de la prueba. El valor predeterminado es «TestResults».
- `test-results-json`— Marca para determinar si se genera un informe JSON de Cucumber resultante escrito en el disco. El valor predeterminado es true (verdadero).
- `test-results-log`— Marca para determinar si la salida de la consola se genera escrita en el disco. El valor predeterminado es falso.
- `test-results-xml`— Marcador para determinar si se genera un informe XML de JUnit resultante escrito en el disco. El valor predeterminado es true (verdadero).
- `test-temp-path`— Directorio para generar artefactos de prueba locales. El valor predeterminado es un directorio temporal aleatorio con el prefijo gg-testing.
- `timeout-multiplier`— Se proporciona un multiplicador para todos los tiempos de espera de las pruebas. El valor predeterminado es 1.0.

Ejemplos de archivos de configuración CLI de GDK

Puede hacer referencia a los siguientes ejemplos de archivos de configuración CLI de GDK para ayudarle a configurar los entornos de componentes de Greengrass.

Hola mundo (Python)

El siguiente archivo de configuración CLI de GDK admite un componente Hello World que ejecuta un script de Python. Este archivo de configuración utiliza el sistema de `zip` compilación para empaquetar el script de Python del componente en un archivo ZIP que la CLI de GDK carga como un artefacto.

```
{
  "component": {
    "com.example.PythonHelloWorld": {
      "author": "Amazon",
      "version": "NEXT_PATCH",
      "build": {
        "build_system" : "zip",
        "options": {
           "excludes": [".*"]
        }
      },
      "publish": {
        "bucket": "greengrass-component-artifacts",
        "region": "us-west-2",
        "options": {
           "file_upload_args": {
              "Metadata": {
                 "some-key": "some-value"
              }
           }
        }
      }
    },
  "test-e2e":{
    "build":{
        "build_system": "maven"
    },
    "gtf_version": "1.1.0",
    "gtf_options": {
         "tags": "Sample"
     }
```

```
  },
  "gdk_version": "1.6.1"
  }
}
```

Hello World (Java)

El siguiente archivo de configuración CLI de GDK admite un componente Hello World que ejecuta una aplicación Java. Este archivo de configuración utiliza el sistema de `maven` compilación para empaquetar el código fuente Java del componente en un archivo JAR que la CLI de GDK carga como un artefacto.

```
{
  "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"
  }
```

```
}
```

Componentes de la comunidad

Varios componentes de la comunidad del catálogo de software de Greengrass utilizan la CLI de GDK. Puede explorar los archivos de configuración de la CLI de GDK en los repositorios de estos componentes.

Para ver los archivos de configuración de la CLI de GDK de los componentes de la comunidad

1. Ejecute el siguiente comando para enumerar los componentes de la comunidad que utilizan la CLI de GDK.

   ```
   gdk component list --repository
   ```

   La respuesta muestra el nombre del GitHub repositorio de cada componente de la comunidad que usa la CLI de GDK. Cada repositorio existe en la `awslabs` organización.

   ```
   [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. Abre el GitHub repositorio de un componente de la comunidad en la siguiente URL. *`community-component-name`*Sustitúyalo por el nombre de un componente de la comunidad del paso anterior.

   ```
   https://github.com/awslabs/community-component-name
   ```

Interfaz de línea de comandos Greengrass

La interfaz de línea de comandos (CLI) de Greengrass le permite interactuar con AWS IoT Greengrass Core en su dispositivo para desarrollar componentes y depurar problemas de forma

local. Por ejemplo, puede usar la CLI de Greengrass para crear una implementación local y reiniciar un componente en el dispositivo principal.

Implemente el componente CLI de Greengrass (`aws.greengrass.Cli`) para instalar la CLI de Greengrass en su dispositivo principal.

> **Important**
>
> Le recomendamos que utilice este componente únicamente en entornos de desarrollo, no en entornos de producción. Este componente proporciona acceso a información y operaciones que normalmente no necesitará en un entorno de producción. Siga el principio de privilegios mínimos implementando este componente solo en los dispositivos principales donde lo necesite.

Temas

- Instalación de la CLI de Greengrass
- Comandos CLI de Greengrass

Instalación de la CLI de Greengrass

Puede instalar la CLI de Greengrass de una de las siguientes maneras:

- Utilice `--deploy-dev-tools` este argumento la primera vez que configure el software AWS IoT Greengrass Core en su dispositivo. También debe especificar `--provision true` la aplicación de este argumento.
- Implemente el componente CLI de Greengrass (`aws.greengrass.Cli`) en su dispositivo.

En esta sección se describen los pasos para implementar el componente CLI de Greengrass. Para obtener información sobre la instalación de la CLI de Greengrass durante la configuración inicial, consulte. Tutorial: Introducción a AWS IoT Greengrass V2

Requisitos previos

Para implementar el componente CLI de Greengrass, debe cumplir los siguientes requisitos:

- AWS IoT Greengrass El software principal está instalado y configurado en su dispositivo principal. Para obtener más información, consulte Tutorial: Introducción a AWS IoT Greengrass V2.

- Para utilizar el AWS CLI para implementar la CLI de Greengrass, debe haber instalado y configurado el. AWS CLI Para obtener más información, consulte Configuración de la AWS CLI en la Guía del usuario de AWS Command Line Interface .
- Debe estar autorizado a utilizar la CLI de Greengrass para interactuar con el software AWS IoT Greengrass principal. Realice una de las siguientes acciones para usar la CLI de Greengrass:
  - Utilice el usuario del sistema que ejecuta el software AWS IoT Greengrass Core.
  - Utilice un usuario con permisos root o administrativos. En los dispositivos principales de Linux, puede utilizarlos `sudo` para obtener permisos de root.
  - Utilice un usuario del sistema que esté en un grupo que especifique en los parámetros de `AuthorizedWindowsGroups` configuración `AuthorizedPosixGroups` o al implementar el componente. Para obtener más información, consulte Configuración de componentes CLI de Greengrass.

Implemente el componente CLI de Greengrass

Complete los siguientes pasos para implementar el componente CLI de Greengrass en su dispositivo principal:

Para implementar el componente CLI de Greengrass (consola)

1. Inicie sesión en la consola de AWS IoT Greengrass.
2. En el menú de navegación, elija Componentes.
3. En la página Componentes, en la pestaña Componentes públicos, elija `aws.greengrass.Cli`.
4. En la página aws.greengrass.Cli, elija Implementar.
5. En Añadir a la implementación, elija Crear nueva implementación.
6. En la página Especificar destino, en Objetivos de despliegue, en la lista Nombre de destino, elija el grupo de Greengrass en el que desee realizar el despliegue y elija Siguiente.
7. En la página Seleccionar componentes, compruebe que el aws.greengrass.Clicomponente esté seleccionado y elija Siguiente.
8. En la página Configurar componentes, conserve los valores de configuración predeterminados y seleccione Siguiente.
9. En la página Configurar los ajustes avanzados, conserve los valores de configuración predeterminados y seleccione Siguiente.
10. En la página de revisión, haga clic en Implementar

Para implementar el componente CLI de Greengrass ()AWS CLI

1. En su dispositivo, cree un `deployment.json` archivo para definir la configuración de despliegue del componente CLI de Greengrass. Este archivo debería tener el siguiente aspecto:

```
{
  "targetArn":"targetArn",
  "components": {
    "aws.greengrass.Cli": {
      "componentVersion": "2.12.6",
      "configurationUpdate": {
        "merge": "{\"AuthorizedPosixGroups\":\"<group1>,<group2>,...,<groupN>\",
\"AuthorizedWindowsGroups\":\"<group1>,<group2>,...,<groupN>\"}"
      }
    }
  }
}
```

   - En el campo `target`, sustituya *`targetArn`* por el nombre de recurso de Amazon (ARN) de la cosa o grupo de cosas a la que apunte la implementación, en el siguiente formato:
     - Cosa: `arn:aws:iot:`*`region`*`:`*`account-id`*`:thing/`*`thingName`*
     - Grupo de cosas: `arn:aws:iot:`*`region`*`:`*`account-id`*`:thinggroup/`*`thingGroupName`*
   - En el objeto `aws.greengrass.Cli` componente, especifique los valores de la siguiente manera:

     `version`

     La versión del componente CLI de Greengrass.

     `configurationUpdate.AuthorizedPosixGroups`

     (Opcional) Cadena que contiene una lista de grupos de sistemas separados por comas. Usted autoriza a estos grupos de sistemas a utilizar la CLI de Greengrass para interactuar con el software AWS IoT Greengrass principal. Puede especificar los nombres o los ID de los grupos. Por ejemplo, `group1,1002,group3` autoriza a tres grupos de sistemas (`group11002`, y`group3`) a utilizar la CLI de Greengrass.

     Si no especifica ningún grupo para autorizarlo, puede usar la CLI de Greengrass como usuario root (`sudo`) o como usuario del sistema que ejecuta el software AWS IoT Greengrass Core.

`configurationUpdate.AuthorizedWindowsGroups`

(Opcional) Cadena que contiene una lista de grupos de sistemas separados por comas. Usted autoriza a estos grupos de sistemas a utilizar la CLI de Greengrass para interactuar con el software AWS IoT Greengrass principal. Puede especificar los nombres o los ID de los grupos. Por ejemplo, `group1,1002,group3` autoriza a tres grupos de sistemas (`group11002`, `ygroup3`) a utilizar la CLI de Greengrass.

Si no especifica ningún grupo para autorizarlo, puede usar la CLI de Greengrass como administrador o como usuario del sistema que ejecuta el software AWS IoT Greengrass principal.

2. Ejecute el siguiente comando para implementar el componente CLI de Greengrass en el dispositivo:

```
$ aws greengrassv2 create-deployment --cli-input-json file://path/
to/deployment.json
```

Durante la instalación, el componente añade un enlace simbólico a `greengrass-cli` la */greengrass/v2*`/bin` carpeta del dispositivo y usted ejecuta la CLI de Greengrass desde esta ruta. Para ejecutar la CLI de Greengrass sin su ruta absoluta, añada la */greengrass/v2*`/bin` carpeta a la variable PATH. Para comprobar la instalación de la CLI de Greengrass, ejecute el siguiente comando:

Linux or Unix

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

Windows

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

Debería ver los siguientes datos de salida:

```
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.
```

Si `greengrass-cli` no lo encuentra, es posible que la implementación no haya podido instalar la CLI de Greengrass. Para obtener más información, consulte Solución de problemas AWS IoT Greengrass V2.

Comandos CLI de Greengrass

La CLI de Greengrass proporciona una interfaz de línea de comandos para interactuar localmente con su dispositivo AWS IoT Greengrass principal. Los comandos CLI de Greengrass utilizan el siguiente formato.

```
$ greengrass-cli <command> <subcommand> [arguments]
```

De forma predeterminada, el archivo `greengrass-cli` ejecutable de la */greengrass/v2*`/bin/` carpeta interactúa con la versión del software AWS IoT Greengrass principal que se ejecuta en la */greengrass/v2* carpeta. Si llama a un archivo ejecutable que no se encuentra en esta ubicación o si desea interactuar con el software de AWS IoT Greengrass Core en una ubicación diferente, debe utilizar uno de los siguientes métodos para especificar de forma explícita la ruta raíz del software de AWS IoT Greengrass Core con el que desea interactuar:

- Establezca la variable de entorno `GGC_ROOT_PATH` en */greengrass/v2*.
- Añada el `--ggcRootPath` */greengrass/v2* argumento al comando, tal y como se muestra en el siguiente ejemplo.

  ```
  greengrass-cli --ggcRootPath /greengrass/v2 <command> <subcommand> [arguments]
  ```

Puede usar los siguientes argumentos con cualquier comando:

- Se utiliza `--help` para obtener información sobre un comando CLI específico de Greengrass.
- Se utiliza `--version` para obtener información sobre la versión CLI de Greengrass.

En esta sección se describen los comandos de la CLI de Greengrass y se proporcionan ejemplos de estos comandos. La sinopsis de cada comando muestra sus argumentos y su uso. Los argumentos opcionales se muestran entre corchetes.

Comandos disponibles

- componente
- Implementación
- registros
- get-debug-password

componente

Utilice el `component` comando para interactuar con los componentes locales del dispositivo principal.

Subcomandos

- details
- Lista
- reiniciar
- stop

details

Recupera la versión, el estado y la configuración de un componente.

Sinopsis

```
greengrass-cli component details --name <component-name>
```

Arguments (Argumentos)

`--name,-n`. El nombre del componente.

Salida

El siguiente ejemplo muestra el resultado que se produce al ejecutar este comando.

```
$ sudo greengrass-cli component details --name MyComponent

Component Name: MyComponent
Version: 1.0.0
State: RUNNING
Configuration: null
```

Lista

Recupera el nombre, la versión, el estado y la configuración de cada componente instalado en el dispositivo.

Sinopsis

```
greengrass-cli component list
```

Arguments (Argumentos)

Ninguno

Salida

El siguiente ejemplo muestra el resultado que se produce al ejecutar este comando.

```
$ sudo greengrass-cli component list

Components currently running in Greengrass:
Component Name: FleetStatusService
Version: 0.0.0
State: RUNNING
Configuration: {"periodicUpdateIntervalSec":86400.0}
Component Name: UpdateSystemPolicyService
Version: 0.0.0
State: RUNNING
Configuration: null
Component Name: aws.greengrass.Nucleus
Version: 2.0.0
```

```
State: FINISHED
Configuration: {"awsRegion":"region","runWithDefault":
{"posixUser":"ggc_user:ggc_group"},"telemetry":{}}
Component Name: DeploymentService
Version: 0.0.0
State: RUNNING
Configuration: null
Component Name: TelemetryAgent
Version: 0.0.0
State: RUNNING
Configuration: null
Component Name: aws.greengrass.Cli
Version: 2.0.0
State: RUNNING
Configuration: {"AuthorizedPosixGroups":"ggc_user"}
```

reiniciar

Reinicie los componentes.

Sinopsis

```
greengrass-cli component restart --names <component-name>,...
```

Arguments (Argumentos)

`--names,-n`. El nombre del componente. Se requiere al menos un nombre de componente. Puede especificar nombres de componentes adicionales, separando cada nombre con una coma.

Salida

Ninguno

stop

Deje de ejecutar los componentes.

Sinopsis

```
greengrass-cli component stop --names <component-name>,...
```

Arguments (Argumentos)

`--names,-n`. El nombre del componente. Se requiere al menos un nombre de componente. Puede especificar nombres de componentes adicionales si es necesario, separando cada nombre con una coma.

Salida

Ninguno

Implementación

Utilice el `deployment` comando para interactuar con los componentes locales de su dispositivo principal.

Para supervisar el progreso de una implementación local, utilice el `status` subcomando. No puede supervisar el progreso de una implementación local mediante la consola.

Subcomandos

- crear
- cancelar
- list
- estado

crear

Cree o actualice una implementación local mediante recetas de componentes, artefactos y argumentos de tiempo de ejecución específicos.

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]>]
```

Arguments

- `--recipeDir,-r`. La ruta completa a la carpeta que contiene los archivos de recetas de los componentes.
- `--artifactDir,-a`. La ruta completa a la carpeta que contiene los archivos de artefactos que desea incluir en la implementación. La carpeta de artefactos debe contener la siguiente estructura de directorios:

  ```
  /path/to/artifact/folder/<component-name>/<component-version>/<artifacts>
  ```

- `--update-config,-c`. Los argumentos de configuración de la implementación, proporcionados como una cadena JSON o un archivo JSON. La cadena JSON debe tener el siguiente formato:

  ```
  { \
    "componentName": { \
      "MERGE": {"config-key": "config-value"}, \
      "RESET": ["path/to/reset/"] \
    } \
  }
  ```

  `MERGE`y `RESET` distinguen entre mayúsculas y minúsculas y deben estar en mayúsculas.
- `--groupId,-g`. El grupo objetivo del despliegue.
- `--merge,-m`. El nombre y la versión del componente de destino que desea añadir o actualizar. Debe proporcionar la información del componente en el formato`<component>`=`<version>`. Utilice un argumento diferente para cada componente adicional que desee especificar. Si es necesario, utilice el `--runWith` argumento para proporcionar el `posixUserposixGroup`, y la `windowsUser` información para ejecutar el componente.
- `--runWith`. El `posixUserposixGroup`, y la `windowsUser` información para ejecutar un componente genérico o de Lambda. Debe proporcionar esta información en el formato`<component>`:`{posixUser|windowsUser}`=`<user>`[:`<=posixGroup>`]. Por ejemplo, puede especificar **`HelloWorld:posixUser=ggc_user:ggc_group`** o**`HelloWorld:windowsUser=ggc_user`**. Utilice un argumento diferente para cada opción adicional que desee especificar.

  Para obtener más información, consulte Configure el usuario que ejecuta los componentes.

- `--systemLimits`. Los límites de recursos del sistema se aplicarán a los procesos de los componentes Lambda genéricos y no contenerizados en el dispositivo principal. Puede configurar la cantidad máxima de uso de CPU y RAM que pueden utilizar los procesos de cada componente. Especifique un objeto JSON serializado o la ruta de un archivo a un archivo JSON. El objeto JSON debe tener el siguiente formato.

  ```
  {  \
    "componentName": { \
      "cpus": cpuTimeLimit, \
      "memory": memoryLimitInKb \
    } \
  }
  ```

  Puede configurar los siguientes límites de recursos del sistema para cada componente:

  - `cpus`— La cantidad máxima de tiempo de CPU que los procesos de este componente pueden utilizar en el dispositivo principal. El tiempo total de CPU de un dispositivo principal equivale a la cantidad de núcleos de CPU del dispositivo. Por ejemplo, en un dispositivo principal con 4 núcleos de CPU, puede establecer este valor 2 para limitar los procesos de este componente al 50 por ciento de uso de cada núcleo de CPU. En un dispositivo con 1 núcleo de CPU, puede establecer este valor `0.25` para limitar los procesos de este componente al 25 por ciento de uso de la CPU. Si establece este valor en un número superior al número de núcleos de la CPU, el software AWS IoT Greengrass Core no limita el uso de la CPU del componente.
  - `memory`— La cantidad máxima de RAM (en kilobytes) que los procesos de este componente pueden utilizar en el dispositivo principal.

  Para obtener más información, consulte Configure los límites de recursos del sistema para los componentes.

  Esta función está disponible para la versión 2.4.0 y versiones posteriores del componente núcleo de Greengrass y la CLI de Greengrass en los dispositivos principales de Linux. AWS IoT Greengrass actualmente no admite esta función en los dispositivos principales de Windows.

- `--remove`. El nombre del componente de destino que desea eliminar de una implementación local. Para eliminar un componente que se fusionó de una implementación en la nube, debe proporcionar el ID de grupo del grupo de cosas de destino en el siguiente formato:

Greengrass nucleus v2.4.0 and later

```
--remove <component-name> --groupId <group-name>
```

Earlier than v2.4.0

```
--remove <component-name> --groupId thinggroup/<group-name>
```

- `--failure-handling-policy`. Define la acción que se lleva a cabo cuando se produce un error en una implementación. Hay dos acciones que puede especificar:
  - `ROLLBACK` –
  - `DO_NOTHING` –

  Esta función está disponible para la versión 2.11.0 y versiones posteriores de. Núcleo de Greengrass

Salida

El siguiente ejemplo muestra el resultado que se produce al ejecutar este comando.

```
$ sudo greengrass-cli deployment create \
    --merge MyApp1=1.0.0 \
    --merge MyApp2=1.0.0 --runWith MyApp2:posixUser=ggc_user \
    --remove MyApp3 \
    --recipeDir recipes/ \
    --artifactDir artifacts/

Local deployment has been submitted! Deployment Id: 44d89f46-1a29-4044-
ad89-5151213dfcbc
```

cancelar

Cancela la implementación especificada.

Sinopsis

```
greengrass-cli deployment cancel
    -i <deployment-id>
```

Argumentos

`-i`. El identificador único del despliegue que se va a cancelar. El identificador de despliegue se devuelve en el resultado del `create` comando.

Salida

- Ninguna

list

Recupera el estado de las últimas 10 implementaciones locales.

Sinopsis

```
greengrass-cli deployment list
```

Arguments

Ninguna

Salida

El siguiente ejemplo muestra el resultado que se produce al ejecutar este comando. Según el estado de la implementación, el resultado muestra uno de los siguientes valores de estado:`IN_PROGRESS`,`SUCCEEDED`, o`FAILED`.

```
$ sudo greengrass-cli deployment list

44d89f46-1a29-4044-ad89-5151213dfcbc: SUCCEEDED
Created on: 6/27/23 11:05 AM
```

estado

Recupera el estado de una implementación específica.

Sinopsis

```
greengrass-cli deployment status -i <deployment-id>
```

Arguments

`-i`. El ID de la implementación.

Salida

El siguiente ejemplo muestra el resultado que se produce al ejecutar este comando. Según el estado de la implementación, el resultado muestra uno de los siguientes valores de estado:`IN_PROGRESS`,`SUCCEEDED`, o`FAILED`.

```
$ sudo greengrass-cli deployment status -i 44d89f46-1a29-4044-ad89-5151213dfcbc

44d89f46-1a29-4044-ad89-5151213dfcbc: FAILED
Created on: 6/27/23 11:05 AM
Detailed Status: <Detailed deployment status>
Deployment Error Stack: List of error codes
Deployment Error Types: List of error types
Failure Cause: Cause
```

registros

Utilice el `logs` comando para analizar los registros de Greengrass en su dispositivo principal.

Subcomandos

- get
- lista de palabras clave
- list-log-files

get

Recopile, filtre y visualice los archivos de registro de Greengrass. Este comando solo admite archivos de registro con formato JSON. Puede especificar el formato de registro en la configuración del núcleo.

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> ]
```

Arguments

- `--log-dir`,`-ld`. La ruta al directorio para comprobar si hay archivos de registro, por ejemplo**/*greengrass/v2*/logs**. No lo use con`--syslog`. Utilice un argumento diferente para cada directorio adicional que desee especificar. Debe utilizar al menos uno de los siguientes `--log-dir` valores`--log-file`: También puede usar ambos argumentos en un solo comando.
- `--log-file`,`-lf`. Las rutas a los directorios de registro que desea utilizar. Utilice un argumento diferente para cada directorio adicional que desee especificar. Debe utilizar al menos uno de los siguientes `--log-dir` valores`--log-file`: También puede usar ambos argumentos en un solo comando.
- `--follow`,`-fol`. Muestra las actualizaciones de registro a medida que se producen. La CLI de Greengrass continúa ejecutándose y lee los registros especificados. Si especifica un intervalo de tiempo, la CLI de Greengrass deja de supervisar los registros una vez finalizados todos los intervalos de tiempo.
- `--filter`,`-f`. La palabra clave, las expresiones regulares o el par clave-valor que se va a utilizar como filtro. Proporcione este valor como cadena, expresión regular o par clave-valor. Utilice un argumento diferente para cada filtro adicional que desee especificar.

  Cuando se evalúan, los filtros múltiples especificados en un único argumento se separan mediante operadores OR y los filtros especificados en argumentos adicionales se combinan con los operadores AND. Por ejemplo, si su comando incluye`--filter "installed" --filter "name=alpha,name=beta"`, la CLI de Greengrass filtrará y mostrará los mensajes de registro que contengan tanto la palabra clave `installed` como una `name` clave que contenga los valores `alpha` o. `beta`
- `--time-window`,`-t`. El intervalo de tiempo durante el que se mostrará la información del registro. Puede utilizar tanto marcas de tiempo exactas como compensaciones relativas. Debe proporcionar esta información en el formato. *<begin-time>*,*<end-time>* Si no especifica ni la hora de inicio ni la hora de finalización, el valor de esa opción se establece de forma predeterminada en la fecha y hora actuales del sistema. Utilice un argumento diferente para cada ventana de tiempo adicional que desee especificar.

La CLI de Greengrass admite los siguientes formatos para las marcas de tiempo:

- `yyyy-MM-DD`, por ejemplo,. `2020-06-30` La hora predeterminada es 00:00:00 cuando se utiliza este formato.

  `yyyyMMDD`, por ejemplo,. `20200630` La hora predeterminada es 00:00:00 cuando se utiliza este formato.

  `HH:mm:ss`, por ejemplo,. `15:30:45` La fecha por defecto es la fecha actual del sistema cuando se utiliza este formato.

  `HH:mm:ssSSS`, por ejemplo,`15:30:45`. La fecha establece de forma predeterminada la fecha actual del sistema cuando se utiliza este formato.

  `YYYY-MM-DD'T'HH:mm:ss'Z'`, por ejemplo,`2020-06-30T15:30:45Z`.

  `YYYY-MM-DD'T'HH:mm:ss`, por ejemplo,`2020-06-30T15:30:45`.

  `yyyy-MM-dd'T'HH:mm:ss.SSS`, por ejemplo,`2020-06-30T15:30:45.250`.

  Los desfases relativos especifican un desfase de período de tiempo con respecto a la hora actual del sistema. La CLI de Greengrass admite el siguiente formato para las compensaciones relativas:. `+|-[<value>h|hr|hours][value m|min|minutes][value]s|sec|seconds`

  Por ejemplo, el siguiente argumento para especificar un intervalo de tiempo entre 1 hora y 2 horas y 15 minutos antes de la hora actual es. `--time-window -2h15min,-1hr`
- `--verbose`. Muestra todos los campos de los mensajes de registro. No lo use con`--syslog`.
- `--no-color`,`-nc`. Eliminar el código de colores. El código de colores predeterminado para los mensajes de registro utiliza texto rojo en negrita. Solo admite terminales tipo Unix porque utiliza secuencias de escape ANSI.
- `--before-b`,. El número de líneas que se deben mostrar antes de una entrada de registro coincidente. El valor predeterminado es 0.
- `--after`,`-a`. El número de líneas que se muestran después de una entrada de registro coincidente. El valor predeterminado es 0.
- `--syslog`. Procese todos los archivos de registro mediante el protocolo syslog definido en el RFC3164. No lo use con y. `--log-dir` `--verbose` El protocolo syslog usa el siguiente formato:. `"<$Priority>$Timestamp $Host $Logger ($Class): $Message"` Si no especifica un archivo de registro, la CLI de Greengrass lee los mensajes de registro

de las siguientes ubicaciones: `/var/log/messages`/`var/log/syslog`, o. `/var/log/system.log`

AWS IoT Greengrassactualmente no admite esta función en los dispositivos principales de Windows.

- `--max-log-queue-size`,`-m`. El número máximo de entradas de registro que se van a asignar a la memoria. Use esta opción para optimizar el uso de la memoria. El valor predeterminado es 100.

Salida

El siguiente ejemplo muestra el resultado que se produce al ejecutar este comando.

```
$ sudo greengrass-cli logs get --verbose \
    --log-file /greengrass/v2/logs/greengrass.log \
    --filter deployment,serviceName=DeploymentService \
    --filter level=INFO \
    --time-window 2020-12-08T01:11:17,2020-12-08T01:11:22

2020-12-08T01:11:17.615Z [INFO] (pool-2-thread-14)
 com.aws.greengrass.deployment.DeploymentService: Current deployment finished.
 {DeploymentId=44d89f46-1a29-4044-ad89-5151213dfcbc, serviceName=DeploymentService,
 currentState=RUNNING}
2020-12-08T01:11:17.675Z [INFO] (pool-2-thread-14)
 com.aws.greengrass.deployment.IotJobsHelper: Updating status of persisted
 deployment. {Status=SUCCEEDED, StatusDetails={detailed-deployment-
status=SUCCESSFUL}, ThingName=MyThing, JobId=22d89f46-1a29-4044-ad89-5151213dfcbc
```

lista de palabras clave

Muestra las palabras clave sugeridas que puedes usar para filtrar los archivos de registro.

Sinopsis

```
greengrass-cli logs list-keywords [arguments]
```

Arguments

Ninguna

Salida

Los ejemplos siguientes muestran el resultado que se produce al ejecutar este comando.

```
$ sudo greengrass-cli logs list-keywords

Here is a list of suggested keywords for Greengrass log:
level=$str
thread=$str
loggerName=$str
eventType=$str
serviceName=$str
error=$str
```

```
$ sudo greengrass-cli logs list-keywords --syslog

Here is a list of suggested keywords for syslog:
priority=$int
host=$str
logger=$str
class=$str
```

list-log-files

Muestra los archivos de registro ubicados en un directorio especificado.

Sinopsis

```
greengrass-cli logs list-log-files [arguments]
```

Arguments

`--log-dir`,`-ld`. La ruta al directorio para comprobar los archivos de registro.

Salida

El siguiente ejemplo muestra el resultado que se produce al ejecutar este comando.

```
$ sudo greengrass-cli logs list-log-files -ld /greengrass/v2/logs/

/greengrass/v2/logs/aws.greengrass.Nucleus.log
/greengrass/v2/logs/main.log
```

```
/greengrass/v2/logs/greengrass.log
Total 3 files found.
```

get-debug-password

Usar`get-debug-password`para imprimir una contraseña generada aleatoriamente paracomponente de consola de depuración local(`aws.greengrass.LocalDebugConsole`). La contraseña caduca 8 horas después de generarse.

Sinopsis

```
greengrass-cli get-debug-password
```

Arguments (Argumentos)

Ninguno

Salida

El siguiente ejemplo muestra la salida producida al ejecutar este comando.

```
$ sudo greengrass-cli get-debug-password

Username: debug
Password: bEDp3MOHdj8ou2w5de_sCBI2XAaguy3a8XxREXAMPLE
Password expires at: 2021-04-01T17:01:43.921999931-07:00
The local debug console is configured to use TLS security. The certificate is self-
signed so you will need to bypass your web browser's security warnings to open the
 console.
Before you bypass the security warning, verify that the certificate fingerprint
 matches the following fingerprints.
SHA-256: 15 0B 2C E2 54 8B 22 DE 08 46 54 8A B1 2B 25 DE FB 02 7D 01 4E 4A 56 67 96
 DA A6 CC B1 D2 C4 1B
SHA-1: BC 3E 16 04 D3 80 70 DA E0 47 25 F9 90 FA D6 02 80 3E B5 C1
```

Utilice el marco AWS IoT Greengrass de pruebas

Greengrass Testing Framework (GTF) es un conjunto de componentes básicos que respaldan la end-to-end automatización desde la perspectiva del cliente. GTF utiliza Cucumber como motor de funciones. AWS IoT Greengrass utiliza los mismos componentes básicos para calificar los cambios

de software en varios dispositivos. Para obtener más información, consulte Greengrass Testing Framework en Github.

El GTF se implementa utilizando Cucumber, una herramienta que se utiliza para ejecutar pruebas automatizadas, a fin de fomentar un desarrollo impulsado por el comportamiento (BDD) de los componentes. En Cucumber, las características de este sistema se describen en un tipo especial de archivo llamado. `feature` Cada característica se describe en un formato legible por humanos denominado escenarios, que son especificaciones que se pueden convertir en pruebas automatizadas. Cada escenario se describe como una serie de pasos que definen las interacciones y los resultados del sistema que se está probando utilizando un lenguaje de dominio específico llamado Gherkin. Un paso de Gherkin se vincula al código de programación mediante un método denominado definición de pasos que conecta la especificación al flujo de prueba. Las definiciones de pasos en GTF se implementan con Java.

Temas

- Funcionamiento
- Registros de cambios
- Opciones de configuración de Greengrass Testing Framework
- Tutorial: Ejecute end-to-end pruebas con Greengrass Testing Framework y Greengrass Development Kit
- Tutorial: Utilice una prueba de confianza del conjunto de pruebas de confianza

Funcionamiento

AWS IoT Greengrass distribuye el GTF como un JAR independiente que consta de varios módulos de Java. Para utilizar el GTF para end-to-end probar componentes, debe implementar las pruebas en un proyecto de Java. Añadir el JAR compatible con las pruebas como dependencia en su proyecto de Java le permite utilizar la funcionalidad existente del GTF y ampliarla escribiendo sus propios casos de prueba personalizados. Para ejecutar los casos de prueba personalizados, puede crear su proyecto Java y ejecutar el JAR de destino con las opciones de configuración que se describen en. Opciones de configuración de Greengrass Testing Framework

JAR independiente de GTF

Greengrass usa Cloudfront como repositorio de Maven para alojar diferentes versiones del JAR independiente de GTF. Para obtener una lista completa de las versiones de GTF, consulte las versiones de GTF.

El JAR independiente de GTF incluye los siguientes módulos. No se limita solo a estos módulos. Puede seleccionar cada una de estas dependencias por separado en su proyecto o incluirlas todas a la vez en el archivo JAR independiente de prueba.

- `aws-greengrass-testing-resources`: Este módulo proporciona una abstracción para gestionar el ciclo de vida de un AWS recurso durante el transcurso de una prueba. Puedes usarlo para definir tus AWS recursos personalizados mediante la `ResourceSpec` abstracción, de modo que GTF pueda encargarse de crear y eliminar esos recursos por ti.
- `aws-greengrass-testing-platform`: Este módulo proporciona una abstracción a nivel de plataforma para el dispositivo que se está probando durante el ciclo de vida de la prueba. Contiene las API que se utilizan para interactuar con el sistema operativo independientemente de la plataforma y se puede utilizar para simular los comandos que se ejecutan en la carcasa del dispositivo.
- `aws-greengrass-testing-components`: Este módulo consta de componentes de muestra que se utilizan para probar las funciones principales de Greengrass, como las implementaciones, el IPC y otras funciones.
- `aws-greengrass-testing-features`: Este módulo consta de pasos comunes reutilizables y sus definiciones que se utilizan para realizar pruebas en el entorno de Greengrass.

Temas

Registros de cambios

La siguiente tabla describe los cambios en cada versión del GTF. Para obtener más información, consulte la página de versiones del GTF en. GitHub

Versión	Cambios
1.2.0	Nuevas características • Añade pasos relacionados con la red para configurar el MQTT y la conectividad a la red de Internet durante las pruebas. • Agrega pasos métricos del sistema para monitorear el uso de la RAM y la CPU del dispositivo. Mejoras y correcciones de errores • El paso de despliegue local de la CLI de Greengrass se vuelve a intentar hasta que se realiza correctamente. • Las pruebas detienen con elegancia el núcleo de Greengrass en lugar de matarlo. • Se ha añadido una mejora en la que GTF sondea el punto final de AWS IoT credenciales hasta que se puedan recuperar las credenciales del alias del objeto y del rol. • Corrige los artefactos y directorios de recetas que faltaban. Esta versión también corrige las versiones de los componentes que faltan. • Soluciona un problema por el que GTF fallaba durante la limpieza de la imagen del docker si la imagen del docker no existe. • Añade la palabra clave CURRENT como versión del componente.
1.1.0	Nuevas características • Añade la posibilidad de instalar un componente personalizado con la configuración. Esto requiere una receta para el componente personali zado. • Añade la posibilidad de actualizar una implementación local con una configuración personalizada. Mejoras y correcciones de errores • Corrige el problema de inconsistencia de la versión GTF en el contexto del registro.
1.0.0	Versión inicial.

Opciones de configuración de Greengrass Testing Framework

opciones de configuración de GTF

Greengrass Testing Framework (GTF) permite configurar determinados parámetros durante el lanzamiento del end-to-end proceso de prueba para organizar el flujo de prueba. Puede especificar estas opciones de configuración como argumentos de la CLI para el JAR independiente del GTF.

La versión 1.1.0 y posteriores del GTF proporcionan las siguientes opciones de configuración.

- `additional-plugins`— (Opcional) Plugins adicionales de Cucumber
- `aws-region`— Se dirige a puntos finales regionales específicos paraAWSservicios. El valor predeterminado es lo queAWSEl SDK descubre.
- `credentials-path`— OpcionalAWSruta de credenciales de perfil. El valor predeterminado son las credenciales descubiertas en el entorno anfitrión.
- `credentials-path-rotation`— Duración de rotación opcional paraAWScredenciales. El valor predeterminado es 15 minutos o`PT15M`.
- `csr-path`— La ruta de la CSR mediante la cual se generará el certificado del dispositivo.
- `device-mode`— El dispositivo objetivo que se está probando. El valor predeterminado es el dispositivo local.
- `env-stage`— Se dirige al entorno de despliegue de Greengrass. El valor predeterminado es de producción.
- `existing-device-cert-arn`— El ARN de un certificado existente que desea utilizar como certificado de dispositivo para Greengrass.
- `feature-path`— Archivo o directorio que contiene archivos de funciones adicionales. El valor predeterminado es que no se utilizan archivos de funciones adicionales.
- `gg-cli-version`— Anula la versión de la CLI de Greengrass. El valor predeterminado es el que se encuentra en`ggc.version`.
- `gg-component-bucket`— El nombre de un depósito de Amazon S3 existente que aloja los componentes de Greengrass.
- `gg-component-overrides`— Una lista de anulaciones de componentes de Greengrass.
- `gg-persist`— Una lista de los elementos de prueba que se conservan tras una ejecución de la prueba. El comportamiento predeterminado es no conservar nada. Los valores aceptados son:`aws.resources`,`installed.software`, y`generated.files`.

- `gg-runtime`— Una lista de valores para influir en la forma en que la prueba interactúa con los recursos de la prueba. Estos valores sustituyen a los`gg.persist`parámetro. Si el valor predeterminado está vacío, se asume que todos los recursos de prueba se administran por caso de prueba, incluido el entorno de ejecución de Greengrass instalado. Los valores aceptados son:`aws.resources`,`installed.software`, y`generated.files`.
- `ggc-archive`— La ruta hacia el componente del núcleo archivado de Greengrass.
- `ggc-install-root`— Directorio para instalar el componente núcleo de Greengrass. Los valores predeterminados son test.temp.path y test run folder.
- `ggc-log-level`— Defina el nivel de registro del núcleo de Greengrass para la ejecución de la prueba. El valor predeterminado es «INFO».
- `ggc-tes-rolename`— La función de IAM queAWS IoT GreengrassCore asumirá el accesoAWSservicios. Si no existe un rol con un nombre de pila, se creará uno con una política de acceso predeterminada.
- `ggc-trusted-plugins`— La lista separada por comas de las rutas (en el servidor) de los complementos de confianza que se deben añadir a Greengrass. Para indicar la ruta en el propio DUT, añada el prefijo «dut:»
- `ggc-user-name`— El valor posixUser user:group para el núcleo de Greengrass. El valor predeterminado es el nombre de usuario actual con el que se ha iniciado sesión.
- `ggc-version`— Anula la versión del componente núcleo de Greengrass en ejecución. El valor predeterminado es el que se encuentra en ggc.archive.
- `log-level`— Nivel de registro de la ejecución de la prueba. El valor predeterminado es «INFO».
- `parallel-config`— Conjunto del índice de lotes y el número de lotes como cadena JSON. El valor predeterminado del índice de lotes es 0 y el número de lotes es 1.
- `proxy-url`— Configure todas las pruebas para enrutar el tráfico a través de esta URL.
- `tags`— Ejecute únicamente etiquetas de funciones. Se puede intersecar con '&'
- `test-id-prefix`— Un prefijo común que se aplica a todos los recursos específicos de la prueba, incluidosAWSnombres y etiquetas de los recursos. El prefijo predeterminado es «gg».
- `test-log-path`— Directorio que contendrá los resultados de toda la ejecución de la prueba. El valor predeterminado es «TestResults».
- `test-results-json`— Marca para determinar si se genera un informe JSON de Cucumber resultante escrito en el disco. El valor predeterminado es true (verdadero).
- `test-results-log`— Marca para determinar si la salida de la consola se genera escrita en el disco. El valor predeterminado es falso.

- `test-results-xml`— Marcador para determinar si se genera un informe XML de JUnit resultante escrito en el disco. El valor predeterminado es true (verdadero).
- `test-temp-path`— Directorio para generar artefactos de prueba locales. El valor predeterminado es un directorio temporal aleatorio con el prefijo gg-testing.
- `timeout-multiplier`— Se proporciona un multiplicador para todos los tiempos de espera de las pruebas. El valor predeterminado es 1.0.

Tutorial: Ejecute end-to-end pruebas con Greengrass Testing Framework y Greengrass Development Kit

AWS IoT GreengrassTesting Framework (GTF) y Greengrass Development Kit (GDK) ofrecen a los desarrolladores formas de ejecutar pruebas. end-to-end Puede completar este tutorial para inicializar un proyecto de GDK con un componente, inicializar un proyecto de GDK con un módulo de prueba y crear un caso de end-to-end prueba personalizado. Después de crear tu caso de prueba personalizado, puedes ejecutar la prueba.

En este tutorial, aprenderá a hacer lo siguiente:

1. Inicializa un proyecto de GDK con un componente.
2. Inicialice un proyecto de GDK con un módulo de prueba. end-to-end
3. Cree un caso de prueba personalizado.
4. Añada una etiqueta al nuevo caso de prueba.
5. Construye el JAR de prueba.
6. Ejecute la prueba .

Temas

Requisitos previos

Necesitará lo siguiente para completar este tutorial:

- GDK versión 1.3.0 o posterior
- Java
- Maven
- Git

Paso 1: inicializar un proyecto de GDK con un componente

- Inicialice una carpeta vacía con un proyecto de GDK. Descargue el `HelloWorld` componente implementado en Python ejecutando el siguiente comando.

  ```
  gdk component init -t HelloWorld -l python -n HelloWorld
  ```

  Este comando crea un nuevo directorio con `HelloWorld` el nombre del directorio actual.

Paso 2: inicializar un proyecto de GDK con un end-to-end módulo de prueba

- GDK le permite descargar la plantilla del módulo de pruebas que consta de una función y de una implementación por etapas. Ejecute el siguiente comando para abrir el `HelloWorld` directorio e inicializar el proyecto GDK existente mediante un módulo de pruebas.

  ```
  cd HelloWorld
  gdk test-e2e init
  ```

  Este comando crea un nuevo directorio con un nombre `gg-e2e-tests` dentro del `HelloWorld` directorio. Este directorio de pruebas es un proyecto de Maven que depende del JAR independiente de pruebas de Greengrass.

Paso 3: Cree un caso de prueba personalizado

La redacción de un caso de prueba personalizado consta, en líneas generales, de dos pasos: crear un archivo de características con un escenario de prueba e implementar las definiciones de los pasos. Para ver un ejemplo de cómo crear un caso de prueba personalizado, consulteEjemplo:

crea un caso de prueba personalizado. Siga los siguientes pasos para crear su caso de prueba personalizado:

1. Cree un archivo de funciones con un escenario de prueba

   Por lo general, una función describe una funcionalidad específica del software que se está probando. En Cucumber, cada función se especifica como un archivo de funciones individual con un título, una descripción detallada y uno o más ejemplos de casos específicos denominados escenarios. Cada escenario consta de un título, una descripción detallada y una serie de pasos que definen las interacciones y los resultados esperados. Los escenarios se escriben en un formato estructurado con las palabras clave «dado», «cuándo» y «entonces».
2. Implemente definiciones de pasos

   Una definición de paso vincula el paso de Gherkin en lenguaje sencillo con el código programático. Cuando Cucumber identifique un paso de Gherkin en un escenario, buscará una definición de paso coincidente para ejecutarlo.

Paso 4: Añade una etiqueta al nuevo caso de prueba

- Puede asignar etiquetas a las características y escenarios para organizar el proceso de prueba. Puede usar etiquetas para categorizar los subconjuntos de escenarios y también seleccionar los ganchos de forma condicional para que se ejecuten. Las entidades y los escenarios pueden tener varias etiquetas separadas por un espacio.

  En este ejemplo, estamos usando el `HelloWorld` componente.

  En el archivo de características, añada una nueva etiqueta con un nombre `@HelloWorld` junto a la `@Sample` etiqueta.

  ```
  @Sample @HelloWorld
  Scenario: As a developer, I can create a component and deploy it on my device
  ....
  ```

Paso 5: Construye el JAR de prueba

1. Construya el componente. Debe compilar el componente antes de crear el módulo de prueba.

   ```
   gdk component build
   ```

2. Cree el módulo de prueba mediante el siguiente comando. Este comando generará el JAR de prueba en la `greengrass-build` carpeta.

```
gdk test-e2e build
```

Paso 6: Ejecuta la prueba

Cuando ejecutas un caso de prueba personalizado, el GTF automatiza el ciclo de vida de la prueba y gestiona los recursos que se crearon durante la prueba. Primero aprovisiona un dispositivo bajo prueba (DUT) como una AWS IoT cosa e instala el software principal de Greengrass en él. A continuación, creará un nuevo componente denominado `HelloWorld` con la receta especificada en esa ruta. A continuación, el `HelloWorld` componente se despliega en el dispositivo principal mediante un despliegue de Greengrass Thing. A continuación, se verificará si el despliegue se ha realizado correctamente. El estado de la implementación cambiará a `COMPLETED` dentro de 3 minutos si la implementación se realiza correctamente.

1. Vaya al `gdk-config.json` archivo del directorio del proyecto para seleccionar las pruebas con la `HelloWorld` etiqueta. Actualice la `test-e2e` clave con el siguiente comando.

```
"test-e2e":{
  "gtf_options" : {
       "tags":"HelloWorld"
   }
}
```

2. Antes de ejecutar las pruebas, debe proporcionar AWS las credenciales al dispositivo anfitrión. El GTF usa estas credenciales para administrar los AWS recursos durante el proceso de prueba. Asegúrese de que el rol que proporcione tenga permisos para automatizar las operaciones necesarias que se incluyen en la prueba.

   Ejecute los siguientes comandos para proporcionar las AWS credenciales.

   - 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. Ejecute la prueba con el siguiente comando.

```
gdk test-e2e run
```

Este comando descarga la última versión del núcleo de Greengrass de la `greengrass-build` carpeta y ejecuta pruebas con ella. Este comando también se dirige solo a los escenarios con la `HelloWorld` etiqueta y genera un informe para esos escenarios. Verá que los AWS recursos que se crearon durante esta prueba se descartan al final de la prueba.

Ejemplo: crea un caso de prueba personalizado

Example

El módulo de pruebas descargado en el proyecto GDK consta de una función de muestra y un archivo de implementación gradual.

En el siguiente ejemplo, creamos un archivo de funciones para probar la función de despliegue de cosas del software Greengrass. Probamos parcialmente la funcionalidad de esta función con un escenario que realiza el despliegue de un componente a través de GreengrassNube de AWS. Se trata de una serie de pasos que nos ayudan a entender las interacciones y los resultados esperados de este caso de uso.

1. Cree un archivo de funciones

   Navegue hasta la `gg-e2e-tests/src/main/resources/greengrass/features` carpeta del directorio actual. Encontrará un ejemplo `component.feature` parecido al siguiente.

   En este archivo de funciones, puede probar la función de despliegue de Thing del software Greengrass. Puede probar parcialmente la funcionalidad de esta función con un escenario

que realice el despliegue de un componente a través de la nube de Greengrass. El escenario consiste en una serie de pasos que ayudan a comprender las interacciones y los resultados esperados de este caso de uso.

```
Feature: Testing features of Greengrassv2 component

Background:
    Given my device is registered as a Thing
    And my device is running Greengrass

@Sample
Scenario: As a developer, I can create a component and deploy it on my device
    When I create a Greengrass deployment with components
        HelloWorld | /path/to/recipe/file
    And I deploy the Greengrass deployment configuration
    Then the Greengrass deployment is COMPLETED on the device after 180 seconds
    And I call my custom step
```

El GTF contiene las definiciones de todos los pasos siguientes, excepto el paso denominado:`And I call my custom step`.

2. Implemente las definiciones de pasos

   El JAR independiente de GTF contiene las definiciones de todos los pasos excepto uno:. `And I call my custom step` Puede implementar este paso en el módulo de pruebas.

   Navegue hasta el código fuente del archivo de prueba. Puede vincular su paso personalizado mediante una definición de paso mediante el siguiente comando.

```
@And("I call my custom step")
public void customStep() {
    System.out.println("My custom step was called ");
}
```

Tutorial: Utilice una prueba de confianza del conjunto de pruebas de confianza

AWS IoT GreengrassTesting Framework (GTF) y Greengrass Development Kit (GDK) ofrecen a los desarrolladores formas de ejecutar pruebas. end-to-end Puede completar este tutorial para inicializar un proyecto de GDK con un componente, inicializar un proyecto de GDK con un módulo de prueba

y utilizar una end-to-end prueba de confianza del conjunto de pruebas de confianza. Una vez que hayas creado tu caso de prueba personalizado, podrás ejecutar la prueba.

Una prueba de confianza es una prueba genérica proporcionada por Greengrass que valida los comportamientos de los componentes fundamentales. Estas pruebas se pueden modificar o ampliar para adaptarlas a necesidades de componentes más específicas.

Para este tutorial utilizaremos un HelloWorld componente. Si está utilizando otro componente, sustituya el HelloWorld componente por el suyo.

En este tutorial, aprenderá a hacer lo siguiente:

1. Inicialice un proyecto de GDK con un componente.
2. Inicialice un proyecto de GDK con un módulo de prueba. end-to-end
3. Utilice una prueba del conjunto de pruebas de confianza.
4. Añada una etiqueta al nuevo caso de prueba.
5. Construye el JAR de prueba.
6. Ejecute la prueba .

Temas

Requisitos previos

Necesitará lo siguiente para completar este tutorial:

- GDK versión 1.6.0 o posterior
- Java

- Maven
- Git

Paso 1: inicializar un proyecto de GDK con un componente

- Inicialice una carpeta vacía con un proyecto de GDK. Descargue el `HelloWorld` componente implementado en Python ejecutando el siguiente comando.

  ```
  gdk component init -t HelloWorld -l python -n HelloWorld
  ```

  Este comando crea un nuevo directorio con `HelloWorld` el nombre del directorio actual.

Paso 2: inicializar un proyecto de GDK con un end-to-end módulo de prueba

- GDK le permite descargar la plantilla del módulo de pruebas que consta de una función y una implementación escalonada. Ejecute el siguiente comando para abrir el `HelloWorld` directorio e inicializar el proyecto GDK existente mediante un módulo de pruebas.

  ```
  cd HelloWorld
  gdk test-e2e init
  ```

  Este comando crea un nuevo directorio con un nombre `gg-e2e-tests` dentro del `HelloWorld` directorio. Este directorio de pruebas es un proyecto de Maven que depende del JAR independiente de pruebas de Greengrass.

Paso 3: Utilice una prueba del conjunto de pruebas de confianza

La redacción de un caso de prueba de confianza consiste en utilizar el archivo de funciones proporcionado y, si es necesario, modificar los escenarios. Para ver un ejemplo del uso de una prueba de confianza, consulteEjemplo: crea un caso de prueba personalizado. Siga los siguientes pasos para utilizar una prueba de confianza:

- Utilice el archivo de funciones proporcionado.

  Navegue hasta la `gg-e2e-tests/src/main/resources/greengrass/features` carpeta del directorio actual. Abra el `confidenceTest.feature` archivo de muestra para utilizar la prueba de confianza.

Paso 4: Añadir una etiqueta al nuevo caso de prueba

- Puede asignar etiquetas a las características y escenarios para organizar el proceso de prueba. Puede usar etiquetas para categorizar los subconjuntos de escenarios y también seleccionar los ganchos de forma condicional para que se ejecuten. Las entidades y los escenarios pueden tener varias etiquetas separadas por un espacio.

  En este ejemplo, estamos usando el `HelloWorld` componente.

  Cada escenario está etiquetado con`@ConfidenceTest`. Cambie o añada etiquetas si desea ejecutar solo un subconjunto del conjunto de pruebas. Cada escenario de prueba se describe en la parte superior de cada prueba de confianza. El escenario consiste en una serie de pasos que ayudan a comprender las interacciones y los resultados esperados de cada caso de prueba. Puede ampliar estas pruebas añadiendo sus propios pasos o modificando los existentes.

  ```
  @ConfidenceTest
  Scenario: As a Developer, I can deploy GDK_COMPONENT_NAME to my device and see it
   is working as expected
  ....
  ```

Paso 5: Construye el JAR de prueba

1. Construya el componente. Debe compilar el componente antes de crear el módulo de prueba.

   ```
   gdk component build
   ```

2. Cree el módulo de prueba mediante el siguiente comando. Este comando generará el JAR de prueba en la `greengrass-build` carpeta.

   ```
   gdk test-e2e build
   ```

Paso 6: Ejecuta la prueba

Cuando realizas una prueba de confianza, el GTF automatiza el ciclo de vida de la prueba y gestiona los recursos que se crearon durante la prueba. Primero aprovisiona un dispositivo bajo prueba (DUT) como una AWS IoT cosa e instala el software principal de Greengrass en él. A continuación, creará un nuevo componente denominado `HelloWorld` con la receta especificada en esa ruta. A continuación, el `HelloWorld` componente se despliega en el dispositivo principal mediante un

despliegue de Greengrass Thing. A continuación, se verificará si el despliegue se ha realizado correctamente. El estado de la implementación cambiará a `COMPLETED` dentro de 3 minutos si la implementación se realiza correctamente.

1. Vaya al `gdk-config.json` archivo del directorio del proyecto para seleccionar las pruebas con la `ConfidenceTest` etiqueta o la etiqueta que haya especificado en el paso 4. Actualice la `test-e2e` clave con el siguiente comando.

   ```
   "test-e2e":{
     "gtf_options" : {
          "tags":"ConfidenceTest"
      }
   }
   ```

2. Antes de ejecutar las pruebas, debe proporcionar AWS las credenciales al dispositivo anfitrión. El GTF usa estas credenciales para administrar los AWS recursos durante el proceso de prueba. Asegúrese de que el rol que proporcione tenga permisos para automatizar las operaciones necesarias que se incluyen en la prueba.

   Ejecute los siguientes comandos para proporcionar las AWS credenciales.

   - 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. Ejecute la prueba con el siguiente comando.

   ```
   gdk test-e2e run
   ```

Este comando descarga la última versión del núcleo de Greengrass de la `greengrass-build` carpeta y ejecuta pruebas con ella. Este comando también se dirige solo a los escenarios con la `ConfidenceTest` etiqueta y genera un informe para esos escenarios. Verá que los AWS recursos que se crearon durante esta prueba se descartan al final de la prueba.

Ejemplo: utilice una prueba de confianza

Example

El módulo de pruebas descargado en el proyecto GDK consta de un archivo de funciones proporcionado.

En el siguiente ejemplo, utilizamos un archivo de funciones para probar la función de despliegue de elementos del software Greengrass. Probamos parcialmente la funcionalidad de esta función con un escenario que realiza el despliegue de un componente a través de GreengrassNube de AWS. Se trata de una serie de pasos que nos ayudan a entender las interacciones y los resultados esperados de este caso de uso.

- Utilice el archivo de funciones proporcionado.

  Navegue hasta la `gg-e2e-tests/src/main/resources/greengrass/features` carpeta del directorio actual. Encontrará un ejemplo `confidenceTest.feature` parecido al siguiente.

```
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
```

Cada escenario de prueba se describe en la parte superior de cada prueba de confianza. El escenario consiste en una serie de pasos que ayudan a comprender las interacciones y los resultados esperados de cada caso de prueba. Puede ampliar estas pruebas añadiendo sus propios pasos o modificando los existentes. Cada uno de los escenarios incluye comentarios que le ayudan a realizar estos ajustes.

Desarrolle AWS IoT Greengrass componentes

Puede desarrollar y probar componentes en su dispositivo principal de Greengrass. Como resultado, puede crear e iterar su AWS IoT Greengrass software sin interactuar con el. Nube de AWS Cuando termine una versión de su componente, podrá subirla a AWS IoT Greengrass la nube para que usted y su equipo puedan implementar el componente en otros dispositivos de su flota. Para obtener más información sobre cómo implementar componentes, consulteImplemente AWS IoT Greengrass componentes en los dispositivos.

Cada componente se compone de una receta y artefactos.

- Recetas

  Cada componente contiene un archivo de recetas, que define sus metadatos. La receta también especifica los parámetros de configuración del componente, las dependencias de los componentes, el ciclo de vida y la compatibilidad de la plataforma. El ciclo de vida del componente define los comandos que instalan, ejecutan y apagan el componente. Para obtener más información, consulte AWS IoT Greengrass referencia de recetas de componentes.

  Puede definir recetas en formato JSON o YAML.
- Artefactos

  Los componentes pueden tener cualquier número de artefactos, que son componentes binarios. Los artefactos pueden incluir scripts, código compilado, recursos estáticos y cualquier otro archivo que consuma un componente. Los componentes también pueden consumir artefactos de las dependencias de los componentes.

AWS IoT Greengrassproporciona componentes prediseñados que puede usar en sus aplicaciones e implementar en sus dispositivos. Por ejemplo, puedes usar el componente administrador de

transmisiones para cargar datos en varios AWS servicios, o puedes usar el componente de CloudWatch métricas para publicar métricas personalizadas en Amazon CloudWatch. Para obtener más información, consulte AWS-componentes proporcionados.

AWS IoT Greengrassselecciona un índice de componentes de Greengrass, denominado Catálogo de software de Greengrass. Este catálogo rastrea los componentes de Greengrass desarrollados por la comunidad de Greengrass. Desde este catálogo, puede descargar, modificar e implementar componentes para crear sus aplicaciones de Greengrass. Para obtener más información, consulte Componentes de la comunidad.

El software AWS IoT Greengrass Core ejecuta los componentes como usuario y grupo del sistema, como `ggc_user` y`ggc_group`, que usted configura en el dispositivo principal. Esto significa que los componentes tienen los permisos de ese usuario del sistema. Si utiliza un usuario del sistema sin un directorio principal, los componentes no pueden utilizar comandos de ejecución ni código que utilice un directorio principal. Esto significa que no puede usar el `pip install some-library --user` comando para instalar paquetes de Python, por ejemplo. Si has seguido el tutorial de introducción para configurar tu dispositivo principal, el usuario del sistema no tiene un directorio principal. Para obtener más información sobre cómo configurar el usuario y el grupo que ejecutan los componentes, consulteConfigure el usuario que ejecuta los componentes.

> **Note**
>
> AWS IoT Greengrassutiliza versiones semánticas para los componentes. Las versiones semánticas siguen un sistema de números de principal.secundario.parche. Por ejemplo, la versión `1.0.0` representa la primera versión principal de un componente. Para obtener más información, consulte la especificación de la versión semántica.

Temas

- Vida útil de los componentes
- Tipos de componentes
- Crear AWS IoT Greengrass componentes
- Pruebe AWS IoT Greengrass los componentes con despliegues locales
- Publique componentes para desplegarlos en sus dispositivos principales
- Interactúa con AWS los servicios
- Ejecute un contenedor Docker

- AWS IoT Greengrass referencia de recetas de componentes
- Referencia de variable de entorno de componentes

Vida útil de los componentes

El ciclo de vida de los componentes define las etapas que el software AWS IoT Greengrass principal utiliza para instalar y ejecutar los componentes. Cada etapa define un script y otra información que especifica cómo se comporta el componente. Por ejemplo, al instalar un componente, el software AWS IoT Greengrass principal ejecuta el script del `Install` ciclo de vida de ese componente. Los componentes de los dispositivos principales tienen los siguientes estados de ciclo de vida:

- `NEW`— La receta y los artefactos del componente se cargan en el dispositivo principal, pero el componente no está instalado. Cuando un componente entra en este estado, ejecuta su script de instalación.
- `INSTALLED`— El componente está instalado en el dispositivo principal. El componente entra en este estado después de ejecutar su script de instalación.
- `STARTING`— El componente se está iniciando en el dispositivo principal. El componente entra en este estado cuando ejecuta su script de inicio. Si el inicio se realiza correctamente, el componente entra en ese `RUNNING` estado.
- `RUNNING`— El componente se está ejecutando en el dispositivo principal. El componente entra en este estado cuando ejecuta su script de ejecución o cuando tiene procesos en segundo plano activos desde su script de inicio.
- `FINISHED`— El componente se ejecutó correctamente y completó su ejecución.
- `STOPPING`— El componente se detiene. El componente entra en este estado cuando ejecuta su script de apagado.
- `ERRORED`— El componente ha detectado un error. Cuando el componente entra en este estado, ejecuta su script de recuperación. A continuación, el componente se reinicia para intentar volver a su uso normal. Si el componente entra en ese `ERRORED` estado tres veces sin ejecutarse correctamente, pasa `BROKEN` a ser.
- `BROKEN`— El componente ha detectado errores varias veces y no se puede recuperar. Debe volver a implementar el componente para solucionarlo.

Tipos de componentes

El tipo de componente especifica cómo el software AWS IoT Greengrass principal ejecuta el componente. Los componentes pueden tener los siguientes tipos:

- Núcleo (`aws.greengrass.nucleus`)

  El núcleo de Greengrass es el componente que proporciona la funcionalidad mínima del software AWS IoT Greengrass Core. Para obtener más información, consulte Núcleo de Greengrass.

- Complemento () `aws.greengrass.plugin`

  El núcleo de Greengrass ejecuta un componente de complemento en la misma máquina virtual Java (JVM) que el núcleo. El núcleo se reinicia al cambiar la versión de un componente del complemento en un dispositivo principal. Para instalar y ejecutar los componentes del complemento, debe configurar el núcleo de Greengrass para que se ejecute como un servicio del sistema. Para obtener más información, consulte Configurar el núcleo de Greengrass como un servicio del sistema.

  Varios componentes proporcionados por AWS son componentes de complementos, lo que les permite interactuar directamente con el núcleo de Greengrass. Los componentes del plugin utilizan el mismo archivo de registro que el núcleo de Greengrass. Para obtener más información, consulte Supervisar AWS IoT Greengrass registros.

- Genérico () `aws.greengrass.generic`

  El núcleo de Greengrass ejecuta los scripts de ciclo de vida de un componente genérico, si el componente define un ciclo de vida.

  Este tipo es el tipo predeterminado para los componentes personalizados.

- Lambda () `aws.greengrass.lambda`

  El núcleo de Greengrass ejecuta un componente de función Lambda mediante el componente lanzador Lambda.

  Al crear un componente a partir de una función Lambda, el componente tiene este tipo. Para obtener más información, consulte AWS LambdaFunciones de ejecución.

> **Note**
>
> No se recomienda especificar el tipo de componente en una receta. AWS IoT Greengrassestablece el tipo automáticamente al crear un componente.

Crear AWS IoT Greengrass componentes

Puede desarrollar AWS IoT Greengrass componentes personalizados en un ordenador de desarrollo local o en un dispositivo central de Greengrass. AWS IoT Greengrass proporciona la interfaz de línea de comandos (CLI de GDK) del kit de AWS IoT Greengrass desarrollo para ayudarlo a crear, compilar y publicar componentes a partir de plantillas de componentes predefinidas y componentes de comunidad. También puede ejecutar comandos de shell integrados para crear, compilar y publicar componentes. Elija entre las siguientes opciones para crear componentes personalizados de Greengrass:

- Utilice la CLI del kit de desarrollo Greengrass

  Utilice la CLI de GDK para desarrollar componentes en un ordenador de desarrollo local. La CLI de GDK crea y empaqueta el código fuente de los componentes en una receta y artefactos que puede publicar como un componente privado en el AWS IoT Greengrass servicio. Puede configurar la CLI de GDK para que actualice automáticamente la versión del componente y los URI del artefacto al publicar el componente, de modo que no necesite actualizar la receta cada vez. Para desarrollar un componente mediante la CLI de GDK, puede partir de una plantilla o un componente comunitario del catálogo de software de Greengrass. Para obtener más información, consulte AWS IoT GreengrassInterfaz de línea de comandos del kit de desarrollo.
- Ejecute los comandos de shell integrados

  Puede ejecutar comandos de shell integrados para desarrollar componentes en un ordenador de desarrollo local o en un dispositivo central de Greengrass. Los comandos de shell se utilizan para copiar o compilar el código fuente de los componentes para convertirlos en artefactos. Cada vez que cree una nueva versión de un componente, debe crear o actualizar la receta con la nueva versión del componente. Al publicar el componente en el AWS IoT Greengrass servicio, debe actualizar el URI de cada artefacto componente de la receta.

Temas

- Crear un componente (GDK CLI)

- Cree un componente (comandos de shell)

Crear un componente (GDK CLI)

Siga las instrucciones de esta sección para crear y compilar un componente mediante la CLI de GDK.

Para desarrollar un componente de Greengrass (GDK CLI)

1. Si aún no lo ha hecho, instale la CLI de GDK en su ordenador de desarrollo. Para obtener más información, consulte Instalar o actualizar la interfaz de línea AWS IoT Greengrass de comandos del kit de desarrollo.
2. Cambie a la carpeta en la que desee crear las carpetas de componentes.

   Linux or Unix

   ```
   mkdir ~/greengrassv2
   cd ~/greengrassv2
   ```

   Windows Command Prompt (CMD)

   ```
   mkdir %USERPROFILE%\greengrassv2
   cd %USERPROFILE%\greengrassv2
   ```

   PowerShell

   ```
   mkdir ~/greengrassv2
   cd ~/greengrassv2
   ```

3. Elija una plantilla de componentes o un componente de comunidad para descargarlos. La CLI de GDK descarga la plantilla o el componente de la comunidad, por lo que puede empezar con un ejemplo funcional. Utilice el comando component list para recuperar la lista de plantillas o componentes de la comunidad disponibles.
   - Para enumerar las plantillas de componentes, ejecute el siguiente comando. Cada línea de la respuesta incluye el nombre y el lenguaje de programación de la plantilla.

     ```
     gdk component list --template
     ```

   - Para enumerar los componentes de la comunidad, ejecute el siguiente comando.

```
gdk component list --repository
```

4. Cree y cambie a una carpeta de componentes en la que la CLI de GDK descargue la plantilla o el componente de la comunidad. *HelloWorld*Sustitúyalo por el nombre del componente o por otro nombre que ayude a identificar esta carpeta de componentes.

   Linux or Unix

   ```
   mkdir HelloWorld
   cd HelloWorld
   ```

   Windows Command Prompt (CMD)

   ```
   mkdir HelloWorld
   cd HelloWorld
   ```

   PowerShell

   ```
   mkdir HelloWorld
   cd HelloWorld
   ```

5. Descargue la plantilla o el componente de la comunidad en la carpeta actual. Utilice el comando component init.

   - Para crear una carpeta de componentes a partir de una plantilla, ejecute el siguiente comando. *HelloWorld*Sustituya por el nombre de la plantilla y sustituya *python* por el nombre del lenguaje de programación.

     ```
     gdk component init --template HelloWorld --language python
     ```

   - Para crear una carpeta de componentes a partir de un componente de la comunidad, ejecute el siguiente comando. *ComponentName*Sustitúyalo por el nombre del componente de la comunidad.

     ```
     gdk component init --repository ComponentName
     ```

> **Note**
>
> Si usa GDK CLI v1.0.0, debe ejecutar este comando en una carpeta vacía. La CLI de GDK descarga la plantilla o el componente de la comunidad en la carpeta actual.
> Si usa la versión 1.1.0 de la CLI de GDK o una versión posterior, puede especificar el `--name` argumento para especificar la carpeta en la que la CLI de GDK descarga la plantilla o el componente de la comunidad. Si usa este argumento, especifique una carpeta que no exista. La CLI de GDK crea la carpeta por usted. Si no especifica este argumento, la CLI de GDK utilizará la carpeta actual, que debe estar vacía.

6. La CLI de GDK lee el archivo de configuración de la CLI de GDK, denominado`gdk-config.json`, para crear y publicar componentes. Este archivo de configuración existe en la raíz de la carpeta del componente. El paso anterior crea este archivo automáticamente. En este paso, se actualiza `gdk-config.json` con información sobre el componente. Haga lo siguiente:

   a. Abra `gdk-config.json` en un editor de texto.

   b. (Opcional) Cambie el nombre del componente. El nombre del componente es la clave del `component` objeto.

   c. Cambie el autor del componente.

   d. (Opcional) Cambie la versión del componente. Especifique uno de los siguientes valores:

      - `NEXT_PATCH`— Al elegir esta opción, la CLI de GDK establece la versión al publicar el componente. La CLI de GDK consulta el AWS IoT Greengrass servicio para identificar la última versión publicada del componente. A continuación, establece la versión en la siguiente versión del parche posterior a esa versión. Si no ha publicado el componente antes, la CLI de GDK usa la versión`1.0.0`.

        Si elige esta opción, no podrá usar la CLI de Greengrass para implementar y probar localmente el componente en su computadora de desarrollo local que ejecuta el software AWS IoT Greengrass Core. Para habilitar las implementaciones locales, debe especificar una versión semántica en su lugar.

      - Una versión semántica, como. **`1.0.0`** Las versiones semánticas utilizan una principal. menor. sistema de numeración de parches. Para obtener más información, consulte la especificación de la versión semántica.

Si desarrolla componentes en un dispositivo principal de Greengrass en el que desee implementar y probar el componente, elija esta opción. Debe compilar el componente con una versión específica para crear despliegues locales con la CLI de Greengrass.

e. (Opcional) Cambie la configuración de compilación del componente. La configuración de compilación define cómo la CLI de GDK crea la fuente del componente en artefactos. Elija una de las siguientes opciones para`build_system`:

- `zip`— Empaqueta la carpeta del componente en un archivo ZIP para definirla como el único artefacto del componente. Seleccione esta opción para los siguientes tipos de componentes:
  - Componentes que utilizan lenguajes de programación interpretados, como Python o JavaScript.
  - Componentes que empaquetan archivos distintos del código, como modelos de aprendizaje automático u otros recursos.

  La CLI de GDK comprime la carpeta del componente en un archivo zip con el mismo nombre que la carpeta del componente. Por ejemplo, si el nombre de la carpeta del componente es`HelloWorld`, la CLI de GDK crea un archivo zip denominado`HelloWorld.zip`.

  > **Note**
  >
  > Si utiliza la versión 1.0.0 de la CLI de GDK en un dispositivo Windows, los nombres de las carpetas y los archivos zip de los componentes deben contener solo letras minúsculas.

  Cuando la CLI de GDK comprime la carpeta del componente en un archivo zip, omite los siguientes archivos:

  - El archivo `gdk-config.json`
  - El archivo de recetas (o) `recipe.json` `recipe.yaml`
  - Cree carpetas, como `greengrass-build`
- `maven`— Ejecuta el `mvn clean package` comando para convertir la fuente del componente en artefactos. Elija esta opción para los componentes que utilizan Maven, como los componentes de Java.

En los dispositivos Windows, esta función está disponible para GDK CLI v1.1.0 y versiones posteriores.

- `gradle`— Ejecuta el `gradle build` comando para convertir la fuente del componente en artefactos. Elige esta opción para los componentes que usan Gradle. Esta función está disponible para GDK CLI v1.1.0 y versiones posteriores.

  El sistema de `gradle` compilación admite Kotlin DSL como archivo de compilación. Esta función está disponible para GDK CLI v1.2.0 y versiones posteriores.

- `gradlew`— Ejecuta el `gradlew` comando para convertir la fuente del componente en artefactos. Elija esta opción para los componentes que utilizan el Gradle Wrapper.

  Esta función está disponible para GDK CLI v1.2.0 y versiones posteriores.

- `custom`— Ejecuta un comando personalizado para convertir la fuente del componente en una receta y artefactos. Especifique el comando personalizado en el `custom_build_command` parámetro.

f. Si especifica `custom` para`build_system`, `custom_build_command` agréguelo al `build` objeto. En`custom_build_command`, especifique una sola cadena o lista de cadenas, donde cada cadena sea una palabra del comando. Por ejemplo, para ejecutar un comando de compilación personalizado para un componente de C++, puede especificar**`["cmake", "--build", "build", "--config", "Release"]`**.

g. Si utiliza la versión 1.1.0 de la CLI de GDK o una versión posterior, puede especificar el `--bucket` argumento para especificar el depósito de S3 en el que la CLI de GDK carga los artefactos del componente. Si no especificas este argumento, la CLI de GDK se carga en el bucket de S3 cuyo nombre es*`bucket`*`-`*`region`*`-`*`accountId`*, donde *`bucket`* y *`region`* son los valores que especificas y *`AccountID`* es tu ID. `gdk-config.json` Cuenta de AWS La CLI de GDK crea el bucket si no existe.

   Cambie la configuración de publicación del componente. Haga lo siguiente:

   i. Especifique el nombre del depósito de S3 que se utilizará para alojar los artefactos de los componentes.

   ii. Especifique el Región de AWS lugar donde la CLI de GDK publica el componente.

Cuando haya terminado con este paso, el `gdk-config.json` archivo podría tener un aspecto similar al del siguiente ejemplo.

```
{
  "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. Actualice el archivo de recetas del componente, denominado `recipe.yaml` `orecipe.json`. Haga lo siguiente:

   a. Si has descargado una plantilla o un componente de la comunidad que utiliza el sistema de `zip` compilación, comprueba que el nombre del artefacto zip coincide con el nombre de la carpeta del componente. La CLI de GDK comprime la carpeta del componente en un archivo zip con el mismo nombre que la carpeta del componente. La receta contiene el nombre del artefacto zip en la lista de artefactos componentes y en los scripts de ciclo de vida que utilizan archivos del artefacto zip. Actualice las `Lifecycle` definiciones `Artifacts` y de forma que el nombre del archivo zip coincida con el nombre de la carpeta del componente. Los siguientes ejemplos de recetas parciales resaltan el nombre del fichero zip en las `Lifecycle` definiciones `Artifacts` y.

      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. (Opcional) Actualice la descripción del componente, la configuración predeterminada, los artefactos, los scripts del ciclo de vida y el soporte de la plataforma. Para obtener más información, consulte AWS IoT Greengrass referencia de recetas de componentes.

Cuando haya terminado con este paso, el archivo de recetas podría tener un aspecto similar al de los ejemplos siguientes.

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. Desarrolle y construya el componente Greengrass. El comando de construcción del componente produce una receta y artefactos en la `greengrass-build` carpeta de la carpeta del componente. Ejecute el siguiente comando de la .

```
gdk component build
```

Cuando esté listo para probar el componente, utilice la CLI de GDK para publicarlo en el AWS IoT Greengrass servicio. A continuación, puede implementar el componente en los dispositivos principales de Greengrass. Para obtener más información, consulte Publique componentes para desplegarlos en sus dispositivos principales.

Cree un componente (comandos de shell)

Siga las instrucciones de esta sección para crear carpetas de recetas y artefactos que contengan el código fuente y los artefactos de varios componentes.

Para desarrollar un componente de Greengrass (comandos de shell)

1. Cree una carpeta para sus componentes con subcarpetas para recetas y artefactos. Ejecute los siguientes comandos en su dispositivo principal de Greengrass para crear estas carpetas y cambiarlas a la carpeta de componentes. Sustituya *~/greengrassv2* o *%USERPROFILE%\greengrassv2* por la ruta a la carpeta que se utilizará para el desarrollo local.

   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. Utilice un editor de texto para crear un archivo de recetas que defina los metadatos, los parámetros, las dependencias, el ciclo de vida y la capacidad de la plataforma de su componente. Incluya la versión del componente en el nombre del archivo de recetas para poder identificar qué receta refleja qué versión del componente. Puedes elegir el formato YAML o JSON para tu receta.

   Por ejemplo, en un sistema basado en Linux, puede ejecutar el siguiente comando para usar GNU nano para crear el archivo.

   JSON

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

   YAML

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

   > **Note**
   >
   > AWS IoT Greengrass usa versiones semánticas para los componentes. Las versiones semánticas siguen un sistema de números de principal.secundario.parche. Por ejemplo, la versión `1.0.0` representa la primera versión principal de un componente. Para obtener más información, consulte la especificación de la versión semántica.

3. Defina la receta de su componente. Para obtener más información, consulte AWS IoT Greengrass referencia de recetas de componentes.

   Su receta podría tener un aspecto similar al siguiente ejemplo de receta de Hello World.

   JSON

   ```
   {
     "RecipeFormatVersion": "2020-01-25",
   ```

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

YAML

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

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

Esta receta ejecuta un script de Python de Hello World, que podría tener un aspecto similar al siguiente script de ejemplo.

```
import sys

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

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

4. Cree una carpeta para desarrollar la versión del componente. Le recomendamos que utilice una carpeta independiente para los artefactos de cada versión del componente, de modo que pueda identificar los artefactos de cada versión del componente. Ejecute el siguiente comando de la .

   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**
>
> Debe utilizar el siguiente formato para la ruta de la carpeta de artefactos. Incluya el nombre y la versión del componente que especifique en la receta.
>
> ```
> artifacts/componentName/componentVersion/
> ```

5. Cree los artefactos del componente en la carpeta que creó en el paso anterior. Los artefactos pueden incluir software, imágenes y cualquier otro binario que utilice el componente.

   Cuando el componente esté listo, pruébelo.

Pruebe AWS IoT Greengrass los componentes con despliegues locales

Si desarrolla un componente de Greengrass en un dispositivo principal, puede crear una implementación local para instalarlo y probarlo. Siga los pasos de esta sección para crear una implementación local.

Si desarrolla el componente en un equipo diferente, como un equipo de desarrollo local, no podrá crear un despliegue local. En su lugar, publique el componente en el AWS IoT Greengrass servicio para poder implementarlo en los dispositivos principales de Greengrass para probarlo. Para obtener más información, consulte Publique componentes para desplegarlos en sus dispositivos principales y Implemente AWS IoT Greengrass componentes en los dispositivos.

Para probar un componente en un dispositivo principal de Greengrass

1. El dispositivo principal registra eventos como las actualizaciones de componentes. Puede ver este archivo de registro para detectar y solucionar errores en su componente, como una receta no válida. Este archivo de registro también muestra los mensajes que el componente imprime en formato estándar (stdout). Le recomendamos que abra una sesión de terminal adicional en su dispositivo principal para observar los nuevos mensajes de registro en tiempo real. Abre una nueva sesión de terminal, por ejemplo, mediante SSH, y ejecuta el siguiente comando para ver los registros. */greengrass/v2*Sustitúyala por la ruta a la carpeta AWS IoT Greengrass raíz.

Linux or Unix

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

PowerShell

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

También puede ver el archivo de registro de su componente.

Linux or Unix

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

PowerShell

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

2. En la sesión de terminal original, ejecute el siguiente comando para actualizar el dispositivo principal con su componente. Reemplace por */greengrass/v2* la ruta a la carpeta AWS IoT Greengrass raíz y reemplace *~/greengrassv2* por la ruta a su carpeta de desarrollo local.

   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**
>
> También puede usar el `greengrass-cli deployment create` comando para establecer el valor de los parámetros de configuración de su componente. Para obtener más información, consulte crear.

3. Utilice el `greengrass-cli deployment status` comando para supervisar el progreso de la implementación del componente.

   Unix or Linux

   ```
   sudo /greengrass/v2/bin/greengrass-cli deployment status \
     -i deployment-id
   ```

   Windows Command Prompt (CMD)

   ```
   C:\greengrass\v2\bin\greengrass-cli deployment status ^
     -i deployment-id
   ```

   PowerShell

   ```
   C:\greengrass\v2\bin\greengrass-cli deployment status `
     -i deployment-id
   ```

4. Pruebe su componente mientras se ejecuta en el dispositivo principal de Greengrass. Cuando termine esta versión de su componente, podrá subirla al AWS IoT Greengrass servicio. A continuación, puede implementar el componente en otros dispositivos principales. Para obtener más información, consulte Publique componentes para desplegarlos en sus dispositivos principales.

Publique componentes para desplegarlos en sus dispositivos principales

Después de crear o completar una versión de un componente, puede publicarla en el AWS IoT Greengrass servicio. Luego, puede implementarlo en los dispositivos principales de Greengrass.

Si utiliza la CLI del kit de desarrollo de Greengrass (CLI de GDK) para desarrollar y crear un componente, puede utilizar la CLI de GDK para publicar el componente en. Nube de AWS De lo contrario, utilice los comandos de shell integrados y los AWS CLI para publicar el componente.

También se puede utilizar AWS CloudFormation para crear componentes y otros AWS recursos a partir de plantillas. Para obtener más información, consulte ¿Qué es AWS CloudFormation? y AWS::GreengrassV2::ComponentVersionen la Guía AWS CloudFormation del usuario.

Temas

- Publicar un componente (GDK CLI)
- Publicar un componente (comandos de shell)

Publicar un componente (GDK CLI)

Siga las instrucciones de esta sección para publicar un componente mediante la CLI de GDK. La CLI de GDK carga los artefactos de construcción en un bucket de S3, actualiza los URI de los artefactos de la receta y crea el componente a partir de la receta. Debe especificar el bucket y la región de S3 que se van a utilizar en el archivo de configuración de la CLI de GDK.

Si utiliza la versión 1.1.0 de la CLI de GDK o una versión posterior, puede especificar el `--bucket` argumento para especificar el depósito de S3 en el que la CLI de GDK carga los artefactos del componente. Si no especificas este argumento, la CLI de GDK se carga en el bucket de S3 cuyo nombre es*`bucket`*`-`*`region`*`-`*`accountId`*, donde *`bucket`* y *`region`* son los valores que especificas y *`AccountID`* es tu ID. `gdk-config.json` Cuenta de AWS La CLI de GDK crea el bucket si no existe.

> **Important**
>
> De forma predeterminada, las funciones principales de los dispositivos no permiten el acceso a los depósitos de S3. Si es la primera vez que utiliza este depósito de S3, debe añadir permisos al rol para que los dispositivos principales puedan recuperar los artefactos de los componentes de este depósito de S3. Para obtener más información, consulte Permita el acceso a los depósitos de S3 para los artefactos de los componentes.

Para publicar un componente de Greengrass (GDK CLI)

1. Abra la carpeta del componente en una línea de comandos o en una terminal.
2. Si aún no lo ha hecho, cree el componente Greengrass. El comando de construcción del componente produce una receta y artefactos en la `greengrass-build` carpeta de la carpeta del componente. Ejecute el siguiente comando de la .

   ```
   gdk component build
   ```

3. Publique el componente en Nube de AWS. El comando component publish carga los artefactos del componente en Amazon S3 y actualiza la receta del componente con el URI de cada artefacto. A continuación, crea el componente en el AWS IoT Greengrass servicio.

   > **Note**
   >
   > AWS IoT Greengrass calcula el resumen de cada artefacto al crear el componente. Esto significa que no puede modificar los archivos de artefactos de su bucket de S3 después de crear un componente. Si lo hace, las implementaciones que incluyan este componente fallarán porque el resumen del archivo no coincide. Si modifica un archivo de artefactos, debe crear una nueva versión del componente.

   Si especifica `NEXT_PATCH` la versión del componente en el archivo de configuración de la CLI de GDK, la CLI de GDK utilizará la siguiente versión del parche que aún no exista en el AWS IoT Greengrass servicio.

   Ejecute el siguiente comando de la .

   ```
   gdk component publish
   ```

   El resultado indica la versión del componente que creó la CLI de GDK.

   Después de publicar el componente, puede implementarlo en los dispositivos principales. Para obtener más información, consulte Implemente AWS IoT Greengrass componentes en los dispositivos.

Publicar un componente (comandos de shell)

Utilice el siguiente procedimiento para publicar un componente mediante los comandos de shell y el AWS Command Line Interface (AWS CLI). Al publicar un componente, haga lo siguiente:

1. Publique los artefactos de los componentes en un bucket de S3.
2. Añada el URI de Amazon S3 de cada artefacto a la receta del componente.
3. Cree una versión del componente AWS IoT Greengrass a partir de la receta del componente.

> **Note**
>
> Cada versión de componente que cargue debe ser única. Asegúrese de cargar la versión del componente correcta, ya que no podrá editarla después de cargarla.

Puede seguir estos pasos para publicar un componente desde su ordenador de desarrollo o su dispositivo principal de Greengrass.

Para publicar un componente (comandos de shell)

1. Si el componente usa una versión que existe en el AWS IoT Greengrass servicio, debe cambiarla. Abra la receta en un editor de texto, incremente la versión y guarde el archivo. Elija una nueva versión que refleje los cambios que ha realizado en el componente.

   > **Note**
   >
   > AWS IoT Greengrass utiliza versiones semánticas para los componentes. Las versiones semánticas siguen un sistema de números de principal.secundario.parche. Por ejemplo, la versión `1.0.0` representa la primera versión principal de un componente. Para obtener más información, consulte la especificación de la versión semántica.

2. Si el componente tiene artefactos, haga lo siguiente:

   a. Publique los artefactos del componente en un depósito de S3 de su Cuenta de AWS.

> **Tip**
>
> Le recomendamos que incluya el nombre y la versión del componente en la ruta al artefacto del bucket de S3. Este esquema de nomenclatura puede ayudarte a conservar los artefactos que utilizan las versiones anteriores del componente, de forma que puedas seguir admitiendo las versiones anteriores del componente.

Ejecute el siguiente comando para publicar un archivo de artefactos en un bucket de S3. *Sustituya DOC-EXAMPLE-BUCKET por el nombre del depósito y sustituya artifacts/com.example. HelloWorld/1.0.0/artifact.py* por la ruta al archivo del artefacto.

```
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**
>
> De forma predeterminada, las funciones principales de los dispositivos no permiten el acceso a los depósitos de S3. Si es la primera vez que utiliza este depósito de S3, debe añadir permisos al rol para que los dispositivos principales puedan recuperar los artefactos de los componentes de este depósito de S3. Para obtener más información, consulte Permita el acceso a los depósitos de S3 para los artefactos de los componentes.

b. Agrega una lista `Artifacts` con el nombre de la receta del componente si no está presente. La `Artifacts` lista aparece en cada manifiesto, donde se definen los requisitos del componente en cada plataforma compatible (o los requisitos predeterminados del componente para todas las plataformas).

c. Añada cada artefacto a la lista de artefactos o actualice el URI de los artefactos existentes. El URI de Amazon S3 está compuesto por el nombre del bucket y la ruta al objeto artefacto del bucket. Los URI de Amazon S3 de sus artefactos deberían tener un aspecto similar al del siguiente ejemplo.

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

Tras completar estos pasos, la receta debería tener una `Artifacts` lista similar a la siguiente.

JSON

```
{
  ...
  "Manifests": [
    {
      "Lifecycle": {
        ...
      },
      "Artifacts": [
        {
          "URI": "s3://DOC-EXAMPLE-BUCKET/artifacts/MyGreengrassComponent/1.0.0/
artifact.py",
          "Unarchive": "NONE"
        }
      ]
    }
  ]
}
```

> **Note**
>
> Puede añadir la `"Unarchive": "ZIP"` opción de utilizar un artefacto ZIP para configurar el software AWS IoT Greengrass principal de forma que descomprima el artefacto cuando se despliegue el componente.

YAML

```
...
Manifests:
  - Lifecycle:
      ...
    Artifacts:
      - URI: s3://DOC-EXAMPLE-BUCKET/artifacts/MyGreengrassComponent/1.0.0/
artifact.py
        Unarchive: NONE
```

> **Note**
>
> Puede utilizar la `Unarchive: ZIP` opción de configurar el software AWS IoT Greengrass Core para descomprimir un artefacto ZIP cuando se despliegue el componente. Para obtener más información sobre cómo utilizar los artefactos ZIP en un componente, consulte la variable de receta Artifacts:decompressedPath.

Para obtener más información acerca de las recetas, consulte AWS IoT Greengrass referencia de recetas de componentes.

3. Utilice la AWS IoT Greengrass consola para crear un componente a partir del archivo de recetas.

   Ejecute el siguiente comando para crear el componente a partir de un archivo de recetas. Este comando crea el componente y lo publica como un AWS IoT Greengrass componente privado en su Cuenta de AWS. Sustituya *`path/to/recipeFile por la ruta del archivo`* de recetas.

   ```
   aws greengrassv2 create-component-version --inline-recipe fileb://path/to/
   recipeFile
   ```

   Copie la `arn` información de la respuesta para comprobar el estado del componente en el siguiente paso.

   > **Note**
   >
   > AWS IoT Greengrass calcula el resumen de cada artefacto al crear el componente. Esto significa que no puede modificar los archivos de artefactos de su bucket de S3 después de crear un componente. Si lo hace, las implementaciones que incluyan este componente fallarán porque el resumen del archivo no coincide. Si modifica un archivo de artefactos, debe crear una nueva versión del componente.

4. Cada componente del AWS IoT Greengrass servicio tiene un estado. Ejecute el siguiente comando para confirmar el estado de la versión del componente que publique en este procedimiento. Sustituya *`com.example. HelloWorld`*y *`1.0.0`* con la versión del componente que se va a consultar. Sustituya el `arn` por el ARN del paso anterior.

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

La operación devuelve una respuesta que contiene los metadatos del componente. Los metadatos contienen un `status` objeto que contiene el estado del componente y cualquier error, si corresponde.

Cuando el estado del componente es`DEPLOYABLE`, puede implementarlo en los dispositivos. Para obtener más información, consulte Implemente AWS IoT Greengrass componentes en los dispositivos.

Interactúa con AWS los servicios

Los dispositivos principales de Greengrass utilizan certificados X.509 para conectarse AWS IoT Core mediante protocolos de autenticación mutua TLS. Estos certificados permiten que los dispositivos interactúen AWS IoT sin AWS credenciales, que normalmente incluyen un identificador de clave de acceso y una clave de acceso secreta. Otros AWS servicios requieren AWS credenciales en lugar de certificados X.509 para realizar llamadas a las operaciones de la API en los puntos finales del servicio. AWS IoT Coretiene un proveedor de credenciales que permite a los dispositivos utilizar su certificado X.509 para autenticar las solicitudes. AWS El proveedor de AWS IoT credenciales autentica los dispositivos mediante un certificado X.509 y emite AWS las credenciales en forma de token de seguridad temporal con privilegios limitados. Los dispositivos pueden usar este token para firmar y autenticar cualquier solicitud. AWS Esto elimina la necesidad de almacenar AWS las credenciales en los dispositivos principales de Greengrass. Para obtener más información, consulte Autorizar llamadas directas a AWS los servicios en la Guía para AWS IoT Coredesarrolladores.

Para obtener las credenciales de GreengrassAWS IoT, los dispositivos principales utilizan AWS IoT un alias de rol que apunta a un rol de IAM. Esta función de IAM se denomina función de intercambio de fichas. El alias del rol y el rol de intercambio de tokens se crean al instalar el software AWS IoT Greengrass Core. Para especificar el alias de rol que utiliza un dispositivo principal, configure el `iotRoleAlias` parámetro delNúcleo de Greengrass.

El proveedor de AWS IoT credenciales asume la función de intercambio de fichas en su nombre para proporcionar AWS credenciales a los dispositivos principales. Puede adjuntar las políticas de IAM adecuadas a esta función para permitir que sus dispositivos principales accedan a sus AWS recursos, como los componentes y artefactos de los depósitos de S3. Para obtener más información

sobre cómo configurar la función de intercambio de tokens, consulte. Autorizar a los dispositivos principales a interactuar con AWS los servicios

Los dispositivos principales de Greengrass almacenan AWS las credenciales en la memoria y, de forma predeterminada, las credenciales caducan después de una hora. Si el software AWS IoT Greengrass Core se reinicia, debe volver a buscar las credenciales. Puede utilizar la UpdateRoleAliasoperación para configurar el tiempo de validez de las credenciales.

AWS IoT Greengrassproporciona un componente público, el componente del servicio de intercambio de fichas, que puede definir como una dependencia en su componente personalizado para interactuar con AWS los servicios. El servicio de intercambio de tokens proporciona al componente una variable de entorno que define el URI de un servidor local que proporciona AWS las credenciales. `AWS_CONTAINER_CREDENTIALS_FULL_URI` Al crear un cliente de AWS SDK, el cliente comprueba esta variable de entorno y se conecta al servidor local para recuperar AWS las credenciales y las utiliza para firmar las solicitudes de API. Esto le permite usar AWS los SDK y otras herramientas para llamar a AWS los servicios de sus componentes. Para obtener más información, consulte Servicio de intercambio de fichas.

> **Important**
>
> El 13 de julio de 2016 se agregó a los AWS SDK el soporte para adquirir AWS credenciales de esta manera. Su componente debe usar una versión AWS del SDK que se haya creado en esa fecha o después. Para obtener más información, consulte Uso de un AWS SDK compatible en la Guía para desarrolladores de Amazon Elastic Container Service.

Para adquirir AWS credenciales en su componente personalizado, defínalas `aws.greengrass.TokenExchangeService` como una dependencia en la receta del componente. La siguiente receta de ejemplo define un componente que instala boto3 y ejecuta un script de Python que usa AWS las credenciales del servicio de intercambio de tokens para enumerar los buckets de Amazon S3.

> **Note**
>
> Para ejecutar este componente de ejemplo, el dispositivo debe tener el permiso. `s3:ListAllMyBuckets` Para obtener más información, consulte Autorizar a los dispositivos principales a interactuar con AWS los servicios.

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
```

Este componente de ejemplo ejecuta el siguiente script de Python, `list_s3_buckets.py` que muestra una lista de los buckets de 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)
```

Ejecute un contenedor Docker

Puede configurar AWS IoT Greengrass los componentes para que ejecuten un contenedor Docker a partir de imágenes almacenadas en las siguientes ubicaciones:

- Repositorios de imágenes públicos y privados en Amazon Elastic Container Registry (Amazon ECR)
- Repositorio público de Docker Hub
- Registro público de confianza de Docker
- Bucket de S3

En su componente personalizado, incluya el URI de la imagen de Docker como un artefacto para recuperar la imagen y ejecutarla en el dispositivo principal. En el caso de las imágenes de Amazon ECR y Docker Hub, puede utilizar el componente gestor de aplicaciones de Docker para descargar las imágenes y gestionar las credenciales de los repositorios privados de Amazon ECR.

Temas

Requisitos

Para ejecutar un contenedor de Docker en un componente, necesita lo siguiente:

- Un dispositivo central de Greengrass. Si no dispone de una, consulte Tutorial: Introducción a AWS IoT Greengrass V2.

- Docker Engine 1.9.1 o posterior instalado en el dispositivo principal de Greengrass. Se ha comprobado que la versión 20.10 es la última versión que funciona con el software Core. AWS IoT Greengrass Debe instalar Docker directamente en el dispositivo principal antes de implementar los componentes que ejecutan contenedores de Docker.

  > **Tip**
  >
  > También puede configurar el dispositivo principal para instalar Docker Engine cuando se instale el componente. Por ejemplo, el siguiente script de instalación instala Docker Engine antes de cargar la imagen de Docker. Este script de instalación funciona en distribuciones de Linux basadas en Debian, como Ubuntu. Si configura el componente para instalar Docker Engine con este comando, puede que tenga que configurarlo `true` en el script del ciclo de vida `RequiresPrivilege` para ejecutar la instalación y los comandos. `docker` Para obtener más información, consulte AWS IoT Greengrass referencia de recetas de componentes.
  >
  > ```
  > apt-get install docker-ce docker-ce-cli containerd.io && docker load -i
  >  {artifacts:path}/hello-world.tar
  > ```

- El usuario del sistema que ejecute un componente contenedor de Docker debe tener permisos de raíz o administrador, o bien debe configurar Docker para que se ejecute como un usuario no de raíz o no administrador.
  - En los dispositivos Linux, puede añadir un usuario al `docker` grupo sin necesidad `sudo` de invocar `docker` comandos.
  - En los dispositivos Windows, puede añadir un usuario al `docker-users` grupo para que invoque `docker` comandos sin privilegios de administrador.

  Linux or Unix

  Para añadir `ggc_user` al `docker` grupo el usuario no root que utiliza para ejecutar los componentes del contenedor de Docker, ejecute el siguiente comando.

  ```
  sudo usermod -aG docker ggc_user
  ```

  Para obtener más información, consulta Administrar Docker como usuario no root.

Windows Command Prompt (CMD)

Para añadir `ggc_user` al `docker-users` grupo o el usuario que utiliza para ejecutar los componentes del contenedor de Docker, ejecute el siguiente comando como administrador.

```
net localgroup docker-users ggc_user /add
```

Windows PowerShell

Para añadir `ggc_user` al `docker-users` grupo o al usuario que utiliza para ejecutar los componentes del contenedor de Docker, ejecute el siguiente comando como administrador.

```
Add-LocalGroupMember -Group docker-users -Member ggc_user
```

- Los archivos a los que accede el componente contenedor de Docker están montados como un volumen en el contenedor de Docker.
- Si configura el software AWS IoT Greengrass Core para usar un proxy de red, debe configurar Docker para que use el mismo servidor proxy.

Además de estos requisitos, también debe cumplir los siguientes requisitos si se aplican a su entorno:

- Para usar Docker Compose para crear e iniciar sus contenedores de Docker, instale Docker Compose en su dispositivo principal de Greengrass y cargue el archivo de Docker Compose en un bucket de S3. Debe almacenar el archivo de Compose en un depósito de S3 en el mismo Cuenta de AWS componente y como él. Región de AWS Para ver un ejemplo en el que se usa el `docker-compose up` comando en un componente personalizado, consulteEjecute un contenedor de Docker desde una imagen pública en Amazon ECR o Docker Hub.
- Si utiliza un proxy AWS IoT Greengrass de red, configure el daemon de Docker para que utilice un servidor proxy.
- Si sus imágenes de Docker están almacenadas en Amazon ECR o Docker Hub, incluya el componente administrador de componentes de Docker como una dependencia en su componente de contenedor de Docker. Debe iniciar el daemon de Docker en el dispositivo principal antes de implementar el componente.

Además, incluya los URI de la imagen como artefactos de los componentes. Los URI de imagen deben tener el formato que `docker:registry/image[:tag|@digest]` se muestra en los siguientes ejemplos:

- Imagen privada de Amazon ECR: `docker:account-id.dkr.ecr.region.amazonaws.com/repository/image[:tag|@digest]`
- Imagen pública de Amazon ECR: `docker:public.ecr.aws/repository/image[:tag|@digest]`
- Imagen pública de Docker Hub: `docker:name[:tag|@digest]`

Para obtener más información sobre cómo ejecutar contenedores Docker a partir de imágenes almacenadas en repositorios públicos, consulte. Ejecute un contenedor de Docker desde una imagen pública en Amazon ECR o Docker Hub

- Si sus imágenes de Docker están almacenadas en un repositorio privado de Amazon ECR, debe incluir el componente del servicio de intercambio de tokens como una dependencia en el componente de contenedor de Docker. Además, el rol de dispositivo de Greengrass debe permitir las `ecr:GetDownloadUrlForLayer` acciones, y `ecr:GetAuthorizationTokenecr:BatchGetImage`, como se muestra en el siguiente ejemplo de política de IAM.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "ecr:GetAuthorizationToken",
        "ecr:BatchGetImage",
        "ecr:GetDownloadUrlForLayer"
      ],
      "Resource": [
        "*"
      ],
      "Effect": "Allow"
    }
  ]
}
```

Para obtener información sobre la ejecución de contenedores Docker a partir de imágenes almacenadas en un repositorio privado de Amazon ECR, consulte. Ejecute un contenedor de Docker desde una imagen privada en Amazon ECR

- Para usar imágenes de Docker almacenadas en un repositorio privado de Amazon ECR, el repositorio privado debe estar en el Región de AWS mismo lugar que el dispositivo principal.
- Si las imágenes de Docker o los archivos de Compose están almacenados en un bucket de S3, el rol de dispositivo de Greengrass debe conceder `s3:GetObject` el permiso para que los dispositivos principales descarguen las imágenes como artefactos componentes, como se muestra en el siguiente ejemplo de política de IAM.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "s3:GetObject"
      ],
      "Resource": [
        "*"
      ],
      "Effect": "Allow"
    }
  ]
}
```

Para obtener información sobre la ejecución de contenedores Docker a partir de imágenes almacenadas en Amazon S3, consulteEjecute un contenedor de Docker a partir de una imagen en Amazon S3.

- Para usar la comunicación entre procesos (IPC), AWS las credenciales o el administrador de transmisiones en el componente de contenedor de Docker, debe especificar opciones adicionales al ejecutar el contenedor de Docker. Para más información, consulte los siguientes temas:
  - Utilice la comunicación entre procesos en los componentes del contenedor de Docker
  - Utilice AWS las credenciales en los componentes del contenedor de Docker (Linux)
  - Utilice el administrador de flujos en los componentes del contenedor de Docker (Linux)

Ejecute un contenedor de Docker desde una imagen pública en Amazon ECR o Docker Hub

En esta sección, se describe cómo crear un componente personalizado que utilice Docker Compose para ejecutar un contenedor de Docker a partir de imágenes de Docker almacenadas en Amazon ECR y Docker Hub.

Para ejecutar un contenedor de Docker mediante Docker Compose

1. Cree y cargue un archivo de Docker Compose en un bucket de Amazon S3. Asegúrese de que el rol de dispositivo de Greengrass otorgue el `s3:GetObject` permiso para permitir que el dispositivo acceda al archivo de composición. El archivo Compose de ejemplo que se muestra en el siguiente ejemplo incluye la imagen de Amazon CloudWatch Agent de Amazon ECR y la imagen de MySQL de Docker Hub.

   ```
   version: "3"
   services:
     cloudwatchagent:
       image: "public.ecr.aws/cloudwatch-agent/cloudwatch-agent:latest"
     mysql:
       image: "mysql:8.0"
   ```

2. Cree un componente personalizado en su dispositivo AWS IoT Greengrass principal. La receta de ejemplo que se muestra en el siguiente ejemplo tiene las siguientes propiedades:

   - El componente del administrador de aplicaciones de Docker como dependencia. Este componente permite AWS IoT Greengrass descargar imágenes de los repositorios públicos de Amazon ECR y Docker Hub.
   - Un artefacto componente que especifica una imagen de Docker en un repositorio público de Amazon ECR.
   - Un artefacto componente que especifica una imagen de Docker en un repositorio público de Docker Hub.
   - Un artefacto componente que especifica el archivo de Docker Compose que incluye los contenedores para las imágenes de Docker que quieres ejecutar.
   - Un script de ejecución de ciclo de vida que usa docker-compose up para crear e iniciar un contenedor a partir de las imágenes especificadas.

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**
>
> Para usar la comunicación entre procesos (IPC), AWS las credenciales o el administrador de flujos en el componente de contenedor de Docker, debe especificar opciones adicionales al ejecutar el contenedor de Docker. Para más información, consulte los siguientes temas:
>
> - Utilice la comunicación entre procesos en los componentes del contenedor de Docker
> - Utilice AWS las credenciales en los componentes del contenedor de Docker (Linux)
> - Utilice el administrador de flujos en los componentes del contenedor de Docker (Linux)

3. Pruebe el componente para comprobar que funciona según lo esperado.

> **Important**
>
> Debe instalar e iniciar el daemon de Docker antes de implementar el componente.

Tras implementar el componente localmente, puede ejecutar el comando docker container ls para comprobar que el contenedor se ejecuta.

```
docker container ls
```

4. Cuando el componente esté listo, cárguelo para AWS IoT Greengrass implementarlo en otros dispositivos principales. Para obtener más información, consulte Publique componentes para desplegarlos en sus dispositivos principales.

Ejecute un contenedor de Docker desde una imagen privada en Amazon ECR

En esta sección se describe cómo puede crear un componente personalizado que ejecute un contenedor de Docker a partir de una imagen de Docker almacenada en un repositorio privado de Amazon ECR.

Para ejecutar un contenedor de Docker

1. Cree un componente personalizado en su dispositivo AWS IoT Greengrass principal. Utilice la siguiente receta de ejemplo, que tiene las siguientes propiedades:

   - El componente del administrador de aplicaciones de Docker como dependencia. Este componente permite AWS IoT Greengrass administrar las credenciales para descargar imágenes de repositorios privados.
   - El componente del servicio de intercambio de fichas como dependencia. Este componente permite AWS IoT Greengrass recuperar AWS credenciales para interactuar con Amazon ECR.
   - Un artefacto componente que especifica una imagen de Docker en un repositorio privado de Amazon ECR.
   - Un script de ejecución del ciclo de vida que usa docker run para crear e iniciar un contenedor a partir de la imagen.

   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**
>
> Para usar la comunicación entre procesos (IPC), AWS las credenciales o el administrador de flujos en el componente de contenedor de Docker, debe especificar opciones adicionales al ejecutar el contenedor de Docker. Para más información, consulte los siguientes temas:
>
> - Utilice la comunicación entre procesos en los componentes del contenedor de Docker
> - Utilice AWS las credenciales en los componentes del contenedor de Docker (Linux)
> - Utilice el administrador de flujos en los componentes del contenedor de Docker (Linux)

2. Pruebe el componente para comprobar que funciona según lo esperado.

   > **Important**
   >
   > Debe instalar e iniciar el daemon de Docker antes de implementar el componente.

   Tras implementar el componente localmente, puede ejecutar el comando docker container ls para comprobar que el contenedor se ejecuta.

   ```
   docker container ls
   ```

3. Cargue el componente AWS IoT Greengrass para implementarlo en otros dispositivos principales. Para obtener más información, consulte Publique componentes para desplegarlos en sus dispositivos principales.

Ejecute un contenedor de Docker a partir de una imagen en Amazon S3

En esta sección, se describe cómo ejecutar un contenedor de Docker en un componente desde una imagen de Docker almacenada en Amazon S3.

Para ejecutar un contenedor de Docker en un componente desde una imagen en Amazon S3

1. Ejecute el comando docker save para crear una copia de seguridad de un contenedor Docker. Esta copia de seguridad se proporciona como un artefacto componente en el que ejecutar el contenedor. AWS IoT Greengrass Sustituya *hello-world* por el nombre de la imagen y sustituya *hello-world.tar* por el nombre del archivo comprimido que desee crear.

   ```
   docker save hello-world > artifacts/com.example.MyDockerComponent/1.0.0/hello-world.tar
   ```

2. Cree un componente personalizado en su dispositivo AWS IoT Greengrass principal. Utilice la siguiente receta de ejemplo, que tiene las siguientes propiedades:

   - Un script de instalación del ciclo de vida que usa Docker Load para cargar una imagen de Docker desde un archivo.
   - Un script de ejecución del ciclo de vida que usa docker run para crear e iniciar un contenedor a partir de la imagen. La `--rm` opción limpia el contenedor al salir.

   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**
>
> Para usar la comunicación entre procesos (IPC), AWS las credenciales o el administrador de flujos en el componente de contenedor de Docker, debe especificar opciones adicionales al ejecutar el contenedor de Docker. Para más información, consulte los siguientes temas:
>
> - Utilice la comunicación entre procesos en los componentes del contenedor de Docker
> - Utilice AWS las credenciales en los componentes del contenedor de Docker (Linux)
> - Utilice el administrador de flujos en los componentes del contenedor de Docker (Linux)

3. Pruebe el componente para comprobar que funciona según lo esperado.

   Tras implementar el componente localmente, puede ejecutar el comando docker container ls para comprobar que el contenedor se ejecuta.

   ```
   docker container ls
   ```

4. Cuando el componente esté listo, sube el archivo de imágenes de Docker a un bucket de S3 y añade su URI a la receta del componente. A continuación, puede cargar el componente para AWS IoT Greengrass implementarlo en otros dispositivos principales. Para obtener más información, consulte Publique componentes para desplegarlos en sus dispositivos principales.

   Cuando termines, la receta del componente debería tener el aspecto que se muestra en el siguiente ejemplo.

   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
```

Utilice la comunicación entre procesos en los componentes del contenedor de Docker

Puede utilizar la biblioteca de comunicación entre procesos (IPC) de Greengrass SDK para dispositivos con AWS IoT para comunicarse con el núcleo de Greengrass, otros componentes de Greengrass y. AWS IoT Core Para obtener más información, consulte Úselo SDK para dispositivos con AWS IoT para comunicarse con el núcleo de Greengrass, otros componentes y AWS IoT Core.

Para usar el IPC en un componente de contenedor de Docker, debe ejecutar el contenedor de Docker con los siguientes parámetros:

- Monte el socket IPC en el contenedor. El núcleo de Greengrass proporciona la ruta del archivo del socket IPC en la `AWS_GG_NUCLEUS_DOMAIN_SOCKET_FILEPATH_FOR_COMPONENT` variable de entorno.
- Establezca las variables `SVCUID` y de `AWS_GG_NUCLEUS_DOMAIN_SOCKET_FILEPATH_FOR_COMPONENT` entorno en los valores que el núcleo de Greengrass proporciona a los componentes. Su componente utiliza estas variables de entorno para autenticar las conexiones al núcleo de Greengrass.

Example Receta de ejemplo: publicar un mensaje MQTT en AWS IoT Core (Python)

La siguiente receta define un ejemplo de componente contenedor de Docker en el que se publica un mensaje MQTT. AWS IoT Core Esta receta tiene las siguientes propiedades:

- Una política de autorización (`accessControl`) que permite al componente publicar mensajes MQTT AWS IoT Core sobre todos los temas. Para obtener más información, consulte Autorización de Autorice a los componentes a realizar operaciones de IPC IPC en AWS IoT Core MQTT.
- Un artefacto componente que especifica una imagen de Docker como un archivo TAR en Amazon S3.
- Un script de instalación del ciclo de vida que carga la imagen de Docker desde el archivo TAR.
- Un script de ejecución del ciclo de vida que ejecuta un contenedor Docker desde la imagen. El comando Docker run tiene los siguientes argumentos:
  - El `-v` argumento monta el conector IPC de Greengrass en el contenedor.
  - Los dos primeros `-e` argumentos establecen las variables de entorno necesarias en el contenedor Docker.
  - Los `-e` argumentos adicionales establecen las variables de entorno utilizadas en este ejemplo.
  - El `--rm` argumento limpia el contenedor al salir.

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
```

Utilice AWS las credenciales en los componentes del contenedor de Docker (Linux)

Puede usar el componente del servicio de intercambio de fichas para interactuar con AWS los servicios de los componentes de Greengrass. Este componente proporciona AWS las credenciales de la función de intercambio de fichas del dispositivo principal mediante un servidor contenedor local. Para obtener más información, consulte Interactúa con AWS los servicios.

> **Note**
>
> El ejemplo de esta sección solo funciona en los dispositivos principales de Linux.

Para usar AWS las credenciales del servicio de intercambio de fichas en un componente de contenedor de Docker, debe ejecutar el contenedor de Docker con los siguientes parámetros:

- Proporcione acceso a la red host mediante el `--network=host` argumento. Esta opción permite que el contenedor de Docker se conecte al servicio de intercambio de tokens local para recuperar AWS las credenciales. Este argumento solo funciona en Docker para Linux.

  > **Warning**
  >
  > Esta opción proporciona al contenedor acceso a todas las interfaces de red locales del host, por lo que es menos segura que si se ejecutan contenedores de Docker sin este acceso a la red del host. Tenga esto en cuenta al desarrollar y ejecutar componentes de contenedores de Docker que utilizan esta opción. Para obtener más información, consulte Network: host en la documentación de Docker.

- Establezca las variables `AWS_CONTAINER_CREDENTIALS_FULL_URI` y de `AWS_CONTAINER_AUTHORIZATION_TOKEN` entorno en los valores que el núcleo de Greengrass proporciona a los componentes. AWS Los SDK utilizan estas variables de entorno para recuperar AWS las credenciales.

Example Receta de ejemplo: Listar buckets de S3 en un componente contenedor de Docker (Python)

La siguiente receta define un ejemplo de componente de contenedor de Docker que muestra los buckets de S3 de su. Cuenta de AWS Esta receta tiene las siguientes propiedades:

- El componente del servicio de intercambio de fichas como dependencia. Esta dependencia permite al componente recuperar AWS credenciales para interactuar con otros AWS servicios.
- Un artefacto componente que especifica una imagen de Docker como un archivo tar en Amazon S3.
- Un script de instalación del ciclo de vida que carga la imagen de Docker desde el archivo TAR.
- Un script de ejecución del ciclo de vida que ejecuta un contenedor Docker desde la imagen. El comando Docker run tiene los siguientes argumentos:
  - El `--network=host` argumento proporciona al contenedor acceso a la red host, de modo que el contenedor puede conectarse al servicio de intercambio de fichas.
  - El `-e` argumento establece las variables de entorno necesarias en el contenedor de Docker.
  - El `--rm` argumento limpia el contenedor al salir.

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
```

Utilice el administrador de flujos en los componentes del contenedor de Docker (Linux)

Puede usar el componente administrador de flujos para administrar flujos de datos en los componentes de Greengrass. Este componente le permite procesar flujos de datos y transferir datos de IoT de gran volumen al Nube de AWS. AWS IoT Greengrass proporciona un SDK de administrador de transmisiones que se utiliza para interactuar con el componente de administrador de transmisiones. Para obtener más información, consulte Gestione los flujos de datos en los dispositivos principales de Greengrass.

> **Note**
>
> El ejemplo de esta sección solo funciona en los dispositivos principales de Linux.

Para usar el SDK del administrador de transmisiones en un componente de contenedor de Docker, debe ejecutar el contenedor de Docker con los siguientes parámetros:

- Proporcione acceso a la red host mediante el `--network=host` argumento. Esta opción permite que el contenedor Docker interactúe con el componente del administrador de transmisiones a través de una conexión TLS local. Este argumento solo funciona en Docker para Linux

> **Warning**
>
> Esta opción proporciona al contenedor acceso a todas las interfaces de red locales del host, por lo que es menos segura que si se ejecutan contenedores de Docker sin este acceso a la red del host. Tenga esto en cuenta al desarrollar y ejecutar componentes de contenedores de Docker que utilizan esta opción. Para obtener más información, consulte Network: host en la documentación de Docker.

- Si configura el componente del administrador de transmisiones para que requiera autenticación, que es el comportamiento predeterminado, establezca la variable de `AWS_CONTAINER_CREDENTIALS_FULL_URI` entorno en el valor que el núcleo de Greengrass proporciona a los componentes. Para obtener más información, consulte la configuración del administrador de transmisiones.
- Si configura el componente del administrador de flujos para que utilice un puerto que no sea el predeterminado, utilice la comunicación entre procesos (IPC) para obtener el puerto desde la configuración del componente del administrador de flujos. Debe ejecutar el contenedor Docker con opciones adicionales para usar el IPC. Para más información, consulte los siguientes temas:
  - Conéctese al administrador de transmisiones en el código de la aplicación
  - Utilice la comunicación entre procesos en los componentes del contenedor de Docker

Example Receta de ejemplo: Transmita un archivo a un bucket de S3 en un componente contenedor de Docker (Python)

La siguiente receta define un ejemplo de componente contenedor de Docker que crea un archivo y lo transmite a un bucket de S3. Esta receta tiene las siguientes propiedades:

- El componente del administrador de flujos como dependencia. Esta dependencia permite que el componente utilice el SDK del administrador de transmisiones para interactuar con el componente de administrador de transmisiones.
- Un artefacto componente que especifica una imagen de Docker como un archivo TAR en Amazon S3.
- Un script de instalación del ciclo de vida que carga la imagen de Docker desde el archivo TAR.
- Un script de ejecución del ciclo de vida que ejecuta un contenedor Docker desde la imagen. El comando Docker run tiene los siguientes argumentos:

- El `--network=host` argumento proporciona al contenedor acceso a la red host, de modo que el contenedor puede conectarse al componente del administrador de flujos.
- El primer `-e` argumento establece la variable de `AWS_CONTAINER_AUTHORIZATION_TOKEN` entorno requerida en el contenedor de Docker.
- Los `-e` argumentos adicionales establecen las variables de entorno utilizadas en este ejemplo.
- El `-v` argumento monta la carpeta de trabajo del componente en el contenedor. En este ejemplo, se crea un archivo en la carpeta de trabajo para subirlo a Amazon S3 mediante el administrador de transmisiones.
- El `--rm` argumento limpia el contenedor al salir.

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 referencia de recetas de componentes

La receta del componente es un archivo que define los detalles, las dependencias, los artefactos y los ciclos de vida de un componente. El ciclo de vida del componente especifica los comandos que se deben ejecutar para instalar, ejecutar y apagar el componente, por ejemplo. El AWS IoT Greengrass núcleo utiliza los ciclos de vida que se definen en la receta para instalar y ejecutar los componentes. El AWS IoT Greengrass servicio utiliza la receta para identificar las dependencias y los artefactos que se van a implementar en los dispositivos principales al implementar el componente.

En la receta, puede definir dependencias y ciclos de vida únicos para cada plataforma compatible con un componente. Puede utilizar esta capacidad para implementar un componente en dispositivos con varias plataformas que tengan requisitos diferentes. También puede usarla para AWS IoT Greengrass evitar que se instale un componente en dispositivos que no lo admiten.

Cada receta contiene una lista de manifiestos. Cada manifiesto especifica un conjunto de requisitos de plataforma, así como el ciclo de vida y los artefactos que se utilizarán en los dispositivos principales cuya plataforma cumpla con esos requisitos. El dispositivo principal usa el primer manifiesto con los requisitos de plataforma que cumple el dispositivo. Especifique un manifiesto sin ningún requisito de plataforma que coincida con ningún dispositivo principal.

También puedes especificar un ciclo de vida global que no esté en un manifiesto. En el ciclo de vida global, puedes usar claves de selección que identifiquen las subsecciones del ciclo de vida. Luego, puedes especificar estas claves de selección en un manifiesto para usar esas secciones del ciclo de vida global además del ciclo de vida del manifiesto. El dispositivo principal usa las claves de selección del manifiesto solo si el manifiesto no define un ciclo de vida. Puedes usar la `all` selección de un manifiesto para hacer coincidir las secciones del ciclo de vida global sin claves de selección.

Una vez que el software AWS IoT Greengrass principal selecciona un manifiesto que coincide con el dispositivo principal, hace lo siguiente para identificar los pasos del ciclo de vida que se deben seguir:

- Si el manifiesto seleccionado define un ciclo de vida, el dispositivo principal utiliza ese ciclo de vida.
- Si el manifiesto seleccionado no define un ciclo de vida, el dispositivo principal utiliza el ciclo de vida global. El dispositivo principal hace lo siguiente para identificar qué secciones del ciclo de vida global debe utilizar:

- Si el manifiesto define las claves de selección, el dispositivo principal utiliza las secciones del ciclo de vida global que contienen las claves de selección del manifiesto.
- Si el manifiesto no define las claves de selección, el dispositivo principal utiliza las secciones del ciclo de vida global que no tienen claves de selección. Este comportamiento equivale a un manifiesto que define la `all` selección.

> **Important**
>
> Un dispositivo principal debe cumplir como mínimo los requisitos de plataforma de un manifiesto para instalar el componente. Si ningún manifiesto coincide con el dispositivo principal, el software AWS IoT Greengrass principal no instala el componente y se produce un error en la implementación.

Puede definir recetas en formato JSON o YAML. La sección de ejemplos de recetas incluye recetas en cada formato.

Temas

- Validación de recetas
- Formato de receta
- Variables de receta
- Ejemplos de recetas

Validación de recetas

Greengrass valida la receta de un componente JSON o YAML al crear una versión del componente. Esta validación de recetas comprueba la receta de sus componentes JSON o YAML para detectar errores comunes a fin de evitar posibles problemas de implementación. La validación comprueba la receta para detectar errores comunes (p. ej., falta de comas, corchetes y campos) y se asegura de que la receta esté bien formada.

Si recibes un mensaje de error al validar una receta, comprueba si en ella faltan comas, corchetes o campos. Comprueba que no te falta ningún campo consultando el formato de la receta.

Formato de receta

Al definir una receta para un componente, se especifica la siguiente información en el documento de receta. La misma estructura se aplica a las recetas en los formatos YAML y JSON.

`RecipeFormatVersion`

La versión de plantilla de la receta. Elija la opción siguiente:

- `2020-01-25`

`ComponentName`

El nombre del componente que define esta receta. El nombre del componente debe ser único Cuenta de AWS en cada región.

> **Consejos**
>
> - Utilice el formato de nombre de dominio inverso para evitar colisiones de nombres dentro de su empresa. Por ejemplo, si su empresa es propietaria de un proyecto de energía solar `example.com` y usted trabaja en él, puede ponerle un nombre a su componente `com.example.solar.HelloWorld` Hello World. Esto ayuda a evitar colisiones entre los nombres de los componentes dentro de su empresa.
> - Evite el `aws.greengrass` prefijo en los nombres de sus componentes. AWS IoT Greengrass usa este prefijo para los componentes públicos que proporciona. Si elige el mismo nombre que un componente público, el componente reemplaza a ese componente. A continuación, AWS IoT Greengrass proporciona su componente en lugar del componente público cuando despliega componentes que dependen de ese componente público. Esta función le permite anular el comportamiento de los componentes públicos, pero también puede interrumpir otros componentes si no tiene intención de anular un componente público.

`ComponentVersion`

Esta es la versión del componente. El valor máximo para los valores principales, secundarios y de parche es 999999.

> **Note**
>
> AWS IoT Greengrass utiliza versiones semánticas para los componentes. Las versiones semánticas siguen un sistema de números de principal.secundario.parche. Por ejemplo, la versión `1.0.0` representa la primera versión principal de un componente. Para obtener más información, consulte la especificación de la versión semántica.

`ComponentDescription`

(Opcional) La descripción del componente.

`ComponentPublisher`

El publicador o el autor del componente.

`ComponentConfiguration`

(Opcional) Un objeto que define la configuración o los parámetros del componente. Define la configuración predeterminada y, a continuación, cuando despliega el componente, puede especificar el objeto de configuración que se va a proporcionar al componente. La configuración de los componentes admite parámetros y estructuras anidados. Este objeto contiene la siguiente información:

`DefaultConfiguration`

Objeto que define la configuración por defecto del componente. Usted define la estructura de este objeto.

> **Note**
>
> AWS IoT Greengrass usa JSON para los valores de configuración. JSON especifica un tipo de número, pero no diferencia entre números enteros y flotantes. Como resultado, los valores de configuración pueden convertirse en valores flotantes. AWS IoT Greengrass Para garantizar que su componente utilice el tipo de datos correcto, le recomendamos que defina los valores de configuración numéricos como cadenas. A continuación, pida a su componente que los analice como enteros o flotantes. Esto garantiza que los valores de configuración sean del mismo tipo en la configuración y en el dispositivo principal.

`ComponentDependencies`

(Opcional) Un diccionario de objetos, cada uno de los cuales define una dependencia de componente para el componente. La clave de cada objeto identifica el nombre de la dependencia del componente. AWS IoT Greengrass instala las dependencias de los componentes cuando se instala el componente. AWS IoT Greengrass espera a que se inicien las dependencias antes de iniciar el componente. Cada objeto contiene la siguiente información:

`VersionRequirement`

La restricción de versión semántica de estilo npm que define las versiones de los componentes compatibles para esta dependencia. Puede especificar una versión o un rango de versiones. Para obtener más información, consulte la calculadora de versiones semánticas de npm.

`DependencyType`

(Opcional) El tipo de esta dependencia. Puede elegir entre las siguientes opciones.

- `SOFT` — El componente no se reinicia si la dependencia cambia de estado.
- `HARD` — El componente se reinicia si la dependencia cambia de estado.

El valor predeterminado es `HARD`.

`ComponentType`

(Opcional) El tipo de componente.

> **Note**
>
> No se recomienda especificar el tipo de componente en una receta. AWS IoT Greengrass establece el tipo automáticamente al crear un componente.

El tipo puede ser de los siguientes tipos:

- `aws.greengrass.generic`— El componente ejecuta comandos o proporciona artefactos.
- `aws.greengrass.lambda`— El componente ejecuta una función Lambda mediante el componente Lambda launcher. El `ComponentSource` parámetro especifica el ARN de la función Lambda que ejecuta este componente.

No se recomienda utilizar esta opción, ya que se establece AWS IoT Greengrass al crear un componente a partir de una función Lambda. Para obtener más información, consulte AWS LambdaFunciones de ejecución.

- `aws.greengrass.plugin`— El componente se ejecuta en la misma máquina virtual Java (JVM) que el núcleo de Greengrass. Si despliega o reinicia un componente del complemento, el núcleo de Greengrass se reinicia.

  Los componentes del plugin utilizan el mismo archivo de registro que el núcleo de Greengrass. Para obtener más información, consulte Supervisar AWS IoT Greengrass registros.

  No se recomienda utilizar esta opción en las recetas de componentes, ya que está AWS pensada para componentes proporcionados y escritos en Java que interactúan directamente con el núcleo de Greengrass. Para obtener más información sobre qué componentes públicos son complementos, consulte. AWS-componentes proporcionados
- `aws.greengrass.nucleus`— El componente del núcleo. Para obtener más información, consulte Núcleo de Greengrass.

  No se recomienda utilizar esta opción en las recetas de componentes. Está diseñado para el componente núcleo de Greengrass, que proporciona la funcionalidad mínima del software AWS IoT Greengrass Core.

El valor predeterminado es `aws.greengrass.generic` cuando se crea un componente a partir de una receta o `aws.greengrass.lambda` cuando se crea un componente a partir de una función Lambda.

Para obtener más información, consulte Tipos de componentes.

`ComponentSource`

(Opcional) El ARN de la función Lambda que ejecuta un componente.

No se recomienda especificar la fuente del componente en una receta. AWS IoT Greengrass establece este parámetro cuando crea un componente a partir de una función Lambda. Para obtener más información, consulte AWS LambdaFunciones de ejecución.

`Manifests`

Una lista de objetos, cada uno de los cuales define el ciclo de vida, los parámetros y los requisitos del componente para una plataforma. Si un dispositivo principal cumple con los requisitos de plataforma de varios manifiestos, AWS IoT Greengrass utiliza el primer manifiesto que coincida

con el dispositivo principal. Para garantizar que los dispositivos principales utilicen el manifiesto correcto, defina primero los manifiestos con requisitos de plataforma más estrictos. Un manifiesto que se aplique a todas las plataformas debe ser el último manifiesto de la lista.

> **Important**
>
> Un dispositivo principal debe cumplir como mínimo los requisitos de plataforma de un manifiesto para instalar el componente. Si ningún manifiesto coincide con el dispositivo principal, el software AWS IoT Greengrass principal no instala el componente y se produce un error en la implementación.

Cada objeto contiene la siguiente información:

`Name`

(Opcional) Un nombre descriptivo para la plataforma que define este manifiesto.

Si omites este parámetro, AWS IoT Greengrass crea un nombre a partir de la plataforma `os` `yarchitecture`.

`Platform`

(Opcional) Un objeto que define la plataforma a la que se aplica este manifiesto. Omita este parámetro para definir un manifiesto que se aplique a todas las plataformas.

Este objeto especifica los pares clave-valor sobre la plataforma en la que se ejecuta un dispositivo principal. Al implementar este componente, el software AWS IoT Greengrass Core compara estos pares clave-valor con los atributos de la plataforma en el dispositivo principal. El software AWS IoT Greengrass principal siempre define `os` `yarchitecture`, y puede definir, atributos adicionales. Puede especificar atributos de plataforma personalizados para un dispositivo principal al implementar el componente núcleo de Greengrass. Para obtener más información, consulte el parámetro de anulación de plataforma del componente núcleo de Greengrass.

Para cada par clave-valor, puede especificar uno de los siguientes valores:

- Un valor exacto, como `linux` o. `windows` Los valores exactos deben empezar por una letra o un número.
- `*`, que coincide con cualquier valor. Esto también coincide cuando un valor no está presente.

- Una expresión regular de estilo Java, como. `/windows|linux/` La expresión regular debe empezar y terminar con un carácter de barra (). `/` Por ejemplo, la expresión regular `/.+/` coincide con cualquier valor que no esté en blanco.

Este objeto contiene la siguiente información:

`os`

(Opcional) El nombre del sistema operativo de la plataforma compatible con este manifiesto. Las plataformas más comunes incluyen los siguientes valores:

- `linux`
- `windows`
- `darwin` (macOS)

`architecture`

(Opcional) La arquitectura del procesador de la plataforma compatible con este manifiesto. Las arquitecturas comunes incluyen los siguientes valores:

- `amd64`
- `arm`
- `aarch64`
- `x86`

`architecture.detail`

(Opcional) El detalle de la arquitectura del procesador de la plataforma compatible con este manifiesto. Los detalles comunes de la arquitectura incluyen los siguientes valores:

- `arm61`
- `arm71`
- `arm81`

`key`

(Opcional) Un atributo de plataforma que definas para este manifiesto. Sustituya la *`clave`* por el nombre del atributo de la plataforma. El software AWS IoT Greengrass Core hace coincidir este atributo de plataforma con los pares clave-valor que especifique en la configuración del componente núcleo de Greengrass. Para obtener más información, consulte el parámetro de anulación de plataforma del componente núcleo de Greengrass.

> **Tip**
>
> Utilice el formato de nombre de dominio inverso para evitar colisiones de nombres dentro de su empresa. Por ejemplo, si su empresa es propietaria de un proyecto de radio `example.com` y usted trabaja en él, puede asignar un nombre a un atributo `com.example.radio.RadioModule` de plataforma personalizado. Esto ayuda a evitar colisiones entre los nombres de los atributos de la plataforma en su empresa.

Por ejemplo, puede definir un atributo de plataforma,`com.example.radio.RadioModule`, para especificar un manifiesto diferente en función del módulo de radio que esté disponible en un dispositivo principal. Cada manifiesto puede incluir diferentes artefactos que se apliquen a distintas configuraciones de hardware, de modo que se pueda implementar el conjunto mínimo de software en el dispositivo principal.

`Lifecycle`

Un objeto o cadena que define cómo instalar y ejecutar el componente en la plataforma que define este manifiesto. También puede definir un ciclo de vida global que se aplique a todas las plataformas. El dispositivo principal utiliza el ciclo de vida global solo si el manifiesto que se va a utilizar no especifica un ciclo de vida.

> **Note**
>
> Este ciclo de vida se define en un manifiesto. Los pasos del ciclo de vida que especifique aquí se aplican únicamente a la plataforma que define este manifiesto. También puede definir un ciclo de vida global que se aplique a todas las plataformas.

Este objeto o cadena contiene la siguiente información:

`Setenv`

(Opcional) Un diccionario de variables de entorno para proporcionarlo a todos los scripts del ciclo de vida. Puede anular estas variables de entorno `Setenv` en cada script de ciclo de vida.

`install`

(Opcional) Un objeto o una cadena que define el script que se ejecutará cuando se instale el componente. El software AWS IoT Greengrass principal también ejecuta este paso del ciclo de vida cada vez que se inicia el software.

Si el `install` script sale con un código de éxito, el componente entra en ese `INSTALLED` estado.

Este objeto o cadena contiene la siguiente información:

`Script`

El script que se va a ejecutar.

`RequiresPrivilege`

(Opcional) Puede ejecutar el script con privilegios de root. Si establece esta opción en`true`, el software AWS IoT Greengrass principal ejecutará este script de ciclo de vida como root en lugar de como el usuario del sistema que usted configure para ejecutar este componente. El valor predeterminado es `false`.

`Skipif`

(Opcional) La comprobación para determinar si se debe ejecutar o no el script. Puede definirlo para comprobar si hay un ejecutable en la ruta o si existe un archivo. Si el resultado es verdadero, el software AWS IoT Greengrass Core omite el paso. Seleccione una de las siguientes comprobaciones:

- `onpath` *`runnable`*— Compruebe si hay un ejecutable en la ruta del sistema. Por ejemplo, úselo **`onpath python3`** para omitir este paso del ciclo de vida si Python 3 está disponible.
- `exists` *`file`*— Comprueba si existe un archivo. Por ejemplo, úselo **`exists /tmp/my-configuration.db`** para omitir este paso del ciclo de vida si `/tmp/my-configuration.db` está presente.

`Timeout`

(Opcional) El tiempo máximo en segundos que puede ejecutarse el script antes de que el software AWS IoT Greengrass principal finalice el proceso.

Predeterminado: 120 segundos

`Setenv`

(Opcional) El diccionario de variables de entorno que se proporcionará al script. Estas variables de entorno anulan las variables que usted proporciona. `Lifecycle.Setenv`

`run`

(Opcional) Un objeto o una cadena que define el script que se ejecutará cuando se inicie el componente.

El componente entra en ese `RUNNING` estado cuando se ejecuta este paso del ciclo de vida. Si el `run` script sale con un código de éxito, el componente entra en ese `STOPPING` estado. Si se especifica un `shutdown` script, se ejecuta; de lo contrario, el componente entra en ese `FINISHED` estado.

Los componentes que dependen de este componente se inician cuando se ejecuta este paso del ciclo de vida. Para ejecutar un proceso en segundo plano, como un servicio que utilizan los componentes dependientes, utilice el paso del `startup` ciclo de vida en su lugar.

Al implementar componentes con un `run` ciclo de vida, el dispositivo principal puede declarar que el despliegue se ha completado en cuanto se ejecute este script de ciclo de vida. Como resultado, la implementación puede completarse y realizarse correctamente incluso si el script de `run` ciclo de vida falla poco después de ejecutarse. Si desea que el estado de la implementación dependa del resultado del script de inicio del componente, utilice el paso del `startup` ciclo de vida en su lugar.

> **Note**
>
> Puede definir solo uno `startup` o un `run` ciclo de vida.

Este objeto o cadena contiene la siguiente información:

`Script`

El script que se va a ejecutar.

`RequiresPrivilege`

(Opcional) Puede ejecutar el script con privilegios de root. Si establece esta opción en`true`, el software AWS IoT Greengrass principal ejecutará este script de ciclo de

vida como root en lugar de como el usuario del sistema que usted configure para ejecutar este componente. El valor predeterminado es `false`.

`Skipif`

(Opcional) La comprobación para determinar si se debe ejecutar o no el script. Puede definirlo para comprobar si hay un ejecutable en la ruta o si existe un archivo. Si el resultado es verdadero, el software AWS IoT Greengrass Core omite el paso. Seleccione una de las siguientes comprobaciones:

- `onpath` *`runnable`*— Compruebe si hay un ejecutable en la ruta del sistema. Por ejemplo, úselo **`onpath python3`** para omitir este paso del ciclo de vida si Python 3 está disponible.
- `exists` *`file`*— Comprueba si existe un archivo. Por ejemplo, úselo **`exists /tmp/my-configuration.db`** para omitir este paso del ciclo de vida si `/tmp/my-configuration.db` está presente.

`Timeout`

(Opcional) El tiempo máximo en segundos que puede ejecutarse el script antes de que el software AWS IoT Greengrass principal finalice el proceso.

Este paso del ciclo de vida no agota el tiempo de espera de forma predeterminada. Si omite este tiempo de espera, el `run` script se ejecutará hasta que finalice.

`Setenv`

(Opcional) El diccionario de variables de entorno que se proporcionará al script. Estas variables de entorno anulan las variables que usted proporciona. `Lifecycle.Setenv`

`startup`

(Opcional) Un objeto o cadena que define el proceso en segundo plano que se ejecutará cuando se inicie el componente.

Se utiliza `startup` para ejecutar un comando que debe cerrarse correctamente o para actualizar el estado del componente `RUNNING` antes de que puedan iniciarse los componentes dependientes. Utilice la operación UpdateStateIPC para establecer el estado del componente como `RUNNING` o `ERRORED` cuando el componente inicie un script que no se cierre. Por ejemplo, puede definir un `startup` paso que inicie el proceso de MySQL con`/etc/init.d/mysqld start`.

El componente entra en ese `STARTING` estado cuando se ejecuta este paso del ciclo de vida. Si el `startup` script sale con un código de éxito, el componente entra en ese `RUNNING` estado. A continuación, se pueden iniciar los componentes dependientes.

Al implementar componentes con un `startup` ciclo de vida, el dispositivo principal puede declarar que la implementación se ha completado una vez que este script de ciclo de vida finaliza o informa de su estado. En otras palabras, el estado de la implementación es `IN_PROGRESS` hasta que los scripts de inicio de todos los componentes se cierren o notifiquen un estado.

> **Note**
>
> Solo puede definir uno `startup` o un `run` ciclo de vida.

Este objeto o cadena contiene la siguiente información:

`Script`

El script que se va a ejecutar.

`RequiresPrivilege`

(Opcional) Puede ejecutar el script con privilegios de root. Si establece esta opción en`true`, el software AWS IoT Greengrass principal ejecutará este script de ciclo de vida como root en lugar de como el usuario del sistema que usted configure para ejecutar este componente. El valor predeterminado es `false`.

`Skipif`

(Opcional) La comprobación para determinar si se debe ejecutar o no el script. Puede definirlo para comprobar si hay un ejecutable en la ruta o si existe un archivo. Si el resultado es verdadero, el software AWS IoT Greengrass Core omite el paso. Seleccione una de las siguientes comprobaciones:

- `onpath` *`runnable`*— Compruebe si hay un ejecutable en la ruta del sistema. Por ejemplo, úselo **`onpath python3`** para omitir este paso del ciclo de vida si Python 3 está disponible.
- `exists` *`file`*— Comprueba si existe un archivo. Por ejemplo, úselo **`exists /tmp/my-configuration.db`** para omitir este paso del ciclo de vida si `/tmp/my-configuration.db` está presente.

`Timeout`

(Opcional) El tiempo máximo en segundos que puede ejecutarse el script antes de que el software AWS IoT Greengrass principal finalice el proceso.

Predeterminado: 120 segundos

`Setenv`

(Opcional) El diccionario de variables de entorno que se proporcionará al script. Estas variables de entorno anulan las variables que usted proporciona. `Lifecycle.Setenv`

`shutdown`

(Opcional) Un objeto o una cadena que define el script que se ejecutará cuando el componente se apague. Utilice el ciclo de vida de apagado para ejecutar el código que desee ejecutar cuando el componente esté en ese `STOPPING` estado. El ciclo de vida de apagado se puede utilizar para detener un proceso iniciado por los `run` scripts `startup` o.

Si inicia un proceso en segundo plano`startup`, utilice el `shutdown` paso para detener ese proceso cuando el componente se apague. Por ejemplo, puede definir un `shutdown` paso que detenga el proceso de MySQL con`/etc/init.d/mysqld stop`.

El `shutdown` script se ejecuta después de que el componente entre en `STOPPING` ese estado. Si el script se completa correctamente, el componente entra en el `FINISHED` estado.

Este objeto o cadena contiene la siguiente información:

`Script`

El script que se va a ejecutar.

`RequiresPrivilege`

(Opcional) Puede ejecutar el script con privilegios de root. Si establece esta opción en`true`, el software AWS IoT Greengrass principal ejecutará este script de ciclo de vida como root en lugar de como el usuario del sistema que usted configure para ejecutar este componente. El valor predeterminado es `false`.

`Skipif`

(Opcional) La comprobación para determinar si se debe ejecutar o no el script. Puede definirlo para comprobar si hay un ejecutable en la ruta o si existe un archivo. Si

el resultado es verdadero, el software AWS IoT Greengrass Core omite el paso. Seleccione una de las siguientes comprobaciones:

- `onpath` *`runnable`*— Compruebe si hay un ejecutable en la ruta del sistema. Por ejemplo, úselo **`onpath python3`** para omitir este paso del ciclo de vida si Python 3 está disponible.
- `exists` *`file`*— Comprueba si existe un archivo. Por ejemplo, úselo **`exists / tmp/my-configuration.db`** para omitir este paso del ciclo de vida si `/tmp/my-configuration.db` está presente.

`Timeout`

(Opcional) El tiempo máximo en segundos que puede ejecutarse el script antes de que el software AWS IoT Greengrass principal finalice el proceso.

Predeterminado: 15 segundos.

`Setenv`

(Opcional) El diccionario de variables de entorno que se proporcionará al script. Estas variables de entorno anulan las variables que usted proporciona. `Lifecycle.Setenv`

`recover`

(Opcional) Un objeto o una cadena que define el script que se ejecutará cuando el componente detecte un error.

Este paso se ejecuta cuando un componente entra en el `ERRORED` estado. Si el componente pasa `ERRORED` tres veces sin recuperarse correctamente, el componente cambia al `BROKEN` estado. Para corregir un `BROKEN` componente, debe volver a desplegarlo.

Este objeto o cadena contiene la siguiente información:

`Script`

El script que se va a ejecutar.

`RequiresPrivilege`

(Opcional) Puede ejecutar el script con privilegios de root. Si establece esta opción en`true`, el software AWS IoT Greengrass principal ejecutará este script de ciclo de vida como root en lugar de como el usuario del sistema que usted configure para ejecutar este componente. El valor predeterminado es `false`.

`Skipif`

(Opcional) La comprobación para determinar si se debe ejecutar o no el script. Puede definirlo para comprobar si hay un ejecutable en la ruta o si existe un archivo. Si el resultado es verdadero, el software AWS IoT Greengrass Core omite el paso. Seleccione una de las siguientes comprobaciones:

- `onpath` *`runnable`*— Compruebe si hay un ejecutable en la ruta del sistema. Por ejemplo, úselo **`onpath python3`** para omitir este paso del ciclo de vida si Python 3 está disponible.
- `exists` *`file`*— Comprueba si existe un archivo. Por ejemplo, úselo **`exists /tmp/my-configuration.db`** para omitir este paso del ciclo de vida si `/tmp/my-configuration.db` está presente.

`Timeout`

(Opcional) El tiempo máximo en segundos que puede ejecutarse el script antes de que el software AWS IoT Greengrass principal finalice el proceso.

Valor predeterminado: 60 segundos.

`Setenv`

(Opcional) El diccionario de variables de entorno que se va a proporcionar al script. Estas variables de entorno anulan las variables que usted proporciona. `Lifecycle.Setenv`

`bootstrap`

(Opcional) Objeto o cadena que define un script que requiere el reinicio del software AWS IoT Greengrass principal o del dispositivo principal. Esto le permite desarrollar un componente que se reinicie después de instalar las actualizaciones del sistema operativo o las actualizaciones del tiempo de ejecución, por ejemplo.

> **Note**
>
> Para instalar actualizaciones o dependencias que no requieran que el software o dispositivo AWS IoT Greengrass principal se reinicie, utilice el ciclo de vida de instalación.

Este paso del ciclo de vida se ejecuta antes del paso del ciclo de vida de la instalación en los siguientes casos cuando el software AWS IoT Greengrass principal implementa el componente:

- El componente se implementa en el dispositivo principal por primera vez.
- La versión del componente cambia.
- El script de arranque cambia como resultado de una actualización de la configuración del componente.

Una vez que el software AWS IoT Greengrass principal complete el paso de arranque de todos los componentes que tengan un paso de arranque en una implementación, el software se reinicia.

> **Important**
>
> Debe configurar el software AWS IoT Greengrass Core como un servicio del sistema para reiniciar el software AWS IoT Greengrass Core o el dispositivo principal. Si no configura el software AWS IoT Greengrass principal como un servicio del sistema, el software no se reiniciará. Para obtener más información, consulte Configurar el núcleo de Greengrass como un servicio del sistema.

Este objeto o cadena contiene la siguiente información:

`BootstrapOnRollback`

> **Note**
>
> Cuando esta función está habilitada, solo `BootstrapOnRollback` se ejecutará en los componentes que hayan completado o intentado ejecutar los pasos del ciclo de vida del arranque como parte de una implementación de destino fallida. Esta función está disponible para las versiones 2.12.0 y posteriores del núcleo de Greengrass.

(Opcional) Puede ejecutar los pasos del ciclo de vida de bootstrap como parte de una implementación de reversión. Si configuras esta opción en`true`, se ejecutarán los pasos del ciclo de vida del arranque definidos en una implementación de reversión. Cuando se produce un error en una implementación, la versión anterior del ciclo de

vida de arranque del componente se volverá a ejecutar durante una implementación de reversión.

El valor predeterminado es `false`.

`Script`

El script que se va a ejecutar. El código de salida de este script define la instrucción de reinicio. Utilice los siguientes códigos de salida:

- `0`— No reinicie el software AWS IoT Greengrass principal ni el dispositivo principal. El software AWS IoT Greengrass principal se sigue reiniciando después del arranque de todos los componentes.
- `100`— Solicitud para reiniciar el software AWS IoT Greengrass Core.
- `101`— Solicitud para reiniciar el dispositivo principal.

Los códigos de salida 100 a 199 están reservados para un comportamiento especial. Otros códigos de salida representan errores de script.

`RequiresPrivilege`

(Opcional) Puede ejecutar el script con privilegios de root. Si establece esta opción en`true`, el software AWS IoT Greengrass principal ejecutará este script de ciclo de vida como root en lugar de como el usuario del sistema que usted configure para ejecutar este componente. El valor predeterminado es `false`.

`Timeout`

(Opcional) El tiempo máximo en segundos que puede ejecutarse el script antes de que el software AWS IoT Greengrass principal finalice el proceso.

Predeterminado: 120 segundos

`Setenv`

(Opcional) El diccionario de variables de entorno que se proporcionará al script. Estas variables de entorno anulan las variables que usted proporciona. `Lifecycle.Setenv`

`Selections`

(Opcional) Una lista de claves de selección que especifican las secciones del ciclo de vida global que se van a ejecutar en este manifiesto. En el ciclo de vida global, puedes definir los pasos del ciclo de vida con claves de selección en cualquier nivel para seleccionar subsecciones del ciclo de vida. A continuación, el dispositivo principal utiliza las secciones

que coinciden con las claves de selección de este manifiesto. Para obtener más información, consulta los ejemplos del ciclo de vida global.

> **Important**
>
> El dispositivo principal utiliza las selecciones del ciclo de vida global solo si este manifiesto no define un ciclo de vida.

Puedes especificar la clave de `all` selección para ejecutar secciones del ciclo de vida global que no tienen claves de selección.

`Artifacts`

(Opcional) Una lista de objetos, cada uno de los cuales define un artefacto binario para el componente de la plataforma que define este manifiesto. Por ejemplo, puede definir el código o las imágenes como artefactos.

Cuando el componente se despliega, el software AWS IoT Greengrass principal descarga el artefacto en una carpeta del dispositivo principal. También puede definir los artefactos como archivos de almacenamiento que el software extrae después de descargarlos.

Puede utilizar variables de receta para obtener las rutas a las carpetas en las que se instalan los artefactos en el dispositivo principal.

- Archivos normales: utilice la variable de receta artifacts:path para obtener la ruta a la carpeta que contiene los artefactos. Por ejemplo, especifique `{artifacts:path}/my_script.py` en una receta la ruta a un artefacto que tenga el URI. `s3://DOC-EXAMPLE-BUCKET/path/to/my_script.py`
- Archivos extraídos: utilice la variable de receta Artifacts:decompressedPath para obtener la ruta a la carpeta que contiene los artefactos de archivo extraídos. El software AWS IoT Greengrass principal extrae cada archivo a una carpeta con el mismo nombre que el archivo. Por ejemplo, especifique `{artifacts:decompressedPath}/my_archive/my_script.py` en una receta la ruta del artefacto de archivo que contiene el URI`s3://DOC-EXAMPLE-BUCKET/path/to/my_archive.zip`. `my_script.py`

> **Note**
>
> Al desarrollar un componente con un artefacto de archivo en un dispositivo central local, es posible que no disponga de un URI para ese artefacto. Para probar el

componente con una `Unarchive` opción que extraiga el artefacto, especifique un URI en el que el nombre del archivo coincida con el nombre del archivo del artefacto archivado. Puede especificar el URI en el que desea cargar el artefacto de archivo o puede especificar un nuevo URI de marcador de posición. Por ejemplo, para extraer el `my_archive.zip` artefacto durante una implementación local, puedes especificarlo. `s3://DOC-EXAMPLE-BUCKET/my_archive.zip`

Cada objeto contiene la siguiente información:

`URI`

El URI de un artefacto en un bucket de S3. El software AWS IoT Greengrass principal busca el artefacto de este URI cuando se instala el componente, a menos que el artefacto ya exista en el dispositivo. Cada artefacto debe tener un nombre de archivo único en cada manifiesto.

`Unarchive`

(Opcional) El tipo de archivo que se va a desempaquetar. Puede elegir entre las siguientes opciones:

- `NONE`— El archivo no es un archivo para desempaquetar. El software AWS IoT Greengrass Core instala el artefacto en una carpeta del dispositivo principal. Puede usar la variable de receta artifacts:path para obtener la ruta a esta carpeta.
- `ZIP`— El fichero es un archivo ZIP. El software AWS IoT Greengrass Core extrae el archivo a una carpeta con el mismo nombre que el archivo. Puede utilizar la variable de receta Artifacts:decompressedPath para obtener la ruta a la carpeta que contiene esta carpeta.

El valor predeterminado es `NONE`.

`Permission`

(Opcional) Un objeto que define los permisos de acceso que se van a establecer para este archivo de artefactos. Puede establecer el permiso de lectura y el permiso de ejecución.

> **Note**
>
> No puedes configurar el permiso de escritura porque el software AWS IoT Greengrass Core no permite que los componentes editen los archivos de

artefactos de la carpeta de artefactos. Para editar un archivo de artefactos de un componente, cópielo en otra ubicación o publique e implemente un nuevo archivo de artefactos.

Si define un artefacto como un archivo para desempaquetar, el software AWS IoT Greengrass Core establece estos permisos de acceso en los archivos que desempaquete del archivo. El software AWS IoT Greengrass Core establece los permisos de acceso de la carpeta en para `ALL` y. `Read` `Execute` Esto permite a los componentes ver los archivos desempaquetados de la carpeta. Para establecer los permisos en los archivos individuales del archivo, puede configurarlos en el script del ciclo de vida de la instalación.

Este objeto contiene la siguiente información:

`Read`

(Opcional) El permiso de lectura que se debe configurar para este archivo de artefactos. Para permitir que otros componentes accedan a este artefacto, como los componentes que dependen de este componente, especifique. `ALL` Puede elegir entre las siguientes opciones:

- `NONE`— El archivo no se puede leer.
- `OWNER`— El usuario del sistema que haya configurado para ejecutar este componente puede leer el archivo.
- `ALL`— Todos los usuarios pueden leer el archivo.

El valor predeterminado es `OWNER`.

`Execute`

(Opcional) El permiso de ejecución que se debe configurar para este archivo de artefactos. El `Execute` permiso implica el `Read` permiso. Por ejemplo, si especifica `ALL` para`Execute`, todos los usuarios podrán leer y ejecutar este archivo de artefactos.

Puede elegir entre las siguientes opciones:

- `NONE`— El archivo no se puede ejecutar.
- `OWNER`— El usuario del sistema que usted configure para ejecutar el componente puede ejecutar el archivo.
- `ALL`— Todos los usuarios pueden ejecutar el archivo.

El valor predeterminado es `NONE`.

`Digest`

(Solo lectura) El hash criptográfico resumido del artefacto. Al crear un componente, AWS IoT Greengrass utiliza un algoritmo de hash para calcular el hash del archivo del artefacto. A continuación, al implementar el componente, el núcleo de Greengrass calcula el hash del artefacto descargado y lo compara con este resumen para verificar el artefacto antes de la instalación. Si el hash no coincide con el resumen, se produce un error en la implementación.

Si establece este parámetro, AWS IoT Greengrass reemplaza el valor que estableció al crear el componente.

`Algorithm`

(Solo lectura) El algoritmo de hash que se AWS IoT Greengrass utiliza para calcular el hash resumido del artefacto.

Si establece este parámetro, AWS IoT Greengrass reemplaza el valor que estableció al crear el componente.

`Lifecycle`

Objeto que define cómo instalar y ejecutar el componente. El dispositivo principal usa el ciclo de vida global solo si el manifiesto que se va a usar no especifica un ciclo de vida.

> **Note**
>
> Este ciclo de vida se define fuera de un manifiesto. También puedes definir un ciclo de vida del manifiesto que se aplique a las plataformas que coincidan con ese manifiesto.

En el ciclo de vida global, puedes especificar los ciclos de vida que se ejecutan para determinadas claves de selección que especificas en cada manifiesto. Las claves de selección son cadenas que identifican las secciones del ciclo de vida global que se van a ejecutar en cada manifiesto.

La clave `all` de selección es la predeterminada en cualquier sección sin una clave de selección. Esto significa que puedes especificar la clave de `all` selección en un manifiesto para ejecutar las secciones del ciclo de vida global sin claves de selección. No es necesario especificar la clave de `all` selección en el ciclo de vida global.

Si un manifiesto no define un ciclo de vida o claves de selección, el dispositivo principal utilizará la `all` selección de forma predeterminada. Esto significa que, en este caso, el dispositivo principal usa las secciones del ciclo de vida global que no usan claves de selección.

Este objeto contiene la misma información que el ciclo de vida del manifiesto, pero puedes especificar las claves de selección en cualquier nivel para seleccionar subsecciones del ciclo de vida.

> **Tip**
>
> Le recomendamos que utilice solo letras minúsculas para cada clave de selección para evitar conflictos entre las claves de selección y las claves del ciclo de vida. Las claves de ciclo de vida comienzan con una letra mayúscula.

Example Ejemplo de ciclo de vida global con claves de selección de nivel superior

```
Lifecycle:
  key1:
    install:
      Skipif: either onpath executable or exists file
      Script: command1
  key2:
    install:
      Script: command2
  all:
    install:
      Script: command3
```

Example Ejemplo de ciclo de vida global con claves de selección de nivel inferior

```
Lifecycle:
  install:
    Script:
      key1: command1
      key2: command2
      all: command3
```

Example Ejemplo de ciclo de vida global con varios niveles de claves de selección

```
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
```

Variables de receta

Las variables de receta exponen la información del componente y el núcleo actuales para que la utilice en sus recetas. Por ejemplo, puede usar una variable de receta para pasar los parámetros de configuración de los componentes a una aplicación que ejecute en un script de ciclo de vida.

Puede usar variables de receta en las siguientes secciones de recetas de componentes:

- Definiciones del ciclo de vida.
- Definiciones de configuración de componentes, si utiliza Greengrass nucleus v2.6.0 o posterior y establece la interpolateComponentConfigurationopción de configuración en. `true` También puede usar variables de recetas al implementar las actualizaciones de configuración de los componentes.

Las variables de receta utilizan `{recipe_variable}` la sintaxis. Los corchetes indican una variable de receta.

AWS IoT Greengrass admite las siguientes variables de receta:

`component_dependency_name``:configuration:`*`json_pointer`*

El valor de un parámetro de configuración para el componente que define esta receta o para un componente del que depende este componente.

Puede utilizar esta variable para proporcionar un parámetro a un script que ejecute en el ciclo de vida del componente.

> **Note**
>
> AWS IoT Greengrass admite esta variable de receta solo en las definiciones del ciclo de vida de los componentes.

Esta variable de receta tiene las siguientes entradas:

- `component_dependency_name`— (Opcional) El nombre de la dependencia del componente que se va a consultar. Omita este segmento para consultar el componente que define esta receta. Solo puede especificar las dependencias directas.
- `json_pointer`— El puntero JSON al valor de configuración que se va a evaluar. Los punteros JSON comienzan con una barra `/` diagonal. Para identificar un valor en una configuración de componentes anidados, utilice barras diagonales (`/`) para separar las claves de cada nivel de la configuración. Puede usar un número como clave para especificar un índice en una lista. Para obtener más información, consulta la especificación del puntero JSON.

  AWS IoT Greengrass Core usa punteros JSON para recetas en formato YAML.

El puntero JSON puede hacer referencia a los siguientes tipos de nodos:

- Un nodo de valor. AWS IoT Greengrass Core reemplaza la variable de receta por la representación en cadena del valor. Los valores nulos se `null` convierten en una cadena.
- Un nodo de objeto. AWS IoT Greengrass Core reemplaza la variable de receta por la representación de cadena JSON serializada de ese objeto.
- Sin nodo. AWS IoT Greengrass Core no reemplaza la variable de receta.

Por ejemplo, la variable de `{configuration:/Message}` receta recupera el valor de la `Message` clave en la configuración del componente. La variable de `{com.example.MyComponentDependency:configuration:/server/port}` receta recupera el valor del objeto `port` de `server` configuración de una dependencia de un componente.

`component_dependency_name``:artifacts:path`

La ruta raíz de los artefactos del componente que define esta receta o de un componente del que depende este componente.

Cuando se instala un componente, AWS IoT Greengrass copia los artefactos del componente en la carpeta que expone esta variable. Puede utilizar esta variable para identificar la ubicación de un script que se va a ejecutar en el ciclo de vida del componente, por ejemplo.

La carpeta de esta ruta es de solo lectura. Para modificar los archivos de artefactos, cópielos en otra ubicación, como el directorio de trabajo actual (`$PWDo). .` A continuación, modifique los archivos allí.

Para leer o ejecutar un artefacto desde una dependencia de un componente, debe ser el artefacto `Read` o el `Execute` permiso de ese artefacto. `ALL` Para obtener más información, consulta los permisos de artefactos que definas en la receta del componente.

Esta variable de receta tiene las siguientes entradas:

- `component_dependency_name`— (Opcional) El nombre de la dependencia del componente que se va a consultar. Omita este segmento para consultar el componente que define esta receta. Solo puede especificar las dependencias directas.

`component_dependency_name``:artifacts:decompressedPath`

La ruta raíz de los artefactos de archivo descomprimidos para el componente que define esta receta o para un componente del que depende este componente.

Cuando se instala un componente, AWS IoT Greengrass desempaqueta los artefactos de archivo del componente en la carpeta que expone esta variable. Puede utilizar esta variable para identificar la ubicación de un script que se va a ejecutar en el ciclo de vida del componente, por ejemplo.

Cada artefacto se descomprime en una carpeta dentro de la ruta descomprimida, donde la carpeta tiene el mismo nombre que el artefacto menos su extensión. Por ejemplo, un artefacto ZIP denominado se descomprime en la carpeta. `models.zip` `{artifacts:decompressedPath}/models`

La carpeta de esta ruta es de solo lectura. Para modificar los archivos de artefactos, cópielos en otra ubicación, como el directorio de trabajo actual (`$PWDo). .` A continuación, modifique los archivos allí.

Para leer o ejecutar un artefacto desde una dependencia de un componente, debe ser el artefacto `Read` o el `Execute` permiso de ese artefacto. `ALL` Para obtener más información, consulta los permisos de artefactos que definas en la receta del componente.

Esta variable de receta tiene las siguientes entradas:

- `component_dependency_name`— (Opcional) El nombre de la dependencia del componente que se va a consultar. Omita este segmento para consultar el componente que define esta receta. Solo puede especificar las dependencias directas.

component_dependency_name`:work:path`

Esta función está disponible para la versión 2.0.4 y versiones posteriores del componente núcleo de Greengrass.

La ruta de trabajo del componente que define esta receta o de un componente del que depende este componente. El valor de esta variable de receta equivale a la salida de la variable de `$PWD` entorno y del comando pwd cuando se ejecuta desde el contexto del componente.

Puede usar esta variable de receta para compartir archivos entre un componente y una dependencia.

El componente que define esta receta y los demás componentes que se ejecutan como el mismo usuario y grupo pueden leer y escribir en la carpeta de esta ruta.

Esta variable de receta tiene las siguientes entradas:

- `component_dependency_name`— (Opcional) El nombre de la dependencia del componente que se va a consultar. Omita este segmento para consultar el componente que define esta receta. Solo puede especificar las dependencias directas.

`kernel:rootPath`

La ruta raíz AWS IoT Greengrass principal.

`iot:thingName`

Esta función está disponible para la versión 2.3.0 y versiones posteriores del componente núcleo de Greengrass.

El nombre del dispositivo principal. AWS IoT

Ejemplos de recetas

Puede hacer referencia a los siguientes ejemplos de recetas para ayudarle a crear recetas para sus componentes.

AWS IoT Greengrass selecciona un índice de componentes de Greengrass, denominado Catálogo de software de Greengrass. Este catálogo rastrea los componentes de Greengrass desarrollados por la comunidad de Greengrass. Desde este catálogo, puede descargar, modificar e implementar componentes para crear sus aplicaciones de Greengrass. Para obtener más información, consulte Componentes de la comunidad.

Temas

Receta de componentes de Hello World

La siguiente receta describe un componente de Hello World que ejecuta un script de Python. Este componente es compatible con todas las plataformas y acepta un `Message` parámetro que AWS IoT Greengrass pasa como argumento al script de Python. Esta es la receta del componente Hello World del tutorial de introducción.

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}"
```

Ejemplo de componente de tiempo de ejecución de Python

La siguiente receta describe un componente que instala Python. Este componente es compatible con dispositivos Linux de 64 bits.

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
```

Receta de componente que especifica varios campos

La siguiente receta de componentes utiliza varios campos de receta.

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'
```

Referencia de variable de entorno de componentes

El softwareAWS IoT Greengrass principal establece las variables de entorno cuando ejecuta los scripts del ciclo de vida de los componentes. Puede obtener estas variables de entorno en sus componentes para obtener el nombre de la cosa y la versión del núcleo de Greengrass.Región de AWS El software también establece las variables de entorno que el componente necesita para utilizar el SDK de comunicación entre procesos e interactuar conAWS los servicios.

También puede configurar variables de entorno personalizadas para los scripts del ciclo de vida de sus componentes. Para obtener más información, consulte Setenv.

El softwareAWS IoT Greengrass principal establece las siguientes variables de entorno:

`AWS_IOT_THING_NAME`

El nombre de laAWS IoT cosa que representa este dispositivo central de Greengrass.

`AWS_REGION`

ElRegión de AWS lugar donde funciona este dispositivo central de Greengrass.

LosAWS SDK utilizan esta variable de entorno para identificar la región predeterminada que se va a utilizar. Esta variable es equivalente a`AWS_DEFAULT_REGION`.

`AWS_DEFAULT_REGION`

ElRegión de AWS lugar donde funciona este dispositivo central de Greengrass.

AWS CLIUsa esta variable de entorno para identificar la región predeterminada que se va a utilizar. Esta variable es equivalente a`AWS_REGION`.

`GGC_VERSION`

La versión del componente del núcleo de Greengrass que se ejecuta en este dispositivo central de Greengrass.

`GG_ROOT_CA_PATH`

Esta función está disponible para la versión 2.5.5 y versiones posteriores del componente Nucleus de Greengrass.

La ruta al certificado de la autoridad certificadora (CA) raíz que utiliza el núcleo de Greengrass.

`AWS_GG_NUCLEUS_DOMAIN_SOCKET_FILEPATH_FOR_COMPONENT`

La ruta al socket IPC que utilizan los componentes para comunicarse con el softwareAWS IoT Greengrass principal. Para obtener más información, consulte Úselo SDK para dispositivos con AWS IoT para comunicarse con el núcleo de Greengrass, otros componentes y AWS IoT Core.

`SVCUID`

El token secreto que utilizan los componentes para conectarse al socket IPC y comunicarse con el softwareAWS IoT Greengrass Core. Para obtener más información, consulte Úselo SDK para dispositivos con AWS IoT para comunicarse con el núcleo de Greengrass, otros componentes y AWS IoT Core.

`AWS_CONTAINER_AUTHORIZATION_TOKEN`

El token secreto que utilizan los componentes para recuperar las credenciales del componente del servicio de intercambio de tokens.

`AWS_CONTAINER_CREDENTIALS_FULL_URI`

El URI que los componentes solicitan para recuperar las credenciales del componente del servicio de intercambio de tokens.

Implemente AWS IoT Greengrass componentes en los dispositivos

Puede utilizarlos AWS IoT Greengrass para implementar componentes en dispositivos o grupos de dispositivos. Las implementaciones se utilizan para definir los componentes y las configuraciones que se envían a los dispositivos. AWS IoT Greengrassse despliega en objetivos, AWS IoT cosas o grupos de cosas que representan los dispositivos principales de Greengrass. AWS IoT Greengrassutiliza AWS IoT Coretareas para implementarlas en sus dispositivos principales. Puede configurar la forma en que se implementará el trabajo en sus dispositivos.

Implementaciones de dispositivos principales

Cada dispositivo principal ejecuta los componentes de las implementaciones de ese dispositivo. Una nueva implementación en el mismo destino sobrescribe la implementación anterior en el destino. Al

crear una implementación, se definen los componentes y las configuraciones que se van a aplicar al software existente del dispositivo principal.

Al revisar una implementación para un objetivo, se sustituyen los componentes de la revisión anterior por los componentes de la nueva revisión. Por ejemplo, se despliegan los Gestor secreto componentes Gestor de registros y en el grupo de cosas`TestGroup`. A continuación, se crea otra implementación para la `TestGroup` que se especifique únicamente el componente del administrador secreto. Como resultado, los dispositivos principales de ese grupo ya no ejecutan el administrador de registros.

Resolución de la dependencia de la plataforma

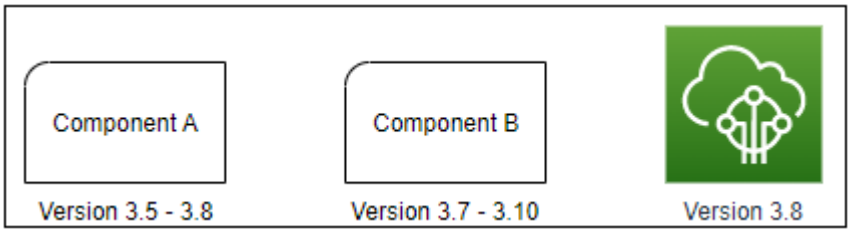
Cuando un dispositivo principal recibe una implementación, comprueba que los componentes son compatibles con el dispositivo principal. Por ejemplo, si lo implementa en Firehose un destino de Windows, la implementación fallará.

Resolución de dependencias de componentes

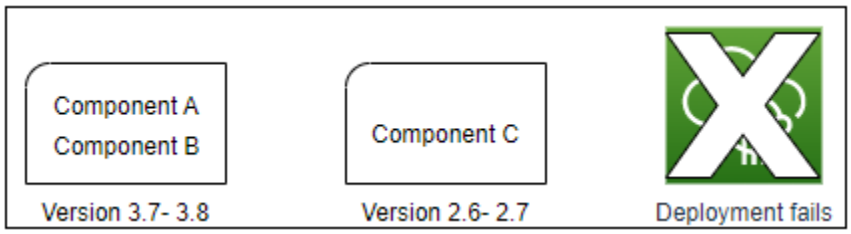
El dispositivo principal también comprueba si las dependencias de cada componente son compatibles con las restricciones de versión para las implementaciones de otros componentes en este grupo. Cuando las restricciones de versión de un componente se superponen, Greengrass utiliza la versión más alta aplicable del componente. Por ejemplo:

- El componente A se implementa en. `TestGroup` El componente A depende de `com.example.PythonRuntime` las versiones 3.5 a 3.10 del componente.
- A continuación, despliega el componente B en`TestGroup`. El componente B depende de `com.example.PythonRuntime` las versiones 3.7 a 3.8 del componente.

Como resultado, los dispositivos principales `TestGroup` determinan que pueden implementar la versión 3.8 del `com.example.PythonRuntime` componente, ya que esta es la versión más aplicable, donde las restricciones de versión se superponen.



A continuación, despliega el componente C en`TestGroup`. El componente C depende de `com.example.PythonRuntime` las versiones 2.6 a 2.7 del componente. Esta implementación falla porque no hay ninguna versión del componente que cumpla con las restricciones 2.6 a 2.7 y 3.7 a 3.8.



Eliminar un dispositivo de un grupo de cosas

Al eliminar un dispositivo principal de un grupo de cosas, el comportamiento de despliegue de los componentes depende de la versión del núcleo de Greengrass que ejecute el dispositivo principal.

2.5.1 and later

Al eliminar un dispositivo principal de un grupo de cosas, el comportamiento depende de si la AWS IoT política concede el `greengrass:ListThingGroupsForCoreDevice` permiso. Para obtener más información sobre este permiso y AWS IoT las políticas para los dispositivos principales, consulteAutenticación y autorización de dispositivos en AWS IoT Greengrass.

- Si la AWS IoT política concede este permiso

  Al eliminar un dispositivo principal de un grupo de cosas, se AWS IoT Greengrass eliminan los componentes del grupo de cosas la próxima vez que se realice una implementación en el dispositivo. Si un componente del dispositivo se incluye en la siguiente implementación, ese componente no se elimina del dispositivo.

- Si la AWS IoT política no concede este permiso

  Cuando se elimina un dispositivo principal de un grupo de cosas, AWS IoT Greengrass no se eliminan los componentes de ese grupo de cosas del dispositivo.

  Para eliminar un componente de un dispositivo, utilice el comando deployment create de la CLI de Greengrass. Especifique el componente que desea eliminar con el `--remove` argumento y especifique el grupo de cosas con el `--groupId` argumento.

2.5.0

Cuando se elimina un dispositivo principal de un grupo de cosas, se AWS IoT Greengrass eliminan los componentes del grupo de cosas la próxima vez que se realice un despliegue en el dispositivo. Si un componente del dispositivo se incluye en la siguiente implementación, ese componente no se elimina del dispositivo.

Este comportamiento requiere que la AWS IoT política del dispositivo principal conceda el `greengrass:ListThingGroupsForCoreDevice` permiso. Si un dispositivo principal no tiene este permiso, no podrá aplicar las implementaciones. Para obtener más información, consulte Autenticación y autorización de dispositivos en AWS IoT Greengrass.

2.0.x - 2.4.x

Cuando se elimina un dispositivo principal de un grupo de cosas, AWS IoT Greengrass no se eliminan los componentes de ese grupo de cosas del dispositivo.

Para eliminar un componente de un dispositivo, utilice el comando deployment create de la CLI de Greengrass. Especifique el componente que desea eliminar con el `--remove` argumento y especifique el grupo de cosas con el `--groupId` argumento.

Implementaciones

Los despliegues son continuos. Al crear una implementación, AWS IoT Greengrass despliega la implementación en los dispositivos de destino que están en línea. Si un dispositivo de destino no está en línea, recibirá la implementación la próxima vez que se conecteAWS IoT Greengrass. Al añadir un dispositivo principal a un grupo de cosas de destino, AWS IoT Greengrass envía al dispositivo la última implementación de ese grupo de cosas.

Antes de que un dispositivo principal despliegue un componente, notifica de forma predeterminada a cada componente del dispositivo. Los componentes de Greengrass pueden responder a la notificación para aplazar el despliegue. Es posible que desee aplazar la implementación si el nivel de batería del dispositivo es bajo o si está ejecutando un proceso que no se puede interrumpir. Para obtener más información, consulte Tutorial: Desarrolle un componente de Greengrass que aplace las actualizaciones de los componentes. Al crear una implementación, puede configurarla para que se despliegue sin notificar a los componentes.

Cada cosa o grupo de cosas objetivo puede tener una implementación a la vez. Esto significa que, al crear una implementación para un objetivo, ya AWS IoT Greengrass no se despliega la revisión anterior de la implementación de ese objetivo.

Opciones de implementación

Las implementaciones ofrecen varias opciones que le permiten controlar qué dispositivos reciben una actualización y cómo se implementa la actualización. Al crear una implementación, puede configurar las siguientes opciones:

- AWS IoT Greengrasscomponentes

  Defina los componentes que se van a instalar y ejecutar en los dispositivos de destino. AWS IoT Greengrasslos componentes son módulos de software que se implementan y ejecutan en los dispositivos principales de Greengrass. Los dispositivos reciben componentes solo si el componente es compatible con la plataforma del dispositivo. Esto le permite realizar la implementación en grupos de dispositivos, incluso si los dispositivos de destino se ejecutan en varias plataformas. Si un componente no es compatible con la plataforma del dispositivo, el componente no se implementa en el dispositivo.

  Puede implementar componentes personalizados y componentes AWS proporcionados en sus dispositivos. Al implementar un componente, AWS IoT Greengrass identifica las dependencias de los componentes y también las despliega. Para obtener más información, consulte Desarrolle AWS IoT Greengrass componentes y AWS-componentes proporcionados.

  Usted define la versión y la actualización de configuración que se va a implementar para cada componente. La actualización de configuración especifica cómo modificar la configuración existente del componente en el dispositivo principal o la configuración predeterminada del componente si el componente no existe en el dispositivo principal. Puede especificar los valores de configuración que desea restablecer a los valores predeterminados y los nuevos valores de configuración que se van a combinar en el dispositivo principal. Cuando un dispositivo principal recibe despliegues para distintos destinos y cada despliegue especifica versiones de componentes compatibles, el dispositivo principal aplica las actualizaciones de configuración en orden en función de la fecha y hora en que se creó el despliegue. Para obtener más información, consulte Actualizar las configuraciones de los componentes.

  > **Important**
  >
  > Al implementar un componente, AWS IoT Greengrass instala las últimas versiones compatibles de todas las dependencias de ese componente. Por este motivo, es posible que las nuevas versiones con parches de los componentes públicos AWS proporcionados se implementen automáticamente en sus dispositivos principales si agrega nuevos

dispositivos a un grupo de cosas o si actualiza la implementación destinada a esos dispositivos. Algunas actualizaciones automáticas, como las actualizaciones de Nucleus, pueden provocar que los dispositivos se reinicien de forma inesperada.
Para evitar actualizaciones no deseadas de un componente que se ejecuta en su dispositivo, le recomendamos que incluya directamente la versión que prefiera de ese componente al crear una implementación. Para obtener más información sobre el comportamiento de las actualizaciones AWS IoT Greengrass del software principal, consulteActualice el software AWS IoT Greengrass principal (OTA).

- Políticas de despliegue

  Defina cuándo es seguro implementar una configuración y qué hacer si la implementación falla. Puede especificar si desea o no esperar a que los componentes informen de que se pueden actualizar. También puede especificar si se restablece o no la configuración anterior de los dispositivos en caso de que apliquen una implementación fallida.

- Detenga la configuración

  Defina cuándo y cómo detener una implementación. El despliegue se detiene y falla si se cumplen los criterios que usted defina. Por ejemplo, puede configurar una implementación para que se detenga si un porcentaje de dispositivos no la aplica después de que la reciba un número mínimo de dispositivos.

- Configuración de despliegue

  Defina la velocidad a la que una implementación se extiende a los dispositivos de destino. Puede configurar un aumento de velocidad exponencial con límites de velocidad mínimos y máximos.

- Configuración de tiempo de espera

  Defina la cantidad máxima de tiempo de que dispone cada dispositivo para aplicar una implementación. Si un dispositivo supera la duración que especifique, no podrá aplicar la implementación.

Important

Los componentes personalizados pueden definir los artefactos en los cubos de S3. Cuando el software AWS IoT Greengrass principal implementa un componente, descarga los artefactos del componente del. Nube de AWS Las funciones principales de los dispositivos no permiten el acceso a los buckets de S3 de forma predeterminada. Para implementar

componentes personalizados que definan los artefactos en un depósito de S3, la función principal del dispositivo debe conceder permisos para descargar los artefactos de ese depósito. Para obtener más información, consulte Permita el acceso a los depósitos de S3 para los artefactos de los componentes.

Temas

Crear implementaciones

Puede crear una implementación que se dirija a una cosa o a un grupo de cosas.

Al crear una implementación, se configuran los componentes de software que se van a implementar y la forma en que el trabajo de implementación se extiende a los dispositivos de destino. Puede definir la implementación en el archivo JSON que proporciona aAWS CLI.

El objetivo de despliegue determina los dispositivos en los que desea ejecutar sus componentes. Para realizar la implementación en un dispositivo principal, especifique algo. Para realizar la implementación en varios dispositivos principales, especifique un grupo de cosas que incluya esos dispositivos. Para obtener más información sobre cómo configurar los grupos de cosas, consulte Grupos de cosas estáticos y Grupos de cosas dinámicos en la Guía para AWS IoT desarrolladores.

Siga los pasos de esta sección para crear un despliegue en un objetivo. Para obtener más información sobre cómo actualizar los componentes de software en un destino que tiene una implementación, consulteRevisar las implementaciones.

⚠ Warning

La CreateDeploymentoperación puede desinstalar componentes de los dispositivos principales. Si un componente está presente en la implementación anterior y no en la nueva, el dispositivo principal desinstala ese componente. Para evitar la desinstalación de los

componentes, utilice primero la ListDeploymentsoperación para comprobar si el destino de la implementación ya tiene una implementación existente. A continuación, utilice la GetDeploymentoperación para empezar desde esa implementación existente cuando cree una nueva implementación.

Para crear una implementación (AWS CLI)

1. Cree un archivo llamado ydeployment.json, a continuación, copie el siguiente objeto JSON en el archivo. Sustituya *TargetARN* por el ARN de la cosa o grupo de cosas a AWS IoT la que se va a destinar la implementación. Los ARN de cosas y grupos de cosas tienen el siguiente formato:

   - Cosa: `arn:aws:iot:`*region*`:`*account-id*`:thing/`*thingName*
   - Grupo de cosas: `arn:aws:iot:`*region*`:`*account-id*`:thinggroup/`*thingGroupName*

   ```
   {
     "targetArn": "targetArn"
   }
   ```

2. Compruebe si el objetivo de la implementación tiene una implementación existente que desee revisar. Haga lo siguiente:

   a. Ejecute el siguiente comando para enumerar las implementaciones del destino de implementación. Sustituya *TargetARN* por el ARN de la cosa o grupo de cosas objetivo. AWS IoT

      ```
      aws greengrassv2 list-deployments --target-arn targetArn
      ```

      La respuesta contiene una lista con la implementación más reciente del objetivo. Si la respuesta está vacía, significa que el destino no tiene una implementación existente y puedes pasar a ella. Step 3 De lo contrario, copie el `deploymentId` de la respuesta para usarlo en el paso siguiente.

      > **Note**
      >
      > También puede revisar una implementación que no sea la última revisión del destino. Especifique el `--history-filter ALL` argumento para enumerar todos

los despliegues del objetivo. A continuación, copie el ID de la implementación que desea revisar.

b. Ejecute el siguiente comando para obtener los detalles de la implementación. Estos detalles incluyen los metadatos, los componentes y la configuración del trabajo. Sustituya *DeploymentiD* por el ID del paso anterior.

```
aws greengrassv2 get-deployment --deployment-id deploymentId
```

La respuesta contiene los detalles de la implementación.

c. Copie cualquiera de los siguientes pares clave-valor de la respuesta del comando anterior en. `deployment.json` Puede cambiar estos valores para la nueva implementación.

- `deploymentName`— El nombre de la implementación.
- `components`— Los componentes de la implementación. Para desinstalar un componente, elimínelo de este objeto.
- `deploymentPolicies`— Las políticas del despliegue.
- `iotJobConfiguration`— La configuración del trabajo de la implementación.
- `tags`— Las etiquetas del despliegue.

3. (Opcional) Defina un nombre para la implementación. Sustituya *DeploymentName* por el nombre de la implementación.

```
{
  "targetArn": "targetArn",
  "deploymentName": "deploymentName"
}
```

4. Agregue cada componente para implementar los dispositivos de destino. Para ello, añada pares clave-valor al `components` objeto, donde la clave es el nombre del componente y el valor es un objeto que contiene los detalles de ese componente. Especifique los siguientes detalles para cada componente que añada:

- `version`— La versión del componente que se va a implementar.
- `configurationUpdate`— La actualización de configuración que se va a implementar. La actualización es una operación de parche que modifica la configuración existente del componente en cada dispositivo de destino o la configuración predeterminada del componente

si no existe en el dispositivo de destino. Puede especificar las siguientes actualizaciones de configuración:

- Restablecer actualizaciones (`reset`): (opcional) una lista de punteros JSON que definen los valores de configuración para restablecer sus valores predeterminados en el dispositivo de destino. El software AWS IoT Greengrass Core aplica las actualizaciones de restablecimiento antes de aplicar las actualizaciones de combinación. Para obtener más información, consulte Restablecer las actualizaciones.
- Fusionar actualizaciones (`merge`): (opcional) un documento JSON que define los valores de configuración que se van a combinar en el dispositivo de destino. Debe serializar el documento JSON como una cadena. Para obtener más información, consulte Fusiona las actualizaciones.

- `runWith`— (Opcional) Las opciones de proceso del sistema que el software AWS IoT Greengrass principal utiliza para ejecutar los procesos de este componente en el dispositivo principal. Si omite un parámetro en el `runWith` objeto, el software AWS IoT Greengrass Core utilizará los valores predeterminados que configure en el componente núcleo de Greengrass.

  Puede especificar cualquiera de las siguientes opciones:

  - `posixUser`— El usuario del sistema POSIX y, opcionalmente, el grupo que se utilizará para ejecutar este componente en los dispositivos principales de Linux. El usuario y el grupo, si se especifican, deben existir en cada dispositivo principal de Linux. Especifique el usuario y el grupo separados por dos puntos (`:`) con el siguiente formato: `user:group`. El grupo es opcional. Si no especifica un grupo, el software AWS IoT Greengrass Core utiliza el grupo principal para el usuario. Para obtener más información, consulte Configure el usuario que ejecuta los componentes.
  - `windowsUser`— El usuario de Windows que se utilizará para ejecutar este componente en los dispositivos principales de Windows. El usuario debe estar en todos los dispositivos principales de Windows y su nombre y contraseña deben estar almacenados en la instancia del administrador de credenciales de la LocalSystem cuenta. Para obtener más información, consulte Configure el usuario que ejecuta los componentes.

    Esta función está disponible para la versión 2.5.0 y versiones posteriores del componente núcleo de Greengrass.
  - `systemResourceLimits`— Los límites de recursos del sistema que se aplicarán a los procesos de este componente. Puede aplicar límites de recursos del sistema a los

componentes Lambda genéricos y no contenerizados. Para obtener más información, consulte Configure los límites de recursos del sistema para los componentes.

Puede especificar cualquiera de las siguientes opciones:

- `cpus`— La cantidad máxima de tiempo de CPU que los procesos de este componente pueden utilizar en el dispositivo principal. El tiempo total de CPU de un dispositivo principal equivale a la cantidad de núcleos de CPU del dispositivo. Por ejemplo, en un dispositivo principal con 4 núcleos de CPU, puede establecer este valor 2 para limitar los procesos de este componente al 50 por ciento de uso de cada núcleo de CPU. En un dispositivo con 1 núcleo de CPU, puede establecer este valor `0.25` para limitar los procesos de este componente al 25 por ciento de uso de la CPU. Si establece este valor en un número superior al número de núcleos de la CPU, el software AWS IoT Greengrass Core no limita el uso de la CPU del componente.
- `memory`— La cantidad máxima de RAM (en kilobytes) que los procesos de este componente pueden utilizar en el dispositivo principal.

Esta función está disponible para la versión 2.4.0 y versiones posteriores del componente núcleo de Greengrass. AWS IoT Greengrassactualmente no admite esta función en los dispositivos principales de Windows.

Example Ejemplo de actualización de configuración básica

El siguiente `components` objeto de ejemplo especifica la implementación de un componente que espera un parámetro de configuración denominado`pythonVersion`. `com.example.PythonRuntime`

```
{
  "targetArn": "targetArn",
  "deploymentName": "deploymentName",
  "components": {
    "com.example.PythonRuntime": {
      "componentVersion": "1.0.0",
      "configurationUpdate": {
        "merge": "{\"pythonVersion\":\"3.7\"}"
      }
    }
  }
```

```
}
```

Example Ejemplo de actualización de configuración con actualizaciones de restablecimiento y fusión

Considere un ejemplo de componente de tablero industrial`com.example.IndustrialDashboard`, que tiene la siguiente configuración predeterminada.

```
{
  "name": null,
  "mode": "REQUEST",
  "network": {
    "useHttps": true,
    "port": {
      "http": 80,
      "https": 443
    },
  },
  "tags": []
}
```

La siguiente actualización de configuración especifica las siguientes instrucciones:

1. Restablezca la configuración HTTPS a su valor predeterminado (`true`).
2. Restablezca la lista de etiquetas industriales a una lista vacía.
3. Combine una lista de etiquetas industriales que identifiquen los flujos de datos de temperatura y presión de dos calderas.

```
{
  "reset": [
    "/network/useHttps",
    "/tags"
  ],
  "merge": {
    "tags": [
      "/boiler/1/temperature",
      "/boiler/1/pressure",
      "/boiler/2/temperature",
      "/boiler/2/pressure"
```

```
    ]
  }
}
```

El siguiente `components` objeto de ejemplo especifica la implementación de este componente del panel de control industrial y la actualización de la configuración.

```
{
  "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. (Opcional) Defina las políticas de implementación para la implementación. Puede configurar cuándo los dispositivos principales pueden aplicar una implementación de forma segura o qué hacer si un dispositivo principal no puede aplicar la implementación. Para ello, añada un `deploymentPolicies` objeto y`deployment.json`, a continuación, realice una de las siguientes acciones:

   1. (Opcional) Especifique la política de actualización del componente (`componentUpdatePolicy`). Esta política define si la implementación permite o no a los componentes aplazar una actualización hasta que estén listos para actualizarse. Por ejemplo, es posible que los componentes deban agotar recursos o completar acciones críticas antes de poder reiniciarse para aplicar una actualización. Esta política también define el tiempo del que disponen los componentes para responder a una notificación de actualización.

      Esta política es un objeto con los siguientes parámetros:

- `action`— (Opcional) Si se debe notificar o no a los componentes y esperar a que informen cuando estén listos para actualizarse. Puede elegir entre las siguientes opciones:
  - `NOTIFY_COMPONENTS`: la implementación notifica a cada componente antes de que se detenga y actualice ese componente. Los componentes pueden usar la operación SubscribeToComponentUpdates IPC para recibir estas notificaciones.
  - `SKIP_NOTIFY_COMPONENTS`: la implementación no notifica a los componentes ni espera a que se confirme que se pueden actualizar con seguridad.

  El valor predeterminado es `NOTIFY_COMPONENTS`.
- `timeoutInSeconds`El tiempo en segundos que cada componente tiene para responder a una notificación de actualización con la operación de DeferComponentUpdate IPC. Si el componente no responde dentro de este período de tiempo, la implementación continúa en el dispositivo principal.

  El valor predeterminado es 60 segundos.

2. (Opcional) Especifique la política de validación de la configuración (`configurationValidationPolicy`). Esta política define cuánto tiempo tiene cada componente para validar una actualización de configuración de una implementación. Los componentes pueden usar la operación SubscribeToValidateConfigurationUpdates IPC para suscribirse a las notificaciones de sus propias actualizaciones de configuración. A continuación, los componentes pueden utilizar la operación SendConfigurationValidityReport IPC para indicar al software AWS IoT Greengrass principal si la actualización de configuración es válida. Si la actualización de la configuración no es válida, se produce un error en la implementación.

   Esta política es un objeto con el siguiente parámetro:

   - `timeoutInSeconds`(Opcional) La cantidad de tiempo en segundos que cada componente tiene para validar una actualización de configuración. Si el componente no responde dentro de este período de tiempo, la implementación continúa en el dispositivo principal.

     El valor predeterminado es 30 segundos.

3. (Opcional) Especifique la política de gestión de errores (`failureHandlingPolicy`). Esta política es una cadena que define si se deben revertir o no los dispositivos en caso de que se produzca un error en la implementación. Puede elegir entre las siguientes opciones:

   - `ROLLBACK`— Si la implementación falla en un dispositivo principal, el software AWS IoT Greengrass Core restaura ese dispositivo principal a su configuración anterior.

- `DO_NOTHING`— Si la implementación falla en un dispositivo principal, el software AWS IoT Greengrass Core conserva la nueva configuración. Esto puede provocar que los componentes se rompan si la nueva configuración no es válida.

El valor predeterminado es `ROLLBACK`.

Su implementación `deployment.json` puede tener un aspecto similar al del siguiente ejemplo:

```
{
  "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. (Opcional) Defina cómo se detiene, se despliega o se agota el tiempo de espera de la implementación. AWS IoT Greengrassusa los AWS IoT Core trabajos para enviar las implementaciones a los dispositivos principales, por lo que estas opciones son idénticas a las opciones de configuración de los AWS IoT Core trabajos. Para obtener más información,

consulte Configuración de implementación y cancelación de trabajos en la Guía para AWS IoT desarrolladores.

Para definir las opciones de trabajo, añada un `iotJobConfiguration` objeto a. `deployment.json` A continuación, defina las opciones que desee configurar.

Su implementación `deployment.json` puede tener un aspecto similar al siguiente ejemplo:

```
{
  "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. (Opcional) Agregue etiquetas (`tags`) para la implementación. Para obtener más información, consulte Etiquetar los recursos de AWS IoT Greengrass Version 2.
8. Ejecute el siguiente comando para crear la implementación desde`deployment.json`.

   ```
   aws greengrassv2 create-deployment --cli-input-json file://deployment.json
   ```

   La respuesta incluye una `deploymentId` que identifica esta implementación. Puede usar el ID de implementación para comprobar el estado de la implementación. Para obtener más información, consulte Verificar el estado de implementación.

Actualizar las configuraciones de los componentes

Las configuraciones de los componentes son objetos JSON que definen los parámetros de cada componente. La receta de cada componente define su configuración predeterminada, que se modifica al implementar los componentes en los dispositivos principales.

Al crear una implementación, puede especificar la actualización de configuración que se aplicará a cada componente. Las actualizaciones de configuración son operaciones de parches, lo que significa que la actualización modifica la configuración del componente que existe en el dispositivo principal. Si el dispositivo principal no tiene el componente, la actualización de configuración modifica y aplica la configuración predeterminada para esa implementación.

La actualización de configuración define las actualizaciones de restablecimiento y las actualizaciones de fusión. Las actualizaciones de restablecimiento definen qué valores de configuración se deben restablecer a sus valores predeterminados o eliminar. Las actualizaciones de combinación definen los nuevos valores de configuración que se deben establecer para el componente. Al implementar una actualización de configuración, el software AWS IoT Greengrass principal ejecuta la actualización de restablecimiento antes de la actualización de fusión.

Los componentes pueden validar las actualizaciones de configuración que se implementen. El componente se suscribe para recibir una notificación cuando una implementación cambia su configuración y puede rechazar una configuración que no sea compatible. Para obtener más información, consulte Interactúa con la configuración de los componentes.

Temas

- Restablecer las actualizaciones
- Fusiona las actualizaciones
- Ejemplos

Restablecer las actualizaciones

Las actualizaciones de restablecimiento definen qué valores de configuración se deben restablecer a sus valores predeterminados en el dispositivo principal. Si un valor de configuración no tiene un valor predeterminado, la actualización de restablecimiento elimina ese valor de la configuración del componente. Esto puede ayudarle a reparar un componente que se interrumpe como resultado de una configuración no válida.

Utilice una lista de punteros JSON para definir qué valores de configuración desea restablecer. Los punteros JSON comienzan con una barra diagonal. / Para identificar un valor en una configuración de componentes anidados, utilice barras diagonales (/) para separar las claves de cada nivel de la configuración. Para obtener más información, consulta la especificación del puntero JSON.

> **Note**
>
> Solo puede restablecer los valores predeterminados de una lista completa. No puedes usar las actualizaciones de restablecimiento para restablecer un elemento individual de una lista.

Para restablecer toda la configuración de un componente a sus valores predeterminados, especifique una sola cadena vacía como actualización de restablecimiento.

```
"reset": [""]
```

Fusiona las actualizaciones

Las actualizaciones de combinación definen los valores de configuración que se van a insertar en la configuración de los componentes del núcleo. La actualización de fusión es un objeto JSON que el software AWS IoT Greengrass principal fusiona después de restablecer los valores de las rutas que se especificaron en la actualización de restablecimiento. Cuando utilices los AWS CLI o los AWS SDK, debes serializar este objeto JSON como una cadena.

Puede combinar un par clave-valor que no exista en la configuración predeterminada del componente. También puede combinar un par clave-valor que tenga un tipo diferente al valor con la misma clave. El nuevo valor reemplaza al valor anterior. Esto significa que puede cambiar la estructura del objeto de configuración.

Puede combinar valores nulos y cadenas, listas y objetos vacíos.

> **Note**
>
> No puede utilizar las actualizaciones de combinación para insertar o añadir un elemento a una lista. Puede reemplazar una lista completa o definir un objeto en el que cada elemento tenga una clave única.
> AWS IoT Greengrass usa JSON para los valores de configuración. JSON especifica un tipo de número, pero no diferencia entre números enteros y flotantes. Como resultado, los valores de configuración pueden convertirse en valores flotantes. AWS IoT Greengrass Para garantizar que su componente utilice el tipo de datos correcto, le recomendamos que defina los valores de configuración numéricos como cadenas. A continuación, pida a su componente que los analice como enteros o flotantes. Esto garantiza que los valores de configuración sean del mismo tipo en la configuración y en el dispositivo principal.

Utilice variables de receta en las actualizaciones de fusión

Esta función está disponible para la versión 2.6.0 y versiones posteriores del componente núcleo de Greengrass.

Si establece la opción de ComponentConfiguration configuración de interpolación del núcleo de Greengrass `entrue`, puede utilizar variables de receta distintas de la variable de receta en las

actualizaciones de *component_dependency_name*:configuration:*json_pointer* fusión. Por ejemplo, puede usar la variable de {iot:thingName} receta en una actualización de fusión para incluir el nombre del dispositivo principal en el valor de AWS IoT la configuración de un componente, como una política de autorización de comunicación entre procesos (IPC).

Ejemplos

En el siguiente ejemplo, se muestran las actualizaciones de configuración de un componente del panel de mandos que tiene la siguiente configuración predeterminada. Este componente de ejemplo muestra información sobre el equipo industrial.

```
{
  "name": null,
  "mode": "REQUEST",
  "network": {
    "useHttps": true,
    "port": {
      "http": 80,
      "https": 443
    },
  },
  "tags": []
}
```

Receta de componentes de salpicadero industrial

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 Ejemplo 1: actualización de combinación

Se crea una implementación que aplica la siguiente actualización de configuración, que especifica una actualización de fusión pero no una actualización de restablecimiento. Esta actualización de configuración indica al componente que muestre el panel de control en el puerto HTTP 8080 con los datos de dos calderas.

Console

Configuración que se va a fusionar

```
{
  "name": "Factory 2A",
  "network": {
    "useHttps": false,
    "port": {
      "http": 8080
    }
  },
  "tags": [
    "/boiler/1/temperature",
    "/boiler/1/pressure",
    "/boiler/2/temperature",
    "/boiler/2/pressure"
  ]
}
```

AWS CLI

El siguiente comando crea una implementación en un dispositivo principal.

```
aws greengrassv2 create-deployment --cli-input-json file://dashboard-deployment.json
```

El `dashboard-deployment.json` archivo contiene el siguiente documento JSON.

```
{
  "targetArn": "arn:aws:iot:us-west-2:123456789012:thing/MyGreengrassCore",
  "deploymentName": "Deployment for MyGreengrassCore",
  "components": {
    "com.example.IndustrialDashboard": {
      "componentVersion": "1.0.0",
      "configurationUpdate": {
        "merge": "{\"name\":\"Factory 2A\",\"network\":{\"useHttps\":false,\"port
\":{\"http\":8080}},\"tags\":[\"/boiler/1/temperature\",\"/boiler/1/pressure\",\"/
boiler/2/temperature\",\"/boiler/2/pressure\"]}"
      }
    }
  }
}
```

Greengrass CLI

El siguiente comando CLI de Greengrass crea una implementación local en un dispositivo principal.

```
sudo greengrass-cli deployment create \
  --recipeDir recipes \
  --artifactDir artifacts \
  --merge "com.example.IndustrialDashboard=1.0.0" \
  --update-config dashboard-configuration.json
```

El `dashboard-configuration.json` archivo contiene el siguiente documento JSON.

```
{
  "com.example.IndustrialDashboard": {
    "MERGE": {
      "name": "Factory 2A",
      "network": {
        "useHttps": false,
```

```
        "port": {
          "http": 8080
        }
      },
      "tags": [
        "/boiler/1/temperature",
        "/boiler/1/pressure",
        "/boiler/2/temperature",
        "/boiler/2/pressure"
      ]
    }
  }
}
```

Tras esta actualización, el componente del panel de mandos tiene la siguiente configuración.

```
{
  "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 Ejemplo 2: restablecer y combinar las actualizaciones

A continuación, se crea una implementación que aplica la siguiente actualización de configuración, que especifica una actualización de restablecimiento y una actualización de fusión. Estas actualizaciones especifican mostrar el panel de control en el puerto HTTPS predeterminado con datos de diferentes fuentes. Estas actualizaciones modifican la configuración resultante de las actualizaciones de configuración del ejemplo anterior.

Console

Restablecer las rutas

```
[
  "/network/useHttps",
  "/tags"
]
```

Configuración para fusionar

```
{
  "tags": [
    "/boiler/3/temperature",
    "/boiler/3/pressure",
    "/boiler/4/temperature",
    "/boiler/4/pressure"
  ]
}
```

AWS CLI

El siguiente comando crea una implementación en un dispositivo principal.

```
aws greengrassv2 create-deployment --cli-input-json file://dashboard-
deployment2.json
```

El `dashboard-deployment2.json` archivo contiene el siguiente documento JSON.

```
{
  "targetArn": "arn:aws:iot:us-west-2:123456789012:thing/MyGreengrassCore",
  "deploymentName": "Deployment for MyGreengrassCore",
  "components": {
    "com.example.IndustrialDashboard": {
      "componentVersion": "1.0.0",
      "configurationUpdate": {
        "reset": [
          "/network/useHttps",
          "/tags"
        ],
```

```
        "merge": "{\"tags\":[\"/boiler/3/temperature\",\"/boiler/3/pressure\",\"/
boiler/4/temperature\",\"/boiler/4/pressure\"]}"
      }
    }
  }
}
```

Greengrass CLI

El siguiente comando CLI de Greengrass crea una implementación local en un dispositivo principal.

```
sudo greengrass-cli deployment create \
  --recipeDir recipes \
  --artifactDir artifacts \
  --merge "com.example.IndustrialDashboard=1.0.0" \
  --update-config dashboard-configuration2.json
```

El `dashboard-configuration2.json` archivo contiene el siguiente documento JSON.

```
{
  "com.example.IndustrialDashboard": {
    "RESET": [
      "/network/useHttps",
      "/tags"
    ],
    "MERGE": {
      "tags": [
        "/boiler/3/temperature",
        "/boiler/3/pressure",
        "/boiler/4/temperature",
        "/boiler/4/pressure"
      ]
    }
  }
}
```

Tras esta actualización, el componente del panel de mandos tiene la siguiente configuración.

```
{
  "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",
  ]
}
```

Crear subdespliegues

> **Note**
>
> La función de subdespliegue está disponible en la versión 2.9.0 y posteriores del núcleo de Greengrass. No es posible implementar una configuración en una subimplementación con versiones de componentes anteriores del núcleo de Greengrass.

Una subimplementación es una implementación que se dirige a un subconjunto más pequeño de dispositivos dentro de una implementación principal. Puede usar las subimplementaciones para implementar una configuración en un subconjunto más pequeño de dispositivos. También puede crear subimplementaciones para volver a intentar una implementación principal que no funciona cuando fallan uno o más dispositivos de esa implementación principal. Con esta función, puede seleccionar los dispositivos que fallaron en esa implementación principal y crear una subimplementación para probar las configuraciones hasta que la subimplementación se realice correctamente. Una vez que la subimplementación se haya realizado correctamente, puede volver a implementar esa configuración en la implementación principal.

Siga los pasos de esta sección para crear una subimplementación y comprobar su estado. Para obtener más información sobre cómo crear despliegues, consulte Crear despliegues.

Para crear una subimplementación () AWS CLI

1. Ejecute el siguiente comando para recuperar las últimas implementaciones de un grupo de cosas. Sustituya el ARN del comando por el ARN del grupo de cosas que se va a consultar. `--history-filter`**`LATEST_ONLY`**Configúrelo para ver la última implementación de ese grupo de cosas.

   ```
   aws greengrassv2 list-deployments --target-arn arn:aws:iot:region:account-id:thinggroup/thingGroupName --history-filter LATEST_ONLY
   ```

2. Copia el fragmento `deploymentId` de la respuesta al list-deployments comando para usarlo en el siguiente paso.

3. Ejecute el siguiente comando para recuperar el estado de una implementación. *`deploymentId`*Sustitúyalo por el ID de la implementación que se va a consultar.

   ```
   aws greengrassv2 get-deployment --deployment-id deploymentId
   ```

4. Copie el `iotJobId` de la respuesta al get-deployment comando para usarlo en el siguiente paso.

5. Ejecute el siguiente comando para recuperar la lista de ejecuciones de tareas para la tarea especificada. Sustituya *`JobID por`* el `iotJobId` del paso anterior. *`Sustituya el estado por el que desee filtrar.`* Puede filtrar los resultados con los siguientes estados:

   - `QUEUED`
   - `IN_PROGRESS`
   - `SUCCEEDED`
   - `FAILED`
   - `TIMED_OUT`
   - `REJECTED`
   - `REMOVED`
   - `CANCELED`

   ```
   aws iot list-job-executions-for-job --job-id jobID --status status
   ```

6. Cree un grupo de AWS IoT cosas nuevo o utilice un grupo de cosas existente para la subimplementación. A continuación, añada un AWS IoT elemento a este grupo de elementos.

Los grupos de cosas se utilizan para gestionar las flotas de los dispositivos principales de Greengrass. Al implementar componentes de software en sus dispositivos, puede dirigirlos a dispositivos individuales o a grupos de dispositivos. Puede añadir un dispositivo a un grupo de cosas con una implementación activa de Greengrass. Una vez agregado, puede implementar los componentes de software de ese grupo de cosas en ese dispositivo.

Para crear un nuevo grupo de cosas y añadirle tus dispositivos, haz lo siguiente:

a. Crea un grupo de AWS IoT cosas. *MyGreengrassCoreGroup*Sustitúyalo por el nombre del nuevo grupo de cosas. No se pueden usar dos puntos (:)) en el nombre de un grupo de cosas.

> **Note**
>
> Si un grupo de cosas de un subdespliegue se usa con un`parentTargetArn`, no se puede reutilizar con una flota principal diferente. Si un grupo de cosas ya se ha utilizado para crear un subdespliegue para otra flota, la API devolverá un error.

```
aws iot create-thing-group --thing-group-name MyGreengrassCoreGroup
```

Si la solicitud se realiza correctamente, la respuesta es similar a la del siguiente ejemplo:

```
{
  "thingGroupName": "MyGreengrassCoreGroup",
  "thingGroupArn": "arn:aws:iot:us-
west-2:123456789012:thinggroup/MyGreengrassCoreGroup",
  "thingGroupId": "4df721e1-ff9f-4f97-92dd-02db4e3f03aa"
}
```

b. Añada un núcleo de Greengrass aprovisionado a su grupo de cosas. Ejecute el siguiente comando con estos parámetros:

- *MyGreengrassCore*Sustitúyalo por el nombre del núcleo de Greengrass aprovisionado.
- *MyGreengrassCoreGroup*Sustitúyalo por el nombre de su grupo de cosas.

```
aws iot add-thing-to-thing-group --thing-name MyGreengrassCore --thing-group-
name MyGreengrassCoreGroup
```

El comando no tiene ningún resultado si la solicitud se realiza correctamente.

7. Cree un archivo llamado y`deployment.json`, a continuación, copie el siguiente objeto JSON en el archivo. Sustituya *`TargetARN`* por el ARN del grupo de cosas al que apuntar AWS IoT la subimplementación. Un objetivo de subdespliegue solo puede ser un grupo de cosas. Los ARN de los grupos de cosas tienen el siguiente formato:

   - Grupo de cosas: `arn:aws:iot:`*`region`*`:`*`account-id`*`:thinggroup/`*`thingGroupName`*

   ```
   {
     "targetArn": "targetArn"
   }
   ```

8. Vuelva a ejecutar el siguiente comando para obtener los detalles de la implementación original. Estos detalles incluyen los metadatos, los componentes y la configuración del trabajo. Sustituya *`DeploymentiD`* por el ID de. Step 1 Puede usar esta configuración de despliegue para configurar su subdespliegue y realizar los cambios necesarios.

   ```
   aws greengrassv2 get-deployment --deployment-id deploymentId
   ```

   La respuesta contiene los detalles de la implementación. Copie cualquiera de los siguientes pares clave-valor de la respuesta del get-deployment comando a. `deployment.json` Puede cambiar estos valores para la subimplementación. Para obtener más información sobre los detalles de este comando, consulte GetDeployment.

   - `components`— Los componentes de la implementación. Para desinstalar un componente, elimínelo de este objeto.
   - `deploymentName`— El nombre de la implementación.
   - `deploymentPolicies`— Las políticas de la implementación.
   - `iotJobConfiguration`— La configuración del trabajo de la implementación.
   - `parentTargetArn`— El objetivo de la implementación principal.
   - `tags`— Las etiquetas del despliegue.

9. Ejecute el siguiente comando para crear la subimplementación desde`deployment.json`. Sustituya *`subDeploymentName por un nombre`* para la subimplementación.

```
aws greengrassv2 create-deployment --deployment-name subdeploymentName --cli-input-
json file://deployment.json
```

La respuesta incluye un `deploymentId` que identifica esta subimplementación. Puede usar el ID de despliegue para comprobar el estado del despliegue. Para obtener más información, consulte Comprobar el estado de la implementación.

10. Si la subimplementación se realiza correctamente, puede usar su configuración para revisar la implementación principal. Copie la `deployment.json` que utilizó en el paso anterior. Sustituya `targetArn` el del archivo JSON por el ARN de la implementación principal y ejecute el siguiente comando para crear la implementación principal con esta nueva configuración.

> **Note**
>
> Si crea una nueva revisión de despliegue de la flota principal, sustituirá a todas las revisiones y subdespliegues del despliegue principal. Para obtener más información, consulte Revisar las implementaciones.

```
aws greengrassv2 create-deployment --cli-input-json file://deployment.json
```

La respuesta incluye una `deploymentId` que identifica esta implementación. Puede usar el ID de implementación para comprobar el estado de la implementación. Para obtener más información, consulte Verificar el estado de implementación.

Revisar las implementaciones

Cada cosa o grupo de cosas objetivo puede tener una implementación activa a la vez. Al crear una implementación para un destino que ya tiene una implementación, los componentes de software de la nueva implementación reemplazan a los de la implementación anterior. Si la nueva implementación no define un componente que definió la implementación anterior, el software AWS IoT Greengrass principal elimina ese componente de los dispositivos principales de destino. Puede revisar una implementación existente para no eliminar los componentes que se ejecutan en los dispositivos principales de una implementación anterior a un destino.

Para revisar una implementación, debe crear una implementación que comience con los mismos componentes y configuraciones que existían en una implementación anterior. Se

utiliza la CreateDeploymentoperación, que es la misma operación que se utiliza para crear las implementaciones.

Para revisar una implementación () AWS CLI

1. Ejecute el siguiente comando para enumerar las implementaciones del destino de implementación. Sustituya *TargetARN* por el ARN de la cosa o grupo de cosas objetivo. AWS IoT

   ```
   aws greengrassv2 list-deployments --target-arn targetArn
   ```

   La respuesta contiene una lista con la implementación más reciente del objetivo. Copia el `deploymentId` fragmento de la respuesta para usarlo en el siguiente paso.

   > **Note**
   >
   > También puede revisar una implementación que no sea la última revisión del destino. Especifique el `--history-filter ALL` argumento para enumerar todos los despliegues del objetivo. A continuación, copie el ID de la implementación que desea revisar.

2. Ejecute el siguiente comando para obtener los detalles de la implementación. Estos detalles incluyen los metadatos, los componentes y la configuración del trabajo. Sustituya *DeploymentiD* por el ID del paso anterior.

   ```
   aws greengrassv2 get-deployment --deployment-id deploymentId
   ```

   La respuesta contiene los detalles de la implementación.

3. Cree un archivo llamado `deployment.json` y copie la respuesta del comando anterior en el archivo.

4. Elimine los siguientes pares de clave-valor del objeto JSON en `deployment.json`:

   - `deploymentId`
   - `revisionId`
   - `iotJobId`
   - `iotJobArn`
   - `creationTimestamp`

- `isLatestForTarget`
- `deploymentStatus`

La CreateDeploymentoperación espera una carga útil con la siguiente estructura.

```
{
  "targetArn": "String",
  "components": Map of components,
  "deploymentPolicies": DeploymentPolicies,
  "iotJobConfiguration": DeploymentIoTJobConfiguration,
  "tags": Map of tags
}
```

5. En `deployment.json`, realice una de las siguientes acciones:
   - Cambie el nombre de la implementación (`deploymentName`).
   - Cambie los componentes de la implementación (`components`).
   - Cambie las políticas de la implementación (`deploymentPolicies`).
   - Cambie la configuración de tareas de la implementación (`iotJobConfiguration`).
   - Cambie las etiquetas de la implementación (`tags`).

   Para obtener más información sobre cómo definir estos detalles de despliegue, consulteCrear implementaciones.
6. Ejecute el siguiente comando para crear la implementación desde`deployment.json`.

```
aws greengrassv2 create-deployment --cli-input-json file://deployment.json
```

   La respuesta incluye una `deploymentId` que identifica esta implementación. Puede usar el ID de implementación para comprobar el estado de la implementación. Para obtener más información, consulte Verificar el estado de implementación.

Cancelar implementaciones

Puede cancelar una implementación activa para evitar que sus componentes de software se instalen en los dispositivosAWS IoT Greengrass principales. Si cancela una implementación dirigida a un grupo de cosas, los dispositivos principales que añada al grupo no recibirán esa implementación continua. Si un dispositivo principal ya ejecuta la implementación, no cambiarás los componentes de

ese dispositivo cuando canceles la implementación. Debe crear una implementación nueva o revisar la implementación para modificar los componentes que se ejecutan en los dispositivos principales que recibieron la implementación cancelada.

Para cancelar una implementación (AWS CLI)

1. Ejecute el siguiente comando para buscar el ID de la última revisión de implementación de un destino. La última revisión es la única implementación que puede estar activa para un destino, ya que las implementaciones anteriores se cancelan al crear una nueva revisión. Reemplace *targetArn* por el ARN del objeto o el grupo del objeto de destinoAWS IoT.

   ```
   aws greengrassv2 list-deployments --target-arn targetArn
   ```

   La respuesta contiene una lista con la implementación más reciente para el objetivo. `deploymentId`Copie el objeto de la respuesta para usarlo en el paso siguiente.
2. Ejecute el comando siguiente para cancelar la implementación. *Sustituya DeploymentID* por el ID del paso anterior.

   ```
   aws greengrassv2 cancel-deployment --deployment-id deploymentId
   ```

   Si la operación es exitosa, el estado del despliegue cambia a`CANCELED`.

Verificar el estado de implementación

Puede comprobar el estado de una implementación en la que creeAWS IoT Greengrass. También puede comprobar el estado de lasAWS IoT tareas que implementan la implementación en cada dispositivo principal. Mientras una implementación esté activa, el estado delAWS IoT trabajo es`IN_PROGRESS`. Tras crear una nueva revisión de una implementación, el estado delAWS IoT trabajo de la revisión anterior cambia a`CANCELLED`.

Temas

- Verificar el estado de implementación
- Compruebe el estado de despliegue del dispositivo

Verificar el estado de implementación

Puede comprobar el estado de una implementación que identifique por su objetivo o su ID.

Para comprobar el estado de la implementación por target (AWS CLI)

- Ejecute el siguiente comando para recuperar el estado de la implementación más reciente de un destino. Sustituya *targetArn* por el nombre de recurso de Amazon (ARN) de laAWS IoT cosa o el grupo de cosas al que se dirige la implementación.

  ```
  aws greengrassv2 list-deployments --target-arn targetArn
  ```

  La respuesta contiene una lista con la implementación más reciente para el objetivo. Este objeto de despliegue incluye el estado del despliegue.

Para comprobar el estado de la implementación por ID (AWS CLI)

- Ejecute el siguiente comando para recuperar el estado de una implementación. Sustituya *DeploymentID* por el ID de la implementación a consultar.

  ```
  aws greengrassv2 get-deployment --deployment-id deploymentId
  ```

  La respuesta contiene el estado de la implementación.

Compruebe el estado de despliegue del dispositivo

Puede comprobar el estado de un trabajo de implementación que se aplica a un dispositivo de núcleo individual. También puede comprobar el estado de una tarea de implementación para una implementación de un grupo de cosas.

Para comprobar el estado de los trabajos de implementación de un dispositivo principal (AWS CLI)

- Ejecute el siguiente comando para recuperar el estado de todos los trabajos de implementación de un dispositivo de núcleo. *coreDeviceName*Sustitúyalo por el nombre del dispositivo principal que se va a consultar.

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

  La respuesta contiene la lista de trabajos de implementación del dispositivo principal. Puede identificar el trabajo de una implementación por su nombre`deploymentId` o`targetArn`. Cada tarea de implementación contiene el estado de la tarea en el dispositivo principal.

Para comprobar los estados de implementación de un grupo de cosas (AWS CLI)

1. Ejecute el siguiente comando para recuperar el ID de una implementación existente. Reemplace *targetArn* por el ARN del grupo de cosas de destino.

   ```
   aws greengrassv2 list-deployments --target-arn targetArn
   ```

   La respuesta contiene una lista con la implementación más reciente para el objetivo. Copie el`deploymentId` de la respuesta para usarlo en el siguiente paso.

   > **Note**
   >
   > También puede incluir una implementación que no sea la implementación más reciente para el destino. Especifique el`--history-filter ALL` argumento para enumerar todas las implementaciones del objetivo. A continuación, copie el ID de la implementación cuyo estado desea comprobar.

2. Ejecute el siguiente comando para obtener los detalles de la implementación. *Reemplace Dep* por el ID del paso anterior.

   ```
   aws greengrassv2 get-deployment --deployment-id deploymentId
   ```

   La respuesta contiene información acerca de la implementación. Copie el`iotJobId` de la respuesta para usarlo en el paso siguiente.

3. Ejecute el siguiente comando para describir la ejecución del trabajo de un dispositivo principal para la implementación. Reemplace *iotJobId*y asigne un *coreDeviceThingnombre* por el ID de trabajo del paso anterior y el dispositivo principal del que desea comprobar el estado.

   ```
   aws iot describe-job-execution --job-id iotJobId --thing-name coreDeviceThingName
   ```

   La respuesta contiene el estado de la ejecución del trabajo de despliegue del dispositivo principal y detalles sobre el estado. `detailsMap`Contiene la siguiente información:

   - `detailed-deployment-status`— El estado del resultado de la implementación, que puede tener uno de los siguientes valores:
     - `SUCCESSFUL`— El despliegue se ha realizado correctamente.

- `FAILED_NO_STATE_CHANGE`— La implementación falló mientras el dispositivo principal se preparaba para aplicar la implementación.
- `FAILED_ROLLBACK_NOT_REQUESTED`— La implementación falló y la implementación no especificó volver a una configuración de trabajo anterior, por lo que es posible que el dispositivo principal no funcione correctamente.
- `FAILED_ROLLBACK_COMPLETE`— Se produjo un error en la implementación y el dispositivo principal se restableció correctamente a una configuración de trabajo anterior.
- `FAILED_UNABLE_TO_ROLLBACK`— La implementación falló y el dispositivo principal no pudo volver a una configuración de trabajo anterior, por lo que es posible que el dispositivo principal no funcione correctamente.

Si la implementación falló, compruebe el`deployment-failure-cause` valor y los archivos de registro del dispositivo principal para identificar el problema. Para obtener más información acerca de cómo acceder a los archivos de registro del dispositivo principal, consulteSupervisar AWS IoT Greengrass registros.

- `deployment-failure-cause`— Un mensaje de error que proporciona detalles adicionales sobre el motivo del error en la ejecución del trabajo.

La respuesta tiene un aspecto similar al siguiente ejemplo.


```
{
  "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 Cuenta de 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
  }
}
```

Registro y monitorización en AWS IoT Greengrass

La supervisión es un aspecto importante del mantenimiento de la fiabilidad, la disponibilidad y el rendimiento de AWS IoT Greengrass y sus soluciones de AWS. Debe recopilar datos de monitorización de todas las partes de su solución de AWS para que le resulte más sencillo depurar cualquier error que se produzca en distintas partes del código, en caso de que ocurra. Antes de empezar a monitorear AWS IoT Greengrass, debe crear un plan de monitoreo que incluya respuestas a las siguientes preguntas:

- ¿Cuáles son los objetivos de la monitorización?
- ¿Qué recursos va a monitorizar?
- ¿Con qué frecuencia va a monitorizar estos recursos?
- ¿Qué herramientas de monitorización va a utilizar?
- ¿Quién se encargará de realizar las tareas de monitoreo?
- ¿Quién debería recibir una notificación cuando surjan problemas?

Temas

Herramientas de monitoreo

AWS proporciona herramientas que puede utilizar para monitorizar AWS IoT Greengrass. Puede configurar algunas de estas herramientas para que realicen la monitorización por usted. Algunas de las herramientas requieren intervención manual. Le recomendamos que automatice las tareas de supervisión en la medida de lo posible.

Puede utilizar las siguientes herramientas de monitorización automatizada para monitorizar AWS IoT Greengrass e informar de los problemas:

- Amazon CloudWatch Logs: supervise, almacene y acceda a sus archivos de registro desde AWS CloudTrail u otras fuentes. Para obtener más información, consulta Supervisión de archivos de registro en la Guía del CloudWatch usuario de Amazon.
- AWS CloudTrailSupervisión de registros: comparta archivos de registro entre cuentas, supervise los archivos de CloudTrail registro en tiempo real enviándolos a CloudWatch Logs, cree aplicaciones de procesamiento de registros en Java y valide que sus archivos de registro no hayan cambiado después de su entrega CloudTrail. Para obtener más información, consulte Trabajar con archivos de CloudTrail registro en la Guía del AWS CloudTrail usuario.
- Telemetría de salud del sistema Greengrass: suscríbase para recibir los datos de telemetría enviados desde el núcleo de Greengrass. Para obtener más información, consulte the section called “Recopile datos de telemetría de estado del sistema”.
- Notificaciones de estado del dispositivo Crea eventos con Amazon EventBridge para recibir actualizaciones de estado relacionadas con las implementaciones y los componentes. Para obtener más información, consulte Reciba notificaciones de despliegue y estado de los componentes.
- Servicio de estado de la flota: utilice las operaciones de la API de estado de la flota para comprobar el estado de los dispositivos principales y sus componentes de Greengrass. También puede ver la información sobre el estado de la flota en la AWS IoT Greengrass consola. Para obtener más información, consulte Compruebe el estado del dispositivo principal de Greengrass.

Supervisar AWS IoT Greengrass registros

AWS IoT Greengrass consta del servicio de nube y el software de AWS IoT Greengrass Core. El software AWS IoT Greengrass Core puede escribir registros en Amazon CloudWatch Logs y en el sistema de archivos local del dispositivo principal. Los componentes de Greengrass que se ejecutan en el dispositivo principal también pueden escribir registros en Logs y en el sistema de archivos local. CloudWatch Puede utilizar registros para monitorizar eventos y solucionar problemas. Todas las entradas de registro de AWS IoT Greengrass incluyen una marca temporal, un nivel de registro e información sobre el evento.

De forma predeterminada, el software AWS IoT Greengrass Core escribe los registros únicamente en el sistema de archivos local. Puede ver los registros del sistema de archivos en tiempo real para poder depurar los componentes de Greengrass que desarrolle e implemente. También puede configurar un dispositivo principal para escribir registros en los CloudWatch registros, de modo que pueda solucionar los problemas del dispositivo principal sin acceso al sistema de archivos local. Para obtener más información, consulte Habilite el registro en los CloudWatch registros.

Temas

Acceda a los registros del sistema de archivos

El software AWS IoT Greengrass Core almacena los registros en la *`/greengrass/v2`*`/logs` carpeta de un dispositivo principal, donde *`/greengrass/v2`* se encuentra la ruta a la carpeta AWS IoT Greengrass raíz. La carpeta de registros tiene la siguiente estructura.

```
/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`— El archivo de registro del software AWS IoT Greengrass principal. Utilice este archivo de registro para ver información en tiempo real sobre los componentes y las implementaciones. Este archivo de registro incluye los registros del núcleo de Greengrass, que es el núcleo del software principal, y los AWS IoT Greengrass componentes del complemento, como el administrador de registros y el administrador de secretos.
- *`ComponentName`*`.log`— Archivos de registro de componentes de Greengrass. Utilice los archivos de registro de componentes para ver información en tiempo real sobre un componente de Greengrass que se ejecuta en el dispositivo principal. Los componentes genéricos y los componentes Lambda escriben la salida estándar (stdout) y el error estándar (stderr) en estos archivos de registro.
- `main.log`— El archivo de registro del `main` servicio que gestiona los ciclos de vida de los componentes. Este archivo de registro siempre estará vacío.

Para obtener más información sobre las diferencias entre los componentes de complemento, genéricos y de Lambda, consulte. Tipos de componentes

Las siguientes consideraciones se aplican cuando se utilizan los registros del sistema de archivos:

- Permisos de usuario root

  Debe tener permisos de raíz para leer registros de AWS IoT Greengrass en el sistema de archivos.

- Rotación de archivos de registro

  El software AWS IoT Greengrass Core rota los archivos de registro cada hora o cuando superan un límite de tamaño de archivo. Los archivos de registro rotados contienen una marca de tiempo en el nombre del archivo. Por ejemplo, un archivo de registro del software AWS IoT Greengrass Core girado podría tener un nombre. `greengrass_2021_09_14_15_0.log` El límite de tamaño de archivo predeterminado es de 1024 KB (1 MB). Puede configurar el límite de tamaño de archivo en el componente núcleo de Greengrass.

- Eliminación del archivo de registro

  El software AWS IoT Greengrass Core limpia los archivos de registro anteriores cuando el tamaño de los archivos de registro del software AWS IoT Greengrass Core o de los archivos de registro de los componentes de Greengrass, incluidos los archivos de registro rotados, supera el límite de espacio en disco. El límite de espacio en disco predeterminado para el registro del software AWS IoT Greengrass principal y para cada registro de componentes es de 10 240 KB (10 MB). Puede configurar el límite de espacio en disco de registro del software AWS IoT Greengrass principal en el componente núcleo de Greengrass o en el componente de administrador de registros. Puede configurar el límite de espacio en disco de registro de cada componente en el componente del administrador de registros.

Para ver el archivo de registro del software AWS IoT Greengrass principal

- Ejecute el siguiente comando para ver el archivo de registro en tiempo real. */greengrass/v2*Sustitúyalo por la ruta a la carpeta AWS IoT Greengrass raíz.

  Linux or Unix

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

Windows Command Prompt (CMD)

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

El `type` comando escribe el contenido del archivo en la terminal. Ejecute este comando varias veces para observar los cambios en el archivo.

PowerShell

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

Para ver el archivo de registro de un componente

- Ejecute el siguiente comando para ver el archivo de registro en tiempo real. Sustituya */greengrass/v2* o *C:\greengrass\v2* por la ruta de acceso a la carpeta AWS IoT Greengrass raíz y sustituya *com.example. HelloWorld*por el nombre del componente.

  Linux or Unix

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

  PowerShell

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

También puede usar el `logs` comando de la CLI de Greengrass para analizar los registros de Greengrass en un dispositivo principal. Para usar el `logs` comando, debe configurar el núcleo de Greengrass para que genere archivos de registro en formato JSON. Para obtener más información, consulte Interfaz de línea de comandos Greengrass y registros.

Registros de acceso CloudWatch

Puede implementar el componente de administrador de registros para configurar el dispositivo principal para que escriba en CloudWatch los registros. Para obtener más información, consulte Habilite el registro en los CloudWatch registros. A continuación, puedes ver los registros en la página Logs de la CloudWatch consola de Amazon o mediante la API CloudWatch Logs.

Nombre del grupo de registros

```
/aws/greengrass/componentType/region/componentName
```

El nombre del grupo de registros utiliza las siguientes variables:

- `componentType`— El tipo de componente, que puede ser uno de los siguientes:
  - `GreengrassSystemComponent`— Este grupo de registros incluye los registros de los componentes del núcleo y del complemento, que se ejecutan en la misma JVM que el núcleo de Greengrass. El componente forma parte del núcleo de Greengrass.
  - `UserComponent`— Este grupo de registros incluye registros de componentes genéricos, componentes de Lambda y otras aplicaciones del dispositivo. El componente no forma parte del núcleo de Greengrass.

  Para obtener más información, consulte Tipos de componentes.
- `region`— La AWS región que utiliza el dispositivo principal.
- `componentName`— El nombre del componente. Para los registros del sistema, este valor es`System`.

Nombre del flujo de registro

```
/date/thing/thingName
```

El nombre del flujo de registro utiliza las siguientes variables:

- `date`— La fecha del registro, por ejemplo`2020/12/15`. El componente del administrador de registros usa el `yyyy/MM/dd` formato.
- `thingName`— El nombre del dispositivo principal.

> **Note**
>
> Si el nombre de un elemento contiene dos puntos (`:`), el administrador de registros los sustituye por un signo más (`+`).

Cuando se utiliza el componente gestor de registros para escribir en los CloudWatch registros, se tienen en cuenta las siguientes consideraciones:

- Retrasos de registro

> **Note**
>
> Le recomendamos que actualice a la versión 2.3.0 del administrador de registros, ya que reduce las demoras en el registro de los archivos de registro activos y rotados. Cuando actualice a log manager 2.3.0, le recomendamos que también actualice a Greengrass nucleus 2.9.1.

La versión 2.2.8 (y anteriores) del componente gestor de registros procesa y carga los registros únicamente a partir de archivos de registro rotados. De forma predeterminada, el software AWS IoT Greengrass Core rota los archivos de registro cada hora o después de que ocupen 1024 KB. Como resultado, el componente del administrador de registros carga los registros solo después de que el software AWS IoT Greengrass Core o un componente de Greengrass hayan escrito registros con un valor superior a 1024 KB. Puede configurar un límite de tamaño de archivo de registro inferior para que los archivos de registro roten con más frecuencia. Esto hace que el componente del administrador de registros cargue registros en CloudWatch Logs con más frecuencia.

La versión 2.3.0 (y posteriores) del componente gestor de registros procesa y carga todos los registros. Al escribir un registro nuevo, la versión 2.3.0 (y posteriores) del administrador de registros procesa y carga directamente el archivo de registro activo en lugar de esperar a que se rote. Esto significa que puede ver el nuevo registro en 5 minutos o menos.

El componente del administrador de registros carga nuevos registros periódicamente. De forma predeterminada, el componente del administrador de registros carga nuevos registros cada 5 minutos. Puede configurar un intervalo de carga más bajo, de modo que el componente del administrador de registros cargue los registros en los CloudWatch registros con más frecuencia configurando el. `periodicUploadIntervalSec` Para obtener más información sobre cómo configurar este intervalo periódico, consulte Configuración.

Los registros se pueden cargar prácticamente en tiempo real desde el mismo sistema de archivos de Greengrass. Si necesita observar los registros en tiempo real, considere la posibilidad de utilizar los registros del sistema de archivos.

> **Note**
>
> Si utiliza distintos sistemas de archivos para escribir los registros, el administrador de registros vuelve al comportamiento de las versiones 2.2.8 y anteriores de los componentes del administrador de registros. Para obtener información sobre cómo acceder a los registros del sistema de archivos, consulte Acceder a los registros del sistema de archivos.

- Inclinación del reloj

  El componente de gestión de registros utiliza el proceso de firma estándar de la versión 4 de Signature para crear solicitudes de API a CloudWatch los registros. Si la hora del sistema en un dispositivo principal está desincronizada durante más de 15 minutos, CloudWatch Logs rechaza las solicitudes. Para obtener más información, consulte Proceso de firma Signature Version 4 en la Referencia general de AWS.

Acceda a los registros de servicios del sistema

Si configura el software AWS IoT Greengrass principal como un servicio del sistema, puede ver los registros de servicio del sistema para solucionar problemas, como el hecho de que el software no se inicie.

Para ver los registros de servicio del sistema (CLI)

1. Ejecute el siguiente comando para ver los registros de servicio del sistema del software AWS IoT Greengrass principal.

   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. En los dispositivos Windows, el software AWS IoT Greengrass Core crea un archivo de registro independiente para los errores del servicio del sistema. Ejecute el siguiente comando para ver los registros de errores del servicio del sistema.

   Windows Command Prompt (CMD)

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

   PowerShell

   ```
   gc C:\greengrass\v2\logs\greengrass.err.log
   ```

En los dispositivos Windows, también puede utilizar la aplicación Event Viewer para ver los registros de servicio del sistema.

Para ver los registros de servicio de Windows (Visor de eventos)

1. Abra la aplicación Event Viewer.
2. Seleccione Registros de Windows para expandirlo.
3. Elija Aplicación para ver los registros de servicio de la aplicación.
4. Busque y abra los registros de eventos cuya fuente sea greengrass.

Habilite el registro en los CloudWatch registros

Puede implementar el componente de administrador de registros para configurar un dispositivo principal para escribir registros en los CloudWatch registros. Puede habilitar CloudWatch los registros para los registros del software AWS IoT Greengrass principal y puede habilitar CloudWatch los registros para componentes específicos de Greengrass.

> **Note**
>
> La función de intercambio de tokens del dispositivo principal de Greengrass debe permitir que el dispositivo principal escriba en los CloudWatch registros, como se muestra en el siguiente ejemplo de política de IAM. Si instaló el software AWS IoT Greengrass Core con el aprovisionamiento automático de recursos, su dispositivo principal tiene estos permisos.
>
> ```
> {
> ```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:DescribeLogStreams"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:logs:*:*:*"
    }
  ]
}
```

Para configurar un dispositivo principal para que escriba los registros del software AWS IoT Greengrass principal en CloudWatch Logs, cree una implementación que especifique una actualización de configuración que se establezca `uploadToCloudWatch` `true` para el `aws.greengrass.LogManager` componente. AWS IoT Greengrass Los registros del software principal incluyen los registros del núcleo y los componentes del complemento de Greengrass.

```
{
  "logsUploaderConfiguration": {
    "systemLogsConfiguration": {
      "uploadToCloudWatch": "true"
    }
  }
}
```

Para configurar un dispositivo principal para escribir los registros de un componente de Greengrass en Logs, cree una implementación que especifique una actualización de configuración que añada el componente a la lista de configuraciones de registro de componentes. CloudWatch Al agregar un componente a esta lista, el componente del administrador de registros escribe sus registros en CloudWatch Logs. Los registros de componentes incluyen registros de componentes genéricos y componentes Lambda.

```
{
  "logsUploaderConfiguration": {
    "componentLogsConfigurationMap": {
```

```
      "com.example.HelloWorld": {

      }
    }
  }
}
```

Al implementar el componente de administrador de registros, también puede configurar los límites de espacio en disco y determinar si el dispositivo principal eliminará los archivos de registro después de escribirlos en los CloudWatch registros. Para obtener más información, consulte Configuración de registro en AWS IoT Greengrass.

Configuración de registro en AWS IoT Greengrass

Puede configurar las siguientes opciones para personalizar el registro de los dispositivos principales de Greengrass. Para configurar estas opciones, cree una implementación que especifique una actualización de configuración para los componentes del núcleo o del administrador de registros de Greengrass.

- Escribir registros en registros CloudWatch

  Para solucionar los problemas de los dispositivos principales de forma remota, puede configurar los dispositivos principales para que escriban los registros AWS IoT Greengrass del software y los componentes principales en los CloudWatch registros. Para ello, implemente y configure el componente del administrador de registros. Para obtener más información, consulte Habilite el registro en los CloudWatch registros.
- Eliminar los archivos de registro cargados

  Para reducir el uso del espacio en disco, puede configurar los dispositivos principales para que eliminen los archivos de registro después de escribirlos en los CloudWatch registros. Para obtener más información, consulte el `deleteLogFileAfterCloudUpload` parámetro del componente del administrador de registros, que puede especificar para los registros del software AWS IoT Greengrass principal y los registros de los componentes.
- Límites de espacio en disco de registro

  Para limitar el uso del espacio en disco, puede configurar el espacio máximo en disco para cada registro, incluidos sus archivos de registro rotados, en un dispositivo principal. Por ejemplo, puede configurar el espacio máximo combinado en disco para los archivos rotados `greengrass.log` y para los archivos. `greengrass.log` Para obtener más información, consulte el parámetro del

componente núcleo de Greengrass y el `logging.totalLogsSizeKB` parámetro del componente del `diskSpaceLimit` administrador de registros, que puede especificar para los registros del software AWS IoT Greengrass principal y los registros de los componentes.

- Límite de tamaño de los archivos de registro

  Puede configurar el tamaño máximo de archivo para cada archivo de registro. Cuando un archivo de registro supera este límite de tamaño de archivo, el software AWS IoT Greengrass Core crea un nuevo archivo de registro. La versión 2.28 (y anteriores) del componente gestor de registros solo graba en los CloudWatch registros los archivos de registro rotados, por lo que puede especificar un límite de tamaño de archivo inferior para escribir los registros en los CloudWatch registros con más frecuencia. La versión 2.3.0 (y posteriores) del componente gestor de registros procesa y carga todos los registros en lugar de esperar a que se roten. Para obtener más información, consulte el parámetro límite de tamaño del archivo de registro del componente núcleo de Greengrass ()`logging.fileSizeKB`.

- Niveles mínimos de registro

  Puede configurar el nivel de registro mínimo que el componente núcleo de Greengrass escribe en los registros del sistema de archivos. Por ejemplo, puede especificar registros de `DEBUG` nivel para facilitar la solución de problemas, o puede especificar registros de `ERROR` nivel para reducir la cantidad de registros que crea un dispositivo central. Para obtener más información, consulte el parámetro de nivel de registro del componente núcleo de Greengrass ()`logging.level`.

  También puede configurar el nivel de registro mínimo que el componente del administrador de CloudWatch registros escribe en Logs. Por ejemplo, puede especificar un nivel de registro superior para reducir los costes de registro. Para obtener más información, consulte el `minimumLogLevel` parámetro del componente del administrador de registros, que puede especificar para los registros del software AWS IoT Greengrass principal y los registros de los componentes.

- Intervalo para comprobar si hay registros para escribirlos en los CloudWatch registros

  Para aumentar o reducir la frecuencia con la que el componente del administrador de registros escribe CloudWatch registros en los registros, puede configurar el intervalo en el que comprueba si hay nuevos archivos de registro que escribir. Por ejemplo, puede especificar un intervalo inferior para ver los registros en los CloudWatch registros antes de lo que lo haría con el intervalo predeterminado de 5 minutos. Puede especificar un intervalo mayor para reducir los costes, ya que el componente del administrador de registros agrupa los archivos de registro en menos solicitudes. Para obtener más información, consulte el parámetro de intervalo de carga del componente de administrador de registros (`periodicUploadIntervalSec`).

- Formato de registro

  Puede elegir si el software AWS IoT Greengrass Core escribe los registros en formato de texto o JSON. Elija el formato de texto si lee los registros o el formato JSON si utiliza una aplicación para leer o analizar los registros. Para obtener más información, consulte el parámetro de formato de registro () `logging.format` del componente núcleo de Greengrass.

- Carpeta de registros del sistema de archivos local

  Puede cambiar la carpeta de registros */greengrass/v2*`/logs` por otra carpeta del dispositivo principal. Para obtener más información, consulte el parámetro del directorio de salida del componente núcleo de Greengrass ()`logging.outputDirectory`.

Registros de AWS CloudTrail

AWS IoT Greengrassse integra conAWS CloudTrail, un servicio que proporciona un registro de las acciones realizadas por un usuario, rol o elementoServicio de AWS. AWS IoT Greengrass Para obtener más información, consulte Registra las llamadas a la AWS IoT Greengrass V2 API con AWS CloudTrail.

Registra las llamadas a la AWS IoT Greengrass V2 API con AWS CloudTrail

AWS IoT Greengrass V2 está integrado con AWS CloudTrail un servicio que proporciona un registro de las acciones realizadas por un usuario, un rol o un AWS servicio en AWS IoT Greengrass Version 2. CloudTrail captura todas las llamadas a la API AWS IoT Greengrass como eventos. Las llamadas que se capturan incluyen las llamadas desde la AWS IoT Greengrass consola y las llamadas en código a las operaciones de la AWS IoT Greengrass API.

Si crea un registro, puede habilitar la entrega continua de CloudTrail eventos a un bucket de S3, incluidos los eventos correspondientes AWS IoT Greengrass. Si no configuras una ruta, podrás ver los eventos más recientes en la CloudTrail consola, en el historial de eventos. Con la información recopilada por usted CloudTrail, puede determinar a AWS IoT Greengrass qué dirección IP se realizó la solicitud, quién la realizó, cuándo se realizó y detalles adicionales.

Para obtener más información al respecto CloudTrail, consulte la Guía AWS CloudTrail del usuario.

AWS IoT Greengrass V2 información en CloudTrail

CloudTrail está habilitada en tu cuenta Cuenta de AWS al crear la cuenta. Cuando se produce una actividad en AWS IoT Greengrass, esa actividad se registra en un CloudTrail evento junto con otros eventos de AWS servicio en el historial de eventos. Puede ver, buscar y descargar eventos recientes en su Cuenta de AWS. Para obtener más información, consulte Visualización de eventos con el historial de CloudTrail eventos.

Para tener un registro continuo de tus eventos Cuenta de AWS, incluidos los eventos para AWS IoT Greengrass ti, crea una ruta. Un rastro permite CloudTrail enviar archivos de registro a un bucket de S3. De forma predeterminada, al crear una ruta en la consola, la ruta se aplica a todos los Región de AWS s. La ruta registra los eventos de todas las regiones de la AWS partición y envía los archivos de registro al depósito de S3 que especifique. Además, puede configurar otros AWS servicios para analizar más a fondo los datos de eventos recopilados en los CloudTrail registros y actuar en función de ellos. Para más información, consulte los siguientes temas:

- Introducción a la creación de registros de seguimiento
- CloudTrail servicios e integraciones compatibles
- Configuración de las notificaciones de Amazon SNS para CloudTrail
- Recibir archivos de CloudTrail registro de varias regiones y recibir archivos de CloudTrail registro de varias cuentas

Todas AWS IoT Greengrass V2 las acciones se registran CloudTrail y se documentan en la referencia de la AWS IoT Greengrass V2 API. Por ejemplo, las llamadas a `CreateDeployment` y `CancelDeployment` las acciones generan entradas en los archivos de CloudTrail registro. `CreateComponentVersion`

Cada entrada de registro o evento contiene información sobre quién generó la solicitud. La información de identidad del usuario le ayuda a determinar lo siguiente:

- Si la solicitud se realizó con credenciales de usuario root o AWS Identity and Access Management (IAM).
- Si la solicitud se realizó con credenciales de seguridad temporales de un rol o fue un usuario federado.
- Si la solicitud la realizó otro AWS servicio.

Para obtener más información, consulte el elemento userIdentity de CloudTrail .

AWS IoT Greengrass eventos de datos en CloudTrail

Los eventos de datos proporcionan información sobre las operaciones de recursos que se realizan en un recurso o dentro de él (por ejemplo, obtener una versión de un componente o la configuración de una implementación). Se denominan también operaciones del plano de datos. Los eventos de datos suelen ser actividades de gran volumen. De forma predeterminada, CloudTrail no registra los eventos de datos. El historial de CloudTrail eventos no registra los eventos de datos.

Se aplican cargos adicionales a los eventos de datos. Para obtener más información sobre CloudTrail los precios, consulta AWS CloudTrail Precios.

Puede registrar eventos de datos para los tipos de AWS IoT Greengrass recursos mediante la CloudTrail consola o las operaciones de la CloudTrail API. AWS CLI La tabla de esta sección muestra los tipos de recursos disponibles para AWS IoT Greengrass.

- Para registrar eventos de datos mediante la CloudTrail consola, cree un almacén de datos de rutas o eventos para registrar eventos de datos, o actualice un banco de datos de seguimiento o evento existente para registrar eventos de datos.

  1. Elija Eventos de datos para registrar los eventos de datos.
  2. En la lista de tipos de eventos de datos, elija el tipo de recurso para el que desea registrar los eventos de datos.
  3. Elija la plantilla de selección de registros que desee utilizar. Puede registrar todos los eventos de datos del tipo de recurso, registrar todos los `readOnly` eventos, registrar todos los `writeOnly` eventos o crear una plantilla de selección de registros personalizada para filtrar `resources.`ARN los campos y. `readOnly` `eventName`

- Para registrar los eventos de datos mediante el AWS CLI, configure el `--advanced-event-selectors` parámetro para que el `eventCategory` campo sea igual al valor del tipo de recurso `Data` y el `resources.type` campo igual al valor del tipo de recurso (consulte la tabla). Puede agregar condiciones para filtrar los valores de los `resources.`ARN campos `readOnly``eventName`, y.
  - Para configurar una ruta para registrar eventos de datos, ejecute el put-event-selectorscomando. Para obtener más información, consulte Registrar eventos de datos para senderos con la AWS CLI.
  - Para configurar un banco de datos de eventos para registrar eventos de datos, ejecute el create-event-data-storecomando para crear un nuevo banco de datos de eventos para registrar eventos de datos, o ejecute el update-event-data-storecomando para actualizar un banco de datos de

eventos existente. Para obtener más información, consulte Registrar eventos de datos para los almacenes de datos de eventos con AWS CLI.

En la siguiente tabla se enumeran los tipos de AWS IoT Greengrass recursos. La columna Tipo de evento de datos (consola) muestra el valor que se puede elegir en la lista de tipos de eventos de datos de la CloudTrail consola. La columna de valores resources.type muestra el `resources.type` valor que se debe especificar al configurar los selectores de eventos avanzados mediante las API o. AWS CLI CloudTrail La CloudTrail columna API de datos en la que se ha registrado muestra las llamadas a la API registradas CloudTrail para el tipo de recurso.

Tipo de evento de datos (consola)	resources.type value	Las API de datos registradas en CloudTrail
Certificado IoT	`AWS::IoT::Certificate`	• VerifyClientDeviceIdentity • VerifyClientDeviceIoTCertificateAssociation
Versión del componente IoT Greengrass	`AWS::GreengrassV2::ComponentVersion`	• ResolveComponentCandidates
Despliegue de IoT Greengrass	`AWS::GreengrassV2::Deployment`	• GetDeploymentConfiguration
Cosa de IoT	`AWS::IoT::Thing`	• ListThingGroupsForCoreDevices • PutCertificateAuthorities • VerifyClientDeviceIoTCertificateAssociation

> **Note**
>
> Greengrass no registra los eventos de acceso denegado.

Puede configurar selectores de eventos avanzados para filtrar según los campos `eventName`, `readOnly` y `resources.ARN` y así registrar solo los eventos que son importantes para usted.

Añada un filtro `eventName` para incluir o excluir API de datos específicas.

Para obtener más información acerca de estos campos, consulte AdvancedFieldSelector.

En los siguientes ejemplos se muestra cómo configurar selectores avanzados mediante. AWS CLI Sustituya *`TrailName`*una *`región`* por su propia información.

Example — Registra eventos de datos para cosas de 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 — Filtrar una API específica de IoT

```
aws cloudtrail put-event-selectors --trail-name TrailName --region region \
--advanced-event-selectors \
'[
    {
            "Name": "Log IoT Greengrass PutCertificateAuthorities API calls",
            "FieldSelectors": [
                { "Field": "eventCategory", "Equals": ["Data"] },
                { "Field": "resources.type", "Equals": ["AWS::IoT::Thing"] },
                { "Field": "eventName", "Equals": ["PutCertificateAuthorities"] }
            ]
    }
]'
```

Example — Registra todos los eventos de datos de 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 eventos de gestión en CloudTrail

Los eventos de administración proporcionan información sobre las operaciones de administración que se realizan en los recursos de su AWS cuenta. Se denominan también operaciones del plano de control. De forma predeterminada, CloudTrail registra los eventos de administración.

AWS IoT Greengrass registra todas las operaciones del plano de AWS IoT Greengrass control como eventos de administración. Para obtener una lista de las operaciones del plano de AWS IoT Greengrass control en las que se AWS IoT Greengrass registra CloudTrail, consulte la referencia de la AWS IoT Greengrass API, versión 2.

Descripción de las entradas de los archivos de AWS IoT Greengrass V2 registro

Un registro es una configuración que permite la entrega de eventos como archivos de registro a un bucket de S3 que usted especifique. CloudTrail Los archivos de registro contienen una o más entradas de registro. Un evento representa una única solicitud desde cualquier origen. Incluye

información sobre la acción solicitada, la fecha y la hora de la acción, los parámetros de la solicitud, etc. CloudTrail Los archivos de registro no son un registro ordenado de las llamadas a las API públicas, por lo que no aparecen en ningún orden específico.

En el siguiente ejemplo, se muestra una entrada de CloudTrail registro que demuestra la `CreateDeployment` acción.


```
{
    "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"
}
```

Recopile datos de telemetría del estado del sistema de los dispositivos principales AWS IoT Greengrass

Los datos de telemetría del estado del sistema son datos de diagnóstico que pueden ayudarlo a monitorear el rendimiento de las operaciones críticas en sus dispositivos principales de Greengrass. Puede crear proyectos y aplicaciones para recuperar, analizar, transformar y generar informes sobre los datos de telemetría de sus dispositivos periféricos. Los expertos de dominio, como los ingenieros de procesos, pueden utilizar estas aplicaciones para obtener información sobre el estado de la flota.

Puede utilizar los siguientes métodos para recopilar datos de telemetría de sus dispositivos principales de Greengrass:

- Componente emisor de telemetría Nucleus: el componente emisor de telemetría Nucleus () de un dispositivo central de Greengrass publica los datos de telemetría `aws.greengrass.telemetry.NucleusEmitter` en el tema de forma predeterminada. `$local/greengrass/telemetry` Puede utilizar los datos publicados en este tema para actuar de forma local en su dispositivo principal, incluso si su conectividad a la nube es limitada. Si lo desea, también puede configurar el componente para publicar datos de telemetría en el tema de AWS IoT Core MQTT que prefiera.

  Debe implementar el componente nucleo-emisor en un dispositivo central para publicar los datos de telemetría. La publicación de datos de telemetría en el periódico local no conlleva ningún coste.

Sin embargo, el uso de un tema de MQTT para publicar datos en el Nube de AWS está sujeto a los precios. AWS IoT Core

AWS IoT Greengrassproporciona varios componentes comunitarios para ayudarlo a analizar y visualizar los datos de telemetría localmente en su dispositivo principal mediante InfluxDB y Grafana. Estos componentes utilizan datos de telemetría del componente nucleo-emisor. Para obtener más información, consulte el archivo README del componente de publicación InfluxDB.

- Agente de telemetría: el agente de telemetría de los dispositivos principales de Greengrass recopila datos de telemetría locales y los publica en Amazon sin necesidad de interacción con el cliente. EventBridge Los dispositivos principales publican los datos de telemetría haciendo el mejor esfuerzo. EventBridge Por ejemplo, es posible que los dispositivos principales no entreguen los datos de telemetría cuando están fuera de línea.

  La función del agente de telemetría está habilitada de forma predeterminada en todos los dispositivos principales de Greengrass. Empezará a recibir datos automáticamente en cuanto configure un dispositivo principal de Greengrass. Además de los costes de enlace de datos, la transferencia de datos desde el dispositivo principal AWS IoT Core es gratuita. Esto se debe a que el agente publica en un tema AWS reservado. Sin embargo, en función de su caso de uso, es posible que incurra en costes cuando reciba o procese los datos.

  > **Note**
  >
  > Amazon EventBridge es un servicio de bus de eventos que puede utilizar para conectar sus aplicaciones con datos de diversas fuentes, como los dispositivos principales de Greengrass. Para obtener más información, consulta ¿Qué es Amazon EventBridge? en la Guía del EventBridge usuario de Amazon.

Para garantizar que el software AWS IoT Greengrass Core funcione correctamente, AWS IoT Greengrass utiliza los datos con fines de desarrollo y mejora de la calidad. Esta función también ayuda a informar sobre las capacidades periféricas nuevas y mejoradas. AWS IoT Greengrassconserva los datos de telemetría durante un máximo de siete días.

En esta sección se describe cómo configurar y utilizar el agente de telemetría. Para obtener información sobre la configuración del componente emisor de telemetría del núcleo, consulte. Emisor de telemetría Nucleus

Temas

Métricas de telemetría

En la siguiente tabla se describen las métricas que publica el agente de telemetría.

Nombre	Descripción	
System (Sistema)		
`SystemMemUsage`	La cantidad de memoria que utilizan actualmente todas las aplicaciones del dispositivo principal de Greengrass, incluido el sistema operativo.	
`CpuUsage`	La cantidad de CPU que utilizan actualmente todas las aplicaciones del dispositivo principal de Greengrass, incluido el sistema operativo.	
`TotalNumberOfFDs`	El número de descriptores de archivos almacenados por el sistema operativo del dispositivo principal de Greengrass. Un descriptor de archivo identifica exclusivamente un archivo abierto.	
Núcleo de Greengrass		
`NumberOfComponentsRunning`	El número de componentes que se ejecutan en el dispositivo principal de Greengrass.	

Nombre	Descripción
`NumberOfComponentsErrored`	El número de componentes que están en estado de error en el dispositivo principal de Greengrass.
`NumberOfComponentsInstalled`	El número de componentes que están instalados en el dispositivo principal de Greengrass.
`NumberOfComponentsStarting`	El número de componentes que se inician en el dispositivo principal de Greengrass.
`NumberOfComponentsNew`	La cantidad de componentes nuevos en el dispositivo principal de Greengrass.
`NumberOfComponentsStopping`	El número de componentes que se detienen en el dispositivo principal de Greengrass.
`NumberOfComponentsFinished`	El número de componentes que están acabados en el dispositivo principal de Greengrass.
`NumberOfComponentsBroken`	La cantidad de componentes que están rotos en el dispositivo principal de Greengrass.
`NumberOfComponentsStateless`	El número de componentes que no tienen estado en el dispositivo principal de Greengrass.

Nombre	Descripción	
Autenticación del dispositivo cliente: esta función requiere la versión 2.4.0 o posterior del componente de autenticación del dispositivo cliente.		
`VerifyClientDeviceIdentity.Success`	El número de veces que se verificó que la identidad del dispositivo cliente se realizó correctamente.	
`VerifyClientDeviceIdentity.Failure`	El número de veces que se verificó que la identidad del dispositivo cliente falló.	
`AuthorizeClientDeviceActions.Success`	El número de veces que el dispositivo cliente está autorizado a completar las acciones solicitadas.	
`AuthorizeClientDeviceActions.Failure`	El número de veces que el dispositivo cliente no está autorizado a completar las acciones solicitadas.	
`GetClientDeviceAuthToken.Success`	El número de veces que el dispositivo cliente se autentica correctamente.	
`GetClientDeviceAuthToken.Failure`	El número de veces que el dispositivo cliente no se puede autenticar.	
`SubscribeToCertificateUpdates.Success`	El número de suscripciones correctas a las actualizaciones de certificados.	

Nombre	Descripción
`SubscribeToCertificateUpdates.Failure`	El número de intentos fallidos de suscribirse a las actualizaciones de certificados.
`ServiceError`	El número de errores internos no controlados en la autenticación del dispositivo cliente.
Administrador de transmisiones: esta función requiere la versión 2.7.0 o posterior del componente núcleo de Greengrass.	
`BytesAppended`	El número de bytes de datos anexos al administrador de secuencias.
`BytesUploadedToIoTAnalytics`	El número de bytes de datos que el administrador de secuencias exporta a los canales en AWS IoT Analytics.
`BytesUploadedToKinesis`	El número de bytes de datos que el administrador de secuencias exporta a las transmisiones de Amazon Kinesis Data Streams.
`BytesUploadedToIoTSiteWise`	El número de bytes de datos que el administrador de secuencias exporta a las propiedades de los activos en AWS IoT SiteWise.

Nombre	Descripción	
`BytesUploadedToS3`	El número de bytes de datos que el administrador de secuencias exporta a objetos de Amazon S3.	

Configure los ajustes del agente de telemetría

El agente de telemetría utiliza los siguientes ajustes predeterminados:

- El agente de telemetría agrega los datos de telemetría cada hora.
- El agente de telemetría publica un mensaje de telemetría cada 24 horas.

El agente de telemetría publica los datos mediante el protocolo MQTT con un nivel de calidad de servicio (QoS) de 0, lo que significa que no confirma la entrega ni reintenta publicar los intentos. Los mensajes de telemetría comparten una conexión MQTT con otros mensajes para las suscripciones destinadas para AWS IoT Core.

Además de los costes de enlace de datos, la transferencia de datos desde el núcleo es gratuita. AWS IoT Core Esto se debe a que el agente publica en un tema AWS reservado. Sin embargo, en función de su caso de uso, es posible que incurra en costes cuando reciba o procese los datos.

Puede activar o desactivar la función de agente de telemetría para cada dispositivo principal de Greengrass. También puede configurar los intervalos durante los cuales el dispositivo principal agrega y publica datos. Para configurar la telemetría, personalice el parámetro de configuración de telemetría al implementar el componente núcleo de Greengrass.

Suscríbase a los datos de telemetría en EventBridge

Puede crear reglas en Amazon EventBridge que definan cómo procesar los datos de telemetría publicados desde el agente de telemetría del dispositivo principal de Greengrass. Cuando EventBridge recibe los datos, invoca las acciones objetivo definidas en sus reglas. Por ejemplo, puede crear reglas de eventos que envíen notificaciones, almacenen información sobre eventos, adopten medidas correctivas o invoquen otros eventos.

Eventos de telemetría

Los eventos de telemetría utilizan el siguiente formato.


```
{
  "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"
          }
        ]
      }
    ]
  }
}
```

La matriz ADP contiene una lista de puntos de datos agregados que tienen las siguientes propiedades:

`TS`

La marca de tiempo del momento en que se recopilaron los datos.

`NS`

El espacio de nombres métrico.

`M`

Lista de métricas Una métrica contiene las siguientes propiedades:

`N`

El nombre de la métrica.

`Sum`

La suma de los valores de la métrica en este evento de telemetría.

`U`

La unidad del valor métrico.

Para obtener más información sobre cada métrica, consulte. Métricas de telemetría

Requisitos previos para crear reglas EventBridge

Antes de crear una EventBridge regla paraAWS IoT Greengrass, debe hacer lo siguiente:

- Familiarícese con los eventos, las reglas y los objetivos en EventBridge.
- Cree y configure los objetivos invocados por sus EventBridge reglas. Las reglas pueden invocar muchos tipos de objetivos, como transmisiones de Amazon Kinesis, funciones AWS Lambda, temas de Amazon SNS y colas de Amazon SQS.

  Tu EventBridge regla y los objetivos asociados deben estar en el Región de AWS lugar donde creaste tus recursos de Greengrass. Para obtener más información, consulte Puntos de enlace y cuotas en la Referencia general de AWS.

Para obtener más información, consulta ¿Qué es Amazon EventBridge? y Introducción a Amazon EventBridge en la Guía del EventBridge usuario de Amazon.

Cree una regla de eventos para obtener datos de telemetría (consola)

Siga los siguientes pasos AWS Management Console para crear una EventBridge regla que reciba los datos de telemetría publicados por el dispositivo principal de Greengrass. De este modo, los servidores web, las direcciones de correo electrónico y otros suscriptores del tema podrán responder al evento. Para obtener más información, consulta Crear una EventBridge regla que se active en un evento desde un AWS recurso en la Guía del EventBridge usuario de Amazon.

1. Abre la EventBridgeconsola de Amazon y selecciona Crear regla.
2. En Name and description (Nombre y descripción), escriba un nombre y descripción para la regla.
3. En Define pattern (Definir patrón), configure el patrón de regla.
   a. Seleccione Event pattern.
   b. Elija Pre-defined pattern by service (Patrón predefinido por servicio).
   c. En Service provider (Proveedor de servicios), elija AWS.
   d. En Service name (Nombre del servicio), elija Greengrass.
   e. En Tipo de evento, seleccione Datos de telemetría de Greengrass.
4. En Select event bus (Seleccionar bus de evento), mantenga las opciones de bus de eventos predeterminadas.
5. En Select targets (Seleccionar destinos), configure su destino. El siguiente ejemplo utiliza una cola de Amazon SQS, pero puede configurar otros tipos de objetivos.
   a. Para Target, elija la cola SQS.
   b. En Queue*, elija la cola de destino.
6. En Tags - optional (Etiquetas - opcional), defina etiquetas para la regla o deje los campos vacíos.
7. Seleccione Crear.

Cree una regla de eventos para obtener datos de telemetría (CLI)

Siga los siguientes pasos AWS CLI para crear una EventBridge regla que reciba los datos de telemetría publicados por los dispositivos principales de Greengrass. De este modo, los servidores web, las direcciones de correo electrónico y otros suscriptores del tema podrán responder al evento.

1. Crear la regla.

- *Sustituya el nombre de la cosa por el nombre* de la cosa del dispositivo principal.

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\"]}}"
```

Las propiedades que se omiten en el patrón no se tienen en cuenta.

2. Añada el tema como destino de la regla. El siguiente ejemplo usa Amazon SQS, pero puede configurar otros tipos de objetivos.

   - Sustituya *queue-arn* por el ARN de su cola de Amazon SQS.

   Linux or Unix

   ```
   aws events put-targets \
     --rule MyGreengrassTelemetryEventRule \
     --targets "Id"="1","Arn"="queue-arn"
   ```

Windows Command Prompt (CMD)

```
aws events put-targets ^
  --rule MyGreengrassTelemetryEventRule ^
  --targets "Id"="1","Arn"="queue-arn"
```

PowerShell

```
aws events put-targets `
  --rule MyGreengrassTelemetryEventRule `
  --targets "Id"="1","Arn"="queue-arn"
```

> **Note**
>
> Para permitir que Amazon EventBridge invoque tu cola de destino, debes añadir a tu tema una política basada en recursos. Para obtener más información, consulte los permisos de Amazon SQS en la Guía EventBridge del usuario de Amazon.

Para obtener más información, consulta Eventos y patrones de eventos EventBridge en la Guía del EventBridge usuario de Amazon.

Reciba notificaciones de despliegue y estado de los componentes

Las reglas de EventBridge eventos de Amazon le proporcionan notificaciones sobre los cambios de estado de las implementaciones de Greengrass que reciben sus dispositivos y de los componentes instalados en sus dispositivos. EventBridge ofrece un flujo casi en tiempo real de los eventos del sistema que describe los cambios en AWS los recursos. AWS IoT Greengrass envía estos eventos haciendo EventBridge el mejor esfuerzo posible. Esto significa que AWS IoT Greengrass intenta enviar todos los eventos EventBridge pero, en algunos casos excepcionales, es posible que no se entregue un evento. Además, AWS IoT Greengrass puede enviar varias copias de un evento determinado, lo que significa que es posible que los oyentes del evento no reciban los eventos en el orden en que se produjeron.

> **Note**
>
> Amazon EventBridge es un servicio de bus de eventos que puede utilizar para conectar sus aplicaciones con datos de diversas fuentes, como los dispositivos principales de Greengrass y las notificaciones de despliegue y componentes. Para obtener más información, consulta ¿Qué es Amazon EventBridge? en la Guía del EventBridge usuario de Amazon.

Temas

Evento de cambio de estado de despliegue

AWS IoT Greengrass emite un evento cuando una implementación entra en los siguientes estados: `FAILED`, `SUCCEEDED`, `COMPLETED`, `REJECTED`, y `CANCELED`. Puede crear una EventBridge regla que se aplique a todas las transiciones de estado o a los estados que especifique. Cuando una implementación entra en un estado que inicia una regla, EventBridge invoca las acciones objetivo definidas en la regla. Esto le permite enviar notificaciones, capturar información sobre el evento, tomar medidas correctivas o iniciar otros eventos en respuesta a un cambio de estado. Por ejemplo, puede crear reglas para los siguientes casos de uso:

- Iniciar operaciones posteriores a la implementación, como descargar recursos y enviar notificaciones al personal.
- Envíe notificaciones en caso de una implementación correcta o con error.
- Publicar métricas personalizadas sobre los eventos de implementación.

El evento de un cambio de estado de la implementación tiene el siguiente formato:

```
{
```

```
    "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=)"
     }
}
```

Puede crear reglas y eventos que le informarán sobre el estado de un despliegue. Un evento se inicia cuando una implementación se completa como `FAILEDSUCCEEDED`,`COMPLETED`,`REJECTED`, o`CANCELED`. Si la implementación falló en el dispositivo principal, recibirá una respuesta detallada que explica por qué la implementación falló. Para obtener más información sobre los códigos de error de despliegue, consulteCódigos de error de implementación detallados.

Estados de implementación

- `FAILED`. La implementación no se ha realizado correctamente
- `SUCCEEDED`. El despliegue dirigido a un grupo de cosas se completó correctamente.
- `COMPLETED`. El despliegue dirigido a algo se completó con éxito.
- `REJECTED`. Se rechazó el despliegue. Para obtener más información, consulte el `statusDetails` campo.
- `CANCELED`. El usuario canceló la implementación.

Es posible que los eventos se dupliquen o estén desordenados. Para determinar el orden de los eventos, utilice la propiedad `time`.

Para obtener una lista completa de los códigos de error en `errorStacks` `yerrorTypes`, consulte Códigos de error de implementación detallados yCódigos de estado detallados de los componentes.

Evento de cambio de estado del componente

Para AWS IoT Greengrass las versiones 2.12.2 y anteriores, Greengrass emite un evento cuando un componente entra en los siguientes estados: y. `ERRORED` `BROKEN` Para las versiones 2.12.3 y posteriores del núcleo de Greengrass, Greengrass emite un evento cuando un componente entra en los siguientes estados:,, y. `ERRORED` `BROKEN` `RUNNING` `FINISHED` Greengrass también emitirá un evento cuando se complete un despliegue. Puede crear una EventBridge regla que se aplique a todas las transiciones de estado o a los estados que especifique. Cuando un componente instalado entra en un estado que inicia una regla, EventBridge invoca las acciones de destino definidas en la regla. Esto le permite enviar notificaciones, capturar información sobre el evento, tomar medidas correctivas o iniciar otros eventos en respuesta a un cambio de estado.

El evento de cambio de estado de un componente utiliza los siguientes formatos:

Greengrass nucleus v2.12.2 and earlier

<title>Estado del componente: ERRORED o BROKEN</title>

```
{
    "version":"0",
    "id":" cd4d811e-ab12-322b-8255-EXAMPLEb1bc8",
    "detail-type":"Greengrass V2 Installed Component Status Change",
    "source":"aws.greengrass",
    "account":"123456789012",
    "region":"us-west-2",
    "time":"2018-03-22T00:38:11Z",
    "resources":["arn:aws:greengrass:us-
east-1:123456789012:coreDevices:MyGreengrassCore"],
    "detail": {
       "components": [
          {
            "componentName": "MyComponent",
            "componentVersion": "1.0.0",
            "root": true,
            "lifecycleState": "ERRORED|BROKEN",
            "lifecycleStatusCodes": ["STARTUP_ERROR"],
```

```
            "lifecycleStateDetails": "An error occurred during startup. The startup
 script exited with code 1."
          }
        ]
    }
}
```

Greengrass nucleus v2.12.3 and later

<title>Estado del componente: ERRORED o BROKEN</title>

```
{
    "version":"0",
    "id":" cd4d811e-ab12-322b-8255-EXAMPLEb1bc8",
    "detail-type":"Greengrass V2 Installed Component Status Change",
    "source":"aws.greengrass",
    "account":"123456789012",
    "region":"us-west-2",
    "time":"2018-03-22T00:38:11Z",
    "resources":["arn:aws:greengrass:us-
east-1:123456789012:coreDevices:MyGreengrassCore"],
    "detail": {
       "components": [
          {
            "componentName": "MyComponent",
            "componentVersion": "1.0.0",
            "root": true,
            "lifecycleState": "ERRORED|BROKEN",
            "lifecycleStatusCodes": ["STARTUP_ERROR"],
            "lifecycleStateDetails": "An error occurred during startup. The startup
 script exited with code 1."
          }
        ]
    }
}
```

<title>Estado del componente: RUNNING o FINISHED</title>

```
{
    "version":"0",
    "id":" cd4d811e-ab12-322b-8255-EXAMPLEb1bc8",
    "detail-type":"Greengrass V2 Installed Component Status Change",
    "source":"aws.greengrass",
```

```
    "account":"123456789012",
    "region":"us-west-2",
    "time":"2018-03-22T00:38:11Z",
    "resources":["arn:aws:greengrass:us-
east-1:123456789012:coreDevices:MyGreengrassCore"],
    "detail": {
       "components": [
          {
            "componentName": "MyComponent",
            "componentVersion": "1.0.0",
            "root": true,
            "lifecycleState": "RUNNING|FINISHED",
            "lifecycleStateDetails": null
          }
        ]
    }
}
```

Puede crear reglas y eventos que le informarán sobre el estado de un componente instalado. Se inicia un evento cuando un componente cambia de estado en el dispositivo. Recibirá una respuesta detallada en la que se explicará por qué un componente tiene errores o está averiado. También recibirá un código de estado que indicará el motivo del fallo. Para obtener más información sobre los códigos de estado de los componentes, consulteCódigos de estado detallados de los componentes.

Requisitos previos para crear reglas EventBridge

Antes de crear una EventBridge regla para AWS IoT Greengrass, haga lo siguiente:

- Familiarícese con los eventos, las reglas y los objetivos de EventBridge.
- Cree y configure los objetivos invocados por sus EventBridge reglas. Las reglas pueden invocar muchos tipos de destinos, entre los que se incluyen:
  - Amazon Simple Notification Service (Amazon SNS)
  - AWS Lambda funciones
  - Amazon Kinesis Video Streams
  - Colas de Amazon Simple Queue Service (Amazon SQS)

Para obtener más información, consulta ¿Qué es Amazon EventBridge? y Introducción a Amazon EventBridge en la Guía del EventBridge usuario de Amazon.

Configura las notificaciones de estado del dispositivo (consola)

Siga los siguientes pasos para crear una EventBridge regla que publique un tema de Amazon SNS cuando cambie el estado de despliegue de un grupo. De este modo, los servidores web, las direcciones de correo electrónico y otros suscriptores del tema podrán responder al evento. Para obtener más información, consulta Crear una EventBridge regla que se active en un evento desde un AWS recurso en la Guía del EventBridge usuario de Amazon.

1. Abre la EventBridgeconsola de Amazon.
2. En el panel de navegación, seleccione Reglas.
3. Elija Crear regla.
4. Escriba un nombre y una descripción para la regla.

   Una regla no puede tener el mismo nombre que otra regla de la misma región y del mismo bus de eventos.
5. En Bus de eventos, seleccione el bus de eventos que desea asociar a esta regla. Si desea que esta regla coincida con eventos procedentes de su cuenta, seleccione Bus de eventos predeterminado de AWS . Cuando un AWS servicio de tu cuenta emite un evento, siempre va al bus de eventos predeterminado de tu cuenta.
6. En Tipo de regla, elija Regla con un patrón de evento.
7. Seleccione Siguiente.
8. En Origen de eventos, seleccione (Eventos de AWS).
9. En Patrón de eventos, seleccione servicios AWS .
10. En Servicio de AWS , elija Greengrass.
11. En Tipo de evento, elige una de las siguientes opciones:

    - Para los eventos de despliegue, elija Greengrass V2 Effective Deployment Status Change.
    - Para los eventos de componentes, elija Greengrass V2 Installed Component Status Change.
12. Seleccione Siguiente.
13. En Tipos de destino (Tipos de destino), elija AWS service.
14. En Seleccionar destinos, configure su destino. En este ejemplo se utiliza un tema de Amazon SNS, pero se pueden configurar otros tipos de destino para enviar notificaciones.

    a. En Destino, elija Tema de SNS.

    b. En Topic (Tema), elija el tema de destino.

c. Elija Siguiente.

15. Seleccione Siguiente.
16. Revise los detalles de la regla y seleccione Crear regla.

Configurar las notificaciones de estado del dispositivo (CLI)

Siga los siguientes pasos para crear una EventBridge regla que publique un tema de Amazon SNS cuando se produzca un evento de cambio de estado de Greengrass. De este modo, los servidores web, las direcciones de correo electrónico y otros suscriptores del tema podrán responder al evento.

1. Crear la regla.

   - Para eventos de cambio de estado de despliegue.

   ```
   aws events put-rule \
     --name TestRule \
     --event-pattern "{\"source\": [\"aws.greengrass\"], \"detail-type\":
    [\"Greengrass V2 Effective Deployment Status Change\"]}"
   ```

   - Para eventos de cambio de estado de componentes.

   ```
   aws events put-rule \
     --name TestRule \
     --event-pattern "{\"source\": [\"aws.greengrass\"], \"detail-type\":
    [\"Greengrass V2 Installed Component Status Change\"]}"
   ```

   Las propiedades que se omiten en el patrón no se tienen en cuenta.

2. Añada el tema como destino de la regla.

   - Sustituya *topic-arn* por el ARN de su tema de Amazon SNS.

   ```
   aws events put-targets \
     --rule TestRule \
     --targets "Id"="1","Arn"="topic-arn"
   ```

> **Note**
>
> Para permitir que Amazon llame EventBridge a tu tema objetivo, debes añadir a tu tema una política basada en recursos. Para obtener más información, consulte los permisos de Amazon SNS en la Guía EventBridge del usuario de Amazon.

Para obtener más información, consulta Eventos y patrones de eventos EventBridge en la Guía del EventBridge usuario de Amazon.

Configura las notificaciones de estado del dispositivo (AWS CloudFormation)

Utilice AWS CloudFormation plantillas para crear EventBridge reglas que envíen notificaciones sobre los cambios de estado para las implementaciones de su grupo de Greengrass. Para obtener más información, consulta la referencia de tipos de EventBridge recursos de Amazon en la Guía del AWS CloudFormation usuario.

Véase también

- Compruebe el estado de despliegue del dispositivo
- ¿Qué es Amazon EventBridge? en la Guía del EventBridge usuario de Amazon

Compruebe el estado del dispositivo principal de Greengrass

Los dispositivos principales de Greengrass informan del estado de sus componentes de software a. AWS IoT Greengrass Puede consultar el resumen del estado de cada dispositivo y comprobar el estado de cada componente de cada dispositivo.

Los dispositivos principales tienen los siguientes estados de salud:

- `HEALTHY`— El software AWS IoT Greengrass principal y todos los componentes se ejecutan sin problemas en el dispositivo principal.
- `UNHEALTHY`— El software AWS IoT Greengrass principal o un componente se encuentra en un estado de error en el dispositivo principal.

> **Note**
>
> AWS IoT Greengrass depende de los dispositivos individuales para enviar actualizaciones de estado al Nube de AWS. Si el software AWS IoT Greengrass principal no se ejecuta en el dispositivo o si el dispositivo no está conectado al Nube de AWS, es posible que el estado informado de ese dispositivo no refleje su estado actual. La marca de tiempo del estado indica cuándo se actualizó por última vez el estado del dispositivo.
> Los dispositivos principales envían actualizaciones de estado en los siguientes momentos:
>
> - Cuando se inicia el software AWS IoT Greengrass Core
> - Cuando el dispositivo principal recibe una implementación del Nube de AWS
> - Para Greengrass nucleus 2.12.2 y versiones anteriores, el dispositivo principal envía actualizaciones de estado cuando el estado de cualquier componente del dispositivo principal pasa a ser o `ERRORED` `BROKEN`
> - Para Greengrass nucleus 2.12.3 y versiones posteriores, el dispositivo principal envía actualizaciones de estado cuando el estado de cualquier componente del dispositivo principal pasa `ERRORED` a ser,, o `BROKEN` `RUNNING` `FINISHED`
> - A un intervalo regular que puede configurar, que por defecto es de 24 horas
>
> En el AWS IoT Greengrass caso de Core v2.7.0 y versiones posteriores, el dispositivo principal envía actualizaciones de estado cuando se realiza una implementación local y una implementación en la nube

Temas

Compruebe el estado de un dispositivo principal

Puede comprobar el estado de los dispositivos principales individuales.

Para comprobar el estado de un dispositivo principal (AWS CLI)

- Ejecute el siguiente comando para recuperar el estado de un dispositivo. *coreDeviceName*Sustitúyalo por el nombre del dispositivo principal que se va a realizar la consulta.

  ```
  aws greengrassv2 get-core-device --core-device-thing-name coreDeviceName
  ```

  La respuesta contiene información sobre el dispositivo principal, incluido su estado.

Compruebe el estado de un grupo de dispositivos principales

Puede comprobar el estado de un grupo de dispositivos principales (un grupo de cosas).

Para comprobar el estado de un grupo de dispositivos (AWS CLI)

- Ejecute el siguiente comando para recuperar el estado de varios dispositivos principales. Sustituya el ARN del comando por el ARN del grupo de cosas que se va a consultar.

  ```
  aws greengrassv2 list-core-devices --thing-group-arn "arn:aws:iot:region:account-
  id:thinggroup/thingGroupName"
  ```

  La respuesta contiene la lista de dispositivos principales del grupo de cosas. Cada entrada de la lista contiene el estado del dispositivo principal.

Compruebe el estado de los componentes principales del dispositivo

Puede comprobar el estado, por ejemplo, el estado del ciclo de vida, de los componentes de software de un dispositivo principal. Para obtener más información sobre los estados del ciclo de vida de los componentes, consulteDesarrolle AWS IoT Greengrass componentes.

Para comprobar el estado de los componentes de un dispositivo principal (AWS CLI)

- Ejecute el siguiente comando para recuperar el estado de los componentes de un dispositivo principal. *coreDeviceName*Sustitúyalo por el nombre del dispositivo principal que se va a realizar la consulta.

  ```
  aws greengrassv2 list-installed-components --core-device-thing-name coreDeviceName
  ```

La respuesta contiene la lista de componentes que se ejecutan en el dispositivo principal. Cada entrada de la lista contiene el estado del ciclo de vida del componente, incluido el estado actual de los datos y la última vez que el dispositivo principal de Greengrass envió un mensaje que contenía un determinado componente a la nube. La respuesta también incluirá la fuente de despliegue más reciente que llevó el componente al dispositivo principal de Greengrass.

> **Note**
>
> Este comando recupera una lista paginada de los componentes que ejecuta un dispositivo principal de Greengrass. De forma predeterminada, esta lista no incluye los componentes que se implementan como dependencias de otros componentes. Puede incluir dependencias en la respuesta configurando el `topologyFilter` parámetro en. `ALL`

AWS LambdaFunciones de ejecución

> **Note**
>
> AWS IoT Greengrassactualmente no admite esta función en los dispositivos principales de Windows.

Puede importar AWS Lambda funciones como componentes que se ejecutan en los dispositivos AWS IoT Greengrass principales. Es posible que desee hacerlo en los siguientes casos:

- Tiene un código de aplicación en las funciones de Lambda que desea implementar en los dispositivos principales.
- Tiene aplicaciones de la AWS IoT Greengrass versión 1 que desea ejecutar en los dispositivos AWS IoT Greengrass V2 principales. Para obtener más información, consulte Paso 2: Crear e implementar AWS IoT Greengrass V2 componentes para migrar aplicaciones AWS IoT Greengrass V1.

Las funciones Lambda incluyen dependencias en los siguientes componentes. No es necesario definir estos componentes como dependencias al importar la función. Al implementar el componente de la función Lambda, el despliegue incluye estas dependencias del componente Lambda.

- El componente Lambda launcher (`aws.greengrass.LambdaLauncher`) gestiona los procesos y la configuración del entorno.
- El componente Lambda manager (`aws.greengrass.LambdaManager`) gestiona la comunicación y el escalado entre procesos.
- El componente de tiempos de ejecución de Lambda (`aws.greengrass.LambdaRuntimes`) proporciona artefactos para cada tiempo de ejecución de Lambda compatible.

Temas

- Importar una función Lambda como componente () AWS CLI

Requisitos

Sus dispositivos principales y las funciones de Lambda deben cumplir los siguientes requisitos para poder ejecutar las funciones en el software AWS IoT Greengrass Core:

- El dispositivo principal debe cumplir los requisitos para ejecutar las funciones de Lambda. Si desea que el dispositivo principal ejecute funciones Lambda en contenedores, el dispositivo debe cumplir los requisitos para hacerlo. Para obtener más información, consulte Requisitos de la función de Lambda.
- Debe instalar los lenguajes de programación que utiliza la función Lambda en sus dispositivos principales.

> **Tip**
>
> Puede crear un componente que instale el lenguaje de programación y, a continuación, especificar ese componente como una dependencia del componente de la función Lambda. Greengrass admite todas las versiones compatibles con Lambda de los tiempos de ejecución de Python, Node.js y Java. Greengrass no aplica ninguna restricción adicional a las versiones de tiempo de ejecución de Lambda obsoletas. Puede ejecutar funciones de Lambda que usen estos tiempos de ejecución obsoletosAWS IoT Greengrass, pero no puede crearlas en. AWS Lambda Para obtener más información sobre la compatibilidad de AWS IoT Greengrass con los tiempos de ejecución Lambda, consulte AWS LambdaFunciones de ejecución.

Configurar el ciclo de vida de una función Lambda

El ciclo de vida de la función de Lambda de Greengrass determina cuándo se inicia una función y cómo crea y utiliza contenedores. El ciclo de vida también determina la forma en que el software AWS IoT Greengrass Core retiene las variables y la lógica de preprocesamiento que se encuentran fuera del controlador de funciones.

AWS IoT Greengrassadmite ciclos de vida bajo demanda (predeterminados) y de larga duración:

- Las funciones bajo demanda se inician cuando se invocan y se detienen cuando no queda ninguna tarea por ejecutar. Cada invocación de la función crea un contenedor independiente, también denominado sandbox, para procesar las invocaciones, a menos que haya un contenedor existente disponible para su reutilización. Es posible que cualquiera de los contenedores procese los datos que envíes a la función.

  Se pueden ejecutar varias invocaciones de una función bajo demanda de forma simultánea.

  Las variables y la lógica de preprocesamiento que defina fuera del controlador de funciones no se conservan cuando se crean nuevos contenedores.
- Las funciones de larga duración (o bloqueadas) se inician cuando se inicia el software AWS IoT Greengrass principal y se ejecutan en un único contenedor. El mismo contenedor procesa todos los datos que se envían a la función.

  Se ponen en cola varias invocaciones hasta que el software AWS IoT Greengrass principal ejecute las invocaciones anteriores.

  Las variables y la lógica de preprocesamiento que defina fuera del controlador de funciones se conservan para cada invocación del controlador.

  Utilice funciones Lambda de larga duración cuando necesite empezar a trabajar sin ninguna entrada inicial. Por ejemplo, una función de larga duración puede cargar y empezar a procesar un modelo de aprendizaje automático para que esté lista cuando la función reciba los datos del dispositivo.

  > **Note**
  >
  > Las funciones de larga duración tienen tiempos de espera asociados a cada invocación de su controlador. Si desea invocar código que se ejecute indefinidamente, debe iniciarlo fuera del controlador. Asegúrese de que no haya ningún código de bloqueo fuera del controlador que pueda impedir que la función se inicialice.
  > Estas funciones se ejecutan a menos que el software AWS IoT Greengrass principal se detenga, por ejemplo, durante una implementación o un reinicio. Estas funciones no se ejecutarán si la función encuentra una excepción no detectada, supera sus límites de memoria o entra en un estado de error, como el tiempo de espera del controlador.

Para obtener más información sobre la reutilización de contenedores, consulta Cómo entender la reutilización de contenedores AWS Lambda en el AWS blog de informática.

Configurar la contenerización de funciones Lambda

De forma predeterminada, las funciones de Lambda se ejecutan dentro de un AWS IoT Greengrass contenedor. Los contenedores Greengrass proporcionan aislamiento entre sus funciones y el anfitrión. Este aislamiento aumenta la seguridad tanto del anfitrión como de las funciones del contenedor.

Se recomienda ejecutar las funciones de Lambda en un contenedor de Greengrass, a menos que su caso de uso requiera que se ejecuten sin contenedorización. Al ejecutar las funciones de Lambda en un contenedor de Greengrass, tiene más control sobre la forma en que restringe el acceso a los recursos.

Puede ejecutar una función Lambda sin contenerización en los siguientes casos:

- Desea ejecutarla AWS IoT Greengrass en un dispositivo que no sea compatible con el modo contenedor. Un ejemplo sería si quisiera usar una distribución especial de Linux o si tiene una versión anterior del núcleo que está desactualizada.
- Desea ejecutar su función de Lambda en otro entorno de contenedor con su propio OverlayFS, pero detecta conflictos de OverlayFS cuando la ejecuta en un contenedor de Greengrass.
- Necesita acceder a recursos locales con rutas que no se puedan determinar en el momento de la implementación o cuyas rutas puedan cambiar después de la implementación. Un ejemplo de este recurso sería un dispositivo conectable.
- Tiene una aplicación anterior que se escribió como un proceso y tiene problemas al ejecutarla en un contenedor de Greengrass.

Diferencias en la creación de contenedores

Creación de contenedores	Notas
Contenedor de Greengrass	• Todas las características de AWS IoT Greengrass están disponibles cuando ejecuta una función de Lambda en un contenedor de Greengrass.

Creación de contenedores	Notas
	• Las funciones de Lambda que se ejecutan en un contenedor de Greengrass no tienen acceso al código desplegado de otras funciones de Lambda, incluso si se ejecutan con el mismo grupo de sistemas. En otras palabras, las funciones Lambda se ejecutan con un mayor aislamiento unas de otras. • Como el software AWS IoT Greengrass Core ejecuta todos los procesos secundarios en el mismo contenedor que la función Lambda, los procesos secundarios se detienen cuando se detiene la función Lambda.

Sin contenedor	• Las siguientes funciones no están disponibl es para las funciones de Lambda no contenedorizadas:
	  • Límites de memoria de funciones de Lambda.
	  • Dispositivos locales y recursos de volumen. Debe acceder a estos recursos utilizando sus rutas de archivo en el dispositivo principal en lugar de hacerlo como recursos de la función Lambda.
	• Si la función de Lambda que no está en un contenedor accede a un recurso de aprendizaje automático, debe identificar a un propietario del recurso y establecer permisos de acceso en el recurso, no en la función de Lambda.
	• Las funciones de Lambda no contenedo rizadas tienen acceso de solo lectura al código implementado de otras funciones de Lambda que se ejecutan con el mismo grupo de sistemas.

Si cambia la contenerización de una función de Lambda al implementarla, es posible que la función no funcione como se esperaba. Si la función Lambda utiliza recursos locales que ya no están disponibles con la nueva configuración de contenedorización, se produce un error en la implementación.

- Al cambiar una función Lambda de ejecutarse en un contenedor de Greengrass a ejecutarse sin contenerización, se descartan los límites de memoria de la función. Debe acceder al sistema de archivos directamente en lugar de utilizar los recursos locales asociados. Debe eliminar todos los recursos adjuntos antes de implementar la función Lambda.
- Al cambiar una función de Lambda de la ejecución sin creación de contenedores a la ejecución en un contenedor, la función de Lambda pierde el acceso directo al sistema de archivos. Debe definir un límite de memoria para cada función o aceptar el límite de memoria predeterminado de 16 MB. Puede configurar estos ajustes para cada función de Lambda al implementarla.

Para cambiar la configuración de contenedorización de un componente de la función Lambda, defina el valor del parámetro de `containerMode` configuración en una de las siguientes opciones al implementar el componente.

- `NoContainer`— El componente no se ejecuta en un entorno de ejecución aislado.
- `GreengrassContainer`— El componente se ejecuta en un entorno de ejecución aislado dentro del AWS IoT Greengrass contenedor.

Para obtener más información sobre cómo implementar y configurar los componentes, consulte Implemente AWS IoT Greengrass componentes en los dispositivos yActualizar las configuraciones de los componentes.

Importación de una función Lambda como componente (consola)

Cuando utiliza la AWS IoT Greengrassconsola para crear un componente de una función Lambda, importa una AWS Lambda función existente y, a continuación, la configura para crear un componente que se ejecute en su dispositivo Greengrass.

Antes de empezar, revise los requisitos para ejecutar las funciones de Lambda en los dispositivos Greengrass.

Tareas

- Paso 1: Elija una función Lambda para importarla

Paso 1: Elija una función Lambda para importarla

1. En el menú de navegación de la AWS IoT Greengrassconsola, seleccione Componentes.
2. En la página Componentes, seleccione Crear componente.
3. En la página Crear componente, en Información del componente, elija Importar función Lambda.
4. En Función Lambda, busque y elija la función Lambda que desee importar.

   AWS IoT Greengrasscrea el componente con el nombre de la función Lambda.
5. En la versión de la función Lambda, elija la versión que desee importar. No puedes elegir alias de Lambda como. `$LATEST`

   AWS IoT Greengrasscrea el componente con la versión de la función Lambda como una versión semántica válida. Por ejemplo, si la versión de la función es `3`, la versión del componente se convierte en `3.0.0`.

Paso 2: Configurar los parámetros de la función Lambda

En la página Crear componente, en Configuración de la función Lambda, configure los siguientes parámetros para utilizarlos en la ejecución de la función Lambda.

1. (Opcional) Añada la lista de fuentes de eventos a las que se suscribe la función Lambda para recibir mensajes de trabajo. Puede especificar las fuentes de eventos para suscribir esta función a los mensajes locales de publicación/suscripción y a los mensajes MQTT. AWS IoT Core Se llama a la función Lambda cuando recibe un mensaje de una fuente de eventos.

   > **Note**
   >
   > Para suscribir esta función a los mensajes de otras funciones o componentes de Lambda, implemente el componente de enrutador de suscripciones heredado al implementar este componente de función de Lambda. Al implementar el componente del

router de suscripciones heredado, especifique las suscripciones que utiliza la función Lambda.

En Fuentes de eventos, haga lo siguiente para agregar una fuente de eventos:

a. Para cada fuente de eventos que añada, especifique las siguientes opciones:

   - Tema: el tema al que suscribirse a los mensajes.
   - Tipo: el tipo de fuente de eventos. Puede elegir entre las siguientes opciones:
     - Publicación/suscripción local: suscríbase a los mensajes de publicación/suscripción locales.

       Si utiliza Greengrass nucleus v2.6.0 o posterior y Lambda manager v2.2.5 o posterior, puede usar comodines (**+**y**#**) de los temas de MQTT en el tema cuando especifique este tipo.
     - AWS IoT CoreAWS IoT CoreMQTT: suscríbase a los mensajes de MQTT.

       Si especifica este tipo, puede utilizar los caracteres comodín (+y#) de los temas de MQTT en el tema.

b. Para añadir otra fuente de eventos, elija Añadir fuente de eventos y repita el paso anterior. Para eliminar una fuente de eventos, selecciona Eliminar junto a la fuente de eventos que deseas eliminar.

2. En Tiempo de espera (segundos), introduzca la cantidad máxima de tiempo en segundos que puede ejecutar una función Lambda no anclada antes de que se agote el tiempo de espera. El valor predeterminado es de 3 segundos.
3. En Fijado, elija si el componente de la función Lambda está anclado. El valor predeterminado es True.

   - Una función Lambda anclada (o de larga duración) se inicia cuando se AWS IoT Greengrass inicia y sigue ejecutándose en su propio contenedor.
   - Una función Lambda no anclada (o bajo demanda) se inicia solo cuando recibe un elemento de trabajo y se cierra después de permanecer inactiva durante un tiempo de inactividad máximo especificado. Si la función tiene varios elementos de trabajo, el software de AWS IoT Greengrass Core crea varias instancias de la función.
4. (Opcional) En Parámetros adicionales, defina los siguientes parámetros de la función Lambda.

- Tiempo de espera de estado (segundos): intervalo en segundos en el que el componente de la función Lambda envía las actualizaciones de estado al componente del administrador de Lambda. Este parámetro solo se aplica a las funciones ancladas. El valor predeterminado es de 60 segundos.
- Tamaño máximo de cola: tamaño máximo de la cola de mensajes para el componente de la función Lambda. El software AWS IoT Greengrass Core almacena los mensajes en una cola FIFO (primero en entrar, primero en salir) hasta que pueda ejecutar la función Lambda para consumir cada mensaje. El valor predeterminado es de 1000 mensajes.
- Número máximo de instancias: el número máximo de instancias que una función Lambda no anclada puede ejecutar al mismo tiempo. El valor predeterminado es 100 instancias.
- Tiempo máximo de inactividad (segundos): cantidad máxima de tiempo en segundos que una función Lambda no anclada puede permanecer inactiva antes de que el software AWS IoT Greengrass principal detenga su proceso. El valor predeterminado es de 60 segundos.
- Tipo de codificación: el tipo de carga útil que admite la función Lambda. Puede elegir entre las siguientes opciones:
  - JSON
  - Binario

  El valor predeterminado es JSON.

5. (Opcional) Especifique la lista de argumentos de la línea de comandos para pasarlos a la función Lambda cuando se ejecute.

   a. En Parámetros adicionales, Argumentos de proceso, elija Añadir argumento.

   b. Para cada argumento que añada, introduzca el argumento que desee pasar a la función.

   c. Para eliminar un argumento, elija Eliminar junto al argumento que desee eliminar.

6. (Opcional) Especifique las variables de entorno que están disponibles para la función Lambda cuando se ejecuta. Las variables de entorno permiten almacenar y actualizar los ajustes de configuración sin necesidad de cambiar el código de la función.

   a. En Parámetros adicionales, Variables de entorno, elija Añadir variable de entorno.

   b. Para cada variable de entorno que añada, especifique las siguientes opciones:

      - Clave: el nombre de la variable.
      - Valor: el valor predeterminado de esta variable.

c. Para eliminar una variable de entorno, elija Eliminar junto a la variable de entorno que desee eliminar.

Paso 3: (opcional) Especifique las plataformas compatibles para la función Lambda

Todos los dispositivos principales tienen atributos para el sistema operativo y la arquitectura. Al implementar el componente de la función Lambda, el software AWS IoT Greengrass Core compara los valores de plataforma que especifique con los atributos de la plataforma en el dispositivo principal para determinar si la función Lambda es compatible con ese dispositivo.

> **Note**
>
> También puede especificar atributos de plataforma personalizados al implementar el componente núcleo de Greengrass en un dispositivo principal. Para obtener más información, consulte el parámetro de anulación de plataforma del componente núcleo de Greengrass.

En Configuración de la función Lambda, Parámetros adicionales, Plataformas, haga lo siguiente para especificar las plataformas que admite esta función Lambda.

1. Para cada plataforma, especifique las siguientes opciones:

   - Sistema operativo: nombre del sistema operativo de la plataforma. Actualmente el único valor admitido es `linux`.
   - Arquitectura: la arquitectura del procesador de la plataforma. Los valores admitidos son:
     - `amd64`
     - `arm`
     - `aarch64`
     - `x86`

2. Para añadir otra plataforma, elija Añadir plataforma y repita el paso anterior. Para eliminar una plataforma compatible, selecciona Eliminar junto a la plataforma que deseas eliminar.

Paso 4: (opcional) Especificar las dependencias de los componentes para la función Lambda

Las dependencias de los componentes identifican los componentes adicionales AWS proporcionados o los componentes personalizados que utiliza la función. Al implementar el componente de la función Lambda, la implementación incluye estas dependencias para que la función se ejecute.

> **Important**
>
> Para importar una función Lambda que haya creado para ejecutarse en la AWS IoT Greengrass V1, debe definir las dependencias de los componentes individuales para las funciones que utiliza la función, como los secretos, las sombras locales y el administrador de flujos. Defina estos componentes como dependencias rígidas para que el componente de la función Lambda se reinicie si la dependencia cambia de estado. Para obtener más información, consulte Importación de funciones Lambda V1.

En Configuración de la función Lambda, Parámetros adicionales, Dependencias de componentes, complete los siguientes pasos para especificar las dependencias de los componentes de la función Lambda.

1. Seleccione Añadir dependencia.
2. Para cada dependencia de componentes que añada, especifique las siguientes opciones:

   - Nombre del componente: el nombre del componente. Por ejemplo, introduzca **`aws.greengrass.StreamManager`** para incluir el componente del administrador de flujos.
   - Requisito de versión: la restricción de versión semántica de estilo npm que identifica las versiones compatibles de la dependencia de este componente. Puede especificar una sola versión o un rango de versiones. Por ejemplo, introduzca **`^1.0.0`** para especificar que esta función Lambda depende de cualquier versión de la primera versión principal del componente administrador de flujos. Para obtener más información sobre las restricciones de la versión semántica, consulte la calculadora npm semver.
   - Tipo: el tipo de dependencia. Puede elegir entre las siguientes opciones:
     - Duro: el componente de la función Lambda se reinicia si la dependencia cambia de estado. Esta es la selección predeterminada.
     - Suave: el componente de la función Lambda no se reinicia si la dependencia cambia de estado.

3. Para eliminar una dependencia de un componente, elija Eliminar junto a la dependencia del componente

Paso 5: (opcional) Ejecute la función Lambda en un contenedor

De forma predeterminada, las funciones Lambda se ejecutan en un entorno de ejecución aislado dentro del software AWS IoT Greengrass Core. También puede optar por ejecutar la función Lambda como un proceso sin aislamiento (es decir, en modo sin contenedor).

En la configuración de procesos de Linux, en el modo de aislamiento, elija una de las siguientes opciones para seleccionar la contenerización de la función Lambda:

- Contenedor Greengrass: la función Lambda se ejecuta en un contenedor. Esta es la selección predeterminada.
- Sin contenedor: la función Lambda se ejecuta como un proceso sin ningún tipo de aislamiento.

Si ejecuta la función Lambda en un contenedor, complete los siguientes pasos para configurar la configuración del proceso para la función Lambda.

1. Configure la cantidad de memoria y los recursos del sistema, como los volúmenes y los dispositivos, para ponerlos a disposición del contenedor.

   En Parámetros del contenedor, haga lo siguiente.

   a. En Tamaño de memoria, introduzca el tamaño de memoria que desee asignar al contenedor. Puede especificar el tamaño de la memoria en MB o KB.

   b. En el caso de la carpeta sys de solo lectura, elija si el contenedor puede leer o no la información de la carpeta del `/sys` dispositivo. El valor predeterminado es False.

2. (Opcional) Configure los volúmenes locales a los que puede acceder la función Lambda en contenedores. Cuando define un volumen, el software de AWS IoT Greengrass Core monta los archivos de origen en el destino dentro del contenedor.

   a. En Volúmenes, elija Añadir volumen.

   b. Para cada volumen que añada, especifique las siguientes opciones:

      - Volumen físico: la ruta a la carpeta de origen del dispositivo principal.
      - Volumen lógico: la ruta a la carpeta de destino del contenedor.

- Permiso: (opcional) el permiso para acceder a la carpeta de origen desde el contenedor. Puede elegir entre las siguientes opciones:
  - Solo lectura: la función Lambda tiene acceso de solo lectura a la carpeta de origen. Esta es la selección predeterminada.
  - Lectura y escritura: la función Lambda tiene acceso de lectura y escritura a la carpeta de origen.
- Agregar propietario del grupo: (opcional) si se debe agregar o no el grupo de sistemas que ejecuta el componente de la función Lambda como propietario de la carpeta de origen. El valor predeterminado es False.

c. Para eliminar un volumen, seleccione Eliminar junto al volumen que desee eliminar.

3. (Opcional) Configure los dispositivos del sistema local a los que puede acceder la función Lambda en contenedor.

   a. En Dispositivos, elija Agregar dispositivo.

   b. Para cada dispositivo que añada, especifique las siguientes opciones:

   - Ruta de montaje: la ruta al dispositivo del sistema en el dispositivo principal.
   - Permiso: (opcional) el permiso para acceder al dispositivo del sistema desde el contenedor. Puede elegir entre las siguientes opciones:
     - Solo lectura: la función Lambda tiene acceso de solo lectura al dispositivo del sistema. Esta es la selección predeterminada.
     - Lectura y escritura: la función Lambda tiene acceso de lectura y escritura a la carpeta de origen.
   - Agregar propietario del grupo: (opcional) si se debe agregar o no el grupo de sistemas que ejecuta el componente de la función Lambda como propietario del dispositivo del sistema. El valor predeterminado es False.

Paso 6: Crear el componente de la función Lambda

Después de configurar los ajustes del componente de la función Lambda, elija Crear para terminar de crear el nuevo componente.

Para ejecutar la función Lambda en su dispositivo principal, puede implementar el nuevo componente en sus dispositivos principales. Para obtener más información, consulte Implemente AWS IoT Greengrass componentes en los dispositivos.

Importar una función Lambda como componente () AWS CLI

Utilice la CreateComponentVersionoperación para crear componentes a partir de funciones Lambda. Cuando llame a esta operación, especifique `lambdaFunction` la importación de una función Lambda.

Tareas

- Paso 1: Definir la configuración de la función Lambda
- Paso 2: Crear el componente de la función Lambda

Paso 1: Definir la configuración de la función Lambda

1. Cree un archivo llamado `ylambda-function-component.json`, a continuación, copie el siguiente objeto JSON en el archivo. Sustituya el `lambdaArn` por el ARN de la función Lambda que se va a importar.

   ```
   {
     "lambdaFunction": {
       "lambdaArn": "arn:aws:lambda:region:account-id:function:HelloWorld:1"
     }
   }
   ```

   > **Important**
   >
   > Debe especificar un ARN que incluya la versión de la función que se va a importar. No puede utilizar alias de versión como `$LATEST`.

2. (Opcional) Especifique el nombre (`componentName`) del componente. Si omite este parámetro, AWS IoT Greengrass crea el componente con el nombre de la función Lambda.

   ```
   {
     "lambdaFunction": {
       "lambdaArn": "arn:aws:lambda:region:account-id:function:HelloWorld:1",
       "componentName": "com.example.HelloWorldLambda"
     }
   }
   ```

3. (Opcional) Especifique la versión (`componentVersion`) del componente. Si omite este parámetro, AWS IoT Greengrass crea el componente con la versión de la función Lambda como una versión semántica válida. Por ejemplo, si la versión de la función es `3`, la versión del componente se convierte en `3.0.0`.

   > **Note**
   >
   > Cada versión de componente que cargue debe ser única. Asegúrese de cargar la versión del componente correcta, ya que no podrá editarla después de cargarla. AWS IoT Greengrassusa versiones semánticas para los componentes. Las versiones semánticas siguen un sistema de números de principal.secundario.parche. Por ejemplo, la versión `1.0.0` representa la primera versión principal de un componente. Para obtener más información, consulte la especificación de la versión semántica.

   ```
   {
     "lambdaFunction": {
       "lambdaArn": "arn:aws:lambda:region:account-id:function:HelloWorld:1",
       "componentName": "com.example.HelloWorldLambda",
       "componentVersion": "1.0.0"
     }
   }
   ```

4. (Opcional) Especifique las plataformas compatibles con esta función Lambda. Cada plataforma contiene un mapa de atributos que la identifican. Todos los dispositivos principales tienen atributos de sistema operativo (`os`) y arquitectura (`architecture`). El software AWS IoT Greengrass principal puede añadir otros atributos de plataforma. También puede especificar atributos de plataforma personalizados al implementar el componente núcleo de Greengrass en un dispositivo principal. Haga lo siguiente:

   a. Añada una lista de plataformas (`componentPlatforms`) a la función Lambda en. `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. Añada cada plataforma compatible a la lista. Cada plataforma tiene un amigable `name` para identificarla y un mapa de atributos. El siguiente ejemplo especifica que esta función es compatible con dispositivos x86 que ejecutan Linux.

```
{
  "name": "Linux x86",
  "attributes": {
    "os": "linux",
    "architecture": "x86"
  }
}
```

`lambda-function-component.json`Puede contener un documento similar al siguiente ejemplo.

```
{
  "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. (Opcional) Especifique las dependencias de los componentes de la función Lambda. Al implementar el componente de la función Lambda, la implementación incluye estas dependencias para que la función se ejecute.

> **Important**
>
> Para importar una función Lambda que haya creado para ejecutarse en la AWS IoT Greengrass V1, debe definir las dependencias de los componentes individuales para las funciones que utiliza la función, como los secretos, las sombras locales y el administrador de flujos. Defina estos componentes como dependencias rígidas para que el componente de la función Lambda se reinicie si la dependencia cambia de estado. Para obtener más información, consulte Importación de funciones Lambda V1.

Haga lo siguiente:

a. Añada un mapa de las dependencias de los componentes (`componentDependencies`) a la función Lambda en. `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. Agregue cada dependencia de los componentes al mapa. Especifique el nombre del componente como clave y especifique un objeto con los siguientes parámetros:

   - `versionRequirement`— La restricción de versión semántica de estilo npm que identifica las versiones compatibles de la dependencia del componente. Puede

especificar una versión única o un rango de versiones. Para obtener más información sobre las restricciones de las versiones semánticas, consulte la calculadora npm semver.

- `dependencyType`— (Opcional) El tipo de dependencia. Elija una de las siguientes opciones:
  - `SOFT`— El componente de la función Lambda no se reinicia si la dependencia cambia de estado.
  - `HARD`— El componente de la función Lambda se reinicia si la dependencia cambia de estado.

  El valor predeterminado es `HARD`.

El siguiente ejemplo especifica que esta función Lambda depende de cualquier versión de la primera versión principal del componente stream manager. El componente de la función Lambda se reinicia cuando el administrador de transmisiones se reinicia o se actualiza.

```
{
  "aws.greengrass.StreamManager": {
    "versionRequirement": "^1.0.0",
    "dependencyType": "HARD"
  }
}
```

`lambda-function-component.json`Puede contener un documento similar al siguiente ejemplo.

```
{
  "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. (Opcional) Configure los parámetros de la función Lambda que se utilizarán para ejecutar la función. Puede configurar opciones como las variables de entorno, las fuentes de eventos de los mensajes, los tiempos de espera y la configuración del contenedor. Haga lo siguiente:

   a. Agregue el objeto de parámetros de Lambda (`componentLambdaParameters`) a la función Lambda en. `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. (Opcional) Especifique las fuentes de eventos a las que se suscribe la función Lambda para los mensajes de trabajo. Puede especificar las fuentes de eventos para suscribir esta función a los mensajes locales de publicación/suscripción y a los mensajes MQTT. AWS IoT Core Se llama a la función Lambda cuando recibe un mensaje de una fuente de eventos.

> **Note**
>
> Para suscribir esta función a los mensajes de otras funciones o componentes de Lambda, implemente el componente de enrutador de suscripciones heredado al implementar este componente de función de Lambda. Al implementar el componente del router de suscripciones heredado, especifique las suscripciones que utiliza la función Lambda.

Haga lo siguiente:

i. Añada la lista de fuentes de eventos (`eventSources`) a los parámetros de la función 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. Añada cada fuente de eventos a la lista. Cada fuente de eventos tiene los siguientes parámetros:

- `topic`— El tema al que suscribirse a los mensajes.
- `type`— El tipo de fuente del evento. Puede elegir entre las siguientes opciones:
  - `PUB_SUB` — Suscribirse a mensajes locales de publicación/suscripción.

    Si usa Greengrass nucleus v2.6.0 o posterior y Lambda manager v2.2.5 o posterior, puede usar los comodines (+y) de los temas MQTT cuando especifique este tipo. `# topic`
  - `IOT_CORE`: Suscribirse a mensajes MQTT de AWS IoT Core.

    Al especificar este tipo, puede utilizar los caracteres comodín de los temas MQTT (+y). `# topic`

En el siguiente ejemplo, se suscribe a AWS IoT Core MQTT los temas que coincidan con el filtro de temas. `hello/world/+`

```
{
  "topic": "hello/world/+",
  "type": "IOT_CORE"
}
```

El tuyo `lambda-function-component.json` podría tener un aspecto similar al del siguiente ejemplo.

```
{
  "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. (Opcional) Especifique cualquiera de los siguientes parámetros en el objeto de parámetros de la función Lambda:

- `environmentVariables`— El mapa de variables de entorno que están disponibles para la función Lambda cuando se ejecuta.
- `execArgs`— La lista de argumentos que se van a pasar a la función Lambda cuando se ejecuta.
- `inputPayloadEncodingType`— El tipo de carga útil que admite la función Lambda. Puede elegir entre las siguientes opciones:
  - `json`
  - `binary`

  Valor predeterminado: `json`
- `pinned`— Si la función Lambda está fija o no. El valor predeterminado es `true`.
  - Una función Lambda anclada (o de larga duración) se inicia cuando se AWS IoT Greengrass inicia y sigue ejecutándose en su propio contenedor.

- Una función Lambda no anclada (o bajo demanda) se inicia solo cuando recibe un elemento de trabajo y se cierra después de permanecer inactiva durante un tiempo de inactividad máximo especificado. Si la función tiene varios elementos de trabajo, el software de AWS IoT Greengrass Core crea varias instancias de la función.

  Se utiliza `maxIdleTimeInSeconds` para establecer el tiempo máximo de inactividad de la función.

- `timeoutInSeconds`— El tiempo máximo en segundos que la función Lambda puede ejecutarse antes de que se agote el tiempo de espera. El valor predeterminado es de 3 segundos.
- `statusTimeoutInSeconds`— El intervalo en segundos en el que el componente de la función Lambda envía las actualizaciones de estado al componente del administrador de Lambda. Este parámetro solo se aplica a las funciones ancladas. El valor predeterminado es de 60 segundos.
- `maxIdleTimeInSeconds`— El tiempo máximo en segundos que una función Lambda no anclada puede permanecer inactiva antes de que el software AWS IoT Greengrass Core detenga su proceso. El valor predeterminado es de 60 segundos.
- `maxInstancesCount`— El número máximo de instancias que una función Lambda no anclada puede ejecutar al mismo tiempo. El valor predeterminado es 100 instancias.
- `maxQueueSize`— El tamaño máximo de la cola de mensajes para el componente de la función Lambda. El software AWS IoT Greengrass Core almacena los mensajes en una cola FIFO (first-in-first-out) hasta que pueda ejecutar la función Lambda para consumir cada mensaje. El valor predeterminado es 1000 mensajes.

`lambda-function-component.json`Puede contener un documento similar al siguiente ejemplo.

```
{
  "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. (Opcional) Configure los ajustes del contenedor para la función Lambda. De forma predeterminada, las funciones Lambda se ejecutan en un entorno de ejecución aislado dentro del software AWS IoT Greengrass Core. También puede optar por ejecutar la función Lambda como un proceso sin ningún tipo de aislamiento. Si ejecuta la función Lambda en un contenedor, configura el tamaño de memoria del contenedor y los recursos del sistema disponibles para la función Lambda. Haga lo siguiente:

   i. Agregue el objeto de parámetros de proceso de Linux (`linuxProcessParams`) al objeto de parámetros de Lambda en. `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. (Opcional) Especifique si la función Lambda se ejecuta o no en un contenedor. Añada el `isolationMode` parámetro al objeto de parámetros del proceso y elija una de las siguientes opciones:

- `GreengrassContainer`— La función Lambda se ejecuta en un contenedor.
- `NoContainer`— La función Lambda se ejecuta como un proceso sin ningún tipo de aislamiento.

El valor predeterminado es `GreengrassContainer`.

iii. (Opcional) Si ejecuta la función Lambda en un contenedor, puede configurar la cantidad de memoria y los recursos del sistema, como los volúmenes y los dispositivos, que se pondrán a disposición del contenedor. Haga lo siguiente:

A. Agregue el objeto de parámetros del contenedor (`containerParams`) al objeto de parámetros de proceso de Linux en`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. (Opcional) Agregue el `memorySizeInKB` parámetro para especificar el tamaño de memoria del contenedor. El valor predeterminado es 16.384 KB (16 MB).

C. (Opcional) Agregue el `mountROSysfs` parámetro para especificar si el contenedor puede leer o no la información de la `/sys` carpeta del dispositivo. El valor predeterminado es `false`.

D. (Opcional) Configure los volúmenes locales a los que puede acceder la función Lambda en contenedores. Cuando define un volumen, el software de AWS IoT Greengrass Core monta los archivos de origen en el destino dentro del contenedor. Haga lo siguiente:

   I. Añada la lista de volúmenes (`volumes`) a los parámetros del contenedor.

```
{
```

```
  "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. Añada cada volumen a la lista. Cada volumen tiene los siguientes parámetros:

- `sourcePath`— La ruta a la carpeta de origen del dispositivo principal.
- `destinationPath`— La ruta a la carpeta de destino del contenedor.
- `permission`— (Opcional) El permiso para acceder a la carpeta de origen desde el contenedor. Puede elegir entre las siguientes opciones:
  - `ro`— La función Lambda tiene acceso de solo lectura a la carpeta de origen.
  - `rw`— La función Lambda tiene acceso de lectura y escritura a la carpeta de origen.

  El valor predeterminado es `ro`.
- `addGroupOwner`— (Opcional) Si se debe añadir o no el grupo de sistemas que ejecuta el componente de la función Lambda como propietario de la carpeta de origen. El valor predeterminado es `false`.

`lambda-function-component.json`Puede contener un documento similar al siguiente ejemplo.

```
{
  "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. (Opcional) Configure los dispositivos del sistema local a los que puede acceder la función Lambda en contenedor. Haga lo siguiente:

I. Añada la lista de dispositivos del sistema (`devices`) a los parámetros del contenedor.

```
{
  "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. Añada cada dispositivo del sistema a la lista. Cada dispositivo del sistema tiene los siguientes parámetros:

- `path`— La ruta al dispositivo del sistema en el dispositivo principal.
- `permission`— (Opcional) El permiso para acceder al dispositivo del sistema desde el contenedor. Puede elegir entre las siguientes opciones:
  - `ro`— La función Lambda tiene acceso de solo lectura al dispositivo del sistema.
  - `rw`— La función Lambda tiene acceso de lectura y escritura al dispositivo del sistema.

  El valor predeterminado es `ro`.

- `addGroupOwner`— (Opcional) Si se debe añadir o no el grupo de sistemas que ejecuta el componente de la función Lambda como propietario del dispositivo del sistema. El valor predeterminado es `false`.

`lambda-function-component.json`Puede contener un documento similar al siguiente ejemplo.

```
{
  "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. (Opcional) Añada etiquetas (`tags`) para el componente. Para obtener más información, consulte Etiquetar los recursos de AWS IoT Greengrass Version 2.

Paso 2: Crear el componente de la función Lambda

1. Ejecute el siguiente comando para crear el componente de la función Lambda desde. `lambda-function-component.json`

```
aws greengrassv2 create-component-version --cli-input-json file://lambda-function-component.json
```

La respuesta es similar a la del siguiente ejemplo si la solicitud se realiza correctamente.

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

`arn`Copie el elemento del resultado para comprobar el estado del componente en el siguiente paso.

2. Al crear un componente, su estado es`REQUESTED`. A continuación, AWS IoT Greengrass valida que el componente sea desplegable. Puede ejecutar el siguiente comando para consultar el estado del componente y comprobar que el componente se puede implementar. Sustituya el `arn` por el ARN del paso anterior.

```
aws greengrassv2 describe-component \
  --arn "arn:aws:greengrass:region:account-
id:components:com.example.HelloWorldLambda:versions:1.0.0"
```

Si el componente se valida, la respuesta indica que el estado del componente es. `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"
      }
    }
  ]
}
```

Una vez creado el componente`DEPLOYABLE`, puede implementar la función Lambda en sus dispositivos principales. Para obtener más información, consulte Implemente AWS IoT Greengrass componentes en los dispositivos.

Úselo SDK para dispositivos con AWS IoT para comunicarse con el núcleo de Greengrass, otros componentes y AWS IoT Core

Los componentes que se ejecutan en su dispositivo principal pueden utilizar la biblioteca de comunicación entre procesos (IPC) del AWS IoT Greengrass núcleo SDK para dispositivos con AWS IoT para comunicarse con el AWS IoT Greengrass núcleo y otros componentes de Greengrass. Para desarrollar y ejecutar componentes personalizados que utilicen la IPC, debe utilizarla SDK para dispositivos con AWS IoT para conectarse al servicio AWS IoT Greengrass Core IPC y realizar las operaciones de IPC.

La interfaz IPC admite dos tipos de operaciones:

- Solicitud/respuesta

  Los componentes envían una solicitud al servicio de IPC y reciben una respuesta que contiene el resultado de la solicitud.
- Suscripción

  Los componentes envían una solicitud de suscripción al servicio de IPC y esperan recibir un flujo de mensajes de eventos como respuesta. Los componentes proporcionan un gestor de suscripciones que gestiona los mensajes de eventos, los errores y el cierre de la transmisión. SDK para dispositivos con AWS IoT Incluye una interfaz de controlador con la respuesta y los tipos de eventos correctos para cada operación de IPC. Para obtener más información, consulte Suscríbase a las transmisiones de eventos del IPC.

Temas

- Publicar/suscribir mensajes MQTT AWS IoT Core
- Interactúe con el ciclo de vida del componente
- Interactúa con la configuración de los componentes
- Recuperar valores secretos
- Interactúa con las sombras locales
- Gestione las implementaciones y los componentes locales
- Autenticar y autorizar los dispositivos cliente

Versiones de cliente IPC

En versiones posteriores de los SDK de Java y Python, AWS IoT Greengrass proporciona una versión mejorada del cliente IPC, denominada cliente IPC V2. Cliente IPC V2:

- Reduce la cantidad de código que debe escribirse para utilizar las operaciones de IPC y ayuda a evitar los errores habituales que pueden producirse con el cliente de IPC V1.
- Realiza llamadas a los controladores de suscripciones en un hilo independiente, por lo que ahora puede ejecutar código de bloqueo, incluidas las llamadas a funciones de IPC adicionales, en las devoluciones de llamadas a los controladores de suscripciones. El cliente de IPC V1 utiliza el mismo hilo para comunicarse con el servidor de IPC y para llamar a los gestores de suscripciones.
- Permite llamar a las operaciones de suscripción mediante expresiones Lambda (Java) o funciones (Python). El cliente IPC V1 requiere que defina las clases de controladores de suscripciones.
- Proporciona versiones síncronas y asíncronas de cada operación de IPC. El cliente IPC V1 proporciona solo versiones asíncronas de cada operación.

Le recomendamos que utilice el cliente IPC V2 para aprovechar estas mejoras. Sin embargo, muchos ejemplos de esta documentación y de algunos contenidos en línea muestran únicamente cómo utilizar el cliente IPC V1. Puede utilizar los siguientes ejemplos y tutoriales para ver ejemplos de componentes que utilizan el cliente IPC V2:

- PublishToTopicejemplos
- SubscribeToTopicejemplos
- Tutorial: Desarrolle un componente de Greengrass que aplace las actualizaciones de los componentes

- Tutorial: Interactúa con dispositivos IoT locales a través de MQTT

Actualmente, la versión 2 SDK para dispositivos con AWS IoT para C++ solo es compatible con el cliente IPC V1.

SDK compatibles para la comunicación entre procesos

Las bibliotecas AWS IoT Greengrass Core IPC se incluyen en las siguientes versiones. SDK para dispositivos con AWS IoT

SDK	Versión mínima	Uso
SDK para dispositivos con AWS IoT para Java v2	v1.6.0	Consulte Úselo SDK para dispositivos con AWS IoT para Java v2 (cliente IPC V2)
SDK para dispositivos con AWS IoT para Python v2	v1.9.0	Consulte Uso SDK para dispositivos con AWS IoT para Python v2 (cliente IPC V2)
SDK para dispositivos con AWS IoT para C++ v2	v1.17.0	Consulte Úselo SDK para dispositivos con AWS IoT para C++ v2
SDK para dispositivos con AWS IoT para v2 JavaScript	v1.12.0	Consulte Úselo SDK para dispositivos con AWS IoT para la JavaScript versión 2 (cliente IPC V1)

Conéctese al AWS IoT Greengrass servicio Core IPC

Para utilizar la comunicación entre procesos en su componente personalizado, debe crear una conexión a un socket de servidor IPC que ejecute el software AWS IoT Greengrass Core. Realice

las siguientes tareas para descargarlo y usarlo SDK para dispositivos con AWS IoT en el idioma que prefiera.

Úselo SDK para dispositivos con AWS IoT para Java v2 (cliente IPC V2)

Para usar la versión 2 SDK para dispositivos con AWS IoT para Java (cliente IPC V2)

1. Descargue la versión SDK para dispositivos con AWS IoT para Java v2 (v1.6.0 o posterior).
2. Realice una de las siguientes acciones para ejecutar el código personalizado en el componente:
   - Cree el componente como un archivo JAR que incluya el SDK para dispositivos con AWS IoT archivo JAR y ejecútelo en la receta del componente.
   - Defina el SDK para dispositivos con AWS IoT JAR como un artefacto de componente y añada ese artefacto a la ruta de clases cuando ejecute la aplicación en la receta de su componente.
3. Utilice el siguiente código para crear el cliente IPC.

```
try (GreengrassCoreIPCClientV2 ipcClient =
 GreengrassCoreIPCClientV2.builder().build()) {
    // Use client.
} catch (Exception e) {
    LOGGER.log(Level.SEVERE, "Exception occurred when using IPC.", e);
    System.exit(1);
}
```

Uso SDK para dispositivos con AWS IoT para Python v2 (cliente IPC V2)

Para usar SDK para dispositivos con AWS IoT para Python v2 (cliente IPC V2)

1. Descargue el SDK para dispositivos con AWS IoT para Python (v1.9.0 o posterior).
2. Añada los pasos de instalación del SDK al ciclo de vida de la instalación en la receta de su componente.
3. Cree una conexión con el servicio AWS IoT Greengrass Core IPC. Utilice el siguiente código para crear el cliente 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)
```

Úselo SDK para dispositivos con AWS IoT para C++ v2

Para compilar la SDK para dispositivos con AWS IoT versión 2 para C++, un dispositivo debe tener las siguientes herramientas:

- C++ 1.1 o posterior
- CMake 3.1 o posterior
- Uno de los siguientes compiladores:
  - GCC 4.8 o posterior
  - Clang 3.9 o posterior
  - MSVC 2015 o posterior

Para usar la versión 2 SDK para dispositivos con AWS IoT para C++

1. Descargue la versión SDK para dispositivos con AWS IoT para C++ (v1.17.0 o posterior).
2. Siga las instrucciones de instalación del archivo README para compilar la versión 2 para C++ a partir del SDK para dispositivos con AWS IoT código fuente.
3. En la herramienta de compilación de C++, vincula la biblioteca IPC de Greengrass que creaste en el paso anterior. `AWS::GreengrassIpc-cpp` En el siguiente `CMakeLists.txt` ejemplo, se vincula la biblioteca IPC de Greengrass a un proyecto que se crea con CMake.

```
cmake_minimum_required(VERSION 3.1)
project (greengrassv2_pubsub_subscriber)

file(GLOB MAIN_SRC
        "*.h"
        "*.cpp"
        )
add_executable(${PROJECT_NAME} ${MAIN_SRC})

set_target_properties(${PROJECT_NAME} PROPERTIES
        LINKER_LANGUAGE CXX
```

```
        CXX_STANDARD 11)
find_package(aws-crt-cpp PATHS ~/sdk-cpp-workspace/build)
find_package(EventstreamRpc-cpp PATHS ~/sdk-cpp-workspace/build)
find_package(GreengrassIpc-cpp PATHS ~/sdk-cpp-workspace/build)
target_link_libraries(${PROJECT_NAME} AWS::GreengrassIpc-cpp)
```

4. En el código del componente, crea una conexión con el servicio AWS IoT Greengrass Core IPC para crear un cliente IPC (). `Aws::Greengrass::GreengrassCoreIpcClient` Debe definir un controlador del ciclo de vida de las conexiones de IPC que gestione los eventos de conexión, desconexión y error de IPC. El siguiente ejemplo crea un cliente de IPC y un controlador del ciclo de vida de las conexiones de IPC que se imprimen cuando el cliente de IPC se conecta, se desconecta y detecta errores.

```
#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. Para ejecutar el código personalizado en el componente, cree el código como un artefacto binario y ejecute el artefacto binario en la receta del componente. Defina el `Execute` permiso del artefacto para `OWNER` permitir que el software AWS IoT Greengrass principal ejecute el artefacto binario.

   La `Manifests` sección de la receta de tus componentes podría tener un aspecto similar al siguiente ejemplo.

   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
```

Úselo SDK para dispositivos con AWS IoT para la JavaScript versión 2 (cliente IPC V1)

Para compilar la JavaScript versión 2 SDK para dispositivos con AWS IoT para usarla con Nodejs, un dispositivo debe tener las siguientes herramientas:

- Nodejs 10.0 o posterior
  - Ejecute `node -v` para comprobar la versión de Node.
- CMake 3.1 o posterior

Para usar la SDK para dispositivos con AWS IoT JavaScript versión 2 (cliente IPC V1)

1. Descargue la JavaScript versión SDK para dispositivos con AWS IoT para la versión 2 (v1.12.10 o posterior).
2. Siga las instrucciones de instalación del archivo README para compilar la versión para la versión 2 a partir del código SDK para dispositivos con AWS IoT fuente JavaScript.
3. Cree una conexión al servicio AWS IoT Greengrass Core IPC. Complete los siguientes pasos para crear el cliente IPC y establecer una conexión.
4. Utilice el siguiente código para crear el cliente IPC.

   ```
   import * as greengrascoreipc from 'aws-iot-device-sdk-v2';

   let client = greengrascoreipc.createClient();
   ```

5. Utilice el siguiente código para establecer una conexión entre el componente y el núcleo de Greengrass.

   ```
   await client.connect();
   ```

Autorice a los componentes a realizar operaciones de IPC

Para permitir que sus componentes personalizados utilicen algunas operaciones de IPC, debe definir políticas de autorización que permitan al componente realizar la operación en determinados

recursos. Cada política de autorización define una lista de operaciones y una lista de recursos que la política permite. Por ejemplo, el servicio IPC de publicación y suscripción de mensajería define las operaciones de publicación y suscripción de los recursos temáticos. Puede utilizar el * comodín para permitir el acceso a todas las operaciones o a todos los recursos.

Las políticas de autorización se definen con el parámetro de `accessControl` configuración, que se puede establecer en la receta del componente o al implementar el componente. El `accessControl` objeto asigna los identificadores del servicio de IPC a listas de políticas de autorización. Puede definir varias políticas de autorización para cada servicio de IPC a fin de controlar el acceso. Cada política de autorización tiene un identificador de política, que debe ser único entre todos los componentes.

> **Tip**
>
> Para crear identificadores de política únicos, puede combinar el nombre del componente, el nombre del servicio de IPC y un contador. Por ejemplo, un componente denominado `com.example.HelloWorld` podría definir dos políticas de autorización de publicación o suscripción con los siguientes identificadores:
>
> - `com.example.HelloWorld:pubsub:1`
> - `com.example.HelloWorld:pubsub:2`

Las políticas de autorización utilizan el siguiente formato. Este objeto es el parámetro `accessControl` de configuración.

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
```

Comodín en las políticas de autorización

Puede utilizar el * comodín en el `resources` elemento de las políticas de autorización del IPC para permitir el acceso a varios recursos en una única política de autorización.

- En todas las versiones del núcleo de Greengrass, puede especificar un solo * carácter como recurso para permitir el acceso a todos los recursos.
- En Greengrass nucleus v2.6.0 y versiones posteriores, puede especificar el * carácter de un recurso para que coincida con cualquier combinación de caracteres. Por ejemplo, puede especificar que se permita el acceso `factory/1/devices/Thermostat*/status` a un tema de estado para todos los dispositivos de termostato de una fábrica, donde comience el nombre de cada dispositivo. `Thermostat`

Al definir políticas de autorización para el servicio IPC de AWS IoT Core MQTT, también puede utilizar los caracteres comodín (+y#) de MQTT para hacer coincidir varios recursos. Para obtener más información, consulte los caracteres comodín de MQTT en las políticas de autorización de IPC de MQTT. AWS IoT Core

Variables de receta en las políticas de autorización

Si usa Greengrass nucleus v2.6.0 o posterior y establece la opción de interpolateComponentConfigurationconfiguración del núcleo de Greengrass en`true`, puede usar la

variable de receta en las políticas de autorización. `{iot:thingName}` Cuando necesite una política de autorización que incluya el nombre del dispositivo principal, como en el caso de los temas de MQTT o los dispositivos ocultos, puede utilizar esta variable de receta para configurar una política de autorización única para un grupo de dispositivos principales. Por ejemplo, puede permitir que un componente acceda al siguiente recurso para realizar operaciones de IPC ocultas.

```
$aws/things/{iot:thingName}/shadow/
```

Caracteres especiales en las políticas de autorización

Para especificar un literal `*` o un `?` carácter en una política de autorización, debe utilizar una secuencia de escape. Las siguientes secuencias de escape indican al software AWS IoT Greengrass Core que utilice el valor literal en lugar del significado especial del carácter. Por ejemplo, el `*` carácter es un comodín que coincide con cualquier combinación de caracteres.

Carácter literal	Secuencia de escape	Notas
`*`	`${*}`	
`?`	`${?}`	AWS IoT Greengrass actualmente no admite el `?` comodín, que coincide con cualquier carácter individual.
`$`	`${$}`	Usa esta secuencia de escape para hacer coincidir un recurso que contenga`${`. Por ejemplo, para que coincida con un recurso denominado `o${resourceName}` , debe especificarlo`${$}{resourceName}` . De lo contrario , para que coincida con un recurso que contiene`$`, puede usar un literal`$`, por ejemplo, para permitir el acceso a un tema que comience por`$aws`.

Ejemplos de políticas de autorización

Puede hacer referencia a los siguientes ejemplos de políticas de autorización para ayudarle a configurar las políticas de autorización para sus componentes.

Example Ejemplo de receta de componentes con una política de autorización

El siguiente ejemplo de receta de componentes incluye un `accessControl` objeto que define una política de autorización. Esta política autoriza al `com.example.HelloWorld` componente a publicar en el `test/topic` tema.

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 Ejemplo de actualización de la configuración de un componente con una política de autorización

El siguiente ejemplo de actualización de configuración en una implementación especifica la configuración de un componente con un `accessControl` objeto que define una política de autorización. Esta política autoriza al `com.example.HelloWorld` componente a publicar en el `test/topic` tema.

Console

Configuración para fusionar

```
{
  "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

El siguiente comando crea una implementación en un dispositivo principal.

```
aws greengrassv2 create-deployment --cli-input-json file://hello-world-
deployment.json
```

El `hello-world-deployment.json` archivo contiene el siguiente documento JSON.

```
{
  "targetArn": "arn:aws:iot:us-west-2:123456789012:thing/MyGreengrassCore",
  "deploymentName": "Deployment for MyGreengrassCore",
  "components": {
    "com.example.HelloWorld": {
      "componentVersion": "1.0.0",
      "configurationUpdate": {
        "merge": "{\"accessControl\":{\"aws.greengrass.ipc.pubsub\":
{\"com.example.HelloWorld:pubsub:1\":{\"policyDescription\":\"Allows access to
 publish to test/topic.\",\"operations\":[\"aws.greengrass#PublishToTopic\"],
\"resources\":[\"test/topic\"]}}}}"
      }
    }
  }
}
```

Greengrass CLI

El siguiente comando CLI de Greengrass crea una implementación local en un dispositivo principal.

```
sudo greengrass-cli deployment create \
  --recipeDir recipes \
  --artifactDir artifacts \
  --merge "com.example.HelloWorld=1.0.0" \
  --update-config hello-world-configuration.json
```

El `hello-world-configuration.json` archivo contiene el siguiente documento JSON.

```
{
  "com.example.HelloWorld": {
    "MERGE": {
      "accessControl": {
        "aws.greengrass.ipc.pubsub": {
          "com.example.HelloWorld:pubsub:1": {
            "policyDescription": "Allows access to publish to test/topic.",
            "operations": [
              "aws.greengrass#PublishToTopic"
            ],
            "resources": [
              "test/topic"
            ]
          }
        }
      }
    }
  }
}
```

Suscríbase a las transmisiones de eventos del IPC

Puede utilizar las operaciones de IPC para suscribirse a las transmisiones de eventos en un dispositivo principal de Greengrass. Para utilizar una operación de suscripción, defina un gestor de suscripciones y cree una solicitud al servicio de IPC. A continuación, el cliente IPC ejecuta las funciones del gestor de suscripciones cada vez que el dispositivo principal transmite un mensaje de evento a su componente.

Puede cerrar una suscripción para dejar de procesar los mensajes de eventos. Para ello, llame a `closeStream()` (Java), `close()` (Python) o `Close()` (C++) al objeto de operación de suscripción que utilizó para abrir la suscripción.

El servicio AWS IoT Greengrass Core IPC admite las siguientes operaciones de suscripción:

- SubscribeToTopic
- SubscribeToIoTCore
- SubscribeToComponentUpdates
- SubscribeToConfigurationUpdate
- SubscribeToValidateConfigurationUpdates

Temas

Defina los gestores de suscripciones

Para definir un gestor de suscripciones, defina las funciones de devolución de llamada que gestionen los mensajes de eventos, los errores y el cierre de transmisiones. Si utiliza el cliente IPC V1, debe definir estas funciones en una clase. Si usa el cliente IPC V2, que está disponible en versiones posteriores de los SDK de Java y Python, puede definir estas funciones sin crear una clase de controlador de suscripciones.

Java

Si utiliza el cliente IPC V1, debe implementar la interfaz genérica. `software.amazon.awssdk.eventstreamrpc.StreamResponseHandler<`*`StreamEventType`*`>` *`StreamEventType`*es el tipo de mensaje de evento para la operación de suscripción. Defina las siguientes funciones para gestionar los mensajes de eventos, los errores y el cierre de transmisiones.

Si usa el cliente IPC V2, puede definir estas funciones fuera de una clase de controlador de suscripciones o usar expresiones lambda.

```
void onStreamEvent(StreamEventType event)
```

La llamada de retorno a la que llama el cliente de IPC cuando recibe un mensaje de evento, como un mensaje MQTT o una notificación de actualización de un componente.

```
boolean onStreamError(Throwable error)
```

La llamada de retorno a la que llama el cliente de IPC cuando se produce un error de transmisión.

Devuelve True para cerrar la transmisión de suscripción como resultado del error, o devuelve false para mantener la transmisión abierta.

```
void onStreamClosed()
```

La llamada de retorno a la que llama el cliente de IPC cuando se cierra la transmisión.

Python

Si utiliza el cliente IPC V1, debe ampliar la clase de controlador de respuesta de la transmisión que corresponde a la operación de suscripción. SDK para dispositivos con AWS IoT Incluye una clase de controlador de suscripciones para cada operación de suscripción. *`StreamEventType`*es el tipo de mensaje de evento para la operación de suscripción. Defina las siguientes funciones para gestionar los mensajes de eventos, los errores y el cierre de transmisiones.

Si usa el cliente IPC V2, puede definir estas funciones fuera de una clase de controlador de suscripciones o usar expresiones lambda.

```
def on_stream_event(self, event: StreamEventType) -> None
```

La llamada de retorno a la que llama el cliente de IPC cuando recibe un mensaje de evento, como un mensaje MQTT o una notificación de actualización de un componente.

```
def on_stream_error(self, error: Exception) -> bool
```

La llamada de retorno a la que llama el cliente de IPC cuando se produce un error de transmisión.

Devuelve True para cerrar la transmisión de suscripción como resultado del error, o devuelve false para mantener la transmisión abierta.

```
def on_stream_closed(self) -> None
```

La llamada de retorno a la que llama el cliente de IPC cuando se cierra la transmisión.

C++

Implemente una clase que se derive de la clase del controlador de respuesta de la transmisión que corresponde a la operación de suscripción. SDK para dispositivos con AWS IoT Incluye una clase base de controlador de suscripciones para cada operación de suscripción. *StreamEventType*es el tipo de mensaje de evento para la operación de suscripción. Defina las siguientes funciones para gestionar los mensajes de eventos, los errores y el cierre de transmisiones.

```
void OnStreamEvent(StreamEventType *event)
```

La llamada de retorno a la que llama el cliente de IPC cuando recibe un mensaje de evento, como un mensaje MQTT o una notificación de actualización de un componente.

```
bool OnStreamError(OperationError *error)
```

La llamada de retorno a la que llama el cliente de IPC cuando se produce un error de transmisión.

Devuelve True para cerrar la transmisión de suscripción como resultado del error, o devuelve false para mantener la transmisión abierta.

```
void OnStreamClosed()
```

La llamada de retorno a la que llama el cliente de IPC cuando se cierra la transmisión.

JavaScript

Implemente una clase que se derive de la clase del controlador de respuesta de la transmisión que corresponde a la operación de suscripción. SDK para dispositivos con AWS IoT Incluye una clase base de controlador de suscripciones para cada operación de suscripción. *StreamEventType*es el tipo de mensaje de evento para la operación de suscripción. Defina las siguientes funciones para gestionar los mensajes de eventos, los errores y el cierre de transmisiones.

```
on(event: 'ended', listener: StreamingOperationEndedListener)
```

La llamada de retorno a la que llama el cliente de IPC cuando se cierra la transmisión.

```
on(event: 'streamError', listener: StreamingRpcErrorListener)
```

La devolución de llamada a la que llama el cliente de IPC cuando se produce un error de transmisión.

Devuelve True para cerrar la transmisión de suscripción como resultado del error, o devuelve false para mantener la transmisión abierta.

`on(event: 'message', listener: (message: InboundMessageType) => void)`

La llamada de retorno a la que llama el cliente de IPC cuando recibe un mensaje de evento, como un mensaje MQTT o una notificación de actualización de un componente.

Ejemplos de gestores de suscripciones

En el siguiente ejemplo, se muestra cómo utilizar la SubscribeToTopic operación y un controlador de suscripciones para suscribirse a los mensajes locales de publicación/suscripción.

Java (IPC client V2)

Example Ejemplo: suscríbase a mensajes de publicación/suscripción locales

```
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 Ejemplo: suscríbase a mensajes de publicación/suscripción locales

```
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 Ejemplo: suscríbase a mensajes de publicación/suscripción locales

```
#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 Ejemplo: suscríbase a mensajes de publicación/suscripción locales

```
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();
```

Mejores prácticas de IPC

Las mejores prácticas para usar el IPC en los componentes personalizados difieren entre el cliente de IPC V1 y el cliente de IPC V2. Siga las prácticas recomendadas para la versión de cliente de IPC que utilice.

IPC client V2

El cliente de IPC V2 ejecuta las funciones de devolución de llamadas en un hilo independiente, por lo que, en comparación con el cliente de IPC V1, hay menos pautas que seguir cuando se utilizan las funciones de IPC y se escriben las funciones de gestión de suscripciones.

- Reutilice un cliente IPC

  Después de crear un cliente de IPC, manténgalo abierto y reutilícelo para todas las operaciones de IPC. La creación de varios clientes consume recursos adicionales y puede provocar pérdidas de recursos.

- Maneje las excepciones

  El cliente IPC V2 registra las excepciones no detectadas en las funciones del gestor de suscripciones. Debe detectar las excepciones en las funciones de su controlador para gestionar los errores que se producen en su código.

IPC client V1

El cliente IPC V1 utiliza un único hilo que se comunica con el servidor IPC y llama a los gestores de suscripciones. Debe tener en cuenta este comportamiento sincrónico al escribir las funciones del controlador de suscripciones.

- Reutilice un cliente de IPC

  Después de crear un cliente de IPC, manténgalo abierto y reutilícelo para todas las operaciones de IPC. La creación de varios clientes consume recursos adicionales y puede provocar pérdidas de recursos.
- Ejecute el código de bloqueo de forma asíncrona

  El cliente IPC V1 no puede enviar nuevas solicitudes ni procesar nuevos mensajes de eventos mientras el hilo está bloqueado. Debe ejecutar el código de bloqueo en un hilo independiente que ejecute desde la función de controlador. El código de bloqueo incluye `sleep` llamadas, bucles que se ejecutan de forma continua y solicitudes de E/S sincrónicas que tardan en completarse.
- Envíe nuevas solicitudes de IPC de forma asíncrona

  El cliente de IPC V1 no puede enviar una nueva solicitud desde las funciones del controlador de suscripciones, ya que la solicitud bloquea la función del controlador si se espera una respuesta. Debe enviar las solicitudes de IPC en un hilo independiente que ejecute desde la función de controlador.
- Gestiona las excepciones

  El cliente IPC V1 no gestiona las excepciones no detectadas en las funciones del gestor de suscripciones. Si tu función de controlador genera una excepción, la suscripción se cierra y la excepción no aparece en los registros de tus componentes. Deberías atrapar las excepciones en las funciones de tu controlador para mantener la suscripción abierta y registrar los errores que se produzcan en tu código.

Publicar/suscribir mensajes locales

La mensajería de publicación/suscripción (pubsub) le permite enviar y recibir mensajes sobre los temas. Los componentes pueden publicar mensajes en los temas para enviar mensajes a otros componentes. A continuación, los componentes que están suscritos a ese tema pueden actuar en función de los mensajes que reciben.

> **Note**
>
> No puede utilizar este servicio de publicación o suscripción de IPC para publicar o suscribirse a MQTT. AWS IoT Core Para obtener más información sobre cómo intercambiar mensajes con AWS IoT Core MQTT, consulte. Publicar/suscribir mensajes MQTT AWS IoT Core

Temas

Versiones mínimas del SDK

En la siguiente tabla se enumeran las versiones mínimas del SDK para dispositivos con AWS IoT que debes usar para publicar y suscribirte a los mensajes de temas locales y desde ellos.

SDK	Versión mínima	
SDK para dispositivos con AWS IoT para Java v2	v1.2.10	
SDK para dispositivos con AWS IoT para Python v2	v1.5.3	
SDK para dispositivos con AWS IoT para C++ v2	v1.17.0	
SDK de AWS IoT Device para JavaScript v2	v1.12.0	

Autorización

Para utilizar la mensajería local de publicación/suscripción en un componente personalizado, debe definir políticas de autorización que permitan a su componente enviar y recibir mensajes sobre los temas. Para obtener información sobre cómo definir las políticas de autorización, consulte. Autorice a los componentes a realizar operaciones de IPC

Las políticas de autorización para publicar o suscribir mensajes tienen las siguientes propiedades.

Identificador del servicio IPC: `aws.greengrass.ipc.pubsub`

Operación	Descripción	Recursos
`aws.greengrass#PublishToTopic`	Permite que un componente publique mensajes en los temas que especifique.	Una cadena de tema, como `test/topic` . Use an * para hacer coincidir cualquier combinación de caracteres de un tema. Esta cadena de tema no admite los caracteres comodín (#y+) de los temas MQTT.
`aws.greengrass#SubscribeToTopic`	Permite que un componente se suscriba a los mensajes de los temas que especifique.	Una cadena de tema, como `test/topic` . Use an * para hacer coincidir cualquier combinación de caracteres de un tema. En Greengrass nucleus v2.6.0 y versiones posteriores, puede suscribirse a temas que contengan comodines (y) de temas MQTT. # + Esta cadena de tema admite los comodines de los temas MQTT como caracteres literales. Por ejemplo, si la política de

Operación	Descripción	Recursos
		autorización de un component e permite el acceso `atest/topic/#` , el component e se puede suscribir`test/topic/#` , pero no se puede suscribir a él. `test/topic/filter`
*	Permite que un componente publique y se suscriba a los mensajes de los temas que especifique.	Una cadena de tema, como`test/topic` . Use an * para hacer coincidir cualquier combinación de caracteres de un tema. En Greengrass nucleus v2.6.0 y versiones posteriores, puede suscribirse a temas que contengan comodines (y) de temas MQTT. # + Esta cadena de tema admite los comodines de los temas MQTT como caracteres literales. Por ejemplo, si la política de autorización de un component e permite el acceso `atest/topic/#` , el component e se puede suscribir`test/topic/#` , pero no se puede suscribir a él. `test/topic/filter`

Ejemplos de políticas de autorización

Puede hacer referencia al siguiente ejemplo de política de autorización para ayudarle a configurar las políticas de autorización para sus componentes.

Example Ejemplo de política de autorización

El siguiente ejemplo de política de autorización permite que un componente publique y se suscriba a todos los temas.

```
{
  "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

Publique un mensaje en un tema.

Solicitud

La solicitud de esta operación tiene los siguientes parámetros:

`topic`

El tema en el que se va a publicar el mensaje.

`publishMessage`(Python:`publish_message`)

El mensaje que se va a publicar. Este objeto,`PublishMessage`, contiene la siguiente información. Debe especificar uno de los siguientes `jsonMessage` valores`binaryMessage`:

`jsonMessage`(Python:`json_message`)

(Opcional) Un mensaje JSON. Este objeto contiene la siguiente información: `JsonMessage`

`message`

El mensaje JSON como objeto.

`context`

El contexto del mensaje, como el tema en el que se publicó el mensaje.

Esta función está disponible para la versión 2.6.0 y versiones posteriores del componente núcleo de Greengrass. En la siguiente tabla se enumeran las versiones mínimas de las SDK para dispositivos con AWS IoT que debe utilizar para acceder al contexto del mensaje.

SDK	Versión mínima	
SDK para dispositivos con AWS IoT para Java v2	v1.9.3	
SDK para dispositivos con AWS IoT para Python v2	v1.11.3	
SDK para dispositivos con AWS IoT para C++ v2	v1.18.4	
SDK de AWS IoT Device para JavaScript v2	v1.12.0	

> **Note**
>
> El software AWS IoT Greengrass Core utiliza los mismos objetos de mensaje en las `PublishToTopic` operaciones y. `SubscribeToTopic` El software AWS IoT Greengrass Core establece este objeto de contexto en los mensajes cuando te suscribes e ignora este objeto de contexto en los mensajes que publicas.

Este objeto contiene `MessageContext` la siguiente información:

`topic`

El tema en el que se publicó el mensaje.

`binaryMessage`(Python:`binary_message`)

(Opcional) Un mensaje binario. Este objeto,`BinaryMessage`, contiene la siguiente información:

`message`

El mensaje binario en forma de blob.

`context`

El contexto del mensaje, como el tema en el que se publicó el mensaje.

Esta función está disponible para la versión 2.6.0 y versiones posteriores del componente núcleo de Greengrass. En la siguiente tabla se enumeran las versiones mínimas de las SDK para dispositivos con AWS IoT que debe utilizar para acceder al contexto del mensaje.

SDK	Versión mínima
SDK para dispositivos con AWS IoT para Java v2	v1.9.3
SDK para dispositivos con AWS IoT para Python v2	v1.11.3
SDK para dispositivos con AWS IoT para C++ v2	v1.18.4
SDK de AWS IoT Device para JavaScript v2	v1.12.0

> **Note**
>
> El software AWS IoT Greengrass Core utiliza los mismos objetos de mensaje en las `PublishToTopic` operaciones y. `SubscribeToTopic` El software AWS IoT Greengrass Core establece este objeto de contexto en los mensajes cuando te suscribes e ignora este objeto de contexto en los mensajes que publicas.

Este objeto contiene `MessageContext` la siguiente información:

`topic`

El tema en el que se publicó el mensaje.

Respuesta

Esta operación no proporciona ninguna información en su respuesta.

Ejemplos

Los siguientes ejemplos muestran cómo llamar a esta operación en el código de un componente personalizado.

Java (IPC client V2)

Example Ejemplo: publicar un mensaje binario

```
package com.aws.greengrass.docs.samples.ipc;

import software.amazon.awssdk.aws.greengrass.GreengrassCoreIPCClientV2;
import software.amazon.awssdk.aws.greengrass.model.BinaryMessage;
import software.amazon.awssdk.aws.greengrass.model.PublishMessage;
import software.amazon.awssdk.aws.greengrass.model.PublishToTopicRequest;
import software.amazon.awssdk.aws.greengrass.model.PublishToTopicResponse;
import software.amazon.awssdk.aws.greengrass.model.UnauthorizedError;

import java.nio.charset.StandardCharsets;

public class PublishToTopicV2 {

    public static void main(String[] args) {
        String topic = args[0];
        String message = args[1];
        try (GreengrassCoreIPCClientV2 ipcClient =
 GreengrassCoreIPCClientV2.builder().build()) {
            PublishToTopicV2.publishBinaryMessageToTopic(ipcClient, topic,
 message);
            System.out.println("Successfully published to topic: " + topic);
        } catch (Exception e) {
            if (e.getCause() instanceof UnauthorizedError) {
                System.err.println("Unauthorized error while publishing to topic: "
 + topic);
```

```
            } else {
                System.err.println("Exception occurred when using IPC.");
            }
            e.printStackTrace();
            System.exit(1);
        }
    }

    public static PublishToTopicResponse publishBinaryMessageToTopic(
            GreengrassCoreIPCClientV2 ipcClient, String topic, String message)
 throws InterruptedException {
        BinaryMessage binaryMessage =
                new
 BinaryMessage().withMessage(message.getBytes(StandardCharsets.UTF_8));
        PublishMessage publishMessage = new
 PublishMessage().withBinaryMessage(binaryMessage);
        PublishToTopicRequest publishToTopicRequest =
                new
 PublishToTopicRequest().withTopic(topic).withPublishMessage(publishMessage);
        return ipcClient.publishToTopic(publishToTopicRequest);
    }
}
```

Python (IPC client V2)

Example Ejemplo: publicar un mensaje binario

```
import sys
import traceback

from awsiot.greengrasscoreipc.clientv2 import GreengrassCoreIPCClientV2
from awsiot.greengrasscoreipc.model import (
    PublishMessage,
    BinaryMessage
)


def main():
    args = sys.argv[1:]
    topic = args[0]
    message = args[1]

    try:
        ipc_client = GreengrassCoreIPCClientV2()
```

```
        publish_binary_message_to_topic(ipc_client, topic, message)
        print('Successfully published to topic: ' + topic)
    except Exception:
        print('Exception occurred', file=sys.stderr)
        traceback.print_exc()
        exit(1)


def publish_binary_message_to_topic(ipc_client, topic, message):
    binary_message = BinaryMessage(message=bytes(message, 'utf-8'))
    publish_message = PublishMessage(binary_message=binary_message)
    return ipc_client.publish_to_topic(topic=topic,
 publish_message=publish_message)


if __name__ == '__main__':
    main()
```

C++

Example Ejemplo: publicar un mensaje binario

```
#include <iostream>

#include <aws/crt/Api.h>
#include <aws/greengrass/GreengrassCoreIpcClient.h>

using namespace Aws::Crt;
using namespace Aws::Greengrass;

class IpcClientLifecycleHandler : public ConnectionLifecycleHandler {
    void OnConnectCallback() override {
        // Handle connection to IPC service.
    }

    void OnDisconnectCallback(RpcError error) override {
        // Handle disconnection from IPC service.
    }

    bool OnErrorCallback(RpcError error) override {
        // Handle IPC service connection error.
        return true;
    }
};
```

```
int main() {
    ApiHandle apiHandle(g_allocator);
    Io::EventLoopGroup eventLoopGroup(1);
    Io::DefaultHostResolver socketResolver(eventLoopGroup, 64, 30);
    Io::ClientBootstrap bootstrap(eventLoopGroup, socketResolver);
    IpcClientLifecycleHandler ipcLifecycleHandler;
    GreengrassCoreIpcClient ipcClient(bootstrap);
    auto connectionStatus = ipcClient.Connect(ipcLifecycleHandler).get();
    if (!connectionStatus) {
        std::cerr << "Failed to establish IPC connection: " <<
 connectionStatus.StatusToString() << std::endl;
        exit(-1);
    }

    String topic("my/topic");
    String message("Hello, World!");
    int timeout = 10;

    PublishToTopicRequest request;
    Vector<uint8_t> messageData({message.begin(), message.end()});
    BinaryMessage binaryMessage;
    binaryMessage.SetMessage(messageData);
    PublishMessage publishMessage;
    publishMessage.SetBinaryMessage(binaryMessage);
    request.SetTopic(topic);
    request.SetPublishMessage(publishMessage);

    auto operation = ipcClient.NewPublishToTopic();
    auto activate = operation->Activate(request, nullptr);
    activate.wait();

    auto responseFuture = operation->GetResult();
    if (responseFuture.wait_for(std::chrono::seconds(timeout)) ==
 std::future_status::timeout) {
        std::cerr << "Operation timed out while waiting for response from Greengrass
 Core." << std::endl;
        exit(-1);
    }

    auto response = responseFuture.get();
    if (!response) {
        // Handle error.
        auto errorType = response.GetResultType();
```

```
        if (errorType == OPERATION_ERROR) {
            auto *error = response.GetOperationError();
            (void)error;
            // Handle operation error.
        } else {
            // Handle RPC error.
        }
    }
    return 0;
}
```

JavaScript

Example Ejemplo: publicar un mensaje binario

```
import * as greengrasscoreipc from "aws-iot-device-sdk-v2/dist/greengrasscoreipc";
import {BinaryMessage, PublishMessage, PublishToTopicRequest} from "aws-iot-device-
sdk-v2/dist/greengrasscoreipc/model";

class PublishToTopic {
    private ipcClient : greengrasscoreipc.Client
    private readonly topic : string;
    private readonly messageString : string;

    constructor() {
        // define your own constructor, e.g.
        this.topic = "<define_your_topic>";
        this.messageString = "<define_your_message_string>";
        this.publishToTopic().then(r => console.log("Started workflow"));
    }

    private async publishToTopic() {
        try {
            this.ipcClient = await getIpcClient();

            const binaryMessage : BinaryMessage = {
                message: this.messageString
            }

            const publishMessage : PublishMessage = {
                binaryMessage: binaryMessage
            }
```

```
            const request : PublishToTopicRequest = {
                topic: this.topic,
                publishMessage: publishMessage
            }

            this.ipcClient.publishToTopic(request).finally(() =>
 console.log(`Published message ${publishMessage.binaryMessage?.message} to topic`))

        } catch (e) {
            // parse the error depending on your use cases
            throw e
        }
    }
}


export async function getIpcClient(){
    try {
        const ipcClient = greengrasscoreipc.createClient();
        await ipcClient.connect()
            .catch(error => {
                // parse the error depending on your use cases
                throw error;
            });
        return ipcClient
    } catch (err) {
        // parse the error depending on your use cases
        throw err
    }
}

// starting point
const publishToTopic = new PublishToTopic();
```

SubscribeToTopic

Suscríbase a los mensajes sobre un tema.

Esta operación es una operación de suscripción en la que te suscribes a un flujo de mensajes de eventos. Para usar esta operación, defina un controlador de respuesta de transmisión con funciones que gestionen los mensajes de eventos, los errores y el cierre de la transmisión. Para obtener más información, consulte Suscríbase a las transmisiones de eventos del IPC.

Tipo de mensaje de evento: `SubscriptionResponseMessage`

Solicitud

La solicitud de esta operación tiene los siguientes parámetros:

`topic`

El tema al que se va a suscribir.

> **Note**
>
> En Greengrass nucleus v2.6.0 y versiones posteriores, este tema admite los comodines (y) de los temas MQTT. # +

`receiveMode`(Python:`receive_mode`)

(Opcional) El comportamiento que especifica si el componente recibe mensajes de sí mismo. Puede cambiar este comportamiento para permitir que un componente actúe sobre sus propios mensajes. El comportamiento predeterminado depende de si el tema contiene un comodín MQTT. Puede elegir entre las siguientes opciones:

- `RECEIVE_ALL_MESSAGES`— Reciba todos los mensajes que coincidan con el tema, incluidos los mensajes del componente al que se suscribe.

  Este modo es la opción por defecto cuando te suscribes a un tema que no contiene un comodín de MQTT.

- `RECEIVE_MESSAGES_FROM_OTHERS`— Reciba todos los mensajes que coincidan con el tema, excepto los del componente al que se suscribe.

  Este modo es la opción por defecto cuando se suscribe a un tema que contiene un comodín MQTT.

Esta función está disponible para la versión 2.6.0 y versiones posteriores del componente núcleo de Greengrass. En la siguiente tabla se enumeran las versiones mínimas del SDK para dispositivos con AWS IoT que debe utilizar para configurar el modo de recepción.

SDK	Versión mínima	
SDK para dispositivos con AWS IoT para Java v2	v1.9.3	
SDK para dispositivos con AWS IoT para Python v2	v1.11.3	
SDK para dispositivos con AWS IoT para C++ v2	v1.18.4	
SDK para dispositivos con AWS IoT para v2 JavaScript	v1.12.0	

Respuesta

La respuesta de esta operación tiene la siguiente información:

`messages`

El flujo de mensajes. Este objeto,`SubscriptionResponseMessage`, contiene la siguiente información. Cada mensaje contiene `jsonMessage` o`binaryMessage`.

`jsonMessage`(Python:`json_message`)

(Opcional) Un mensaje JSON. Este objeto contiene la siguiente información: `JsonMessage`

`message`

El mensaje JSON como objeto.

`context`

El contexto del mensaje, como el tema en el que se publicó el mensaje.

Esta función está disponible para la versión 2.6.0 y versiones posteriores del componente núcleo de Greengrass. En la siguiente tabla se enumeran las versiones mínimas de las SDK para dispositivos con AWS IoT que debe utilizar para acceder al contexto del mensaje.

SDK	Versión mínima	
SDK para dispositivos con AWS IoT para Java v2	v1.9.3	
SDK para dispositivos con AWS IoT para Python v2	v1.11.3	
SDK para dispositivos con AWS IoT para C++ v2	v1.18.4	
SDK de AWS IoT Device para JavaScript v2	v1.12.0	

> **Note**
>
> El software AWS IoT Greengrass Core utiliza los mismos objetos de mensaje en las `PublishToTopic` operaciones y. `SubscribeToTopic` El software AWS IoT Greengrass Core establece este objeto de contexto en los mensajes cuando te suscribes e ignora este objeto de contexto en los mensajes que publicas.

Este objeto contiene `MessageContext` la siguiente información:

`topic`

El tema en el que se publicó el mensaje.

`binaryMessage`(Python:`binary_message`)

(Opcional) Un mensaje binario. Este objeto,`BinaryMessage`, contiene la siguiente información:

`message`

El mensaje binario en forma de blob.

`context`

El contexto del mensaje, como el tema en el que se publicó el mensaje.

Esta función está disponible para la versión 2.6.0 y versiones posteriores del componente núcleo de Greengrass. En la siguiente tabla se enumeran las versiones mínimas de las SDK para dispositivos con AWS IoT que debe utilizar para acceder al contexto del mensaje.

SDK	Versión mínima
SDK para dispositivos con AWS IoT para Java v2	v1.9.3
SDK para dispositivos con AWS IoT para Python v2	v1.11.3
SDK para dispositivos con AWS IoT para C++ v2	v1.18.4
SDK de AWS IoT Device para JavaScript v2	v1.12.0

> **Note**
>
> El software AWS IoT Greengrass Core utiliza los mismos objetos de mensaje en las `PublishToTopic` operaciones y. `SubscribeToTopic` El software AWS IoT Greengrass Core establece este objeto de contexto en los mensajes cuando te suscribes e ignora este objeto de contexto en los mensajes que publicas.

Este objeto contiene `MessageContext` la siguiente información:

`topic`

El tema en el que se publicó el mensaje.

`topicName`(Python:`topic_name`)

El tema en el que se publicó el mensaje.

> **Note**
>
> Esta propiedad no se utiliza actualmente. En Greengrass nucleus v2.6.0 y versiones posteriores, puede obtener el `(jsonMessage|binaryMessage).context.topic` valor de `SubscriptionResponseMessage` a para obtener el tema en el que se publicó el mensaje.

Ejemplos

Los siguientes ejemplos muestran cómo llamar a esta operación en el código de un componente personalizado.

Java (IPC client V2)

Example Ejemplo: suscríbase a mensajes locales de publicación/suscripción

```
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 Ejemplo: suscríbase a mensajes de publicación/suscripción locales

```
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 Ejemplo: suscríbase a mensajes de publicación/suscripción locales

```
#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 Ejemplo: suscríbase a mensajes de publicación/suscripción locales

```
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();
```

Ejemplos

Utilice los siguientes ejemplos para aprender a utilizar el servicio de publicación/suscripción de IPC en sus componentes.

Ejemplo de publicación/suscripción de un editor (Java, cliente IPC V1)

La siguiente receta de ejemplo permite que el componente publique en todos los temas.

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
```

El siguiente ejemplo de aplicación Java demuestra cómo utilizar el servicio IPC de publicación/suscripción para publicar mensajes en otros componentes.

```
/* 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);
        }
    }
}
```

Ejemplo de suscriptor de publicación/suscripción (Java, cliente IPC V1)

La siguiente receta de ejemplo permite que el componente se suscriba a todos los temas.

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
```

El siguiente ejemplo de aplicación Java muestra cómo utilizar el servicio IPC de publicación/suscripción para suscribirse a los mensajes de otros componentes.

```
/* 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.");
        }
    }
}
```

Ejemplo de publicación/suscripción de publicador/suscriptor (Python, cliente IPC V1)

La siguiente receta de ejemplo permite que el componente publique en todos los temas.

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
```

El siguiente ejemplo de aplicación de Python demuestra cómo utilizar el servicio IPC de publicación/suscripción para publicar mensajes en otros componentes.

```
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)
```

Ejemplo de suscriptor de publicación/suscripción (Python, cliente IPC V1)

La siguiente receta de ejemplo permite que el componente se suscriba a todos los temas.

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
```

El siguiente ejemplo de aplicación de Python demuestra cómo utilizar el servicio IPC de publicación/suscripción para suscribirse a los mensajes de otros componentes.

```
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)
```

Ejemplo de publicación/suscripción a un editor (C++)

La siguiente receta de ejemplo permite que el componente publique en todos los temas.

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
```

El siguiente ejemplo de aplicación de C++ demuestra cómo utilizar el servicio IPC de publicación/suscripción para publicar mensajes en otros componentes.

```
#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;
}
```

Ejemplo de suscriptor de publicación/suscripción (C++)

La siguiente receta de ejemplo permite que el componente se suscriba a todos los temas.

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
```

El siguiente ejemplo de aplicación de C++ demuestra cómo utilizar el servicio IPC de publicación/suscripción para suscribirse a los mensajes de otros componentes.

```
#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;
}
```

Publicar/suscribir mensajes MQTT AWS IoT Core

El servicio IPC de mensajería AWS IoT Core MQTT le permite enviar y recibir mensajes MQTT de ida y vuelta. AWS IoT Core Los componentes pueden publicar mensajes en los temas AWS IoT Core y suscribirse a ellos para actuar en función de los mensajes MQTT de otras fuentes. Para obtener más información sobre la AWS IoT Core implementación de MQTT, consulte MQTT en la AWS IoT Core Guía para desarrolladores.

> **Note**
>
> Este servicio IPC de mensajería MQTT le permite intercambiar mensajes con. AWS IoT Core Para obtener más información sobre cómo intercambiar mensajes entre componentes, consulte. Publicar/suscribir mensajes locales

Temas

- Versiones mínimas del SDK
- Autorización
- PublishToIoTCore

- SubscribeToIoTCore
- Ejemplos

Versiones mínimas del SDK

En la siguiente tabla se enumeran las versiones mínimas de las SDK para dispositivos con AWS IoT que debe utilizar para publicar y recibir mensajes de MQTT y suscribirse a AWS IoT Core ellos.

SDK	Versión mínima	
SDK para dispositivos con AWS IoT para Java v2	v1.2.10	
SDK para dispositivos con AWS IoT para Python v2	v1.5.3	
SDK para dispositivos con AWS IoT para C++ v2	v1.17.0	
SDK para dispositivos con AWS IoT para v2 JavaScript	v1.12.0	

Autorización

Para utilizar la mensajería AWS IoT Core MQTT en un componente personalizado, debe definir políticas de autorización que permitan a su componente enviar y recibir mensajes sobre temas. Para obtener información sobre la definición de políticas de autorización, consulteAutorice a los componentes a realizar operaciones de IPC.

Las políticas de autorización para la mensajería AWS IoT Core MQTT tienen las siguientes propiedades.

Identificador de servicio IPC: `aws.greengrass.ipc.mqttproxy`

Operación	Descripción	Recursos
`aws.greengrass#PublishToIoTCore`	Permite que un componente publique mensajes AWS IoT	Una cadena de temas, por ejemplo`test/topic` , o

Operación	Descripción	Recursos
	Core en los temas de MQTT que especifique.	* para permitir el acceso a todos los temas. Puede utilizar los caracteres comodín (#y+) de los temas de MQTT para hacer coincidir varios recursos.
`aws.greengrass#SubscribeToIoTCore`	Permite que un componente se suscriba a los mensajes AWS IoT Core de los temas que especifique.	Una cadena de temas, por ejemplo`test/topic` , o * para permitir el acceso a todos los temas. Puede utilizar los caracteres comodín (#y+) de los temas de MQTT para hacer coincidir varios recursos.
*	Permite que un componente publique y se suscriba a los mensajes de AWS IoT Core MQTT para los temas que especifique.	Una cadena de temas, por ejemplo`test/topic` , o * para permitir el acceso a todos los temas. Puede utilizar los caracteres comodín (#y+) de los temas de MQTT para hacer coincidir varios recursos.

Los caracteres comodín de MQTT en las políticas de autorización de MQTT AWS IoT Core

Puede utilizar caracteres comodín de MQTT en AWS IoT Core las políticas de autorización de IPC de MQTT. Los componentes pueden publicar y suscribirse a temas que coincidan con el filtro de temas que usted permita en una política de autorización. Por ejemplo, si la política de autorización de un componente concede acceso a`test/topic/#`, el componente puede suscribirse`test/topic/#`, publicar y suscribirse`test/topic/filter`.

Variables de receta en las políticas de autorización de AWS IoT Core MQTT

Si usa la versión 2.6.0 o posterior del núcleo de Greengrass, puede usar la variable de `{iot:thingName}` receta en las políticas de autorización. Esta función le permite configurar una política de autorización única para un grupo de dispositivos principales, de forma que cada dispositivo principal solo pueda acceder a los temas que contengan su propio nombre. Por ejemplo, puede permitir que un componente acceda al siguiente recurso temático.

```
devices/{iot:thingName}/messages
```

Para obtener más información, consulte Variables de receta y Utilice variables de receta en las actualizaciones de fusión.

Ejemplos de políticas de autorización

Puede hacer referencia a los siguientes ejemplos de políticas de autorización para ayudarle a configurar las políticas de autorización para sus componentes.

Example Ejemplo de política de autorización con acceso ilimitado

El siguiente ejemplo de política de autorización permite que un componente publique y se suscriba a todos los temas.

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 Ejemplo de política de autorización con acceso limitado

El siguiente ejemplo de política de autorización permite a un componente publicar y suscribirse a dos temas denominados `factory/1/events` y`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 Ejemplo de política de autorización para un grupo de dispositivos principales

> **Important**
>
> En este ejemplo se utiliza una función que está disponible para la versión 2.6.0 y versiones posteriores del componente núcleo de Greengrass. Greengrass nucleus v2.6.0 añade soporte para la mayoría de las variables de receta, por ejemplo, en las configuraciones de componentes`{iot:thingName}`.

El siguiente ejemplo de política de autorización permite a un componente publicar y suscribirse a un tema que contenga el nombre del dispositivo principal que ejecuta el componente.

JSON

```
{
  "accessControl": {
    "aws.greengrass.ipc.mqttproxy": {
      "com.example.MyIoTCorePubSubComponent:mqttproxy:1": {
        "policyDescription": "Allows access to publish/subscribe to all topics.",
        "operations": [
          "aws.greengrass#PublishToIoTCore",
          "aws.greengrass#SubscribeToIoTCore"
        ],
        "resources": [
          "factory/1/devices/{iot:thingName}/controls"
        ]
      }
    }
  }
```

```
}
```

YAML

```
---
accessControl:
  aws.greengrass.ipc.mqttproxy:
    "com.example.MyIoTCorePubSubComponent:mqttproxy:1":
      policyDescription: Allows access to publish/subscribe to all topics.
      operations:
        - aws.greengrass#PublishToIoTCore
        - aws.greengrass#SubscribeToIoTCore
      resources:
        - factory/1/devices/{iot:thingName}/controls
```

PublishToIoTCore

Publica un mensaje MQTT AWS IoT Core sobre un tema.

Al publicar mensajes MQTT en AWS IoT Core, hay una cuota de 100 transacciones por segundo. Si superas esta cuota, los mensajes se ponen en cola para su procesamiento en el dispositivo Greengrass. También hay una cuota de 512 KB de datos por segundo y una cuota de 20 000 publicaciones por segundo en toda la cuenta (2 000 en algunos casos). Regiones de AWS Para obtener más información sobre los límites del agente de mensajes MQTT AWS IoT Core, consulte los límites y cuotas del agente de AWS IoT Core mensajes y el protocolo.

Si superas estas cuotas, el dispositivo Greengrass limita la publicación de mensajes a. AWS IoT Core Los mensajes se almacenan en una bobina de memoria. De forma predeterminada, la memoria asignada al spooler es de 2,5 MB. Si la bobina se llena, se rechazan los mensajes nuevos. Puede aumentar el tamaño de la bobina. Para obtener más información, consulte Configuración en la documentación del Núcleo de Greengrass. Para evitar llenar el espacio y tener que aumentar la memoria asignada, limite las solicitudes de publicación a no más de 100 solicitudes por segundo.

Si su aplicación necesita enviar mensajes a una velocidad mayor o mensajes más grandes, considere utilizarlos Administrador de transmisiones para enviar mensajes a Kinesis Data Streams. El componente Stream Manager está diseñado para transferir grandes volúmenes de datos al. Nube de AWS Para obtener más información, consulte Gestione los flujos de datos en los dispositivos principales de Greengrass.

Solicitud

La solicitud de esta operación tiene los siguientes parámetros:

`topicName`(Python:`topic_name`)

El tema en el que se va a publicar el mensaje.

`qos`

La QoS de MQTT que se va a utilizar. Esta enumeración,`QOS`, tiene los siguientes valores:

- `AT_MOST_ONCE`— QoS 0. El mensaje MQTT se entrega como máximo una vez.
- `AT_LEAST_ONCE`— QoS 1. El mensaje MQTT se entrega al menos una vez.

`payload`

(Opcional) El mensaje se carga como un blob.

Las siguientes funciones están disponibles para la versión 2.10.0 y versiones posteriores cuando se utiliza MQTT 5. Núcleo de Greengrass Estas funciones se ignoran cuando se utiliza MQTT 3.1.1. En la siguiente tabla se muestra la versión mínima del SDK del AWS IoT dispositivo que debe utilizar para acceder a estas funciones.

SDK	Versión mínima
AWS IoT Device SDK para Python v2	v1.15.0
AWS IoT Device SDK para Java v2	v1.13.0
AWS IoT Device SDK para C++ v2	v1.24.0
SDK de AWS IoT Device para JavaScript v2	v1.13.0

`payloadFormat`

(Opcional) El formato de la carga útil del mensaje. Si no estableces el`payloadFormat`, se supone que el tipo es. `BYTES` La enumeración tiene los siguientes valores:

- `BYTES`— El contenido de la carga útil es un blob binario.
- `UTF8`— El contenido de la carga útil es una cadena de caracteres UTF8.

`retain`

(Opcional) Indica si se debe configurar la opción de retención de MQTT al publicar. `true`

`userProperties`

(Opcional) Una lista de `UserProperty` objetos específicos de la aplicación que se van a enviar. El `UserProperty` objeto se define de la siguiente manera:

```
UserProperty:
  key: string
  value: string
```

`messageExpiryIntervalSeconds`

(Opcional) El número de segundos que faltan para que el mensaje caduque y el servidor lo elimine. Si no se establece este valor, el mensaje no caduca.

`correlationData`

(Opcional) Información agregada a la solicitud que se puede usar para asociar una solicitud a una respuesta.

`responseTopic`

(Opcional) El tema que debe usarse para el mensaje de respuesta.

`contentType`

(Opcional) Un identificador específico de la aplicación del tipo de contenido del mensaje.

Respuesta

Esta operación no proporciona ninguna información en su respuesta.

Ejemplos

Los siguientes ejemplos muestran cómo llamar a esta operación en el código de un componente personalizado.

Java (IPC client V2)

Example Ejemplo: publicar un mensaje

```
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 Ejemplo: publicar un mensaje

> **Note**
>
> En este ejemplo se supone que está utilizando la versión 1.5.4 o posterior de SDK para dispositivos con AWS IoT para 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 Ejemplo: publicar un mensaje

> **Note**
>
> En este ejemplo, se utiliza una `IPCUtils` clase para crear una conexión con el servicio AWS IoT Greengrass Core IPC. Para obtener más información, consulte Conéctese al AWS IoT Greengrass servicio 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 Ejemplo: publicar un mensaje

> **Note**
>
> En este ejemplo se supone que está utilizando la versión 1.5.4 o posterior de SDK para dispositivos con AWS IoT para 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 Ejemplo: publicar un mensaje

```
#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 Ejemplo: publicar un mensaje

```
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

Suscríbase a los mensajes de MQTT AWS IoT Core de un tema o filtro de temas. El software AWS IoT Greengrass Core elimina las suscripciones cuando el componente llega al final de su ciclo de vida.

Esta operación es una operación de suscripción en la que se suscribe a un flujo de mensajes de eventos. Para usar esta operación, defina un controlador de respuesta de transmisión con funciones que gestionen los mensajes de eventos, los errores y el cierre de la transmisión. Para obtener más información, consulte Suscríbase a las transmisiones de eventos del IPC.

Tipo de mensaje de evento: `IoTCoreMessage`

Solicitud

La solicitud de esta operación tiene los siguientes parámetros:

`topicName`(Python:`topic_name`)

El tema al que se va a suscribir. Puede utilizar los comodines de los temas de MQTT (#y+) para suscribirse a varios temas.

`qos`

La QoS de MQTT que se va a utilizar. Esta enumeración,`QOS`, tiene los siguientes valores:

- `AT_MOST_ONCE`— QoS 0. El mensaje MQTT se entrega como máximo una vez.
- `AT_LEAST_ONCE`— QoS 1. El mensaje MQTT se entrega al menos una vez.

Respuesta

La respuesta de esta operación contiene la siguiente información:

`messages`

El flujo de mensajes MQTT. Este objeto,`IoTCoreMessage`, contiene la siguiente información:

`message`

El mensaje MQTT. Este objeto,`MQTTMessage`, contiene la siguiente información:

`topicName`(Python:`topic_name`)

El tema en el que se publicó el mensaje.

`payload`

(Opcional) El mensaje se carga como un blob.

Las siguientes funciones están disponibles para la versión 2.10.0 y versiones posteriores cuando se utiliza MQTT 5. Núcleo de Greengrass Estas funciones se ignoran cuando se utiliza MQTT 3.1.1. En la siguiente tabla se muestra la versión mínima del SDK del AWS IoT dispositivo que debe utilizar para acceder a estas funciones.

SDK	Versión mínima
AWS IoT Device SDK para Python v2	v1.15.0
AWS IoT Device SDK para Java v2	v1.13.0
AWS IoT Device SDK para C++ v2	v1.24.0
SDK de AWS IoT Device para JavaScript v2	v1.13.0

`payloadFormat`

(Opcional) El formato de la carga útil del mensaje. Si no estableces el`payloadFormat`, se supone que el tipo es. `BYTES` La enumeración tiene los siguientes valores:

- `BYTES`— El contenido de la carga útil es un blob binario.
- `UTF8`— El contenido de la carga útil es una cadena de caracteres UTF8.

`retain`

(Opcional) Indica si se debe configurar la opción de retención de MQTT al publicar. `true`

`userProperties`

(Opcional) Una lista de `UserProperty` objetos específicos de la aplicación que se van a enviar. El `UserProperty` objeto se define de la siguiente manera:

```
UserProperty:
  key: string
  value: string
```

`messageExpiryIntervalSeconds`

(Opcional) El número de segundos que faltan para que el mensaje caduque y el servidor lo elimine. Si no se establece este valor, el mensaje no caduca.

`correlationData`

(Opcional) Información agregada a la solicitud que se puede usar para asociar una solicitud a una respuesta.

`responseTopic`

(Opcional) El tema que debe usarse para el mensaje de respuesta.

`contentType`

(Opcional) Un identificador específico de la aplicación del tipo de contenido del mensaje.

Ejemplos

Los siguientes ejemplos muestran cómo llamar a esta operación en un código de componente personalizado.

Java (IPC client V2)

Example Ejemplo: suscribirse a los mensajes

```
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 Ejemplo: suscribirse a los mensajes

> **Note**
>
> En este ejemplo se supone que está utilizando la versión 1.5.4 o posterior de SDK para dispositivos con AWS IoT para 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 Ejemplo: suscríbase a los mensajes

> **Note**
>
> En este ejemplo, se utiliza una `IPCUtils` clase para crear una conexión con el servicio AWS IoT Greengrass Core IPC. Para obtener más información, consulte Conéctese al AWS IoT Greengrass servicio 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 Ejemplo: suscribirse a los mensajes

> **Note**
>
> En este ejemplo se supone que está utilizando la versión 1.5.4 o posterior de SDK para dispositivos con AWS IoT para 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 Ejemplo: suscríbase a los mensajes

```
#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 Ejemplo: suscribirse a los mensajes

```
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();
```

Ejemplos

Utilice los siguientes ejemplos para aprender a utilizar el servicio AWS IoT Core MQTT IPC en sus componentes.

Ejemplo de editor AWS IoT Core MQTT (C++)

La siguiente receta de ejemplo permite que el componente publique en todos los temas.

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
```

El siguiente ejemplo de aplicación de C++ demuestra cómo utilizar el servicio IPC de AWS IoT Core MQTT para publicar mensajes en. 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;
}
```

Ejemplo de suscriptor de AWS IoT Core MQTT (C++)

La siguiente receta de ejemplo permite que el componente se suscriba a todos los temas.

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
```

El siguiente ejemplo de aplicación de C++ demuestra cómo utilizar el servicio IPC de AWS IoT Core MQTT para suscribirse a los mensajes de. 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;
}
```

Interactúe con el ciclo de vida del componente

Utilice el servicio IPC del ciclo de vida de los componentes para:

- Actualice el estado del componente en el dispositivo principal.
- Suscríbase a las actualizaciones del estado de los componentes.
- Evite que el núcleo detenga el componente para aplicar una actualización durante una implementación.
- Pausa y reanuda los procesos de los componentes.

Temas

- Versiones mínimas del SDK
- Autorización
- UpdateState
- SubscribeToComponentUpdates
- DeferComponentUpdate
- PauseComponent
- ResumeComponent

Versiones mínimas del SDK

En la siguiente tabla se enumeran las versiones mínimas del SDK para dispositivos con AWS IoT que debe utilizar para interactuar con el ciclo de vida de los componentes.

SDK	Versión mínima	
SDK para dispositivos con AWS IoTpara Java v2	v1.2.10	
SDK para dispositivos con AWS IoTpara Python v2	v1.5.3	
SDK para dispositivos con AWS IoTpara C++ v2	v1.17.0	
SDK para dispositivos con AWS IoTpara v2 JavaScript	v1.12.0	

Autorización

Para pausar o reanudar otros componentes de un componente personalizado, debe definir políticas de autorización que permitan a su componente administrar otros componentes. Para obtener información sobre cómo definir las políticas de autorización, consulteAutorice a los componentes a realizar operaciones de IPC.

Las políticas de autorización para la administración del ciclo de vida de los componentes tienen las siguientes propiedades.

Identificador de servicio IPC: `aws.greengrass.ipc.lifecycle`

Operación	Descripción	Recursos
`aws.greengrass#PauseComponent`	Permite que un componente detenga los componentes que especifique.	Un nombre de componente o * para permitir el acceso a todos los componentes.

Operación	Descripción	Recursos
`aws.greengrass#ResumeComponent`	Permite que un componente reanude los componentes que especifique.	Un nombre de componente o * para permitir el acceso a todos los componentes.
*	Permite que un component e detenga y reanude los componentes que especifique.	Un nombre de componente o * para permitir el acceso a todos los componentes.

Ejemplos de políticas de autorización

Puede hacer referencia al siguiente ejemplo de política de autorización para ayudarle a configurar las políticas de autorización para sus componentes.

Example Ejemplo de política de autorización

El siguiente ejemplo de política de autorización permite a un componente pausar y reanudar todos los componentes.

```
{
  "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

Actualice el estado del componente en el dispositivo principal.

Solicitud

La solicitud de esta operación tiene los siguientes parámetros:

`state`

El estado que se va a establecer. Esta enumeración,`LifecycleState`, tiene los siguientes valores:

- `RUNNING`
- `ERRORED`

Respuesta

Esta operación no proporciona ninguna información en su respuesta.

SubscribeToComponentUpdates

Suscríbase para recibir notificaciones antes de que el software AWS IoT Greengrass principal actualice un componente. La notificación especifica si el núcleo se reiniciará o no como parte de la actualización.

El núcleo envía notificaciones de actualización solo si la política de actualización de componentes de la implementación especifica que se notifique a los componentes. El comportamiento predeterminado es notificar a los componentes. Para obtener más información, consulte Crear implementaciones y el DeploymentComponentUpdatePolicyobjeto que puede proporcionar al llamar a la CreateDeploymentoperación.

> **Important**
>
> Las implementaciones locales no notifican a los componentes antes de las actualizaciones.

Esta operación es una operación de suscripción en la que se suscribe a un flujo de mensajes de eventos. Para usar esta operación, defina un controlador de respuesta de transmisión con funciones que gestionen los mensajes de eventos, los errores y el cierre de la transmisión. Para obtener más información, consulte Suscríbase a las transmisiones de eventos del IPC.

Tipo de mensaje de evento: `ComponentUpdatePolicyEvents`

> **Tip**
>
> Puede seguir un tutorial para aprender a desarrollar un componente que aplace condicionalmente las actualizaciones de los componentes. Para obtener más información, consulte Tutorial: Desarrolle un componente de Greengrass que aplace las actualizaciones de los componentes.

Solicitud

La solicitud de esta operación no tiene ningún parámetro.

Respuesta

La respuesta de esta operación contiene la siguiente información:

`messages`

El flujo de mensajes de notificación. Este objeto,`ComponentUpdatePolicyEvents`, contiene la siguiente información:

`preUpdateEvent`(Python:`pre_update_event`)

(Opcional) Un evento que indica que el núcleo quiere actualizar un componente. Puede responder con la DeferComponentUpdate operación para confirmar o aplazar la actualización hasta que el componente esté listo para reiniciarse. Este objeto contiene `PreComponentUpdateEvent` la siguiente información:

`deploymentId`(Python:`deployment_id`)

El ID de la AWS IoT Greengrass implementación que actualiza el componente.

`isGgcRestarting`(Python:`is_ggc_restarting`)

Si el núcleo necesita o no reiniciarse para aplicar la actualización.

`postUpdateEvent`(Python:`post_update_event`)

(Opcional) Un evento que indica que el núcleo actualizó un componente. Este objeto,`PostComponentUpdateEvent`, contiene la siguiente información:

`deploymentId`(Python:`deployment_id`)

El ID de la AWS IoT Greengrass implementación que actualizó el componente.

> **Note**
>
> Esta función requiere la versión 2.7.0 o posterior del componente núcleo de Greengrass.

DeferComponentUpdate

Confirme o aplace la actualización de un componente que descubra con. SubscribeToComponentUpdates Debe especificar el tiempo que debe transcurrir antes de que el núcleo vuelva a comprobar si el componente está preparado para continuar con la actualización del componente. También puede utilizar esta operación para indicar al núcleo que su componente está listo para la actualización.

Si un componente no responde a la notificación de actualización del componente, el núcleo espera el tiempo que especifique en la política de actualización de componentes de la implementación. Transcurrido ese tiempo de espera, el núcleo continúa con el despliegue. El tiempo de espera predeterminado para la actualización de los componentes es de 60 segundos. Para obtener más información, consulte Crear implementaciones y el DeploymentComponentUpdatePolicyobjeto que puede proporcionar al llamar a la CreateDeploymentoperación.

> **Tip**
>
> Puede seguir un tutorial para aprender a desarrollar un componente que aplace condicionalmente las actualizaciones de los componentes. Para obtener más información, consulte Tutorial: Desarrolle un componente de Greengrass que aplace las actualizaciones de los componentes.

Solicitud

La solicitud de esta operación tiene los siguientes parámetros:

`deploymentId`(Python:`deployment_id`)

El ID de la AWS IoT Greengrass implementación que se va a aplazar.

`message`

(Opcional) El nombre del componente cuyas actualizaciones se van a aplazar.

El valor predeterminado es el nombre del componente que realiza la solicitud.

`recheckAfterMs`(Python:`recheck_after_ms`)

El tiempo en milisegundos durante el que se debe aplazar la actualización. El núcleo espera esa cantidad de tiempo y luego envía otro con el `PreComponentUpdateEvent` que podrás descubrirlo. SubscribeToComponentUpdates

Especifique si `0` desea confirmar la actualización. Esto indica al núcleo que el componente está listo para la actualización.

El valor predeterminado es cero milisegundos, lo que significa confirmar la actualización.

Respuesta

Esta operación no proporciona ninguna información en su respuesta.

PauseComponent

Esta función está disponible para la versión 2.4.0 y versiones posteriores del componente núcleo de Greengrass. AWS IoT Greengrassactualmente no admite esta función en los dispositivos principales de Windows.

Detiene los procesos de un componente en el dispositivo principal. Para reanudar un componente, utilice la ResumeComponentoperación.

Solo puede pausar los componentes genéricos. Si intenta pausar cualquier otro tipo de componente, esta operación arroja un`InvalidRequestError`.

> **Note**
>
> Esta operación no puede pausar los procesos contenerizados, como los contenedores de Docker. Para pausar y reanudar un contenedor de Docker, puede usar los comandos docker pause y docker unpause.

Esta operación no detiene las dependencias de los componentes ni los componentes que dependen del componente que se pausa. Tenga en cuenta este comportamiento al pausar un componente que es una dependencia de otro componente, ya que el componente dependiente puede tener problemas cuando su dependencia está en pausa.

Al reiniciar o apagar un componente en pausa, por ejemplo, durante una implementación, el núcleo de Greengrass reanuda el componente y ejecuta su ciclo de vida de apagado. Para obtener más información sobre el reinicio de un componente, consulte. RestartComponent

> **Important**
>
> Para utilizar esta operación, debe definir una política de autorización que conceda permiso para utilizarla. Para obtener más información, consulte Autorización.

Versiones mínimas del SDK

En la siguiente tabla se enumeran las versiones mínimas SDK para dispositivos con AWS IoT que debes usar para pausar y reanudar los componentes.

SDK	Versión mínima	
SDK para dispositivos con AWS IoTpara Java v2	v1.4.3	
SDK para dispositivos con AWS IoTpara Python v2	v1.6.2	
SDK para dispositivos con AWS IoTpara C++ v2	v1.13.1	
SDK para dispositivos con AWS IoTpara JavaScript v2	v1.12.0	

Solicitud

La solicitud de esta operación tiene los siguientes parámetros:

`componentName`(Python:`component_name`)

El nombre del componente que se va a pausar, que debe ser un componente genérico. Para obtener más información, consulte Tipos de componentes.

Respuesta

Esta operación no proporciona ninguna información en su respuesta.

ResumeComponent

Esta función está disponible para la versión 2.4.0 y versiones posteriores del componente núcleo de Greengrass. AWS IoT Greengrassactualmente no admite esta función en los dispositivos principales de Windows.

Reanuda los procesos de un componente en el dispositivo principal. Para pausar un componente, utilice la PauseComponentoperación.

Solo puede reanudar los componentes pausados. Si intenta reanudar un componente que no está en pausa, esta operación arroja un. `InvalidRequestError`

> **Important**
>
> Para utilizar esta operación, debe definir una política de autorización que conceda permiso para hacerlo. Para obtener más información, consulte Autorización.

Versiones mínimas del SDK

En la siguiente tabla se enumeran las versiones mínimas SDK para dispositivos con AWS IoT que debes usar para pausar y reanudar los componentes.

SDK	Versión mínima	
SDK para dispositivos con AWS IoTpara Java v2	v1.4.3	
SDK para dispositivos con AWS IoTpara Python v2	v1.6.2	
SDK para dispositivos con AWS IoTpara C++ v2	v1.13.1	
SDK para dispositivos con AWS IoTpara JavaScript v2	v1.12.0	

Solicitud

La solicitud de esta operación tiene los siguientes parámetros:

`componentName`(Python:`component_name`)

El nombre del componente que se va a reanudar.

Respuesta

Esta operación no proporciona ninguna información en su respuesta.

Interactúa con la configuración de los componentes

El servicio IPC de configuración de componentes le permite hacer lo siguiente:

- Obtenga y establezca los parámetros de configuración de los componentes.
- Suscríbase a las actualizaciones de configuración de los componentes.
- Valide las actualizaciones de configuración de los componentes antes de que el núcleo las aplique.

Temas

Versiones mínimas del SDK

En la siguiente tabla se enumeran las versiones mínimas del SDK para dispositivos con AWS IoT que debe utilizar para interactuar con la configuración de los componentes.

SDK	Versión mínima	
SDK para dispositivos con AWS IoTpara Java v2	v1.2.10	
SDK para dispositivos con AWS IoTpara Python v2	v1.5.3	
SDK para dispositivos con AWS IoTpara C++ v2	v1.17.0	
SDK para dispositivos con AWS IoTpara v2 JavaScript	v1.12.0	

GetConfiguration

Obtiene un valor de configuración para un componente del dispositivo principal. Usted especifica la ruta clave para la que se va a obtener un valor de configuración.

Solicitud

La solicitud de esta operación tiene los siguientes parámetros:

`componentName`(Python:`component_name`)

(Opcional) El nombre del componente.

El valor predeterminado es el nombre del componente que realiza la solicitud.

`keyPath`(Python:`key_path`)

La ruta clave al valor de configuración. Especifique una lista en la que cada entrada sea la clave de un único nivel del objeto de configuración. Por ejemplo, especifique `["mqtt", "port"]` si desea obtener el valor de `port` en la siguiente configuración.

```
{
  "mqtt": {
    "port": 443
  }
```

```
}
```

Para obtener la configuración completa del componente, especifique una lista vacía.

Respuesta

La respuesta de esta operación contiene la siguiente información:

`componentName`(Python:`component_name`)

El nombre del componente.

`value`

La configuración solicitada como objeto.

UpdateConfiguration

Actualiza un valor de configuración para este componente en el dispositivo principal.

Solicitud

La solicitud de esta operación tiene los siguientes parámetros:

`keyPath`(Python:`key_path`)

(Opcional) La ruta clave al nodo contenedor (el objeto) que se va a actualizar. Especifique una lista en la que cada entrada sea la clave de un único nivel del objeto de configuración. Por ejemplo, especifique la ruta clave `["mqtt"]` y el valor de combinación `{ "port": 443 }` para establecer el valor `port` en la siguiente configuración.

```
{
  "mqtt": {
    "port": 443
  }
}
```

La ruta clave debe especificar un nodo contenedor (un objeto) en la configuración. Si el nodo no existe en la configuración del componente, esta operación lo crea y establece su valor en el objeto en el que se encuentra`valueToMerge`.

El valor predeterminado es la raíz del objeto de configuración.

`timestamp`

El tiempo de época actual de Unix en milisegundos. Esta operación utiliza esta marca de tiempo para resolver las actualizaciones simultáneas de la clave. Si la clave de la configuración del componente tiene una marca de tiempo mayor que la marca de tiempo de la solicitud, la solicitud falla.

`valueToMerge`(Python:`value_to_merge`)

El objeto de configuración que se va a fusionar en la ubicación que especifique`keyPath`. Para obtener más información, consulte Actualizar las configuraciones de los componentes.

Respuesta

Esta operación no proporciona ninguna información en su respuesta.

SubscribeToConfigurationUpdate

Suscríbase para recibir notificaciones cuando se actualice la configuración de un componente. Al suscribirse a una clave, recibirá una notificación cada vez que alguna de las claves secundarias se actualice.

Esta operación es una operación de suscripción en la que te suscribes a un flujo de mensajes de eventos. Para usar esta operación, defina un controlador de respuesta de transmisión con funciones que gestionen los mensajes de eventos, los errores y el cierre de la transmisión. Para obtener más información, consulte Suscríbase a las transmisiones de eventos del IPC.

Tipo de mensaje de evento: `ConfigurationUpdateEvents`

Solicitud

La solicitud de esta operación tiene los siguientes parámetros:

`componentName`(Python:`component_name`)

(Opcional) El nombre del componente.

El valor predeterminado es el nombre del componente que realiza la solicitud.

`keyPath`(Python:`key_path`)

La ruta clave al valor de configuración al que se va a suscribir. Especifique una lista en la que cada entrada sea la clave de un único nivel del objeto de configuración. Por ejemplo, especifique `["mqtt", "port"]` si desea obtener el valor de `port` en la siguiente configuración.

```
{
  "mqtt": {
    "port": 443
  }
}
```

Para suscribirse a las actualizaciones de todos los valores de la configuración del componente, especifique una lista vacía.

Respuesta

La respuesta de esta operación contiene la siguiente información:

`messages`

El flujo de mensajes de notificación. Este objeto,`ConfigurationUpdateEvents`, contiene la siguiente información:

`configurationUpdateEvent`(Python:`configuration_update_event`)

El evento de actualización de la configuración. Este objeto,`ConfigurationUpdateEvent`, contiene la siguiente información:

`componentName`(Python:`component_name`)

El nombre del componente.

`keyPath`(Python:`key_path`)

La ruta clave al valor de configuración que se actualizó.

SubscribeToValidateConfigurationUpdates

Suscríbase para recibir notificaciones antes de que se actualice la configuración de este componente. Esto permite a los componentes validar las actualizaciones de su propia configuración.

Utilice la SendConfigurationValidityReport operación para indicar al núcleo si la configuración es válida o no.

> **Important**
>
> Las implementaciones locales no notifican las actualizaciones a los componentes.

Esta operación es una operación de suscripción en la que se suscribe a un flujo de mensajes de eventos. Para usar esta operación, defina un controlador de respuesta de transmisión con funciones que gestionen los mensajes de eventos, los errores y el cierre de la transmisión. Para obtener más información, consulte Suscríbase a las transmisiones de eventos del IPC.

Tipo de mensaje de evento: `ValidateConfigurationUpdateEvents`

Solicitud

La solicitud de esta operación no tiene ningún parámetro.

Respuesta

La respuesta de esta operación contiene la siguiente información:

`messages`

El flujo de mensajes de notificación. Este objeto,`ValidateConfigurationUpdateEvents`, contiene la siguiente información:

`validateConfigurationUpdateEvent`(Python:`validate_configuration_update_event`)

El evento de actualización de la configuración. Este objeto,`ValidateConfigurationUpdateEvent`, contiene la siguiente información:

`deploymentId`(Python:`deployment_id`)

El ID de la AWS IoT Greengrass implementación que actualiza el componente.

`configuration`

El objeto que contiene la nueva configuración.

SendConfigurationValidityReport

Indique al núcleo si una actualización de configuración de este componente es válida o no. La implementación falla si le dice al núcleo que la nueva configuración no es válida. Utilice la SubscribeToValidateConfigurationUpdates operación de suscripción para validar las actualizaciones de configuración.

Si un componente no responde a una notificación de validación de la actualización de la configuración, el núcleo espera el tiempo que especifique en la política de validación de la configuración de la implementación. Transcurrido ese tiempo de espera, el núcleo continúa con el despliegue. El tiempo de espera de validación de componentes predeterminado es de 20 segundos. Para obtener más información, consulte Crear implementaciones y el DeploymentConfigurationValidationPolicyobjeto que puede proporcionar al llamar a la CreateDeploymentoperación.

Solicitud

La solicitud de esta operación tiene los siguientes parámetros:

`configurationValidityReport`(Python:`configuration_validity_report`)

El informe que indica al núcleo si la actualización de configuración es válida o no. Este objeto,`ConfigurationValidityReport`, contiene la siguiente información:

`status`

El estado de validez. Esta enumeración,`ConfigurationValidityStatus`, tiene los siguientes valores:

- `ACCEPTED`— La configuración es válida y el núcleo puede aplicarla a este componente.
- `REJECTED`— La configuración no es válida y la implementación falla.

`deploymentId`(Python:`deployment_id`)

El ID de la AWS IoT Greengrass implementación que solicitó la actualización de la configuración.

`message`

(Opcional) Un mensaje que informa de los motivos por los que la configuración no es válida.

Respuesta

Esta operación no proporciona ninguna información en su respuesta.

Recuperar valores secretos

Utilice el servicio IPC del administrador secreto para recuperar los valores secretos de los secretos del dispositivo principal. El componente de administrador de secretos se utiliza para implementar secretos cifrados en los dispositivos principales. A continuación, puede utilizar una operación de IPC para descifrar el secreto y utilizar su valor en los componentes personalizados.

Temas

- Versiones mínimas del SDK
- Autorización
- GetSecretValue
- Ejemplos

Versiones mínimas del SDK

En la siguiente tabla se enumeran las versiones mínimas del SDK para dispositivos con AWS IoT que debes usar para recuperar los valores secretos de los secretos del dispositivo principal.

SDK	Versión mínima	
SDK para dispositivos con AWS IoTpara Java v2	v1.2.10	
SDK para dispositivos con AWS IoTpara Python v2	v1.5.3	
SDK para dispositivos con AWS IoTpara C++ v2	v1.17.0	
SDK para dispositivos con AWS IoTpara v2 JavaScript	v1.12.0	

Autorización

Para usar el administrador de secretos en un componente personalizado, debe definir políticas de autorización que permitan a su componente obtener el valor de los secretos que almacena en el dispositivo principal. Para obtener información sobre cómo definir las políticas de autorización, consulteAutorice a los componentes a realizar operaciones de IPC.

Las políticas de autorización del administrador secreto tienen las siguientes propiedades.

Identificador del servicio IPC: `aws.greengrass.SecretManager`

Operación	Descripción	Recursos
`aws.greengrass#GetSecretValue` o `*`	Permite que un componente obtenga el valor de los secretos que están cifrados en el dispositivo principal.	Un ARN secreto de Secrets Manager, o `*` para permitir el acceso a todos los secretos.

Ejemplos de políticas de autorización

Puede hacer referencia al siguiente ejemplo de política de autorización para ayudarle a configurar las políticas de autorización para sus componentes.

Example Ejemplo de política de autorización

El siguiente ejemplo de política de autorización permite a un componente obtener el valor de cualquier secreto del dispositivo principal.

> **Note**
>
> En un entorno de producción, se recomienda reducir el alcance de la política de autorización para que el componente recupere solo los secretos que utiliza. Puede cambiar el `*` comodín por una lista de ARN secretos al implementar el componente.

```
{
  "accessControl": {
```

```
    "aws.greengrass.SecretManager": {
      "com.example.MySecretComponent:secrets:1": {
        "policyDescription": "Allows access to a secret.",
        "operations": [
          "aws.greengrass#GetSecretValue"
        ],
        "resources": [
          "*"
        ]
      }
    }
  }
}
```

GetSecretValue

Obtiene el valor de un secreto que se almacena en el dispositivo principal.

Esta operación es similar a la operación Secrets Manager, que puede utilizar para obtener el valor de un secreto enNube de AWS. Para obtener más información, consulte GetSecretValue en la Referencia de la API de AWS Secrets Manager.

Solicitud

La solicitud de esta operación tiene los siguientes parámetros:

`secretId`(Python:`secret_id`)

El nombre del secreto que hay que obtener. Puede especificar el nombre del recurso de Amazon (ARN) o el nombre descriptivo del secreto.

`versionId`(Python:`version_id`)

(Opcional) El ID de la versión que se va a obtener.

Puede especificar `versionId` o `versionStage`.

Si no se especifica `versionId` o`versionStage`, esta operación toma como valor predeterminado la versión con la `AWSCURRENT` etiqueta.

`versionStage`(Python:`version_stage`)

(Opcional) La etiqueta provisional de la versión que se va a obtener.

Puede especificar `versionId` o `versionStage`.

Si no especifica `versionId` o`versionStage`, esta operación toma como valor predeterminado la versión con la `AWSCURRENT` etiqueta.

Respuesta

La respuesta de esta operación contiene la siguiente información:

`secretId`(Python:`secret_id`)

El ID del secreto.

`versionId`(Python:`version_id`)

El ID de esta versión del secreto.

`versionStage`(Python:`version_stage`)

La lista de etiquetas de puesta en escena adjunta a esta versión del secreto.

`secretValue`(Python:`secret_value`)

El valor de esta versión del secreto. Este objeto,`SecretValue`, contiene la siguiente información.

`secretString`(Python:`secret_string`)

La parte descifrada de la información secreta protegida que proporcionaste a Secrets Manager en forma de cadena.

`secretBinary`(Python:`secret_binary`)

(Opcional) La parte descifrada de la información secreta protegida que proporcionó a Secrets Manager como datos binarios en forma de matriz de bytes. Esta propiedad contiene los datos binarios como una cadena codificada en base64.

Esta propiedad no se utiliza si ha creado el secreto en la consola de Secrets Manager.

Ejemplos

Los siguientes ejemplos muestran cómo llamar a esta operación en un código de componente personalizado.

Java (IPC client V1)

Example Ejemplo: obtener un valor secreto

> **Note**
>
> En este ejemplo, se utiliza una `IPCUtils` clase para crear una conexión con el servicio AWS IoT Greengrass Core IPC. Para obtener más información, consulte Conéctese al AWS IoT Greengrass servicio 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 Ejemplo: obtener un valor secreto

> **Note**
>
> En este ejemplo se supone que está utilizando la versión 1.5.4 o posterior de SDK para dispositivos con AWS IoT para 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 Ejemplo: obtener un valor secreto

```
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();
```

Ejemplos

Utilice los siguientes ejemplos para aprender a utilizar el servicio IPC del administrador secreto en sus componentes.

Ejemplo: secreto de impresión (Python, cliente IPC V1)

Este componente de ejemplo imprime el valor de un secreto que se implementa en el dispositivo principal.

> **Important**
>
> Este componente de ejemplo imprime el valor de un secreto, así que utilícelo solo con los secretos que almacenan datos de prueba. No utilice este componente para imprimir el valor de un secreto que almacena información importante.

Temas

- Receta
- Artefactos
- Uso

Receta

La siguiente receta de ejemplo define un parámetro de configuración de ARN secreto y permite que el componente obtenga el valor de cualquier secreto del dispositivo principal.

> **Note**
>
> En un entorno de producción, se recomienda reducir el alcance de la política de autorización para que el componente recupere solo los secretos que utiliza. Puede cambiar el * comodín por una lista de ARN secretos al implementar el componente.

JSON

```
{
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "com.example.PrintSecret",
  "ComponentVersion": "1.0.0",
  "ComponentDescription": "Prints the value of an AWS Secrets Manager secret.",
  "ComponentPublisher": "Amazon",
  "ComponentDependencies": {
    "aws.greengrass.SecretManager": {
      "VersionRequirement": "^2.0.0",
      "DependencyType": "HARD"
    }
  },
  "ComponentConfiguration": {
    "DefaultConfiguration": {
      "SecretArn": "",
      "accessControl": {
        "aws.greengrass.SecretManager": {
          "com.example.PrintSecret:secrets:1": {
            "policyDescription": "Allows access to a secret.",
            "operations": [
              "aws.greengrass#GetSecretValue"
            ],
            "resources": [
              "*"
            ]
          }
        }
      }
    }
  },
  "Manifests": [
    {
      "Platform": {
```

```
        "os": "linux"
      },
      "Lifecycle": {
        "install": "python3 -m pip install --user awsiotsdk",
        "run": "python3 -u {artifacts:path}/print_secret.py \"{configuration:/
SecretArn}\""
      }
    },
    {
      "Platform": {
        "os": "windows"
      },
      "Lifecycle": {
        "install": "py -3 -m pip install --user awsiotsdk",
        "run": "py -3 -u {artifacts:path}/print_secret.py \"{configuration:/
SecretArn}\""
      }
    }
  ]
}
```

YAML

```
---
RecipeFormatVersion: '2020-01-25'
ComponentName: com.example.PrintSecret
ComponentVersion: 1.0.0
ComponentDescription: Prints the value of a Secrets Manager secret.
ComponentPublisher: Amazon
ComponentDependencies:
  aws.greengrass.SecretManager:
    VersionRequirement: "^2.0.0"
    DependencyType: HARD
ComponentConfiguration:
  DefaultConfiguration:
    SecretArn: ''
    accessControl:
      aws.greengrass.SecretManager:
        com.example.PrintSecret:secrets:1:
          policyDescription: Allows access to a secret.
          operations:
            - aws.greengrass#GetSecretValue
          resources:
```

```
            - "*"
Manifests:
  - Platform:
      os: linux
    Lifecycle:
      install: python3 -m pip install --user awsiotsdk
      run: python3 -u {artifacts:path}/print_secret.py "{configuration:/SecretArn}"
  - Platform:
      os: windows
    Lifecycle:
      install: py -3 -m pip install --user awsiotsdk
      run: py -3 -u {artifacts:path}/print_secret.py "{configuration:/SecretArn}"
```

Artefactos

El siguiente ejemplo de aplicación de Python demuestra cómo utilizar el servicio IPC del administrador secreto para obtener el valor de un secreto en el dispositivo principal.

```
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)
```

Uso

Puede usar este componente de ejemplo con el componente de administrador de secretos para implementar e imprimir el valor de un secreto en su dispositivo principal.

Para crear, implementar e imprimir un secreto de prueba

1. Crea un secreto de Secrets Manager con los datos de las pruebas.

   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"}'
```

Guarde el ARN del secreto para usarlo en los pasos siguientes.

Para obtener más información, consulte Creación de un secreto en la Guía del AWS Secrets Manager usuario.

2. Implemente el componente de administrador de secretos (`aws.greengrass.SecretManager`) con la siguiente actualización de combinación de configuraciones. Especifique el ARN del secreto que creó anteriormente.

```
{
  "cloudSecrets": [
    {
      "arn": "arn:aws:secretsmanager:us-
west-2:123456789012:secret:MyTestGreengrassSecret-abcdef"
    }
  ]
}
```

Para obtener más información, consulte Implemente AWS IoT Greengrass componentes en los dispositivos el comando de despliegue de la CLI de Greengrass.

3. Cree e implemente el componente de ejemplo de esta sección con la siguiente actualización de combinación de configuraciones. Especifique el ARN del secreto que creó anteriormente.

```
{
  "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"
        ]
      }
    }
  }
}
```

Para obtener más información, consulte Crear AWS IoT Greengrass componentes

4. Consulte los registros del software AWS IoT Greengrass principal para comprobar que las implementaciones se han realizado correctamente y consulte el registro de `com.example.PrintSecret` componentes para ver el valor secreto impreso. Para obtener más información, consulte Supervisar AWS IoT Greengrass registros.

Interactúa con las sombras locales

Utilice el servicio IPC oculto para interactuar con las sombras locales de un dispositivo. El dispositivo con el que elija interactuar puede ser su dispositivo principal o un dispositivo cliente conectado.

Para utilizar estas operaciones de IPC, incluya el componente shadow manager como una dependencia en su componente personalizado. A continuación, puede utilizar las operaciones de IPC en sus componentes personalizados para interactuar con las sombras locales del dispositivo a través del administrador de sombras. Para permitir que los componentes personalizados reaccionen ante los cambios en los estados paralelos locales, también puede utilizar el servicio de publicación/suscripción de IPC para suscribirse a eventos paralelos. Para obtener más información sobre el uso del servicio de publicación/suscripción, consulte la. Publicar/suscribir mensajes locales

> **Note**
>
> Para permitir que un dispositivo principal interactúe con las sombras de los dispositivos del cliente, también debe configurar e implementar el componente MQTT bridge. Para obtener más información, consulte Habilitar el administrador de sombras para que se comunique con los dispositivos cliente.

Temas

- Versiones mínimas del SDK
- Autorización
- GetThingShadow
- UpdateThingShadow
- DeleteThingShadow
- ListNamedShadowsForThing

Versiones mínimas del SDK

En la siguiente tabla se enumeran las versiones mínimas del SDK para dispositivos con AWS IoT que debe utilizar para interactuar con las sombras locales.

SDK	Versión mínima
SDK para dispositivos con AWS IoTpara Java v2	v1.4.0
SDK para dispositivos con AWS IoTpara Python v2	v1.6.0
SDK para dispositivos con AWS IoTpara C++ v2	v1.17.0
SDK para dispositivos con AWS IoTpara v2 JavaScript	v1.12.0

Autorización

Para utilizar el servicio IPC oculto en un componente personalizado, debe definir políticas de autorización que permitan que el componente interactúe con las sombras. Para obtener información sobre cómo definir las políticas de autorización, consulteAutorice a los componentes a realizar operaciones de IPC.

Las políticas de autorización para la interacción oculta tienen las siguientes propiedades.

Identificador del servicio IPC: `aws.greengrass.ShadowManager`

Operación	Descripción	Recursos
`aws.greengrass#GetThingShadow`	Permite que un componente recupere la sombra de una cosa.	Una de las siguientes cadenas: • `$aws/things/` *thingName* `/shadow/`, para permitir el acceso a la sombra clásica del dispositivo. • `$aws/things/` *thingName* `/shadow/name/` *shadowName*, para permitir el acceso a una sombra con nombre. • `*` para permitir el acceso a todas las sombras.
`aws.greengrass#UpdateThingShadow`	Permite que un componente actualice la sombra de una cosa.	Una de las siguientes cadenas: • `$aws/things/` *thingName* `/shadow/`, para permitir el acceso a la sombra clásica del dispositivo. • `$aws/things/` *thingName* `/shadow/name/` *shadowName*, para permitir el acceso a una sombra con nombre. • `*` para permitir el acceso a todas las sombras.

Operación	Descripción	Recursos
`aws.greengrass#DeleteThingShadow`	Permite a un componente eliminar la sombra de una cosa.	Una de las siguientes cadenas: • `$aws/things/` *`thingName`* `/shadow/`, para permitir el acceso a la sombra clásica del dispositivo • `$aws/things/` *`thingName`* `/shadow/name/` *`shadowName`* , para permitir el acceso a una sombra con nombre • `*`, para permitir el acceso a todas las sombras.
`aws.greengrass#ListNamedShadowsForThing`	Permite que un componente recupere la lista de sombras con nombre de una cosa.	Cadena con el nombre de una cosa que permite acceder a la cosa para enumerar sus sombras. Se usa * para permitir el acceso a todas las cosas.

Identificador del servicio IPC: `aws.greengrass.ipc.pubsub`

Operación	Descripción	Recursos
`aws.greengrass#SubscribeToTopic`	Permite que un componente se suscriba a los mensajes de los temas que especifique.	Una de las siguientes cadenas de temas: • *`shadowTopicPrefix`* `/get/accepted`

Operación	Descripción	Recursos
		• *shadowTopicPrefix* / get/rejected • *shadowTopicPrefix* / delete/accepted • *shadowTopicPrefix* / delete/rejected • *shadowTopicPrefix* / update/accepted • *shadowTopicPrefix* / update/delta • *shadowTopicPrefix* / update/rejected El valor del prefijo *shadowTopicPrefix* del tema depende del tipo de sombra: • Sombra clásica: $aws/things/ *thingName* /shadow • Sombra con nombre: $aws/things/ *thingName* /shadow/name/ *shadowName* Se usa * para permitir el acceso a todos los temas. En Greengrass nucleus v2.6.0 y versiones posteriores, puede suscribirse a temas que contengan comodines (y) de temas MQTT. # + Esta cadena

Operación	Descripción	Recursos
		de tema admite los comodines de los temas MQTT como caracteres literales. Por ejemplo, si la política de autorización de un component e permite el acceso `atest/topic/#` , el component e se puede suscribir`test/topic/#` , pero no se puede suscribir a él. `test/topic/filter`

Variables de receta en las políticas locales de autorización clandestina

Si usa la versión 2.6.0 o posterior del núcleo de Greengrass y establece la opción de interpolateComponentConfigurationconfiguración del núcleo de Greengrass en`true`, puede usar la variable de receta en las políticas de autorización. `{iot:thingName}` Esta función le permite configurar una política de autorización única para un grupo de dispositivos principales, de forma que cada dispositivo principal solo pueda acceder a su propia sombra. Por ejemplo, puede permitir que un componente acceda al siguiente recurso para realizar operaciones de IPC ocultas.

```
$aws/things/{iot:thingName}/shadow/
```

Ejemplos de políticas de autorización

Puede hacer referencia a los siguientes ejemplos de políticas de autorización para ayudarle a configurar las políticas de autorización para sus componentes.

Example Ejemplo: permitir que un grupo de dispositivos principales interactúe con las sombras locales

> **Important**
>
> En este ejemplo se utiliza una función que está disponible para la versión 2.6.0 y versiones posteriores del componente núcleo de Greengrass. Greengrass nucleus v2.6.0 añade soporte para la mayoría de las variables de receta, por ejemplo, en las configuraciones

de componentes`{iot:thingName}`. Para activar esta función, defina la opción de interpolateComponentConfigurationconfiguración del núcleo de Greengrass en. `true` Para ver un ejemplo que funcione para todas las versiones del núcleo de Greengrass, consulte el ejemplo de política de autorización para un dispositivo de un solo núcleo.

El siguiente ejemplo de política de autorización permite `com.example.MyShadowInteractionComponent` que el componente interactúe con la sombra del dispositivo clásica y con la sombra `myNamedShadow` denominada del dispositivo principal que ejecuta el componente. Esta política también permite que este componente reciba mensajes sobre temas locales relacionados con estas sombras.

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 Ejemplo: permitir que un grupo de dispositivos principales interactúe con las sombras de los dispositivos del cliente

> **Important**
>
> Esta función requiere Greengrass nucleus v2.6.0 o posterior, shadow manager v2.2.0 o posterior y MQTT bridge v2.2.0 o posterior. Debe configurar el puente MQTT para permitir que Shadow Manager se comunique con los dispositivos cliente.

El siguiente ejemplo de política de autorización permite que el componente `com.example.MyShadowInteractionComponent` interactúe con todos los dispositivos ocultos de los dispositivos cliente cuyos nombres comiencen por. `MyClientDevice`

> **Note**
>
> Para permitir que un dispositivo principal interactúe con las sombras de los dispositivos cliente, también debe configurar e implementar el componente MQTT bridge. Para obtener más información, consulte Habilitar el administrador de sombras para que se comunique con los dispositivos cliente.

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 Ejemplo: permitir que un dispositivo de un solo núcleo interactúe con las sombras locales

El siguiente ejemplo de política de autorización permite `com.example.MyShadowInteractionComponent` que el componente interactúe con la sombra clásica del dispositivo y la sombra nombrada `myNamedShadow` para el dispositivo`MyThingName`. Esta política también permite que este componente reciba mensajes sobre temas locales relacionados con estas sombras.

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 Ejemplo: permitir que un grupo de dispositivos principales reaccione a los cambios locales en el estado de sombra

> **Important**
>
> En este ejemplo se utiliza una función que está disponible para la versión 2.6.0 y versiones posteriores del componente núcleo de Greengrass. Greengrass nucleus v2.6.0 añade soporte para la mayoría de las variables de receta, por ejemplo, en las configuraciones de componentes`{iot:thingName}`. Para activar esta función, defina la opción de interpolateComponentConfigurationconfiguración del núcleo de Greengrass en. `true` Para

ver un ejemplo que funcione para todas las versiones del núcleo de Greengrass, consulte el ejemplo de política de autorización para un dispositivo de un solo núcleo.

El siguiente ejemplo de política de control de acceso permite `com.example.MyShadowReactiveComponent` a los usuarios recibir mensajes sobre el `/update/delta` tema correspondiente a la sombra de dispositivo clásica y a la sombra nombrada `myNamedShadow` en cada dispositivo principal en el que se ejecute el componente.

JSON

```
{
  "accessControl": {
    "aws.greengrass.ipc.pubsub": {
      "com.example.MyShadowReactiveComponent:pubsub:1": {
        "policyDescription": "Allows access to shadow pubsub topics",
        "operations": [
          "aws.greengrass#SubscribeToTopic"
        ],
        "resources": [
          "$aws/things/{iot:thingName}/shadow/update/delta",
          "$aws/things/{iot:thingName}/shadow/name/myNamedShadow/update/delta"
        ]
      }
    }
  }
}
```

YAML

```
accessControl:
  aws.greengrass.ipc.pubsub:
    "com.example.MyShadowReactiveComponent:pubsub:1":
      policyDescription: Allows access to shadow pubsub topics
      operations:
        - 'aws.greengrass#SubscribeToTopic'
      resources:
        - $aws/things/{iot:thingName}/shadow/update/delta
        - $aws/things/{iot:thingName}/shadow/name/myNamedShadow/update/delta
```

Example Ejemplo: permitir que un dispositivo de un solo núcleo reaccione a los cambios locales en el estado de sombra

El siguiente ejemplo de política de control de acceso permite `com.example.MyShadowReactiveComponent` a la empresa personalizada recibir mensajes sobre el `/update/delta` tema correspondiente a la sombra clásica del dispositivo y a la sombra denominada `myNamedShadow` correspondiente al dispositivo`MyThingName`.

JSON

```
{
  "accessControl": {
    "aws.greengrass.ipc.pubsub": {
      "com.example.MyShadowReactiveComponent:pubsub:1": {
        "policyDescription": "Allows access to shadow pubsub topics",
        "operations": [
          "aws.greengrass#SubscribeToTopic"
        ],
        "resources": [
          "$aws/things/MyThingName/shadow/update/delta",
          "$aws/things/MyThingName/shadow/name/myNamedShadow/update/delta"
        ]
      }
    }
  }
}
```

YAML

```
accessControl:
  aws.greengrass.ipc.pubsub:
    "com.example.MyShadowReactiveComponent:pubsub:1":
      policyDescription: Allows access to shadow pubsub topics
      operations:
        - 'aws.greengrass#SubscribeToTopic'
      resources:
        - $aws/things/MyThingName/shadow/update/delta
        - $aws/things/MyThingName/shadow/name/myNamedShadow/update/delta
```

GetThingShadow

Obtén la sombra de una cosa específica.

Solicitud

La solicitud de esta operación tiene los siguientes parámetros:

`thingName`(Python:`thing_name`)

El nombre del objeto.

Tipo: `string`

`shadowName`(Python:`shadow_name`)

El nombre de la sombra. Para especificar la sombra clásica de la cosa, defina este parámetro en una cadena vacía (`""`).

> **Warning**
>
> El AWS IoT Greengrass servicio usa el `AWSManagedGreengrassV2Deployment` nombre shadow para administrar las implementaciones que se dirigen a dispositivos principales individuales. Esta sombra denominada está reservada para que la utilice el AWS IoT Greengrass servicio. No actualice ni elimine esta sombra con nombre.

Tipo: `string`

Respuesta

La respuesta de esta operación contiene la siguiente información:

`payload`

El estado de la respuesta se documenta en forma de blob.

Tipo: `object` que contiene la siguiente información:

`state`

La información del estado.

Este objeto contiene la siguiente información.

`desired`

Las propiedades y valores del estado que se solicita actualizar en el dispositivo.

Tipo: `map` de pares clave-valor

`reported`

Las propiedades y valores del estado informados por el dispositivo.

Tipo: `map` de pares clave-valor

`delta`

La diferencia entre las propiedades y los valores de estado deseados y reportados. Esta propiedad solo está presente si los `reported` estados `desired` y son diferentes.

Tipo: `map` de pares clave-valor

`metadata`

Las marcas de tiempo de cada atributo de `reported` las secciones `desired` y permiten determinar cuándo se actualizó el estado.

Tipo: `string`

`timestamp`

La época, fecha y hora en que se generó la respuesta.

Tipo: `integer`

`clientToken`(Python:`clientToken`)

El token que se utiliza para hacer coincidir la solicitud y la respuesta correspondiente

Tipo: `string`

`version`

La versión del documento paralelo local.

Tipo: `integer`

Errores

Esta operación puede devolver los siguientes errores.

`InvalidArgumentsError`

El servicio paralelo local no puede validar los parámetros de la solicitud. Esto puede ocurrir si la solicitud contiene un formato incorrecto de JSON o caracteres no admitidos.

`ResourceNotFoundError`

No se encuentra el documento alternativo local solicitado.

`ServiceError`

Se ha producido un error de servicio interno o el número de solicitudes al servicio de IPC ha superado los límites especificados en `maxLocalRequestsPerSecondPerThing` los parámetros de `maxTotalLocalRequestsRate` configuración del componente shadow manager.

`UnauthorizedError`

La política de autorización del componente no incluye los permisos necesarios para esta operación.

Ejemplos

Los siguientes ejemplos muestran cómo llamar a esta operación en el código de un componente personalizado.

Java (IPC client V1)

Example Ejemplo: obtener la sombra de una cosa

> **Note**
>
> En este ejemplo, se utiliza una `IPCUtils` clase para crear una conexión con el servicio AWS IoT Greengrass Core IPC. Para obtener más información, consulte Conéctese al AWS IoT Greengrass servicio 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 Ejemplo: Consiga una sombra

```
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 Ejemplo: obtener la sombra de una cosa

```
import {
    GetThingShadowRequest
} from 'aws-iot-device-sdk-v2/dist/greengrasscoreipc/model';
import * as greengrasscoreipc from 'aws-iot-device-sdk-v2/dist/greengrasscoreipc';

class GetThingShadow {
    private ipcClient: greengrasscoreipc.Client;
    private thingName: string;
    private shadowName: string;

    constructor() {
        // Define args parameters here
        this.thingName = "<define_your_own_thingName>";
        this.shadowName = "<define_your_own_shadowName>";
```

```
        this.bootstrap();
    }

    async bootstrap() {
        try {
            this.ipcClient = await getIpcClient();
        } catch (err) {
            // parse the error depending on your use cases
            throw err
        }

        try {
            await this.handleGetThingShadowOperation(this.thingName,
                this.shadowName);
        } catch (err) {
            // parse the error depending on your use cases
            throw err
        }
    }

    async handleGetThingShadowOperation(
        thingName: string,
        shadowName: string
    ) {
        const request: GetThingShadowRequest = {
            thingName: thingName,
            shadowName: shadowName
        };
        const response = await this.ipcClient.getThingShadow(request);
    }
}

export async function getIpcClient() {
    try {
        const ipcClient = greengrasscoreipc.createClient();
        await ipcClient.connect()
            .catch(error => {
                // parse the error depending on your use cases
                throw error;
            });
        return ipcClient
    } catch (err) {
        // parse the error depending on your use caseså
        throw err
```

```
    }
}

const startScript = new GetThingShadow();
```

UpdateThingShadow

Actualiza la sombra de la cosa especificada. Si no existe una sombra, se crea una.

Solicitud

La solicitud de esta operación tiene los siguientes parámetros:

`thingName`(Python:`thing_name`)

El nombre del objeto.

Tipo: `string`

`shadowName`(Python:`shadow_name`)

El nombre de la sombra. Para especificar la sombra clásica de la cosa, defina este parámetro en una cadena vacía (`""`).

> **Warning**
>
> El AWS IoT Greengrass servicio usa el `AWSManagedGreengrassV2Deployment` nombre shadow para administrar las implementaciones que se dirigen a dispositivos principales individuales. Esta sombra denominada está reservada para que la utilice el AWS IoT Greengrass servicio. No actualice ni elimine esta sombra con nombre.

Tipo: `string`

`payload`

El documento de estado de la solicitud es un blob.

Tipo: `object` que contiene la siguiente información:

`state`

La información de estado que se va a actualizar. Esta operación de IPC afecta únicamente a los campos especificados.

Este objeto contiene la siguiente información. Normalmente, utilizará la `desired` propiedad o la `reported` propiedad, pero no ambas en la misma solicitud.

`desired`

Las propiedades y valores del estado que se solicita actualizar en el dispositivo.

Tipo: `map` de pares clave-valor

`reported`

Las propiedades y valores del estado informados por el dispositivo.

Tipo: `map` de pares clave-valor

`clientToken`(Python:`client_token`)

(Opcional) El token que se utiliza para hacer coincidir la solicitud y la respuesta correspondiente del token del cliente.

Tipo: `string`

`version`

(Opcional) La versión del documento paralelo local que se va a actualizar. El servicio paralelo procesa la actualización solo si la versión especificada coincide con la última versión que tiene.

Tipo: `integer`

Respuesta

La respuesta de esta operación contiene la siguiente información:

`payload`

El estado de la respuesta se documenta en forma de blob.

Tipo: `object` que contiene la siguiente información:

`state`

La información del estado.

Este objeto contiene la siguiente información.

`desired`

Las propiedades y valores del estado que se solicita actualizar en el dispositivo.

Tipo: `map` de pares clave-valor

`reported`

Las propiedades y valores del estado informados por el dispositivo.

Tipo: `map` de pares clave-valor

`delta`

Las propiedades y valores del estado informados por el dispositivo.

Tipo: `map` de pares clave-valor

`metadata`

Las marcas de tiempo de cada atributo de `reported` las secciones `desired` y permiten determinar cuándo se actualizó el estado.

Tipo: `string`

`timestamp`

La época, fecha y hora en que se generó la respuesta.

Tipo: `integer`

`clientToken`(Python:`client_token`)

El token que se utiliza para hacer coincidir la solicitud y la respuesta correspondiente.

Tipo: `string`

`version`

La versión del documento paralelo local una vez finalizada la actualización.

Tipo: `integer`

Errores

Esta operación puede devolver los siguientes errores.

`ConflictError`

El servicio paralelo local detectó un conflicto de versiones durante la operación de actualización. Esto ocurre cuando la versión de la carga útil de la solicitud no coincide con la versión del último documento paralelo local disponible.

`InvalidArgumentsError`

El servicio paralelo local no puede validar los parámetros de la solicitud. Esto puede ocurrir si la solicitud contiene un formato incorrecto de JSON o caracteres no admitidos.

Un objeto válido `payload` tiene las siguientes propiedades:

- El `state` nodo existe y es un objeto que contiene la información de `reported` estado `desired` o estado.
- Los `reported` nodos `desired` y son objetos o nulos. Al menos uno de estos objetos debe contener información de estado válida.
- La profundidad de los `reported` objetos `desired` y no puede superar los ocho nodos.
- La longitud del `clientToken` valor no puede superar los 64 caracteres.
- El `version` valor debe ser igual 1 o superior.

`ServiceError`

Se ha producido un error de servicio interno o el número de solicitudes al servicio de IPC ha superado los límites especificados en `maxLocalRequestsPerSecondPerThing` los parámetros de `maxTotalLocalRequestsRate` configuración del componente shadow manager.

`UnauthorizedError`

La política de autorización del componente no incluye los permisos necesarios para esta operación.

Ejemplos

Los siguientes ejemplos muestran cómo llamar a esta operación en el código de un componente personalizado.

Java (IPC client V1)

Example Ejemplo: actualizar la sombra de una cosa

> **Note**
>
> En este ejemplo, se utiliza una `IPCUtils` clase para crear una conexión con el servicio AWS IoT Greengrass Core IPC. Para obtener más información, consulte Conéctese al AWS IoT Greengrass servicio 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 Ejemplo: actualizar la sombra de una cosa

```
import awsiot.greengrasscoreipc
import awsiot.greengrasscoreipc.client as client
from awsiot.greengrasscoreipc.model import UpdateThingShadowRequest

TIMEOUT = 10

def sample_update_thing_shadow_request(thingName, shadowName, payload):
    try:
        # set up IPC client to connect to the IPC server
        ipc_client = awsiot.greengrasscoreipc.connect()

        # create the UpdateThingShadow request
        update_thing_shadow_request = UpdateThingShadowRequest()
        update_thing_shadow_request.thing_name = thingName
        update_thing_shadow_request.shadow_name = shadowName
        update_thing_shadow_request.payload = payload

        # retrieve the UpdateThingShadow response after sending the request to the
 IPC server
        op = ipc_client.new_update_thing_shadow()
        op.activate(update_thing_shadow_request)
        fut = op.get_response()

        result = fut.result(TIMEOUT)
        return result.payload

    except InvalidArgumentsError as e:
        # add error handling
    ...
    # except ConflictError | UnauthorizedError | ServiceError
```

JavaScript

Example Ejemplo: actualizar la sombra de una cosa

```
import {
    UpdateThingShadowRequest
} from 'aws-iot-device-sdk-v2/dist/greengrasscoreipc/model';
```

```
import * as greengrasscoreipc from 'aws-iot-device-sdk-v2/dist/greengrasscoreipc';

class UpdateThingShadow {
    private ipcClient: greengrasscoreipc.Client;
    private thingName: string;
    private shadowName: string;
    private shadowDocumentStr: string;

    constructor() {
        // Define args parameters here

        this.thingName = "<define_your_own_thingName>";
        this.shadowName = "<define_your_own_shadowName>";
        this.shadowDocumentStr = "<define_your_own_payload>";

        this.bootstrap();
    }

    async bootstrap() {
        try {
            this.ipcClient = await getIpcClient();
        } catch (err) {
            // parse the error depending on your use cases
            throw err
        }

        try {
            await this.handleUpdateThingShadowOperation(
                this.thingName,
                this.shadowName,
                this.shadowDocumentStr);
        } catch (err) {
            // parse the error depending on your use cases
            throw err
        }
    }

    async handleUpdateThingShadowOperation(
        thingName: string,
        shadowName: string,
        payloadStr: string
    ) {
        const request: UpdateThingShadowRequest = {
            thingName: thingName,
```

```
            shadowName: shadowName,
            payload: payloadStr
        }
        // make the UpdateThingShadow request
        const response = await this.ipcClient.updateThingShadow(request);
    }
}

export async function getIpcClient() {
    try {
        const ipcClient = greengrasscoreipc.createClient();
        await ipcClient.connect()
            .catch(error => {
                // parse the error depending on your use cases
                throw error;
            });
        return ipcClient
    } catch (err) {
        // parse the error depending on your use cases
        throw err
    }
}

const startScript = new UpdateThingShadow();
```

DeleteThingShadow

Elimina la sombra de objeto especificado.

A partir de la versión 2.0.4 de Shadow Manager, al eliminar una sombra se incrementa el número de versión. Por ejemplo, si eliminas la sombra `MyThingShadow` en la versión 1, la versión de la sombra eliminada es la 2. Si después vuelves a crear una sombra con ese nombre`MyThingShadow`, la versión de esa sombra es 3.

Solicitud

La solicitud de esta operación tiene los siguientes parámetros:

`thingName`(Python:`thing_name`)

El nombre del objeto.

Tipo: `string`

`shadowName`(Python:`shadow_name`)

El nombre de la sombra. Para especificar la sombra clásica de la cosa, defina este parámetro en una cadena vacía ("").

> **Warning**
>
> El AWS IoT Greengrass servicio usa el `AWSManagedGreengrassV2Deployment` nombre shadow para administrar las implementaciones que se dirigen a dispositivos principales individuales. Esta sombra denominada está reservada para que la utilice el AWS IoT Greengrass servicio. No actualice ni elimine esta sombra con nombre.

Tipo: `string`

Respuesta

La respuesta de esta operación contiene la siguiente información:

`payload`

Un documento de estado de respuesta vacío.

Errores

Esta operación puede devolver los siguientes errores.

`InvalidArgumentsError`

El servicio paralelo local no puede validar los parámetros de la solicitud. Esto puede ocurrir si la solicitud contiene un formato incorrecto de JSON o caracteres no admitidos.

`ResourceNotFoundError`

No se encuentra el documento alternativo local solicitado.

`ServiceError`

Se ha producido un error de servicio interno o el número de solicitudes al servicio de IPC ha superado los límites especificados en `maxLocalRequestsPerSecondPerThing` los

parámetros de `maxTotalLocalRequestsRate` configuración del componente shadow manager.

`UnauthorizedError`

La política de autorización del componente no incluye los permisos necesarios para esta operación.

Ejemplos

Los siguientes ejemplos muestran cómo llamar a esta operación en el código de un componente personalizado.

Java (IPC client V1)

Example Ejemplo: eliminar la sombra de una cosa

> **Note**
>
> En este ejemplo, se utiliza una `IPCUtils` clase para crear una conexión con el servicio AWS IoT Greengrass Core IPC. Para obtener más información, consulte Conéctese al AWS IoT Greengrass servicio 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 Ejemplo: eliminar la sombra de una cosa

```
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 Ejemplo: Eliminar la sombra de una cosa

```
import {
    DeleteThingShadowRequest
} from 'aws-iot-device-sdk-v2/dist/greengrasscoreipc/model';
import * as greengrasscoreipc from 'aws-iot-device-sdk-v2/dist/greengrasscoreipc';

class DeleteThingShadow {
    private ipcClient: greengrasscoreipc.Client;
    private thingName: string;
    private shadowName: string;

    constructor() {
        // Define args parameters here
        this.thingName = "<define_your_own_thingName>";
        this.shadowName = "<define_your_own_shadowName>";
        this.bootstrap();
    }

    async bootstrap() {
        try {
            this.ipcClient = await getIpcClient();
        } catch (err) {
            // parse the error depending on your use cases
            throw err
        }

        try {
            await this.handleDeleteThingShadowOperation(this.thingName,
 this.shadowName)
        } catch (err) {
            // parse the error depending on your use cases
            throw err
        }
    }

    async handleDeleteThingShadowOperation(thingName: string, shadowName: string) {
```

```
        const request: DeleteThingShadowRequest = {
            thingName: thingName,
            shadowName: shadowName
        }
        // make the DeleteThingShadow request
        const response = await this.ipcClient.deleteThingShadow(request);
    }
}

export async function getIpcClient() {
    try {
        const ipcClient = greengrasscoreipc.createClient();
        await ipcClient.connect()
            .catch(error => {
                // parse the error depending on your use cases
                throw error;
            });
        return ipcClient
    } catch (err) {
        // parse the error depending on your use cases
        throw err
    }
}

const startScript = new DeleteThingShadow();
```

ListNamedShadowsForThing

Enumere las sombras con nombre de la cosa especificada.

Solicitud

La solicitud de esta operación tiene los siguientes parámetros:

`thingName`(Python:`thing_name`)

El nombre del objeto.

Tipo: `string`

`pageSize`(Python:`page_size`)

(Opcional) El número de nombres ocultos que se devolverán en cada llamada.

Tipo: `integer`

Predeterminado: 25

Máximo: 100

`nextToken`(Python:`next_token`)

(Opcional) El token para recuperar el siguiente conjunto de resultados. Este valor se devuelve en los resultados paginados y se utiliza en la llamada que devuelve la página siguiente.

Tipo: `string`

Respuesta

La respuesta de esta operación contiene la siguiente información:

`results`

La lista de nombres ocultos.

Tipo: `array`

`timestamp`

(Opcional) La fecha y la hora en que se generó la respuesta.

Tipo: `integer`

`nextToken`(Python:`next_token`)

(Opcional) El valor del token que se utilizará en las solicitudes paginadas para recuperar la siguiente página de la secuencia. Este token no está presente cuando no hay más nombres ocultos que devolver.

Tipo: `string`

> **Note**
>
> Si el tamaño de página solicitado coincide exactamente con el número de nombres ocultos de la respuesta, entonces este token está presente; sin embargo, cuando se usa, devuelve una lista vacía.

Errores

Esta operación puede devolver los siguientes errores.

`InvalidArgumentsError`

El servicio paralelo local no puede validar los parámetros de la solicitud. Esto puede ocurrir si la solicitud contiene un formato incorrecto de JSON o caracteres no admitidos.

`ResourceNotFoundError`

No se encuentra el documento alternativo local solicitado.

`ServiceError`

Se ha producido un error de servicio interno o el número de solicitudes al servicio de IPC ha superado los límites especificados en `maxLocalRequestsPerSecondPerThing` los parámetros de `maxTotalLocalRequestsRate` configuración del componente shadow manager.

`UnauthorizedError`

La política de autorización del componente no incluye los permisos necesarios para esta operación.

Ejemplos

Los siguientes ejemplos muestran cómo llamar a esta operación en el código de un componente personalizado.

Java (IPC client V1)

Example Ejemplo: Haz una lista con los nombres de sombras de una cosa

> **Note**
>
> En este ejemplo, se utiliza una `IPCUtils` clase para crear una conexión con el servicio AWS IoT Greengrass Core IPC. Para obtener más información, consulte Conéctese al AWS IoT Greengrass servicio 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 Ejemplo: Haz una lista con los nombres de sombras de una cosa

```
import awsiot.greengrasscoreipc
import awsiot.greengrasscoreipc.client as client
from awsiot.greengrasscoreipc.model import ListNamedShadowsForThingRequest

TIMEOUT = 10

def sample_list_named_shadows_for_thing_request(thingName, nextToken, pageSize):
    try:
        # set up IPC client to connect to the IPC server
        ipc_client = awsiot.greengrasscoreipc.connect()

        # create the ListNamedShadowsForThingRequest request
        list_named_shadows_for_thing_request = ListNamedShadowsForThingRequest()
        list_named_shadows_for_thing_request.thing_name = thingName
        list_named_shadows_for_thing_request.next_token = nextToken
        list_named_shadows_for_thing_request.page_size = pageSize

        # retrieve the ListNamedShadowsForThingRequest response after sending the
 request to the IPC server
        op = ipc_client.new_list_named_shadows_for_thing()
        op.activate(list_named_shadows_for_thing_request)
        fut = op.get_response()

        list_result = fut.result(TIMEOUT)

        # additional returned fields
        timestamp = list_result.timestamp
        next_token = result.next_token
        named_shadow_list = list_result.results

        return named_shadow_list, next_token, timestamp

    except InvalidArgumentsError as e:
        # add error handling
    ...
    # except ResourceNotFoundError | UnauthorizedError | ServiceError
```

JavaScript

Example Ejemplo: Haz una lista con los nombres de sombras de una cosa

```
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();
```

Gestione las implementaciones y los componentes locales

> **Note**
>
> Esta función está disponible para la versión 2.6.0 y versiones posteriores del componente núcleo de Greengrass.

Utilice el servicio IPC CLI de Greengrass para gestionar las implementaciones locales y los componentes de Greengrass en el dispositivo principal.

Para usar estas operaciones de IPC, incluya la versión 2.6.0 o posterior del componente CLI de Greengrass como dependencia en su componente personalizado. A continuación, puede utilizar las operaciones de IPC en sus componentes personalizados para hacer lo siguiente:

- Cree despliegues locales para modificar y configurar los componentes de Greengrass en el dispositivo principal.
- Reinicie y detenga los componentes de Greengrass en el dispositivo principal.
- Genere una contraseña que pueda usar para iniciar sesión en la consola de depuración local.

Temas

Versiones mínimas del SDK

En la siguiente tabla se enumeran las versiones mínimas de las SDK para dispositivos con AWS IoT que debe utilizar para interactuar con el servicio CLI IPC de Greengrass.

SDK	Versión mínima	
SDK para dispositivos con AWS IoTpara Java v2	v1.2.10	
SDK para dispositivos con AWS IoTpara Python v2	v1.5.3	

SDK	Versión mínima	
SDK para dispositivos con AWS IoTpara C++ v2	v1.17.0	
SDK para dispositivos con AWS IoTpara v2 JavaScript	v1.12.0	

Autorización

Para utilizar el servicio CLI IPC de Greengrass en un componente personalizado, debe definir políticas de autorización que permitan a su componente gestionar las implementaciones y los componentes locales. Para obtener información sobre cómo definir las políticas de autorización, consulte. Autorice a los componentes a realizar operaciones de IPC

Las políticas de autorización de la CLI de Greengrass tienen las siguientes propiedades.

Identificador de servicio IPC: `aws.greengrass.Cli`

Operación	Descripción	Recursos
`aws.greengrass#CreateLocalDeployment`	Permite que un componente cree una implementación local en el dispositivo principal.	*
`aws.greengrass#ListLocalDeployments`	Permite que un componente enumere las implementaciones locales en el dispositivo principal.	*
`aws.greengrass#GetLocalDeploymentStatus`	Permite que un componente obtenga el estado de una implementación local en el dispositivo principal.	Un ID de implementación local o * para permitir el acceso a todas las implementaciones locales.
`aws.greengrass#ListComponents`	Permite que un componente enumere los componentes del dispositivo principal.	*

Operación	Descripción	Recursos
`aws.greengrass#GetComponentDetails`	Permite que un componente obtenga detalles sobre un componente del dispositivo principal.	Un nombre de componente, por ejemplo `com.example.HelloWorld`, o `*` para permitir el acceso a todos los componentes.
`aws.greengrass#RestartComponent`	Permite que un componente reinicie un componente del dispositivo principal.	Un nombre de componente, por ejemplo `com.example.HelloWorld`, o `*` para permitir el acceso a todos los componentes.
`aws.greengrass#StopComponent`	Permite que un componente detenga un componente del dispositivo principal.	Un nombre de componente, por ejemplo `com.example.HelloWorld`, o `*` para permitir el acceso a todos los componentes.
`aws.greengrass#CreateDebugPassword`	Permite que un componente genere una contraseña para iniciar sesión en el componente de la consola de depuración local.	`*`

Example Ejemplo de política de autorización

Los siguientes ejemplos de políticas de autorización permiten a un componente crear implementaciones locales, ver todas las implementaciones y componentes locales y reiniciar y detener un componente denominado. `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

Cree o actualice una implementación local mediante recetas de componentes, artefactos y argumentos de tiempo de ejecución específicos.

Esta operación proporciona la misma funcionalidad que el comando deploy create de la CLI de Greengrass.

Solicitud

La solicitud de esta operación tiene los siguientes parámetros:

`recipeDirectoryPath`(Python:`recipe_directory_path`)

(Opcional) La ruta absoluta a la carpeta que contiene los archivos de recetas de los componentes.

`artifactDirectoryPath`(Python:`artifact_directory_path`)

(Opcional) La ruta absoluta a la carpeta que contiene los archivos de artefactos que se van a incluir en la implementación. La carpeta de artefactos debe contener la siguiente estructura de carpetas:

```
/path/to/artifact/folder/component-name/component-version/artifacts
```

`rootComponentVersionsToAdd`(Python:`root_component_versions_to_add`)

(Opcional) Las versiones de los componentes que se van a instalar en el dispositivo principal. Este objeto,`ComponentToVersionMap`, es un mapa que contiene los siguientes pares clave-valor:

`key`

El nombre del componente.

`value`

Esta es la versión del componente.

`rootComponentsToRemove`(Python:`root_components_to_remove`)

(Opcional) Los componentes que se van a desinstalar del dispositivo principal. Especifique una lista en la que cada entrada sea el nombre de un componente.

`componentToConfiguration`(Python:`component_to_configuration`)

(Opcional) La configuración se actualiza para cada componente de la implementación. Este objeto,`ComponentToConfiguration`, es un mapa que contiene los siguientes pares clave-valor:

`key`

El nombre del componente.

`value`

El objeto JSON de actualización de configuración para el componente. El objeto JSON debe tener el siguiente formato.

```
{
  "MERGE": {
    "config-key": "config-value"
  },
  "RESET": [
    "path/to/reset/"
  ]
}
```

Para obtener más información sobre las actualizaciones de configuración, consulteActualizar las configuraciones de los componentes.

`componentToRunWithInfo`(Python:`component_to_run_with_info`)

(Opcional) La configuración del tiempo de ejecución de cada componente de la implementación. Esta configuración incluye el usuario del sistema que es propietario de los procesos de cada componente y los límites del sistema que se aplican a cada componente. Este objeto,`ComponentToRunWithInfo`, es un mapa que contiene los siguientes pares clave-valor:

`key`

El nombre del componente.

`value`

La configuración de tiempo de ejecución del componente. Si omite un parámetro de configuración del tiempo de ejecución, el software AWS IoT Greengrass principal utilizará los valores predeterminados que configure en el núcleo de Greengrass. Este objeto contiene `RunWithInfo` la siguiente información:

`posixUser`(Python:`posix_user`)

(Opcional) El usuario y, opcionalmente, el grupo del sistema POSIX que se utilizará para ejecutar este componente en los dispositivos principales de Linux. El usuario y el grupo, si se especifican, deben existir en cada dispositivo principal de Linux. Especifique el usuario y el grupo separados por dos puntos (`:`) con el siguiente formato: `user:group`. El grupo es opcional. Si no especifica un grupo, el software AWS IoT Greengrass Core utiliza el grupo principal para el usuario. Para obtener más información, consulte Configure el usuario que ejecuta los componentes.

`windowsUser`(Python:`windows_user`)

(Opcional) El usuario de Windows que se utilizará para ejecutar este componente en los dispositivos principales de Windows. El usuario debe estar en todos los dispositivos

principales de Windows y su nombre y contraseña deben almacenarse en la instancia del administrador de credenciales de la LocalSystem cuenta. Para obtener más información, consulte Configure el usuario que ejecuta los componentes.

`systemResourceLimits`(Python:`system_resource_limits`)

(Opcional) Los límites de recursos del sistema que se aplicarán a los procesos de este componente. Puede aplicar límites de recursos del sistema a los componentes Lambda genéricos y no contenerizados. Para obtener más información, consulte Configure los límites de recursos del sistema para los componentes.

AWS IoT Greengrassactualmente no admite esta función en los dispositivos principales de Windows.

Este objeto,`SystemResourceLimits`, contiene la siguiente información:

`cpus`

(Opcional) La cantidad máxima de tiempo de CPU que los procesos de este componente pueden utilizar en el dispositivo principal. El tiempo total de CPU de un dispositivo principal equivale a la cantidad de núcleos de CPU del dispositivo. Por ejemplo, en un dispositivo principal con 4 núcleos de CPU, puede establecer este valor 2 para limitar los procesos de este componente al 50 por ciento de uso de cada núcleo de CPU. En un dispositivo con 1 núcleo de CPU, puede establecer este valor `0.25` para limitar los procesos de este componente al 25 por ciento de uso de la CPU. Si establece este valor en un número superior al número de núcleos de la CPU, el software AWS IoT Greengrass Core no limita el uso de la CPU del componente.

`memory`

(Opcional) La cantidad máxima de RAM (en kilobytes) que los procesos de este componente pueden utilizar en el dispositivo principal.

`groupName`(Python:`group_name`)

(Opcional) El nombre del grupo de cosas al que se va a dirigir esta implementación.

Respuesta

La respuesta de esta operación contiene la siguiente información:

`deploymentId`(Python:`deployment_id`)

El ID de la implementación local que creó la solicitud.

ListLocalDeployments

Obtiene el estado de las últimas 10 implementaciones locales.

Esta operación proporciona la misma funcionalidad que el comando deployment list de la CLI de Greengrass.

Solicitud

La solicitud de esta operación no tiene ningún parámetro.

Respuesta

La respuesta de esta operación contiene la siguiente información:

`localDeployments`(Python:`local_deployments`)

La lista de despliegues locales. Cada objeto de esta lista es un `LocalDeployment` objeto que contiene la siguiente información:

`deploymentId`(Python:`deployment_id`)

El ID de la implementación local.

`status`

El estado de la implementación local. Esta enumeración,`DeploymentStatus`, tiene los siguientes valores:

- `QUEUED`
- `IN_PROGRESS`
- `SUCCEEDED`
- `FAILED`

GetLocalDeploymentStatus

Obtiene el estado de una implementación local.

Esta operación proporciona la misma funcionalidad que el comando deployment status de la CLI de Greengrass.

Solicitud

La solicitud de esta operación tiene los siguientes parámetros:

`deploymentId`(Python:`deployment_id`)

El ID de la implementación local que se va a obtener.

Respuesta

La respuesta de esta operación contiene la siguiente información:

`deployment`

El despliegue local. Este objeto,`LocalDeployment`, contiene la siguiente información:

`deploymentId`(Python:`deployment_id`)

El ID de la implementación local.

`status`

El estado de la implementación local. Esta enumeración,`DeploymentStatus`, tiene los siguientes valores:

- `QUEUED`
- `IN_PROGRESS`
- `SUCCEEDED`
- `FAILED`

ListComponents

Obtiene el nombre, la versión, el estado y la configuración de cada componente raíz del dispositivo principal. Un componente raíz es un componente que se especifica en una implementación. Esta respuesta no incluye los componentes que se instalan como dependencias de otros componentes.

Esta operación proporciona la misma funcionalidad que el comando component list de la CLI de Greengrass.

Solicitud

La solicitud de esta operación no tiene ningún parámetro.

Respuesta

La respuesta de esta operación contiene la siguiente información:

`components`

La lista de componentes raíz del dispositivo principal. Cada objeto de esta lista es un `ComponentDetails` objeto que contiene la siguiente información:

`componentName`(Python:`component_name`)

El nombre del componente.

`version`

Esta es la versión del componente.

`state`

El estado del componente. Este estado puede ser uno de los siguientes:

- `BROKEN`
- `ERRORED`
- `FINISHED`
- `INSTALLED`
- `NEW`
- `RUNNING`
- `STARTING`
- `STOPPING`

`configuration`

La configuración del componente como objeto JSON.

GetComponentDetails

Obtiene la versión, el estado y la configuración de un componente del dispositivo principal.

Esta operación proporciona la misma funcionalidad que el comando component details de la CLI de Greengrass.

Solicitud

La solicitud de esta operación tiene los siguientes parámetros:

`componentName`(Python:`component_name`)

El nombre del componente que se va a obtener.

Respuesta

La respuesta de esta operación contiene la siguiente información:

`componentDetails`(Python:`component_details`)

Los detalles del componente. Este objeto,`ComponentDetails`, contiene la siguiente información:

`componentName`(Python:`component_name`)

El nombre del componente.

`version`

Esta es la versión del componente.

`state`

El estado del componente. Este estado puede ser uno de los siguientes:

- `BROKEN`
- `ERRORED`
- `FINISHED`
- `INSTALLED`
- `NEW`
- `RUNNING`
- `STARTING`
- `STOPPING`

`configuration`

La configuración del componente como objeto JSON.

RestartComponent

Reinicia un componente en el dispositivo principal.

> **Note**
>
> Si bien puede reiniciar cualquier componente, le recomendamos que reinicie solo los componentes genéricos.

Esta operación proporciona la misma funcionalidad que el comando component restart de la CLI de Greengrass.

Solicitud

La solicitud de esta operación tiene los siguientes parámetros:

`componentName`(Python:`component_name`)

El nombre del componente.

Respuesta

La respuesta de esta operación tiene la siguiente información:

`restartStatus`(Python:`restart_status`)

El estado de la solicitud de reinicio. El estado de la solicitud puede ser uno de los siguientes:

- `SUCCEEDED`
- `FAILED`

`message`

Un mensaje sobre el motivo por el que el componente no se pudo reiniciar, si la solicitud falló.

StopComponent

Detiene los procesos de un componente en el dispositivo principal.

> **Note**
>
> Si bien puede detener cualquier componente, le recomendamos que detenga solo los componentes genéricos.

Esta operación proporciona la misma funcionalidad que el comando component stop de la CLI de Greengrass.

Solicitud

La solicitud de esta operación tiene los siguientes parámetros:

`componentName`(Python:`component_name`)

El nombre del componente.

Respuesta

La respuesta de esta operación tiene la siguiente información:

`stopStatus`(Python:`stop_status`)

El estado de la solicitud de parada. El estado de la solicitud puede ser uno de los siguientes:

- `SUCCEEDED`
- `FAILED`

`message`

Un mensaje sobre el motivo por el que el componente no se pudo detener, si la solicitud falló.

CreateDebugPassword

Genera una contraseña aleatoria que puede usar para iniciar sesión en el componente de la consola de depuración local. La contraseña caduca 8 horas después de generarse.

Esta operación proporciona la misma funcionalidad que el get-debug-password comando de la CLI de Greengrass.

Solicitud

La solicitud de esta operación no tiene ningún parámetro.

Respuesta

La respuesta de esta operación contiene la siguiente información:

`username`

El nombre de usuario que se va a utilizar para iniciar sesión.

`password`

La contraseña que se va a usar para iniciar sesión.

`passwordExpiration`(Python:`password_expiration`)

Hora en la que caduca la contraseña.

`certificateSHA256Hash`(Python:`certificate_sha256_hash`)

La huella digital SHA-256 del certificado autofirmado que utiliza la consola de depuración local cuando HTTPS está activado. Cuando abra la consola de depuración local, utilice esta huella digital para comprobar que el certificado es legítimo y que la conexión es segura.

`certificateSHA1Hash`(Python:`certificate_sha1_hash`)

La huella digital SHA-1 del certificado autofirmado que utiliza la consola de depuración local cuando HTTPS está activado. Cuando abra la consola de depuración local, utilice esta huella digital para comprobar que el certificado es legítimo y que la conexión es segura.

Autenticar y autorizar los dispositivos cliente

> **Note**
>
> Esta función está disponible para la versión 2.6.0 y versiones posteriores del componente núcleo de Greengrass.

Utilice el servicio IPC de autenticación de dispositivos cliente para desarrollar un componente de intermediario local personalizado al que puedan conectarse los dispositivos IoT locales, como los dispositivos cliente.

Para usar estas operaciones de IPC, incluya la versión 2.2.0 o posterior del componente de autenticación del dispositivo cliente como una dependencia en su componente personalizado. A continuación, puede utilizar las operaciones de IPC en sus componentes personalizados para hacer lo siguiente:

- Compruebe la identidad de los dispositivos cliente que se conectan al dispositivo principal.
- Cree una sesión para que un dispositivo cliente se conecte al dispositivo principal.
- Compruebe si un dispositivo cliente tiene permiso para realizar una acción.
- Reciba una notificación cuando el certificado del servidor del dispositivo principal cambie.

Temas

Versiones mínimas del SDK

En la siguiente tabla se enumeran las versiones mínimas del servicio IPC de autenticación del dispositivo cliente SDK para dispositivos con AWS IoT que debe utilizar.

SDK	Versión mínima	
SDK para dispositivos con AWS IoTpara Java v2	v1.9.3	
SDK para dispositivos con AWS IoTpara Python v2	v1.11.3	

SDK	Versión mínima	
SDK para dispositivos con AWS IoTpara C++ v2	v1.18.3	
SDK para dispositivos con AWS IoTpara v2 JavaScript	v1.12.0	

Autorización

Para utilizar el servicio IPC de autenticación del dispositivo cliente en un componente personalizado, debe definir políticas de autorización que permitan a su componente realizar estas operaciones. Para obtener información sobre cómo definir las políticas de autorización, consulte. Autorice a los componentes a realizar operaciones de IPC

Las políticas de autorización para la autenticación y autorización de los dispositivos cliente tienen las siguientes propiedades.

Identificador del servicio IPC: `aws.greengrass.clientdevices.Auth`

Operación	Descripción	Recursos
`aws.greengrass#VerifyClientDeviceIdentity`	Permite que un componente verifique la identidad de un dispositivo cliente.	*
`aws.greengrass#GetClientDeviceAuthToken`	Permite que un componente valide las credenciales de un dispositivo cliente y cree una sesión para ese dispositivo cliente.	*
`aws.greengrass#AuthorizeClientDeviceAction`	Permite que un componente verifique si un dispositivo cliente tiene permiso para realizar una acción.	*

Operación	Descripción	Recursos
`aws.greengrass#SubscribeToCertificateUpdates`	Permite que un componente reciba notificaciones cuando el certificado del servidor del dispositivo principal rota.	*
*	Permite que un componente realice todas las operaciones del servicio IPC de autenticación del dispositivo cliente.	*

Ejemplos de políticas de autorización

Puede hacer referencia al siguiente ejemplo de política de autorización para ayudarle a configurar las políticas de autorización para sus componentes.

Example Ejemplo de política de autorización

El siguiente ejemplo de política de autorización permite a un componente realizar todas las operaciones IPC de autenticación del dispositivo cliente.

```
{
  "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

Compruebe la identidad de un dispositivo cliente. Esta operación verifica si el dispositivo cliente es válidoAWS IoT.

Solicitud

La solicitud de esta operación tiene los siguientes parámetros:

`credential`

Las credenciales del dispositivo cliente. Este objeto,`ClientDeviceCredential`, contiene la siguiente información:

`clientDeviceCertificate`(Python:`client_device_certificate`)

El certificado de dispositivo X.509 del dispositivo cliente.

Respuesta

La respuesta de esta operación contiene la siguiente información:

`isValidClientDevice`(Python:`is_valid_client_device`)

Si la identidad del dispositivo cliente es válida.

GetClientDeviceAuthToken

Valida las credenciales de un dispositivo cliente y crea una sesión para el dispositivo cliente. Esta operación devuelve un token de sesión que puede usar en solicitudes posteriores para autorizar acciones del dispositivo cliente.

Para conectar correctamente un dispositivo cliente, el componente de autenticación del dispositivo cliente debe conceder el `mqtt:connect` permiso para el ID de cliente que utiliza el dispositivo cliente.

Solicitud

La solicitud de esta operación tiene los siguientes parámetros:

`credential`

Las credenciales del dispositivo cliente. Este objeto,`CredentialDocument`, contiene la siguiente información:

`mqttCredential`(Python:`mqtt_credential`)

Las credenciales MQTT del dispositivo cliente. Especifique el ID de cliente y el certificado que el dispositivo cliente utiliza para conectarse. Este objeto,`MQTTCredential`, contiene la siguiente información:

`clientId`(Python:`client_id`)

El ID de cliente que se utilizará para conectarse.

`certificatePem`(Python:`certificate_pem`)

El certificado del dispositivo X.509 que se utilizará para conectarse.

`username`

> **Note**
>
> Esta propiedad no se utiliza actualmente.

`password`

> **Note**
>
> Esta propiedad no se utiliza actualmente.

Respuesta

La respuesta de esta operación contiene la siguiente información:

`clientDeviceAuthToken`(Python:`client_device_auth_token`)

El token de sesión del dispositivo cliente. Puede usar este token de sesión en solicitudes posteriores para autorizar las acciones de este dispositivo cliente.

AuthorizeClientDeviceAction

Compruebe si un dispositivo cliente tiene permiso para realizar una acción en un recurso. Las políticas de autorización de los dispositivos cliente especifican los permisos que los dispositivos cliente pueden realizar mientras están conectados a un dispositivo principal. Las políticas de autorización de los dispositivos cliente se definen al configurar el componente de autenticación del dispositivo cliente.

Solicitud

La solicitud de esta operación tiene los siguientes parámetros:

`clientDeviceAuthToken`(Python:`client_device_auth_token`)

El token de sesión del dispositivo cliente.

`operation`

La operación que se va a autorizar.

`resource`

El recurso en el que el dispositivo cliente realiza la operación.

Respuesta

La respuesta de esta operación contiene la siguiente información:

`isAuthorized`(Python:`is_authorized`)

Si el dispositivo cliente está autorizado a realizar la operación en el recurso.

SubscribeToCertificateUpdates

Suscríbase para recibir el nuevo certificado de servidor del dispositivo principal cada vez que gire. Cuando el certificado de servidor rota, los intermediarios deben volver a cargarlo con el nuevo certificado de servidor.

De forma predeterminada, el componente de autenticación del dispositivo cliente rota los certificados del servidor cada 7 días. Puede configurar el intervalo de rotación entre 2 y 10 días.

Esta operación es una operación de suscripción en la que se suscribe a un flujo de mensajes de eventos. Para utilizar esta operación, defina un controlador de respuesta de transmisión con funciones que gestionen los mensajes de eventos, los errores y el cierre de la transmisión. Para obtener más información, consulte Suscríbase a las transmisiones de eventos del IPC.

Tipo de mensaje de evento: `CertificateUpdateEvent`

Solicitud

La solicitud de esta operación tiene los siguientes parámetros:

`certificateOptions`(Python:`certificate_options`)

Los tipos de actualizaciones de certificados a las que suscribirse. Este objeto,`CertificateOptions`, contiene la siguiente información:

`certificateType`(Python:`certificate_type`)

El tipo de actualizaciones de certificados a las que suscribirse. Elija la opción siguiente:

- `SERVER`

Respuesta

La respuesta de esta operación contiene la siguiente información:

`messages`

El flujo de mensajes. Este objeto,`CertificateUpdateEvent`, contiene la siguiente información:

`certificateUpdate`(Python:`certificate_update`)

La información sobre el nuevo certificado. Este objeto,`CertificateUpdate`, contiene la siguiente información:

`certificate`

El certificado.

`privateKey`(Python:`private_key`)

La clave privada del certificado.

`publicKey`(Python:`public_key`)

La clave pública del certificado.

`caCertificates`(Python:`ca_certificates`)

La lista de certificados de la autoridad de certificación (CA) de la cadena de certificados de CA del certificado.

Interactúa con dispositivos IoT locales

Los dispositivos cliente son dispositivos IoT locales que se conectan y se comunican con un dispositivo central de Greengrass a través de MQTT. Puede conectar los dispositivos cliente a los dispositivos principales para hacer lo siguiente:

- Interactúa con los mensajes MQTT en los componentes de Greengrass.
- Transmita mensajes y datos entre los dispositivos cliente y. AWS IoT Core
- Interactúe con las sombras de los dispositivos cliente en los componentes de Greengrass.
- Sincronice las sombras de los dispositivos cliente conAWS IoT Core.

Para conectarse a un dispositivo principal, los dispositivos cliente pueden usar la detección en la nube. Los dispositivos cliente se conectan al servicio AWS IoT Greengrass en la nube para recuperar información sobre los dispositivos principales a los que se pueden conectar. Luego, pueden conectarse a un dispositivo central para procesar sus mensajes y sincronizar sus datos con el servicio AWS IoT Core en la nube.

Puedes seguir un tutorial que explica cómo configurar un dispositivo principal para conectarse y comunicarse con cualquier AWS IoT cosa. En este tutorial también se explica cómo desarrollar un componente personalizado de Greengrass que interactúe con los dispositivos cliente. Para obtener más información, consulte Tutorial: Interactúa con dispositivos IoT locales a través de MQTT.

Temas

AWS-componentes de dispositivo cliente proporcionados

AWS IoT Greengrassproporciona los siguientes componentes públicos que puede implementar en los dispositivos principales. Estos componentes permiten que los dispositivos cliente se conecten y se comuniquen con un dispositivo principal.

> **Note**
>
> Varios AWS de los componentes proporcionados dependen de versiones secundarias específicas del núcleo de Greengrass. Debido a esta dependencia, es necesario actualizar estos componentes al actualizar el núcleo de Greengrass a una nueva versión secundaria. Para obtener información sobre las versiones específicas del núcleo de las que depende cada componente, consulte el tema del componente correspondiente. Para obtener más información sobre la actualización del núcleo, consulteActualice el software AWS IoT Greengrass principal (OTA).

Cuando un componente tiene un tipo de componente genérico y Lambda, la versión actual del componente es del tipo genérico y la versión anterior del componente es del tipo Lambda.

Componente	Descripción	Tipo de componente	Sistema operativo admitido	Código abierto
Autenticación del dispositivo cliente	Permite que los dispositivos IoT locales, denominados dispositivos cliente, se conecten al dispositivo principal.	Complemento	Linux, Windows	Sí
Detector de IP	Envía la información de conectividad	Complemento	Linux, Windows	Sí

Componente	Descripción	Tipo de componente	Sistema operativo admitido	Código abierto
	al intermediario MQTT para AWS IoT Greengrass que los dispositivos cliente puedan descubrir cómo conectarse.			
Puente MQTT	Transmite mensajes MQTT entre los dispositivos cliente, AWS IoT Greengrass publica o suscribe de forma local y. AWS IoT Core	Complemento	Linux, Windows	Sí
Bróker MQTT 3.1.1 (Moquette)	Ejecuta un intermediario MQTT 3.1.1 que gestiona los mensajes entre los dispositivos cliente y el dispositivo principal.	Complemento	Linux, Windows	Sí

Componente	Descripción	Tipo de componente	Sistema operativo admitido	Código abierto
Bróker MQTT 5 (EMQX)	Ejecuta un intermediario MQTT 5 que gestiona los mensajes entre los dispositivos cliente y el dispositivo principal.	Genérico	Linux, Windows	No
Gestor en la sombra	Permite la interacción con las sombras del dispositivo principal. Gestiona el almacenamiento de documentos ocultos y también la sincronización de los estados ocultos locales con el servicio AWS IoT Device Shadow.	Complemento	Linux, Windows	Sí

Connect los dispositivos cliente a los dispositivos principales

Puede configurar la detección en la nube para conectar los dispositivos cliente a los dispositivos principales. Al configurar la detección en la nube, los dispositivos cliente se pueden conectar al servicio AWS IoT Greengrass en la nube para recuperar información sobre los dispositivos principales a los que se pueden conectar. A continuación, los dispositivos cliente pueden intentar conectarse a cada dispositivo principal hasta que se conecten correctamente.

Para usar la detección en la nube, debe hacer lo siguiente:

- Asocie los dispositivos cliente a los dispositivos principales a los que se pueden conectar.
- Especifique los puntos finales del broker MQTT donde los dispositivos cliente se pueden conectar a cada dispositivo principal.
- Implemente componentes en el dispositivo principal que permitan la compatibilidad con los dispositivos cliente.

  También puede implementar componentes opcionales para hacer lo siguiente:

  - Retransmita mensajes entre los dispositivos cliente, los componentes de Greengrass y el servicio AWS IoT Core en la nube.
  - Administre automáticamente los terminales MQTT Broker de los dispositivos principales por usted.
  - Gestione las sombras de los dispositivos clientes locales y sincronice las sombras con el servicio en la nubeAWS IoT Core.

También debe revisar y actualizar la AWS IoT política del dispositivo principal para comprobar que tiene los permisos necesarios para conectar los dispositivos cliente. Para obtener más información, consulte Requisitos.

Después de configurar la detección en la nube, puede probar las comunicaciones entre un dispositivo cliente y un dispositivo principal. Para obtener más información, consulte Pruebe las comunicaciones del dispositivo cliente.

Temas

- Configure la detección en la nube () AWS CLI
- Asociar dispositivos cliente
- Autenticación de clientes sin conexión
- Administre los puntos finales de los dispositivos principales
- Elija un bróker MQTT
- Conexión de dispositivos cliente a un dispositivoAWS IoT Greengrass Core con un broker de MQTT
- Pruebe las comunicaciones del dispositivo cliente
- API RESTful de Greengrass Discovery

Requisitos

Para conectar los dispositivos cliente a un dispositivo principal, debe disponer de lo siguiente:

- El dispositivo principal debe ejecutar Greengrass nucleus v2.2.0 o posterior.
- La función de servicio de Greengrass asociada a usted AWS IoT Greengrass Cuenta de AWS en la AWS región en la que opera el dispositivo principal. Para obtener más información, consulte Configurar el rol de servicio de Greengrass.
- La AWS IoT política del dispositivo principal debe permitir los siguientes permisos:
  - `greengrass:PutCertificateAuthorities`
  - `greengrass:VerifyClientDeviceIdentity`
  - `greengrass:VerifyClientDeviceIoTCertificateAssociation`
  - `greengrass:GetConnectivityInfo`
  - `greengrass:UpdateConnectivityInfo`— (Opcional) Este permiso es necesario para utilizar el componente detector de IP, que envía la información de conectividad de red del dispositivo principal al servicio AWS IoT Greengrass en la nube.
  - `iot:GetThingShadowiot:UpdateThingShadow`, y `iot:DeleteThingShadow` — (opcional) Estos permisos son necesarios para utilizar el componente administrador de sombras con el que se sincronizan las sombras de los dispositivos clienteAWS IoT Core. Esta función requiere Greengrass nucleus v2.6.0 o posterior, shadow manager v2.2.0 o posterior y MQTT bridge v2.2.0 o posterior.

  Para obtener más información, consulte Configure la política de AWS IoT cosas.

> **Note**
>
> Si utilizó la AWS IoT política predeterminada al instalar el software AWS IoT Greengrass Core, el dispositivo principal tiene una AWS IoT política que permite el acceso a todas las acciones (). AWS IoT Greengrass `greengrass:*`

- AWS IoTcosas que se pueden conectar como dispositivos cliente. Para obtener más información, consulte Crear AWS IoT recursos en la Guía para AWS IoT Core desarrolladores.
- El dispositivo cliente debe conectarse mediante un ID de cliente. Un ID de cliente es el nombre de una cosa. No se aceptará ningún otro ID de cliente.
- La AWS IoT política de cada dispositivo cliente debe permitir el `greengrass:Discover` permiso. Para obtener más información, consulte AWS IoTPolítica mínima para los dispositivos cliente.

Temas

- Configurar el rol de servicio de Greengrass
- Configure la política de AWS IoT cosas

Configurar el rol de servicio de Greengrass

El rol de servicio de Greengrass es un rol de servicio de AWS Identity and Access Management (IAM) que autoriza a AWS IoT Greengrass a acceder a recursos de servicios de AWS en su nombre. Esta función permite verificar la identidad de AWS IoT Greengrass los dispositivos cliente y administrar la información de conectividad de los dispositivos principales.

Si no ha configurado anteriormente la función de servicio de Greengrass en esta región, debe asociarle una función de servicio de Greengrass en esta región. AWS IoT Greengrass Cuenta de AWS

Cuando utiliza la página Configurar la detección de dispositivos principales de la AWS IoT Greengrassconsola, AWS IoT Greengrass configura el rol de servicio Greengrass automáticamente. De lo contrario, puede configurarlo manualmente mediante la AWS IoTconsola o la AWS IoT Greengrass API.

En esta sección, comprueba si el rol de servicio de Greengrass está configurado. Si no está configurado, crea un nuevo rol de servicio de Greengrass AWS IoT Greengrass para asociarlo con usted Cuenta de AWS en esta región.

Configurar el rol de servicio de Greengrass (consola)

1. Compruebe si la función de servicio de Greengrass está asociada AWS IoT Greengrass a su función Cuenta de AWS en esta región. Haga lo siguiente:

   a. Vaya a la consola de AWS IoT.

   b. En el panel de navegación, seleccione Configuración.

   c. En la sección Función de servicio de Greengrass, busque Función de servicio actual para ver si hay una función de servicio de Greengrass asociada.

      Si tiene un rol de servicio de Greengrass asociado, cumple con este requisito para usar el componente de detector de IP. Vaya a Configure la política de AWS IoT cosas.

2. Si el rol de servicio de Greengrass no está asociado AWS IoT Greengrass para usted Cuenta de AWS en esta región, cree un rol de servicio de Greengrass y asócielo. Haga lo siguiente:

   a. Vaya a la consola de IAM.

   b. Elija Roles.

   c. Elija Crear rol.

   d. En la página Crear rol, haga lo siguiente:

      i. En Tipo de entidad de confianza, elija Servicio de AWS.

      ii. En Caso de uso, Casos de uso para otros Servicios de AWS, elige Greengrass y selecciona Greengrass. Esta opción especifica agregar una entidad AWS IoT Greengrass de confianza que pueda asumir esta función.

      iii. Elija Siguiente.

      iv. En Políticas de permisos, seleccione la AWSGreengrassResourceAccessRolePolicyque desee adjuntar a la función.

      v. Elija Siguiente.

      vi. En Nombre del rol, introduzca un nombre para el rol, como**`Greengrass_ServiceRole`**.

      vii. Elija Crear rol.

   e. Vaya a la consola de AWS IoT.

   f. En el panel de navegación, seleccione Configuración.

   g. En la sección Función de servicio de Greengrass, elija Adjuntar función.

h. En el modal Actualizar rol de servicio Greengrass, seleccione el rol de IAM que creó y, a continuación, elija Adjuntar rol.

Configurar el rol de servicio de Greengrass () AWS CLI

1. Compruebe si la función de servicio de Greengrass está asociada AWS IoT Greengrass a su función Cuenta de AWS en esta región.

```
aws greengrassv2 get-service-role-for-account
```

Si el rol de servicio de Greengrass está asociado, la operación devuelve una respuesta que contiene información sobre el rol.

Si tiene un rol de servicio de Greengrass asociado, cumple con este requisito para usar el componente de detector de IP. Vaya a Configure la política de AWS IoT cosas.

2. Si el rol de servicio de Greengrass no está asociado AWS IoT Greengrass para usted Cuenta de AWS en esta región, cree un rol de servicio de Greengrass y asócielo. Haga lo siguiente:

a. Cree el rol con una política de confianza que permita a AWS IoT Greengrass asumir el rol. Este ejemplo crea un rol denominado `Greengrass_ServiceRole`, pero puede utilizar un nombre distinto. Le recomendamos que incluya también las claves de contexto de condición global `aws:SourceArn` y `aws:SourceAccount` en su política de confianza para ayudar a prevenir el problema de seguridad del suplente confuso. Las claves de contexto de condición restringen el acceso para permitir solo las solicitudes que provienen de la cuenta especificada y del espacio de trabajo de Greengrass. Para obtener más información sobre el problema del suplente confuso, consulte Prevención del suplente confuso entre servicios.

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. Copie el ARN del rol de los metadatos del rol en la salida. Puede utilizar el ARN para asociar el rol a su cuenta.

c. Asocie la política de `AWSGreengrassResourceAccessRolePolicy` al rol.

```
aws iam attach-role-policy --role-name Greengrass_ServiceRole --policy-arn
 arn:aws:iam::aws:policy/service-role/AWSGreengrassResourceAccessRolePolicy
```

d. Asocie la función de servicio de Greengrass con AWS IoT Greengrass su. Cuenta de AWS Sustituya *`role-arn`* por el ARN del rol de servicio.

```
aws greengrassv2 associate-service-role-to-account --role-arn role-arn
```

La operación devuelve la siguiente respuesta si tiene éxito.

```
{
  "associatedAt": "timestamp"
}
```

Configure la política de AWS IoT cosas

Los dispositivos principales utilizan certificados de dispositivo X.509 para autorizar las conexiones. AWS Debe adjuntar AWS IoT políticas a los certificados de los dispositivos para definir los permisos de un dispositivo principal. Para obtener más información, consulte Políticas de AWS IoT para operaciones de plano de datos y AWS IoTPolítica mínima de compatibilidad con los dispositivos cliente.

Para conectar los dispositivos cliente a un dispositivo principal, la AWS IoT política del dispositivo principal debe permitir los siguientes permisos:

- `greengrass:PutCertificateAuthorities`
- `greengrass:VerifyClientDeviceIdentity`
- `greengrass:VerifyClientDeviceIoTCertificateAssociation`
- `greengrass:GetConnectivityInfo`

- `greengrass:UpdateConnectivityInfo`— (Opcional) Este permiso es necesario para usar el componente detector de IP, que envía la información de conectividad de red del dispositivo principal al servicio AWS IoT Greengrass en la nube.
- `iot:GetThingShadow`iot:UpdateThingShadow, y `iot:DeleteThingShadow` — (opcional) Estos permisos son necesarios para utilizar el componente administrador de sombras con el que se sincronizan las sombras de los dispositivos clienteAWS IoT Core. Esta función requiere Greengrass nucleus v2.6.0 o posterior, shadow manager v2.2.0 o posterior y MQTT bridge v2.2.0 o posterior.

En esta sección, revisará AWS IoT las políticas de su dispositivo principal y agregará los permisos necesarios que falten. Si utilizaste el instalador del software AWS IoT Greengrass principal para aprovisionar recursos, tu dispositivo principal tiene una AWS IoT política que permite el acceso a todas AWS IoT Greengrass las acciones (`greengrass:*`). En este caso, solo debe actualizar la AWS IoT política si planea implementar el componente de administrador de sombras para sincronizar las sombras de los dispositivos con élAWS IoT Core. De lo contrario, puedes saltarte esta sección.

Configura la política de AWS IoT cosas (consola)

1. En el menú de navegación de la AWS IoT Greengrassconsola, selecciona Dispositivos principales.
2. En la página de dispositivos principales, selecciona el dispositivo principal que deseas actualizar.
3. En la página de detalles del dispositivo principal, selecciona el enlace al dispositivo principal. Este enlace abre la página de detalles del dispositivo en la AWS IoT consola.
4. En la página de detalles de la cosa, selecciona Certificados.
5. En la pestaña Certificados, selecciona el certificado activo del objeto.
6. En la página de detalles del certificado, selecciona Políticas.
7. En la pestaña Políticas, elija la AWS IoT política que desee revisar y actualizar. Puede añadir los permisos necesarios a cualquier política que esté asociada al certificado activo del dispositivo principal.

> **Note**
>
> Si usó el instalador de software AWS IoT Greengrass Core para aprovisionar recursos, tiene dos AWS IoT políticas. Le recomendamos que elija la política nombrada GreengrassV2IoTThingPolicy, si existe. Los dispositivos principales que cree con el instalador rápido utilizan este nombre de política de forma predeterminada. Si agrega

permisos a esta política, también los otorga a otros dispositivos principales que usan esta política.

8. En la descripción general de la política, selecciona Editar la versión activa.
9. Revisa la política para ver los permisos necesarios y añade los permisos necesarios que falten.

   - `greengrass:PutCertificateAuthorities`
   - `greengrass:VerifyClientDeviceIdentity`
   - `greengrass:VerifyClientDeviceIoTCertificateAssociation`
   - `greengrass:GetConnectivityInfo`
   - `greengrass:UpdateConnectivityInfo`— (Opcional) Este permiso es necesario para utilizar el componente de detección de IP, que envía la información de conectividad de red del dispositivo principal al servicio AWS IoT Greengrass en la nube.
   - `iot:GetThingShadow`iot:UpdateThingShadow, y `iot:DeleteThingShadow` — (opcional) Estos permisos son necesarios para utilizar el componente administrador de sombras con el que se sincronizan las sombras de los dispositivos clienteAWS IoT Core. Esta función requiere Greengrass nucleus v2.6.0 o posterior, shadow manager v2.2.0 o posterior y MQTT bridge v2.2.0 o posterior.
10. (Opcional) Para permitir que el dispositivo principal se sincronice con las sombrasAWS IoT Core, añada la siguiente declaración a la política. Si planea interactuar con las sombras de los dispositivos cliente, pero no sincronizarlas con ellasAWS IoT Core, omita este paso. Sustituye *`la región`* y el *`identificador de cuenta`* por la región que utilizas y tu Cuenta de AWS número.

    - Esta afirmación de ejemplo permite acceder a todas las sombras de los dispositivos. Para seguir las mejores prácticas de seguridad, puede restringir el acceso únicamente al dispositivo principal y a los dispositivos cliente que conecte al dispositivo principal. Para obtener más información, consulte AWS IoTPolítica mínima de compatibilidad con los dispositivos cliente.

```
{
  "Effect": "Allow",
  "Action": [
    "iot:GetThingShadow",
    "iot:UpdateThingShadow",
    "iot:DeleteThingShadow"
  ],
```

```
  "Resource": [
    "arn:aws:iot:region:account-id:thing/*"
  ]
}
```

Tras añadir esta declaración, el documento de política podría tener un aspecto similar al del siguiente ejemplo.

```
{
  "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. Para establecer una nueva versión de la política como la versión activa, en Estado de la versión de la política, seleccione Establecer la versión editada como la versión activa de esta política.
12. Seleccione Guardar como versión nueva.

Configure la política de AWS IoT cosas (AWS CLI)

1. Enumere los principios del AWS IoT dispositivo principal. Los elementos principales pueden ser certificados de dispositivo X.509 u otros identificadores. Ejecute el siguiente comando y *MyGreengrassCore*sustitúyalo por el nombre del dispositivo principal.

   ```
   aws iot list-thing-principals --thing-name MyGreengrassCore
   ```

   La operación devuelve una respuesta en la que se enumeran los elementos principales del dispositivo principal.

   ```
   {
       "principals": [
           "arn:aws:iot:us-west-2:123456789012:cert/certificateId"
       ]
   }
   ```

2. Identifique el certificado activo del dispositivo principal. Ejecute el siguiente comando y sustituya *CertificateID* por el ID de cada certificado del paso anterior hasta que encuentre el certificado activo. El identificador del certificado es la cadena hexadecimal situada al final del ARN del certificado. El `--query` argumento especifica que solo se muestre el estado del certificado.

   ```
   aws iot describe-certificate --certificate-id certificateId --query
    'certificateDescription.status'
   ```

   La operación devuelve el estado del certificado en forma de cadena. Por ejemplo, si el certificado está activo, se genera esta operación`"ACTIVE"`.

3. Enumere las AWS IoT políticas que se adjuntan al certificado. Ejecute el siguiente comando y sustituya el ARN del certificado por el ARN del certificado.

   ```
   aws iot list-principal-policies --principal arn:aws:iot:us-
   west-2:123456789012:cert/certificateId
   ```

   La operación devuelve una respuesta en la que se enumeran las AWS IoT políticas adjuntas al certificado.

   ```
   {
   ```

```
    "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. Elija la política que desee ver y actualizar.

> **Note**
>
> Si ha utilizado el instalador de software AWS IoT Greengrass principal para aprovisionar recursos, tiene dos AWS IoT políticas. Le recomendamos que elija la política nombrada GreengrassV2IoTThingPolicy, si existe. Los dispositivos principales que cree con el instalador rápido utilizan este nombre de política de forma predeterminada. Si agrega permisos a esta política, también los otorga a otros dispositivos principales que usan esta política.

5. Obtenga el documento de la política. Ejecute el siguiente comando y sustituya *GreengrassV2IoT* por el nombre ThingPolicy de la política.

```
aws iot get-policy --policy-name GreengrassV2IoTThingPolicy
```

La operación devuelve una respuesta que contiene el documento de la política y otra información sobre la política. El documento de política es un objeto JSON serializado como una cadena.

```
{
    "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. Utilice un conversor en línea u otra herramienta para convertir la cadena del documento de política en un objeto JSON y, a continuación, guárdela en un archivo denominado`iot-policy.json`.

   Por ejemplo, si tiene instalada la herramienta jq, puede ejecutar el siguiente comando para obtener el documento de política, convertirlo en un objeto JSON y guardar el documento de política como un objeto JSON.

   ```
   aws iot get-policy --policy-name GreengrassV2IoTThingPolicy --query
    'policyDocument' | jq fromjson >> iot-policy.json
   ```

7. Revise la política para ver los permisos necesarios y añada los permisos necesarios que falten.

   Por ejemplo, en un sistema basado en Linux, puede ejecutar el siguiente comando para usar GNU nano y abrir el archivo.

   ```
   nano iot-policy.json
   ```

   - `greengrass:PutCertificateAuthorities`
   - `greengrass:VerifyClientDeviceIdentity`

- `greengrass:VerifyClientDeviceIoTCertificateAssociation`
- `greengrass:GetConnectivityInfo`
- `greengrass:UpdateConnectivityInfo`— (Opcional) Este permiso es necesario para utilizar el componente detector de IP, que envía la información de conectividad de red del dispositivo principal al servicio en la nube. AWS IoT Greengrass
- `iot:GetThingShadow`, `iot:UpdateThingShadow`, y `iot:DeleteThingShadow` — (opcional) Estos permisos son necesarios para utilizar el componente administrador de sombras con el que se sincronizan las sombras de los dispositivos clienteAWS IoT Core. Esta función requiere Greengrass nucleus v2.6.0 o posterior, shadow manager v2.2.0 o posterior y MQTT bridge v2.2.0 o posterior.

8. Guarde los cambios como una nueva versión de la política. Ejecute el siguiente comando y sustituya *`GreengrassV2IoT`* por el nombre ThingPolicy de la política.

```
aws iot create-policy-version --policy-name GreengrassV2IoTThingPolicy --policy-
document file://iot-policy.json --set-as-default
```

La operación devuelve una respuesta similar a la del siguiente ejemplo si tiene éxito.

```
{
    "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
```

```
}
```

Componentes de Greengrass para soporte de dispositivos cliente

> **Important**
>
> El dispositivo principal debe ejecutar Greengrass nucleus v2.2.0 o posterior para que sea compatible con los dispositivos cliente.

Para permitir que los dispositivos cliente se conecten y se comuniquen con un dispositivo principal, debe implementar los siguientes componentes de Greengrass en el dispositivo principal:

- Autenticación del dispositivo cliente (`aws.greengrass.clientdevices.Auth`)

  Implemente el componente de autenticación del dispositivo cliente para autenticar los dispositivos cliente y autorizar las acciones de los dispositivos cliente. Este componente permite que sus AWS IoT cosas se conecten a un dispositivo principal.

  Este componente requiere alguna configuración para poder usarlo. Debe especificar los grupos de dispositivos cliente y las operaciones que cada grupo está autorizado a realizar, como conectarse y comunicarse a través de MQTT. Para obtener más información, consulte la configuración del componente de autenticación del dispositivo cliente.
- Bróker MQTT 3.1.1 (Moquette) (`aws.greengrass.clientdevices.mqtt.Moquette`)

  Implemente el componente de broker MQTT de Moquette para ejecutar un broker MQTT ligero. El broker MQTT de Moquette es compatible con MQTT 3.1.1 e incluye soporte local para QoS 0, QoS 1, QoS 2, mensajes retenidos, mensajes de última voluntad y suscripciones persistentes.

  No es necesario configurar este componente para usarlo. Sin embargo, puede configurar el puerto en el que este componente opera el broker MQTT. De forma predeterminada, utiliza el puerto 8883.
- Bróker MQTT 5 (EMQX) (`aws.greengrass.clientdevices.mqtt.EMQX`)

> **Note**
>
> Para usar el broker EMQX MQTT 5, debe usar Greengrass nucleus v2.6.0 o posterior y la autenticación del dispositivo cliente v2.2.0 o posterior.

Implemente el componente intermediario MQTT de EMQX para utilizar las funciones de MQTT 5.0 en la comunicación entre los dispositivos cliente y el dispositivo principal. El broker MQTT de EMQX es compatible con MQTT 5.0 e incluye soporte para los intervalos de caducidad de las sesiones y los mensajes, las propiedades de los usuarios, las suscripciones compartidas, los alias de los temas y mucho más.

No es necesario configurar este componente para usarlo. Sin embargo, puede configurar el puerto en el que este componente opera el broker MQTT. De forma predeterminada, utiliza el puerto 8883.

- Puente MQTT (`aws.greengrass.clientdevices.mqtt.Bridge`)

  (Opcional) Implemente el componente de puente MQTT para retransmitir mensajes entre los dispositivos cliente (MQTT local), la publicación/suscripción local y el MQTT. AWS IoT Core Configure este componente para sincronizar los dispositivos cliente AWS IoT Core e interactuar con los dispositivos cliente de los componentes de Greengrass.

  Para poder utilizar este componente, es necesario configurarlo. Debe especificar las asignaciones de temas en las que este componente transmite los mensajes. Para obtener más información, consulte Configuración del componente del puente MQTT.

- Detector de IP (`aws.greengrass.clientdevices.IPDetector`)

  (Opcional) Implemente el componente detector de IP para informar automáticamente al servicio en la nube de los puntos finales del agente MQTT del AWS IoT Greengrass dispositivo principal. No puede usar este componente si tiene una configuración de red compleja, como una en la que un router reenvía el puerto intermediario MQTT al dispositivo principal.

  No es necesario configurar este componente para usarlo.

- Gestor en la sombra (`aws.greengrass.ShadowManager`)

> **Note**
>
> Para gestionar las sombras de los dispositivos cliente, debe utilizar Greengrass nucleus v2.6.0 o posterior, shadow manager v2.2.0 o posterior y MQTT bridge v2.2.0 o posterior.

(Opcional) Implemente el componente administrador de sombras para gestionar las sombras de los dispositivos cliente en el dispositivo principal. Los componentes de Greengrass pueden obtener, actualizar y eliminar las sombras de los dispositivos cliente para interactuar con los dispositivos cliente. También puede configurar el componente administrador de sombras para sincronizar las sombras de los dispositivos cliente con el servicio AWS IoT Core en la nube.

Para utilizar este componente con las sombras de los dispositivos cliente, debe configurar el componente puente MQTT para que retransmita los mensajes entre los dispositivos cliente y el administrador de sombras, que utiliza la función de publicación/suscripción local. De lo contrario, no es necesario configurar este componente para su uso, pero sí para sincronizar las sombras de los dispositivos.

> **Note**
>
> Se recomienda implementar solo un componente de broker de MQTT. Los componentes MQTT bridge y IP detector solo funcionan con un componente intermediario MQTT a la vez. Si despliega varios componentes de broker de MQTT, debe configurarlos para que utilicen puertos diferentes.

Configure la detección en la nube (consola)

Puede usar la AWS IoT Greengrass consola para asociar los dispositivos cliente, administrar los puntos finales de los dispositivos principales e implementar componentes para habilitar la compatibilidad con los dispositivos cliente. Para obtener más información, consulte Paso 2: Habilita la compatibilidad con los dispositivos cliente.

Configure la detección en la nube () AWS CLI

Puede usar AWS Command Line Interface (AWS CLI) para asociar los dispositivos cliente, administrar los puntos finales de los dispositivos principales e implementar componentes para

habilitar la compatibilidad con los dispositivos cliente. Para obtener más información, consulte los siguientes temas:

- Administre las asociaciones de dispositivos cliente () AWS CLI
- Administre los puntos finales de los dispositivos principales
- AWS-componentes de dispositivo cliente proporcionados
- Crear implementaciones

Asociar dispositivos cliente

Para usar la detección en la nube, asocie los dispositivos cliente a un dispositivo principal para que puedan detectar el dispositivo principal. Luego, pueden usar la API de descubrimiento de Greengrass para recuperar la información de conectividad y los certificados de sus dispositivos principales asociados.

Del mismo modo, desasocie los dispositivos cliente de un dispositivo principal para evitar que descubran el dispositivo principal.

Temas

- Administre las asociaciones de dispositivos cliente (consola)
- Administre las asociaciones de dispositivos cliente () AWS CLI
- Administre las asociaciones de dispositivos cliente (API)

Administre las asociaciones de dispositivos cliente (consola)

Puede usar la AWS IoT Greengrass consola para ver, agregar y eliminar asociaciones de dispositivos cliente.

Para ver las asociaciones de dispositivos cliente de un dispositivo principal (consola)

1. Vaya a la consola de AWS IoT Greengrass.
2. Seleccione Dispositivos principales.
3. Elija el dispositivo principal que desee administrar.
4. En la página de detalles del dispositivo principal, elija la pestaña Client devices (Dispositivos cliente).

5. En la sección Dispositivos cliente asociados, puede ver qué dispositivos cliente (AWS IoTcosas) están asociados al dispositivo principal.

Para asociar los dispositivos cliente a un dispositivo principal (consola)

1. Vaya a la consola de AWS IoT Greengrass.
2. Seleccione Dispositivos principales.
3. Elija el dispositivo principal que desee administrar.
4. En la página de detalles del dispositivo principal, elija la pestaña Client devices (Dispositivos cliente).
5. En la sección Associated client devices (Dispositivos cliente asociados), seleccione Associate client devices (Asociar dispositivos cliente).
6. En el modal Associate client devices with core device (Asociar dispositivos cliente a dispositivos principales), haga lo siguiente para cada dispositivo cliente que desee asociar:
   a. Introduzca el nombre del elemento AWS IoT que desee asociar como dispositivo cliente.
   b. Elija Add (Añadir).
7. Elija Associate (Asociar).

   Los dispositivos cliente que asoció ahora pueden usar la API de detección de Greengrass para detectar este dispositivo principal.

Para desasociar los dispositivos cliente de un dispositivo principal (consola)

1. Vaya a la consola de AWS IoT Greengrass.
2. Seleccione Dispositivos principales.
3. Elija el dispositivo principal que desee administrar.
4. En la página de detalles del dispositivo principal, elija la pestaña Client devices (Dispositivos cliente).
5. En la sección Dispositivos cliente asociados, seleccione cada dispositivo cliente que desee desasociar.
6. Elija Disociar.
7. En el modal de confirmación, elija Desasociar.

Los dispositivos cliente que desasoció ya no pueden usar la API de descubrimiento de Greengrass para detectar este dispositivo principal.

Administre las asociaciones de dispositivos cliente () AWS CLI

Puede usar AWS Command Line Interface (AWS CLI) para administrar las asociaciones de dispositivos cliente de un dispositivo principal.

Para ver las asociaciones de dispositivos cliente de un dispositivo principal (AWS CLI)

- Utilice el siguiente comando: list-client-devices-associated- with-core-device.

Para asociar dispositivos cliente a un dispositivo principal (AWS CLI)

- Utilice el siguiente comando: batch-associate-client-device- with-core-device.

Para desasociar los dispositivos cliente de un dispositivo principal () AWS CLI

- Utilice el siguiente comando: batch-disassociate-client-device- from-core-device.

Administre las asociaciones de dispositivos cliente (API)

Puede usar la AWS API para administrar las asociaciones de dispositivos cliente de un dispositivo principal.

Para ver las asociaciones de dispositivos cliente de un dispositivo principal (AWSAPI)

- Utilice la siguiente operación: ListClientDevicesAssociatedWithCoreDevice.

Para asociar dispositivos cliente a un dispositivo principal (AWSAPI)

- Utilice la siguiente operación: BatchAssociateClientDeviceWithCoreDevice.

Para desasociar los dispositivos cliente de un dispositivo principal (AWSAPI)

- Utilice la siguiente operación: BatchDisassociateClientDeviceFromCoreDevice.

Autenticación de clientes sin conexión

Con la autenticación sin conexión, puede configurar su dispositivo AWS IoT Greengrass principal para que los dispositivos cliente puedan conectarse a un dispositivo principal, incluso cuando el dispositivo principal no esté conectado a la nube. Cuando utiliza la autenticación sin conexión, sus dispositivos Greengrass pueden seguir funcionando en un entorno parcialmente desconectado.

Para utilizar la autenticación sin conexión en un dispositivo cliente con conexión a la nube, necesita lo siguiente:

- Un dispositivo AWS IoT Greengrass Core con el Autenticación del dispositivo cliente componente implementado. Debe usar la versión 2.3.0 o superior para la autenticación sin conexión.
- Una conexión a la nube para el dispositivo principal durante la conexión inicial de los dispositivos cliente.

Almacenamiento de las credenciales del cliente

Cuando un dispositivo cliente se conecta a un dispositivo principal por primera vez, el dispositivo principal llama al AWS IoT Greengrass servicio. Cuando se llama, Greengrass valida el registro del dispositivo cliente como una cosa. AWS IoT También valida que el dispositivo tenga un certificado válido. A continuación, el dispositivo principal almacena esta información de forma local.

La próxima vez que el dispositivo se conecte, el dispositivo principal de Greengrass intentará validar el dispositivo cliente con el AWS IoT Greengrass servicio. Si no se puede conectar AWS IoT Greengrass, el dispositivo principal utilizará la información del dispositivo almacenada localmente para validar el dispositivo cliente.

Puede configurar el tiempo durante el que el dispositivo principal de Greengrass almacena las credenciales. Puede establecer el tiempo de espera de un minuto a 2.147.483.647 minutos configurando la opción de configuración en la `clientDeviceTrustDurationMinutes` configuración del componente de autenticación del dispositivo cliente. El valor predeterminado es un minuto, lo que desactiva de forma efectiva la autenticación sin conexión. Al establecer este tiempo de espera, le recomendamos que tenga en cuenta sus necesidades de seguridad. También debes tener en cuenta cuánto tiempo esperas que funcionen los dispositivos principales mientras estén desconectados de la nube.

El dispositivo principal actualiza su almacenamiento de credenciales tres veces:

1. Cuando un dispositivo se conecta al dispositivo principal por primera vez.

2. Si el dispositivo principal está conectado a la nube, cuando un dispositivo cliente se vuelve a conectar al dispositivo principal.
3. Si el dispositivo principal está conectado a la nube, una vez al día para actualizar todo el almacén de credenciales.

Cuando el dispositivo principal de Greengrass actualiza su almacén de credenciales, utiliza la operación. ListClientDevicesAssociatedWithCoreDevice Greengrass solo actualiza los dispositivos devueltos por esta operación. Para asociar un dispositivo cliente a un dispositivo principal, consulte. Asociar dispositivos cliente

Para utilizar la `ListClientDevicesAssociatedWithCoreDevice` operación, debe añadir el permiso para la operación al rol AWS Identity and Access Management (IAM) asociado al Cuenta de AWS que se ejecuta AWS IoT Greengrass. Para obtener más información, consulte Autorizar a los dispositivos principales a interactuar con AWS los servicios.

Administre los puntos finales de los dispositivos principales

Al utilizar la detección en la nube, se almacenan los puntos finales del broker MQTT para los dispositivos principales en el servicio en la nube. AWS IoT Greengrass Los dispositivos cliente se conectan AWS IoT Greengrass para recuperar estos puntos finales y otra información para sus dispositivos principales asociados.

Para cada dispositivo principal, puede administrar los puntos finales de forma automática o manual.

- Administre automáticamente los puntos finales con un detector de IP

  Puede implementar el componente detector de IP para administrar automáticamente los puntos finales de los dispositivos principales si tiene una configuración de red no compleja, por ejemplo, si los dispositivos cliente están en la misma red que el dispositivo principal. No puede utilizar el componente detector de IP si el dispositivo principal está detrás de un router que reenvía el puerto intermediario MQTT al dispositivo principal, por ejemplo.

  El componente detector de IP también es útil si se implementa en grupos de cosas, ya que administra los puntos finales de todos los dispositivos principales del grupo de cosas. Para obtener más información, consulte Utilice el detector de IP para gestionar automáticamente los puntos finales.
- Administre manualmente los puntos finales

Si no puede utilizar el componente detector de IP, debe administrar manualmente los puntos finales de los dispositivos principales. Puede actualizar estos puntos finales con la consola o la API. Para obtener más información, consulte Administre manualmente los puntos finales.

Temas

Utilice el detector de IP para gestionar automáticamente los puntos finales

Si tiene una configuración de red sencilla, como los dispositivos cliente en la misma red que el dispositivo principal, puede implementar el componente detector de IP para hacer lo siguiente:

- Supervise la información de conectividad de red local del dispositivo principal de Greengrass. Esta información incluye los puntos finales de la red del dispositivo principal y el puerto en el que opera el intermediario MQTT.
- Informe la información de conectividad del dispositivo principal al servicio en la AWS IoT Greengrass nube.

El componente del detector de IP sobrescribe los puntos finales que se configuran manualmente.

> **Important**
>
> La AWS IoT política del dispositivo principal debe permitir el `greengrass:UpdateConnectivityInfo` permiso para usar el componente detector de IP. Para obtener más información, consulte Políticas de AWS IoT para operaciones de plano de datos y Configure la política de AWS IoT cosas.

Puede realizar una de las siguientes acciones para implementar el componente del detector de IP:

- Utilice la página Configurar la detección de la consola. Para obtener más información, consulte Configure la detección en la nube (consola).
- Cree y revise las implementaciones para incluir el detector de IP. Puede usar la consola o la AWS CLI AWS API para administrar las implementaciones. Para obtener más información, consulte Crear implementaciones.

Implemente el componente de detección de IP (consola)

1. En el menú de navegación de la AWS IoT Greengrassconsola, elija Componentes.
2. En la página Componentes, seleccione la pestaña Componentes públicos y, a continuación, elija aws.greengrass.clientdevices.IPDetector.
3. En la página aws.greengrass.clientdevices.IPDetector, elija Implementar.
4. En Añadir a la implementación, elija una implementación existente para revisarla o cree una nueva y, a continuación, elija Siguiente.
5. Si opta por crear una nueva implementación, elija el dispositivo principal o el grupo de cosas de destino para la implementación. En la página Especificar el destino, en Destino del despliegue, elija un dispositivo principal o un grupo de cosas y, a continuación, elija Siguiente.
6. En la página Seleccionar componentes, compruebe que el aws.greengrass.clientdevices.IPDetectorcomponente esté seleccionado y seleccione Siguiente.
7. En la página Configurar componentes, seleccione y aws.greengrass.clientdevices.IPDetector, a continuación, haga lo siguiente:

   a. Seleccione Configurar componente.

   b. En el aws.greengrass.clientdevices.IPDetector modo Configurar, en Actualización de configuración, en Configuración para fusionar, puede introducir una actualización de configuración para configurar el componente del detector de IP. Puede especificar cualquiera de las siguientes opciones de configuración:

      - `defaultPort`— (Opcional) El puerto del broker MQTT para informar cuando este componente detecta direcciones IP. Debe especificar este parámetro si configura el broker MQTT para que utilice un puerto diferente al puerto predeterminado 8883.
      - `includeIPv4LoopbackAddrs`— (Opcional) Puede activar esta opción para detectar e informar sobre las direcciones de bucle invertido de IPv4. Se trata de direcciones IP, por ejemplo`localhost`, donde un dispositivo puede comunicarse consigo mismo. Utilice esta opción en entornos de prueba en los que el dispositivo principal y el dispositivo cliente se ejecuten en el mismo sistema.
      - `includeIPv4LinkLocalAddrs`— (Opcional) Puede habilitar esta opción para detectar e informar sobre las direcciones IPv4 de enlace local. Utilice esta opción si la red del dispositivo principal no tiene el Protocolo de configuración dinámica de host (DHCP) ni direcciones IP asignadas de forma estática.

La actualización de la configuración puede tener un aspecto similar al del siguiente ejemplo.

```
{
  "defaultPort": "8883",
  "includeIPv4LoopbackAddrs": false,
  "includeIPv4LinkLocalAddrs": false
}
```

c. Elija Confirmar para cerrar el modal y, a continuación, elija Siguiente.

8. En la página Configurar ajustes avanzados, mantenga los ajustes de configuración predeterminados y seleccione Siguiente.
9. En la página Revisar, elija Implementar.

   La implementación puede tardar hasta un minuto en completarse.

Implemente el componente detector de IP (AWS CLI)

Para implementar el componente detector de IP, cree un documento de despliegue que lo incluya `aws.greengrass.clientdevices.IPDetector` en el `components` objeto y especifique la actualización de configuración del componente. Siga las instrucciones Crear implementaciones para crear una nueva implementación o revisar una implementación existente.

Puede especificar cualquiera de las siguientes opciones para configurar el componente del detector de IP al crear el documento de despliegue:

- `defaultPort`— (Opcional) El puerto del broker MQTT para informar cuando este componente detecta direcciones IP. Debe especificar este parámetro si configura el broker MQTT para que utilice un puerto diferente al puerto predeterminado 8883.
- `includeIPv4LoopbackAddrs`— (Opcional) Puede activar esta opción para detectar e informar sobre las direcciones de bucle invertido de IPv4. Se trata de direcciones IP, por ejemplo`localhost`, donde un dispositivo puede comunicarse consigo mismo. Utilice esta opción en entornos de prueba en los que el dispositivo principal y el dispositivo cliente se ejecuten en el mismo sistema.
- `includeIPv4LinkLocalAddrs`— (Opcional) Puede habilitar esta opción para detectar e informar sobre las direcciones IPv4 de enlace local. Utilice esta opción si la red del dispositivo principal no tiene el Protocolo de configuración dinámica de host (DHCP) ni direcciones IP asignadas de forma estática.

En el siguiente ejemplo de documento de despliegue parcial, se especifica que el puerto 8883 es el puerto intermediario de MQTT.

```
{
  ...,
  "components": {
    ...,
    "aws.greengrass.clientdevices.IPDetector": {
      "componentVersion": "2.1.1",
      "configurationUpdate": {
        "merge": "{\"defaultPort\":\"8883\",}"
      }
    }
  }
}
```

Administre manualmente los puntos finales

Puede administrar manualmente los puntos finales de MQTT Broker para los dispositivos principales.

Cada punto final del broker MQTT tiene la siguiente información:

Punto final () `HostAddress`

Una dirección IP o una dirección DNS donde los dispositivos cliente pueden conectarse a un intermediario MQTT en el dispositivo principal.

Port (`PortNumber`)

El puerto en el que opera el broker MQTT en el dispositivo principal.

Puede configurar este puerto en el componente broker MQTT de Moquette, que utiliza de forma predeterminada el puerto 8883.

Metadatos () `Metadata`

Metadatos adicionales para proporcionarlos a los dispositivos cliente que se conectan a este punto final.

Temas

- Administre los puntos finales (consola)
- Gestione los puntos finales () AWS CLI

- Gestione los puntos finales (API)

Administre los puntos finales (consola)

Puede usar la AWS IoT Greengrass consola para ver, actualizar y eliminar los puntos finales de un dispositivo principal.

Para administrar los puntos finales de un dispositivo principal (consola)

1. Vaya a la consola de AWS IoT Greengrass.
2. Elija dispositivos principales.
3. Elija el dispositivo principal que desee administrar.
4. En la página de detalles del dispositivo principal, elija la pestaña Client devices (Dispositivos cliente).
5. En la sección de puntos finales del corredor MQTT, puede ver los puntos finales del corredor MQTT del dispositivo principal. Seleccione Administrar puntos de conexión.
6. En el modal Administrar puntos de conexión, añada o elimine los puntos de conexión de intermediación MQTT para el dispositivo principal.
7. Seleccione Actualizar.

Gestione los puntos finales () AWS CLI

Puede usar AWS Command Line Interface (AWS CLI) para administrar los puntos finales de un dispositivo principal.

> **Note**
>
> Como la compatibilidad con dispositivos cliente AWS IoT Greengrass V2 es retrocompatibleAWS IoT Greengrass V1, puedes utilizar AWS IoT Greengrass V2 nuestras operaciones de AWS IoT Greengrass V1 API para gestionar los puntos finales principales de los dispositivos.

Para obtener puntos finales para un dispositivo principal () AWS CLI

- Utilice uno de los siguientes comandos:
  - greengrass v2: get-connectivity-info

- hierba verde: get-connectivity-info

Para actualizar los puntos finales de un dispositivo principal () AWS CLI

- Utilice uno de los siguientes comandos:
  - greengrass v2: update-connectivity-info
  - hierba verde: update-connectivity-info

Gestione los puntos finales (API)

Puede usar la AWS API para administrar los puntos finales de un dispositivo principal.

> **Note**
>
> Como la compatibilidad con dispositivos cliente AWS IoT Greengrass V2 es retrocompatibleAWS IoT Greengrass V1, puedes utilizar AWS IoT Greengrass V2 nuestras operaciones de AWS IoT Greengrass V1 API para gestionar los puntos finales principales de los dispositivos.

Para obtener puntos finales para un dispositivo principal (API) AWS

- Utilice una de las siguientes operaciones:
  - V2: GetConnectivityInfo
  - V1: GetConnectivityInfo

Para actualizar los puntos finales de un dispositivo principal (AWSAPI)

- Utilice una de las siguientes operaciones:
  - V2: UpdateConnectivityInfo
  - V1: UpdateConnectivityInfo

Elija un bróker MQTT

AWS IoT Greengrassofrece opciones para que pueda elegir qué broker MQTT local desea ejecutar en sus dispositivos principales. Los dispositivos cliente se conectan al agente MQTT que se ejecuta

en un dispositivo principal, así que elija un agente MQTT que sea compatible con los dispositivos cliente a los que desee conectarse.

> **Note**
>
> Se recomienda implementar solo un componente de broker de MQTT. Los componentes MQTT bridge y IP detector solo funcionan con un componente intermediario MQTT a la vez. Si despliega varios componentes de broker de MQTT, debe configurarlos para que utilicen puertos diferentes.

Puede elegir entre los siguientes corredores de MQTT:

- Bróker MQTT 3.1.1 (Moquette) — `aws.greengrass.clientdevices.mqtt.Moquette`

  Elija esta opción si busca un bróker MQTT ligero que cumpla con el estándar MQTT 3.1.1. El broker AWS IoT Core MQTT también es compatible con el estándar MQTT 3.1.1, por lo que puede utilizar estas funciones para crear una aplicación que utilice MQTT 3.1.1 en sus dispositivos y en el. SDK para dispositivos con AWS IoT Nube de AWS

- Broker MQTT 5 (EMQX): `aws.greengrass.clientdevices.mqtt.EMQX`

  Elija esta opción para utilizar las funciones de MQTT 5 en la comunicación entre los dispositivos principales y los dispositivos cliente. Este componente utiliza más recursos que el broker MQTT 3.1.1 de Moquette y, en los dispositivos principales de Linux, requiere Docker.

  MQTT 5 es compatible con versiones anteriores de MQTT 3.1.1, por lo que puede conectar dispositivos cliente que utilicen MQTT 3.1.1 a este broker. Si utiliza el broker MQTT 3.1.1 de Moquette, puede sustituirlo por el broker MQTT 5 de EMQX y los dispositivos cliente podrán seguir conectándose y funcionando como de costumbre.

- Implemente un bróker personalizado

  Elija esta opción para crear un componente de intermediario local personalizado para comunicarse con los dispositivos del cliente. Puede crear un intermediario local personalizado que utilice un protocolo distinto de MQTT. AWS IoT Greengrassproporciona un SDK de componentes que puede utilizar para autenticar y autorizar los dispositivos cliente. Para obtener más información, consulte Úselo SDK para dispositivos con AWS IoT para comunicarse con el núcleo de Greengrass, otros componentes y AWS IoT Core y Autenticar y autorizar los dispositivos cliente.

Conexión de dispositivos cliente a un dispositivoAWS IoT Greengrass Core con un broker de MQTT

Cuando utilizas un intermediario de MQTT en tu dispositivoAWS IoT Greengrass Core, el dispositivo utiliza una autoridad certificadora (CA) del dispositivo principal exclusiva del dispositivo para emitir un certificado al intermediario para establecer conexiones TLS mutuas con los clientes.

AWS IoT Greengrassgenerará automáticamente una CA de dispositivo principal, o puede utilizar sus propias CA propias CA. La CA del dispositivo principal se registraAWS IoT Greengrass cuando se conecta elAutenticación del dispositivo cliente componente. La CA del dispositivo principal generada automáticamente es persistente; el dispositivo seguirá utilizando la misma CA mientras esté configurado el componente de autenticación del dispositivo cliente.

Cuando se inicia el bróker MQTT, solicita un certificado. El componente de autenticación del dispositivo cliente emite un certificado X.509 mediante la CA del dispositivo principal. El certificado se rota cuando se inicia el corredor, cuando caduca o cuando cambia la información de conectividad, como la dirección IP. Para obtener más información, consulte Rotación de certificados en el broker MQTT local.

Para conectar un cliente al bróker MQTT, necesita lo siguiente:

- El dispositivo cliente debe tener la CA del dispositivoAWS IoT Greengrass principal. Puede obtener esta CA mediante la detección en la nube o proporcionándola manualmente. Para obtener más información, consulte Uso de su propia autoridad de certificación.
- El nombre de dominio completo (FQDN) o la dirección IP del dispositivo principal deben estar presentes en el certificado de intermediario emitido por la CA del dispositivo principal. Asegúrese de ello mediante elDetector de IP componente o configurando manualmente la dirección IP. Para obtener más información, consulte Administre los puntos finales de los dispositivos principales.
- El componente de autenticación del dispositivo cliente debe conceder permiso al dispositivo cliente para conectarse al dispositivo principal de Greengrass. Para obtener más información, consulte Autenticación del dispositivo cliente.

Uso de su propia autoridad de certificación

Si sus dispositivos cliente no pueden acceder a la nube para descubrir su dispositivo principal, puede proporcionar una autoridad certificadora (CA) del dispositivo principal. Su dispositivo principal de Greengrass utiliza la CA del dispositivo principal para emitir certificados para su agente

de MQTT. Una vez que haya configurado el dispositivo principal y haya suministrado su CA al dispositivo cliente, los dispositivos cliente podrán conectarse al punto final y verificar el protocolo de enlace de TLS mediante la CA del dispositivo principal (CA proporcionada por usted o generada automáticamente).

Para configurar elAutenticación del dispositivo cliente componente para que utilice la CA de su dispositivo principal, defina el parámetro `certificateAuthority` configuración al implementar el componente. Debe proporcionar los siguientes detalles durante la configuración:

- La ubicación del certificado de CA de un dispositivo principal.
- La clave privada del certificado CA del dispositivo principal.
- (Opcional) La cadena de certificados al certificado raíz si la CA del dispositivo principal es una CA intermedia.

Si proporciona una CA del dispositivo principal,AWS IoT Greengrass registra la CA en la nube.

Puede almacenar los certificados en un módulo de seguridad de hardware o en el sistema de archivos. El siguiente ejemplo muestra una`certificateAuthority` configuración para una CA intermedia almacenada mediante HSM/TPM. Tenga en cuenta que la cadena de certificados solo se puede almacenar en el disco.

```
"certificateAuthority": {
    "certificateUri": "pkcs11:object=CustomerIntermediateCA;type=cert",
    "privateKeyUri": "pkcs11:object=CustomerIntermediateCA;type=private"
    "certificateChainUri": "file:///home/ec2-user/creds/certificateChain.pem",
  }
```

En este ejemplo, el parámetro `certificateAuthority` configuración configura el componente de autenticación del dispositivo cliente para utilizar una CA intermedia del sistema de archivos:

```
"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",
  }
```

Para conectar los dispositivos a su dispositivoAWS IoT Greengrass Core, haga lo siguiente:

1. Entidad de certificación (CA) intermedia para el dispositivo principal de Greengrass utilizando la entidad de certificación (CA) de la organización. Le recomendamos que utilice una CA intermedia como práctica recomendada desde el punto de vista de vista de vista de la seguridad.
2. Proporcione el certificado de CA intermedio, la clave privada y la cadena de certificados de la CA raíz al dispositivo principal de Greengrass. Para obtener más información, consulte Autenticación del dispositivo cliente. La CA intermedia se convierte en la CA del dispositivo central del dispositivo central de Greengrass y el dispositivo registra la CA conAWS IoT Greengrass.
3. Registre el dispositivo cliente como unaAWS IoT cosa. Para obtener más información, consulta Crear un objeto en la Guía paraAWS IoT Core desarrolladores. Agregue la clave privada, la clave pública, el certificado de dispositivo y el certificado de CA raíz a su dispositivo cliente. La forma de añadir la información depende del dispositivo y del software.

Una vez configurado el dispositivo, puede utilizar el certificado y la cadena de claves públicas para conectarse al dispositivo principal de Greengrass. Su software es responsable de encontrar los puntos finales principales del dispositivo. Puede configurar el punto final manualmente para el dispositivo principal. Para obtener más información, consulte Administre manualmente los puntos finales.

Pruebe las comunicaciones del dispositivo cliente

Los dispositivos cliente pueden SDK para dispositivos con AWS IoT utilizarla para detectar, conectarse y comunicarse con un dispositivo principal. Puede usar el cliente de descubrimiento de Greengrass SDK para dispositivos con AWS IoT para usar la API de descubrimiento de Greengrass, que devuelve información sobre los dispositivos principales a los que se puede conectar un dispositivo cliente. La respuesta de la API incluye los puntos finales del broker MQTT para conectarse y los certificados que se utilizan para verificar la identidad de cada dispositivo principal. A continuación, el dispositivo cliente puede probar cada punto final hasta que se conecte correctamente a un dispositivo principal.

Los dispositivos cliente solo pueden detectar los dispositivos principales a los que se asocien. Antes de probar las comunicaciones entre un dispositivo cliente y un dispositivo principal, debe asociar el dispositivo cliente al dispositivo principal. Para obtener más información, consulte Asociar dispositivos cliente.

La API de descubrimiento de Greengrass devuelve los puntos finales del broker MQTT del dispositivo principal que especifique. Puede usar el componente detector de IP para administrar estos puntos

finales por usted, o puede administrarlos manualmente para cada dispositivo principal. Para obtener más información, consulte Administre los puntos finales de los dispositivos principales.

> **Note**
>
> Para usar la API de descubrimiento de Greengrass, un dispositivo cliente debe tener el `greengrass:Discover` permiso. Para obtener más información, consulte AWS IoTPolítica mínima para los dispositivos cliente.

SDK para dispositivos con AWS IoTEstá disponible en varios lenguajes de programación. Para obtener más información, consulte los SDKs de dispositivos de AWS IoT en la Guía para desarrolladores de AWS IoT Core.

Temas

- Comunicaciones de prueba (Python)
- Pruebe las comunicaciones (C++)
- Probar las comunicaciones () JavaScript
- Pruebe las comunicaciones (Java)

Comunicaciones de prueba (Python)

En esta sección, utilizará un ejemplo de descubrimiento de Greengrass en la SDK para dispositivos con AWS IoTversión 2 para Python para probar las comunicaciones entre un dispositivo cliente y un dispositivo principal.

> **Important**
>
> Para usar la SDK para dispositivos con AWS IoT versión 2 para Python, el dispositivo debe ejecutar Python 3.6 o una versión posterior.

Para probar las comunicaciones (SDK para dispositivos con AWS IoTv2 para Python)

1. Descargue e instale la SDK para dispositivos con AWS IoTversión 2 para Python en el AWS IoT dispositivo para conectarlo como dispositivo cliente.

   En el dispositivo cliente, haga lo siguiente:

a. Clona el repositorio SDK para dispositivos con AWS IoT v2 para Python para descargarlo.

```
git clone https://github.com/aws/aws-iot-device-sdk-python-v2.git
```

b. Instale la SDK para dispositivos con AWS IoT versión 2 para Python.

```
python3 -m pip install --user ./aws-iot-device-sdk-python-v2
```

2. Cambie a la carpeta de muestras de la SDK para dispositivos con AWS IoT versión 2 para Python.

```
cd aws-iot-device-sdk-python-v2/samples
```

3. Ejecute la aplicación Greengrass Discovery de muestra. Esta aplicación espera argumentos que especifiquen el nombre del dispositivo cliente, el tema y el mensaje de MQTT que se van a utilizar y los certificados que autentican y protegen la conexión. En el siguiente ejemplo, se envía un mensaje de Hello World al `clients/`*`MyClientDevice1`*`/hello/world` tema.

   - Sustituya *`MyClientDevice1`* por el nombre del dispositivo cliente.
   - Sustituya *`~/certs/ AmazonRoot CA1.pem`* por la ruta al certificado de CA raíz de Amazon en el dispositivo cliente.
   - Sustituya *`~/certs/device.pem.crt por la ruta al certificado del dispositivo cliente`*.
   - Sustituya *`~/certs/private.pem.key`* por la ruta al archivo de clave privada del dispositivo cliente.
   - Sustituya *`us-east-1`* por la región en la que funcionan el dispositivo cliente y el dispositivo principal. AWS

```
python3 basic_discovery.py \\
  --thing_name MyClientDevice1 \\
  --topic 'clients/MyClientDevice1/hello/world' \\
  --message 'Hello World!' \\
  --ca_file ~/certs/AmazonRootCA1.pem \\
  --cert ~/certs/device.pem.crt \\
  --key ~/certs/private.pem.key \\
  --region us-east-1 \\
  --verbosity Warn
```

La aplicación de ejemplo Discovery envía el mensaje 10 veces y se desconecta. También se suscribe al mismo tema en el que publica los mensajes. Si el resultado indica que la aplicación recibió mensajes MQTT sobre el tema, el dispositivo cliente puede comunicarse correctamente con el dispositivo principal.

```
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}'
```

Si, en cambio, la aplicación genera un error, consulte Solución de problemas de detección de Greengrass.

También puede ver los registros de Greengrass en el dispositivo principal para comprobar si el dispositivo cliente se conecta y envía mensajes correctamente. Para obtener más información, consulte Supervisar AWS IoT Greengrass registros.

Pruebe las comunicaciones (C++)

En esta sección, utilizará un ejemplo de descubrimiento de Greengrass en la SDK para dispositivos con AWS IoTversión 2 para C++ para probar las comunicaciones entre un dispositivo cliente y un dispositivo principal.

Para compilar la SDK para dispositivos con AWS IoT versión 2 para C++, un dispositivo debe tener las siguientes herramientas:

- C++ 1.1 o posterior
- CMake 3.1 o posterior
- Uno de los siguientes compiladores:
  - GCC 4.8 o posterior
  - Clang 3.9 o posterior
  - MSVC 2015 o posterior

Para probar las comunicaciones (SDK para dispositivos con AWS IoTv2 para C++)

1. Descargue y cree la SDK para dispositivos con AWS IoTversión 2 para C++ en AWS IoT el dispositivo que desee conectar como dispositivo cliente.

   En el dispositivo cliente, haga lo siguiente:

   a. Cree una carpeta para el espacio de trabajo SDK para dispositivos con AWS IoT de la versión 2 para C++ y cámbiela a ella.

   ```
   cd
   mkdir iot-device-sdk-cpp
   cd iot-device-sdk-cpp
   ```

   b. Clona el SDK para dispositivos con AWS IoT repositorio de la versión 2 para C++ para descargarlo. El `--recursive` indicador especifica la descarga de submódulos.

```
git clone --recursive https://github.com/aws/aws-iot-device-sdk-cpp-v2.git
```

c. Cree una carpeta para el resultado de la compilación SDK para dispositivos con AWS IoT de la versión 2 para C++ y cámbiela a ella.

```
mkdir aws-iot-device-sdk-cpp-v2-build
cd aws-iot-device-sdk-cpp-v2-build
```

d. Compila la SDK para dispositivos con AWS IoT v2 para C++.

```
cmake -DCMAKE_INSTALL_PREFIX="~/iot-device-sdk-cpp" -
DCMAKE_BUILD_TYPE="Release" ../aws-iot-device-sdk-cpp-v2
cmake --build . --target install
```

2. Cree la aplicación de ejemplo Greengrass Discovery en la SDK para dispositivos con AWS IoT versión 2 para C++. Haga lo siguiente:

a. Cambie a la carpeta de ejemplos de Greengrass Discovery en la SDK para dispositivos con AWS IoT versión 2 para C++.

```
cd ../aws-iot-device-sdk-cpp-v2/samples/greengrass/basic_discovery
```

b. Cree una carpeta para el resultado de la compilación de ejemplo de Greengrass Discovery y cámbiese a ella.

```
mkdir build
cd build
```

c. Cree la aplicación de ejemplo Greengrass Discovery.

```
cmake -DCMAKE_PREFIX_PATH="~/iot-device-sdk-cpp" -
DCMAKE_BUILD_TYPE="Release" ..
cmake --build . --config "Release"
```

3. Ejecute la aplicación Greengrass Discovery de muestra. Esta aplicación espera argumentos que especifiquen el nombre del dispositivo cliente, el tema de MQTT que se va a utilizar y los certificados que autentican y protegen la conexión. En el siguiente ejemplo, se suscribe al `clients/`*`MyClientDevice1`*`/hello/world` tema y se publica un mensaje sobre el mismo tema que se introduce en la línea de comandos.

- Sustituya *MyClientDevice1* por el nombre de la cosa del dispositivo cliente.
- Sustituya *~/certs/ AmazonRoot CA1.pem* por la ruta al certificado de CA raíz de Amazon en el dispositivo cliente.
- Sustituya *~/certs/device.pem.crt por la ruta al certificado del dispositivo cliente*.
- Sustituya *~/certs/private.pem.key* por la ruta al archivo de clave privada del dispositivo cliente.
- Sustituya *us-east-1* por la región en la que funcionan el dispositivo cliente y el dispositivo principal. AWS

```
./basic-discovery \
  --thing_name MyClientDevice1 \
  --topic 'clients/MyClientDevice1/hello/world' \
  --ca_file ~/certs/AmazonRootCA1.pem \
  --cert ~/certs/device.pem.crt \
  --key ~/certs/private.pem.key \
  --region us-east-1
```

La aplicación de ejemplo Discovery se suscribe al tema y le pide que introduzca un mensaje para publicarlo.

```
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.
```

Si, en cambio, la aplicación genera un error, consulte Solución de problemas de detección de Greengrass.

4. Introduzca un mensaje, por ejemplo. **`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!
```

Si el resultado indica que la aplicación recibió el mensaje MQTT sobre el tema, el dispositivo cliente puede comunicarse correctamente con el dispositivo principal.

```
Operation on packetId 2 Succeeded
Publish received on topic clients/MyClientDevice1/hello/world
Message:
Hello World!
```

También puede ver los registros de Greengrass en el dispositivo principal para comprobar si el dispositivo cliente se conecta y envía mensajes correctamente. Para obtener más información, consulte Supervisar AWS IoT Greengrass registros.

Probar las comunicaciones () JavaScript

En esta sección, utilizará un ejemplo de descubrimiento de Greengrass en la SDK para dispositivos con AWS IoTversión 2 JavaScript para probar las comunicaciones entre un dispositivo cliente y un dispositivo principal.

> **Important**
>
> Para usar la SDK para dispositivos con AWS IoT versión 2 JavaScript, un dispositivo debe ejecutar Node v10.0 o una versión posterior.

Para probar las comunicaciones (SDK para dispositivos con AWS IoTv2 para) JavaScript

1. Descargue e instale la SDK para dispositivos con AWS IoTversión 2 JavaScript para el AWS IoT dispositivo que desee conectar como dispositivo cliente.

   En el dispositivo cliente, haga lo siguiente:

   a. Clona la SDK para dispositivos con AWS IoT versión 2 JavaScript del repositorio para descargarla.

   ```
   git clone https://github.com/aws/aws-iot-device-sdk-js-v2.git
   ```

   b. Instale la SDK para dispositivos con AWS IoT v2 para JavaScript.

```
cd aws-iot-device-sdk-js-v2
npm install
```

2. Cambie a la carpeta de ejemplos de Greengrass discovery en la SDK para dispositivos con AWS IoT versión 2 para. JavaScript

```
cd samples/node/basic_discovery
```

3. Instale la aplicación de ejemplo Greengrass Discovery.

```
npm install
```

4. Ejecute la aplicación Greengrass Discovery de muestra. Esta aplicación espera argumentos que especifiquen el nombre del dispositivo cliente, el tema y el mensaje de MQTT que se van a utilizar y los certificados que autentican y protegen la conexión. En el siguiente ejemplo, se envía un mensaje de Hello World al `clients/`*`MyClientDevice1`*`/hello/world` tema.

   - Sustituya *`MyClientDevice1`* por el nombre del dispositivo cliente.
   - Sustituya *`~/certs/ AmazonRoot CA1.pem`* por la ruta al certificado de CA raíz de Amazon en el dispositivo cliente.
   - Sustituya *`~/certs/device.pem.crt por la ruta al certificado del dispositivo cliente`*.
   - Sustituya *`~/certs/private.pem.key`* por la ruta al archivo de clave privada del dispositivo cliente.
   - Sustituya *`us-east-1`* por la región en la que funcionan el dispositivo cliente y el dispositivo principal. AWS

```
node dist/index.js \
  --thing_name MyClientDevice1 \
  --topic 'clients/MyClientDevice1/hello/world' \
  --message 'Hello World!' \
  --ca_file ~/certs/AmazonRootCA1.pem \
  --cert ~/certs/device.pem.crt \
  --key ~/certs/private.pem.key \
  --region us-east-1 \
  --verbose warn
```

La aplicación de ejemplo Discovery envía el mensaje 10 veces y se desconecta. También se suscribe al mismo tema en el que publica los mensajes. Si el resultado indica que la aplicación recibió mensajes MQTT sobre el tema, el dispositivo cliente puede comunicarse correctamente con el dispositivo principal.

```
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!
```

Si, en cambio, la aplicación genera un error, consulte Solución de problemas de detección de Greengrass.

También puede ver los registros de Greengrass en el dispositivo principal para comprobar si el dispositivo cliente se conecta y envía mensajes correctamente. Para obtener más información, consulte Supervisar AWS IoT Greengrass registros.

Pruebe las comunicaciones (Java)

En esta sección, utilizará un ejemplo de descubrimiento de Greengrass en la SDK para dispositivos con AWS IoTversión 2 para Java para probar las comunicaciones entre un dispositivo cliente y un dispositivo principal.

> **Important**
>
> Para compilar la SDK para dispositivos con AWS IoT versión 2 para Java, un dispositivo debe tener las siguientes herramientas:
>
> - Java 8 o posterior, `JAVA_HOME` apuntando a la carpeta Java.
> - Apache Maven

Para probar las comunicaciones (SDK para dispositivos con AWS IoTv2 para Java)

1. Descargue y cree la SDK para dispositivos con AWS IoTversión 2 para Java en el AWS IoT dispositivo que desee conectar como dispositivo cliente.

   En el dispositivo cliente, haga lo siguiente:

   a. Clone el SDK para dispositivos con AWS IoT repositorio de la versión 2 para Java para descargarlo.

```
git clone https://github.com/aws/aws-iot-device-sdk-java-v2.git
```

b. Cambie a la carpeta SDK para dispositivos con AWS IoT v2 para Java.

c. Compila la SDK para dispositivos con AWS IoT v2 para Java.

```
cd aws-iot-device-sdk-java-v2
mvn versions:use-latest-versions -Dincludes="software.amazon.awssdk.crt*"
mvn clean install
```

2. Ejecute la aplicación Greengrass Discovery de muestra. Esta aplicación espera argumentos que especifiquen el nombre del dispositivo cliente, el tema de MQTT que se va a utilizar y los certificados que autentican y protegen la conexión. En el siguiente ejemplo, se suscribe al `clients/`*`MyClientDevice1`*`/hello/world` tema y se publica un mensaje sobre el mismo tema que se introduce en la línea de comandos.

   - Sustituya las dos instancias de *`MyClientDevice1`* por el nombre de la cosa del dispositivo cliente.
   - Sustituya *`$HOME/certs/ AmazonRoot CA1.pem`* por la ruta al certificado de CA raíz de Amazon en el dispositivo cliente.
   - Sustituya *`$HOME/certs/device.pem.crt por la ruta al certificado del dispositivo cliente`*.
   - Sustituya *`$HOME/certs/private.pem.key`* por la ruta al archivo de clave privada del dispositivo cliente.
   - Sustituya *`us-east-1`* por el lugar donde funcionan el dispositivo cliente y el dispositivo principal. Región de AWS

```
DISCOVERY_SAMPLE_ARGS="--thing_name MyClientDevice1 \
  --topic 'clients/MyClientDevice1/hello/world' \
  --ca_file $HOME/certs/AmazonRootCA1.pem \
  --cert $HOME/certs/device.pem.crt \
  --key $HOME/certs/private.pem.key \
  --region us-east-1"

mvn exec:java -pl samples/Greengrass \
  -Dexec.mainClass=greengrass.BasicDiscovery \
  -Dexec.args="$DISCOVERY_SAMPLE_ARGS"
```

La aplicación de ejemplo Discovery se suscribe al tema y le pide que introduzca un mensaje para publicarlo.

```
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:
```

Si, en cambio, la aplicación genera un error, consulte Solución de problemas de detección de Greengrass.

3. Introduzca un mensaje, por ejemplo. **`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!
```

Si el resultado indica que la aplicación recibió el mensaje MQTT sobre el tema, el dispositivo cliente puede comunicarse correctamente con el dispositivo principal.

```
Message received on topic clients/MyClientDevice1/hello/world: Hello World!
```

También puede ver los registros de Greengrass en el dispositivo principal para comprobar si el dispositivo cliente se conecta y envía mensajes correctamente. Para obtener más información, consulte Supervisar AWS IoT Greengrass registros.

API RESTful de Greengrass Discovery

AWS IoT Greengrassproporciona la operación de `Discover` API que los dispositivos cliente pueden usar para identificar los dispositivos principales de Greengrass a los que se pueden conectar. Los dispositivos cliente utilizan esta operación del plano de datos para recuperar la información necesaria para conectarse a los dispositivos principales de Greengrass, donde se asocian a la operación de la BatchAssociateClientDeviceWithCoreDeviceAPI. Cuando un dispositivo cliente se conecta a Internet, puede conectarse al servicio AWS IoT Greengrass en la nube y usar la API de detección para encontrar:

- La dirección IP y el puerto de cada dispositivo principal de Greengrass asociado.
- El certificado de CA del dispositivo principal, que los dispositivos cliente pueden usar para autenticar el dispositivo principal de Greengrass.

> **Note**
>
> Los dispositivos cliente también pueden usar el cliente de detección SDK para dispositivos con AWS IoT para descubrir la información de conectividad de los dispositivos principales de Greengrass. El cliente de detección utiliza la API de detección. Para obtener más información, consulte los siguientes temas:
>
> - Pruebe las comunicaciones del dispositivo cliente
> - API RESTful de Greengrass Discovery en la AWS IoT Greengrass Version 1 guía para desarrolladores.

Para usar esta operación de API, envíe solicitudes HTTP a la API de descubrimiento en el punto final del plano de datos de Greengrass. Este punto final de la API tiene el siguiente formato.

```
https://greengrass-ats.iot.region.amazonaws.com:port/greengrass/discover/thing/thing-name
```

Para obtener una lista de los puntos finales Regiones de AWS y los puntos finales compatibles con la API de AWS IoT Greengrass descubrimiento, consulte AWS IoT Greengrass V2los puntos finales y las cuotas en. Referencia general de AWS Esta operación de API solo está disponible en el punto final del plano de datos de Greengrass. El punto final del plano de control que se utiliza para administrar los componentes y las implementaciones es diferente del punto final del plano de datos.

> **Note**
>
> La API de descubrimiento es la misma para AWS IoT Greengrass V1 yAWS IoT Greengrass V2. Si tiene dispositivos cliente que se conectan a un AWS IoT Greengrass V1 núcleo, puede conectarlos a los dispositivos AWS IoT Greengrass V2 principales sin cambiar el código de los dispositivos cliente. Para obtener más información, consulte la API RESTful de Greengrass Discovery en la AWS IoT Greengrass Version 1 Guía para desarrolladores.

Temas

Descubrimiento, autenticación y autorización

Para usar la API de detección para recuperar información de conectividad, un dispositivo cliente debe usar la autenticación mutua TLS con un certificado de cliente X.509 para autenticarse. Para obtener más información, consulte los certificados de cliente X.509 en la Guía para desarrolladores. AWS IoT Core

Un dispositivo cliente también debe tener permiso para realizar la `greengrass:Discover` acción. El siguiente ejemplo AWS IoT de política permite AWS IoT que una cosa denominada `MyClientDevice1` funcione `Discover` por sí misma.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "greengrass:Discover",
      "Resource": [
        "arn:aws:iot:us-west-2:123456789012:thing/MyClientDevice1"
      ]
    }
  ]
}
```

Important

Las variables de política de cosas (`iot:Connection.Thing.*`) no se admiten en las AWS IoT políticas de dispositivos principales ni en las operaciones del plano de datos de Greengrass. En su lugar, puede utilizar un comodín que haga coincidir varios dispositivos con nombres similares. Por ejemplo, puede `MyGreengrassDevice*` especificar que coincida `MyGreengrassDevice1MyGreengrassDevice2`, etc.

Para obtener más información, consulte AWS IoT Corelas políticas en la Guía para AWS IoT Core desarrolladores.

Solicitud

La solicitud contiene los encabezados HTTP estándar y se envía al punto final de descubrimiento de Greengrass, como se muestra en los siguientes ejemplos.

El número de puerto depende de si el dispositivo principal está configurado para enviar tráfico HTTPS a través del puerto 8443 o el puerto 443. Para obtener más información, consulte the section called “Realizar la conexión en el puerto 443 o a través de un proxy de red”.

> **Note**
>
> En estos ejemplos se utiliza el punto de conexión Amazon Trust Services (ATS), que funciona con los certificados de CA raíz ATS recomendados. Los puntos de enlace deben coincidir con el tipo de certificado de CA raíz.

Puerto 8443

```
HTTP GET https://greengrass-ats.iot.region.amazonaws.com:8443/greengrass/discover/
thing/thing-name
```

Puerto 443

```
HTTP GET https://greengrass-ats.iot.region.amazonaws.com:443/greengrass/discover/
thing/thing-name
```

> **Note**
>
> Los clientes que se conecten por el puerto 443 deben implementar la extensión TLS de Negociación de Protocolo de Capa de Aplicación (ALPN) y pasar `x-amzn-http-ca` como el `ProtocolName` en el `ProtocolNameList`. Para obtener más información, consulte Protocolos en la Guía para desarrolladores de AWS IoT.

Respuesta

En caso de éxito, el encabezado de la respuesta incluye el código de estado HTTP 200 y el cuerpo de la respuesta contiene el documento de detección de la respuesta.

> **Note**
>
> Como AWS IoT Greengrass V2 utiliza la misma API de descubrimiento queAWS IoT Greengrass V1, la respuesta organiza la información según AWS IoT Greengrass V1 conceptos, como los grupos de Greengrass. La respuesta contiene una lista de grupos de Greengrass. EnAWS IoT Greengrass V2, cada dispositivo principal está en su propio grupo, donde el grupo contiene solo ese dispositivo principal y su información de conectividad.

Ejemplo de documentos de respuesta de detección

El siguiente documento muestra la respuesta de un dispositivo cliente que está asociado a un dispositivo principal de Greengrass. El dispositivo principal tiene un punto final y un certificado de 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-----"
      ]
    }
  ]
```

```
}
```

El siguiente documento muestra la respuesta de un dispositivo cliente que está asociado a dos dispositivos principales. Los dispositivos principales tienen varios puntos finales y varios certificados de CA de grupo.

```
{
  "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-----"
      ]
    }
  ]
}
```

Pruebe la API de descubrimiento con cURL

Si la ha cURL instalado, puede probar la API de descubrimiento. El siguiente ejemplo especifica los certificados de un dispositivo cliente para autenticar una solicitud en el punto final de la API de descubrimiento de 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**
>
> El `-i` argumento especifica la salida de los encabezados de respuesta HTTP. Puede utilizar esta opción para ayudar a identificar los errores.

Si la solicitud se realiza correctamente, este comando genera una respuesta similar a la del siguiente ejemplo.

```
{
  "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"
      ]
    }
  ]
}
```

Si el comando genera un error, consulte Solución de problemas de detección de Greengrass.

Retransmitir mensajes MQTT entre dispositivos cliente y AWS IoT Core

Puede retransmitir mensajes MQTT y otros datos entre dispositivos cliente y. AWS IoT Core Los dispositivos cliente se conectan al componente intermediario MQTT que se ejecuta en el dispositivo principal. De forma predeterminada, los dispositivos principales no transmiten mensajes o datos MQTT entre los dispositivos cliente y. AWS IoT Core De forma predeterminada, los dispositivos cliente solo se pueden comunicar entre sí a través de MQTT.

Para retransmitir mensajes MQTT entre dispositivos cliente y AWS IoT Core configurar el componente de puente MQTT para que haga lo siguiente:

- Retransmita mensajes desde los dispositivos cliente a. AWS IoT Core
- AWS IoT CoreRetransmita mensajes desde los dispositivos cliente.

> **Note**
>
> El puente MQTT usa QoS 1 para publicar y AWS IoT Core suscribirse, incluso cuando un dispositivo cliente usa QoS 0 para publicar y suscribirse al broker MQTT local. Como resultado, es posible que observe una latencia adicional al retransmitir los mensajes MQTT desde los dispositivos cliente del broker MQTT local. AWS IoT Core Para obtener más información sobre la configuración de MQTT en los dispositivos principales, consulte. Configure los tiempos de espera y los ajustes de caché de MQTT

Temas

Configure e implemente el componente de puente MQTT

El componente MQTT bridge consume una lista de mapeos de temas, cada uno de los cuales especifica un origen y un destino del mensaje. Para retransmitir mensajes entre los dispositivos cliente yAWS IoT Core, implementar el componente de puente MQTT y especificar cada tema de origen y destino en la configuración del componente.

Para implementar el componente MQTT bridge en un dispositivo principal o en un grupo de dispositivos principales, cree una implementación que incluya el `aws.greengrass.clientdevices.mqtt.Bridge` componente. Especifique las asignaciones de temas en la configuración del componente de puente MQTT de la implementación. `mqttTopicMapping`

El siguiente ejemplo define una implementación que configura el componente de puente MQTT para retransmitir mensajes sobre temas que coinciden con el filtro de temas desde los dispositivos cliente hasta los que coinciden con el filtro de `clients/+/hello/world` temas. AWS IoT Core La actualización `merge` de la configuración requiere un objeto JSON serializado. Para obtener más información, consulte Actualizar las configuraciones de los componentes.

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\"}}}"
      }
    }
    ...
  }
}
```

Retransmita mensajes MQTT

Para retransmitir mensajes MQTT entre dispositivos cliente y configurar e AWS IoT Core implementar el componente MQTT Bridge y especificar los temas que se van a retransmitir.

Example Ejemplo: retransmitir mensajes sobre un tema desde los dispositivos cliente a AWS IoT Core

La siguiente configuración de componentes de puente de MQTT especifica la transmisión de mensajes sobre temas que coinciden con el filtro de temas desde los dispositivos cliente a los que coinciden con el filtro de `clients/+/hello/world/event` temas. AWS IoT Core

```
{
  "mqttTopicMapping": {
    "HelloWorldEvent": {
      "topic": "clients/+/hello/world/event",
      "source": "LocalMqtt",
      "target": "IotCore"
```

```
    }
  }
}
```

Example Ejemplo: retransmitir mensajes sobre un tema desde los dispositivos AWS IoT Core cliente

La siguiente configuración de componentes del puente MQTT especifica la transmisión de mensajes sobre temas que coinciden con el filtro de temas desde los que se filtra el `clients/+/hello/world/event/response` tema AWS IoT Core a los dispositivos cliente.

```
{
  "mqttTopicMapping": {
    "HelloWorldEventConfirmation": {
      "topic": "clients/+/hello/world/event/response",
      "source": "IotCore",
      "target": "LocalMqtt"
    }
  }
}
```

Interactúe con los dispositivos cliente en los componentes

Puede desarrollar componentes personalizados de Greengrass que interactúen con los dispositivos cliente conectados a un dispositivo principal. Por ejemplo, puede desarrollar componentes que hagan lo siguiente:

- Utilice los mensajes MQTT de los dispositivos cliente y envíe datos a los Nube de AWS destinos.
- Envíe mensajes MQTT a los dispositivos cliente para iniciar acciones.

Los dispositivos cliente se conectan y se comunican con un dispositivo principal a través del componente intermediario MQTT que se ejecuta en el dispositivo principal. De forma predeterminada, los dispositivos cliente solo pueden comunicarse entre sí a través de MQTT, y los componentes de Greengrass no pueden recibir estos mensajes MQTT ni enviar mensajes a los dispositivos cliente.

Los componentes de Greengrass utilizan la interfaz local de publicación/suscripción para comunicarse en un dispositivo central. Para comunicarse con los dispositivos cliente en los componentes de Greengrass, configure el componente de puente MQTT para que haga lo siguiente:

- Retransmita los mensajes MQTT desde los dispositivos cliente a una plataforma local de publicación/suscripción.
- Transmita los mensajes MQTT desde los dispositivos de publicación o suscripción locales a los dispositivos cliente.

También puede interactuar con las sombras de los dispositivos cliente en los componentes de Greengrass. Para obtener más información, consulte Interactúa con las sombras de los dispositivos cliente y sincronízalas.

Temas

- Configure e implemente el componente puente MQTT
- Reciba mensajes MQTT desde los dispositivos cliente
- Envíe mensajes MQTT a los dispositivos cliente

Configure e implemente el componente puente MQTT

El componente MQTT bridge consume una lista de asignaciones de temas, cada una de las cuales especifica un origen y un destino del mensaje. Para comunicarse con los dispositivos cliente, despliegue el componente MQTT bridge y especifique cada tema de origen y destino en la configuración del componente.

Para implementar el componente MQTT bridge en un dispositivo principal o en un grupo de dispositivos principales, cree una implementación que incluya el `aws.greengrass.clientdevices.mqtt.Bridge` componente. Especifique las asignaciones de temas en la configuración del componente de puente MQTT de la implementación. `mqttTopicMapping`

El siguiente ejemplo define una implementación que configura el componente MQTT bridge para retransmitir el `clients/MyClientDevice1/hello/world` tema desde los dispositivos cliente al agente local de publicación/suscripción. La actualización de `merge` configuración requiere un objeto JSON serializado. Para obtener más información, consulte Actualizar las configuraciones de los componentes.

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\"}}}"
      }
    }
    ...
  }
}
```

Puede utilizar caracteres comodín de temas de MQTT para retransmitir mensajes sobre temas que coincidan con un filtro de temas. Si utilizas MQTT bridge v2.2.0 o una versión posterior, puedes usar comodines de temas MQTT en los filtros de temas cuando el intermediario de origen sea una publicación o suscripción local. Para obtener más información, consulte Configuración de los componentes del puente MQTT.

Reciba mensajes MQTT desde los dispositivos cliente

Puede suscribirse a los temas locales de publicación o suscripción que haya configurado para el componente MQTT bridge para recibir mensajes de los dispositivos cliente.

Para recibir mensajes MQTT desde dispositivos cliente en componentes personalizados

1. Configure e implemente el componente MQTT bridge para retransmitir los mensajes de un tema de MQTT en el que los dispositivos cliente publican a un tema local de publicación o suscripción.

2. Utilice la interfaz IPC local de publicación/suscripción para suscribirse al tema en el que el puente MQTT transmite los mensajes. Para obtener más información, consulte Publicar/suscribir mensajes locales y SubscribeToTopic.

El tutorial Conectar y probar dispositivos cliente incluye una sección en la que se desarrolla un componente que se suscribe a los mensajes de un dispositivo cliente. Para obtener más información, consulte Paso 4: Desarrolle un componente que se comunique con los dispositivos cliente.

Envíe mensajes MQTT a los dispositivos cliente

Puede publicar en los temas locales de publicación o suscripción que haya configurado para el componente MQTT bridge a fin de enviar mensajes a los dispositivos cliente.

Para publicar mensajes MQTT en dispositivos cliente en componentes personalizados

1. Configure e implemente el componente MQTT bridge para retransmitir mensajes de un tema local de publicación o suscripción a un tema de MQTT en el que se suscriban los dispositivos cliente.
2. Utilice la interfaz IPC local de publicación/suscripción para publicar en el tema en el que el puente MQTT transmite los mensajes. Para obtener más información, consulte Publicar/suscribir mensajes locales y PublishToTopic.

Interactúa con las sombras de los dispositivos cliente y sincronízalas

Puede usar el componente administrador de sombras para administrar las sombras locales, incluidas las sombras de los dispositivos cliente. Puede usar el administrador de sombras para hacer lo siguiente:

- Interactúe con las sombras de los dispositivos cliente en los componentes de Greengrass.
- Sincronice las sombras de los dispositivos cliente conAWS IoT Core.

> **Note**
>
> El componente administrador de sombras no sincroniza las sombras con ellas de forma AWS IoT Core predeterminada. Debe configurar el componente de administrador de sombras para especificar qué sombras del dispositivo cliente desea sincronizar.

Temas

Requisitos previos

Para interactuar con las sombras de los dispositivos cliente y sincronizarlas con las sombras de los dispositivos clienteAWS IoT Core, un dispositivo principal debe cumplir los siguientes requisitos:

- El dispositivo principal debe ejecutar los siguientes componentes, además de los componentes de Greengrass para la compatibilidad con los dispositivos cliente:
  - Greengrass nucleus v2.6.0 o posterior
  - Shadow manager v2.2.0 o posterior
  - MQTT bridge v2.2.0 o posterior
- El componente de autenticación del dispositivo cliente debe configurarse para permitir que los dispositivos cliente se comuniquen sobre temas ocultos sobre dispositivos.

Habilite el administrador oculto para que se comunique con los dispositivos cliente

De forma predeterminada, el componente administrador de sombras no administra las sombras de los dispositivos cliente. Para habilitar esta función, debe retransmitir los mensajes MQTT entre los dispositivos cliente y el componente de administrador de sombras. Los dispositivos cliente utilizan los mensajes MQTT para recibir y enviar actualizaciones ocultas a los dispositivos. El componente administrador de sombras se suscribe a la interfaz local de publicación/suscripción de Greengrass,

por lo que puede configurar el componente puente MQTT para retransmitir mensajes MQTT sobre temas ocultos de dispositivos.

El componente MQTT bridge consume una lista de mapeos de temas, cada uno de los cuales especifica un origen y un destino del mensaje. Para permitir que el componente administrador de sombras gestione las sombras de los dispositivos cliente, despliegue el componente puente de MQTT y especifique los temas ocultos para las sombras de los dispositivos cliente. Debe configurar el puente para retransmitir mensajes en ambas direcciones entre el MQTT local y el servicio de publicación/suscripción local.

Para implementar el componente de puente MQTT en un dispositivo principal o grupo de dispositivos principales, cree una implementación que incluya el componente. `aws.greengrass.clientdevices.mqtt.Bridge` Especifique las asignaciones de temas en la configuración del componente de puente MQTT de la implementación. `mqttTopicMapping`

Utilice los siguientes ejemplos para configurar el componente MQTT bridge para permitir la comunicación entre los dispositivos cliente y el componente shadow manager.

> **Note**
>
> Puede utilizar estos ejemplos de configuración en la AWS IoT Greengrass consola. Si utilizas la AWS IoT Greengrass API, la actualización de `merge` configuración requiere un objeto JSON serializado, por lo que debes serializar los siguientes objetos JSON en cadenas. Para obtener más información, consulte Actualizar las configuraciones de los componentes.

Example Ejemplo: administre todas las sombras de los dispositivos cliente

El siguiente ejemplo de configuración de puentes de MQTT permite que el administrador de sombras gestione todas las sombras de todos los dispositivos cliente.

```
{
  "mqttTopicMapping": {
    "ShadowsLocalMqttToPubsub": {
      "topic": "$aws/things/+/shadow/#",
      "source": "LocalMqtt",
      "target": "Pubsub"
    },
    "ShadowsPubsubToLocalMqtt": {
      "topic": "$aws/things/+/shadow/#",
```

```
      "source": "Pubsub",
      "target": "LocalMqtt"
    }
  }
}
```

Example Ejemplo: gestionar las sombras de un dispositivo cliente

El siguiente ejemplo de configuración de un puente de MQTT permite al administrador de sombras gestionar todas las sombras de un dispositivo cliente denominado`MyClientDevice`.

```
{
  "mqttTopicMapping": {
    "ShadowsLocalMqttToPubsub": {
      "topic": "$aws/things/MyClientDevice/shadow/#",
      "source": "LocalMqtt",
      "target": "Pubsub"
    },
    "ShadowsPubsubToLocalMqtt": {
      "topic": "$aws/things/MyClientDevice/shadow/#",
      "source": "Pubsub",
      "target": "LocalMqtt"
    }
  }
}
```

Example Ejemplo: administre una sombra con nombre para todos los dispositivos cliente

El siguiente ejemplo de configuración de un puente de MQTT permite que el administrador de sombras gestione una sombra con el nombre `DeviceConfiguration` de todos los dispositivos cliente.

```
{
  "mqttTopicMapping": {
    "ShadowsLocalMqttToPubsub": {
      "topic": "$aws/things/+/shadow/name/DeviceConfiguration/#",
      "source": "LocalMqtt",
      "target": "Pubsub"
    },
    "ShadowsPubsubToLocalMqtt": {
      "topic": "$aws/things/+/shadow/name/DeviceConfiguration/#",
      "source": "Pubsub",
```

```
      "target": "LocalMqtt"
    }
  }
}
```

Example Ejemplo: administre las sombras sin nombre de todos los dispositivos cliente

El siguiente ejemplo de configuración de puentes de MQTT permite al administrador de sombras gestionar sombras sin nombre, pero no sombras con nombre, para todos los dispositivos cliente.

```
{
  "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"
    }
  }
```

```
}
```

Interactúe con las sombras de los dispositivos cliente en los componentes

Puede desarrollar componentes personalizados que utilicen el servicio paralelo local para leer y modificar los documentos paralelos locales de los dispositivos cliente. Para obtener más información, consulte Interactúa con las sombras de los componentes.

Sincronice las sombras de los dispositivos cliente con AWS IoT Core

Puede configurar el componente de administrador de sombras para sincronizar los estados ocultos de los dispositivos cliente locales con AWS IoT Core ellos. Para obtener más información, consulte Sincronice las sombras de los dispositivos locales con AWS IoT Core.

Solución de problemas de dispositivos cliente

Utilice la información y las soluciones de solución de problemas de esta sección para ayudar a resolver los problemas con los dispositivos cliente y los componentes de los dispositivos cliente de Greengrass.

Temas

- Problemas de descubrimiento de Greengrass
- Problemas de conexión con MQTT

Problemas de descubrimiento de Greengrass

Utilice la siguiente información para solucionar problemas relacionados con la detección de Greengrass. Estos problemas pueden producirse cuando los dispositivos cliente utilizan la API de descubrimiento de Greengrass para identificar un dispositivo principal de Greengrass al que se pueden conectar.

Temas

- Problemas de descubrimiento de Greengrass (API HTTP)
- Problemas de descubrimiento de Greengrass (SDK para dispositivos con AWS IoTversión 2 para Python)
- Problemas de descubrimiento de Greengrass (SDK para dispositivos con AWS IoTversión 2 para C++)

- Problemas de descubrimiento de Greengrass (SDK para dispositivos con AWS IoTversión 2 para) JavaScript
- Problemas de descubrimiento de Greengrass (SDK para dispositivos con AWS IoTversión 2 para Java)

Problemas de descubrimiento de Greengrass (API HTTP)

Utilice la siguiente información para solucionar problemas relacionados con la detección de Greengrass. Es posible que veas estos errores si pruebas la API de descubrimiento con cURL.

Temas

- curl: (52) Empty reply from server
- HTTP 403: {"message":null,"traceId":"a1b2c3d4-5678-90ab-cdef-11111EXAMPLE"}
- HTTP 404: {"errorMessage":"The thing provided for discovery was not found"}

curl: (52) Empty reply from server

Es posible que veas este error si especificas un AWS IoT certificado inactivo en la solicitud.

Compruebe que el dispositivo cliente tiene un certificado adjunto y que el certificado está activo. Para obtener más información, consulte Adjuntar un elemento o una política a un certificado de cliente y Activar o desactivar un certificado de cliente en la Guía para AWS IoT Core desarrolladores.

HTTP 403: {"message":null,"traceId":"a1b2c3d4-5678-90ab-cdef-11111EXAMPLE"}

Es posible que aparezca este error si el dispositivo cliente no tiene permiso para realizar llamadas `greengrass:Discover` por sí mismo.

Compruebe que el certificado del dispositivo cliente tenga una política que lo permita`greengrass:Discover`. No puedes usar variables de política de cosas (`iot:Connection.Thing.*`) en la `Resource` sección correspondiente a este permiso. Para obtener más información, consulte Descubrimiento, autenticación y autorización.

HTTP 404: {"errorMessage":"The thing provided for discovery was not found"}

Es posible que veas este error en los siguientes casos:

- El dispositivo cliente no está asociado a ningún dispositivo o AWS IoT Greengrass V1 grupo principal de Greengrass.

- Ninguno de los dispositivos o AWS IoT Greengrass V1 grupos principales de Greengrass asociados al dispositivo cliente tiene un punto final intermediario MQTT.
- Ninguno de los dispositivos principales de Greengrass asociados al dispositivo cliente ejecuta el componente de autenticación del dispositivo cliente.

Compruebe que el dispositivo cliente esté asociado al dispositivo principal al que desea que se conecte. A continuación, compruebe que el dispositivo principal ejecute el componente de autenticación del dispositivo cliente y que tenga al menos un punto final intermediario MQTT. Para obtener más información, consulte los siguientes temas:

- Asociar dispositivos cliente
- Administre los puntos finales de los dispositivos principales
- Configure la detección en la nube (consola)

Problemas de descubrimiento de Greengrass (SDK para dispositivos con AWS IoTversión 2 para Python)

Utilice la siguiente información para solucionar problemas relacionados con el descubrimiento de Greengrass en SDK para dispositivos con AWS IoTla versión 2 para Python.

Temas

- 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.

Es posible que aparezca este error si especifica un AWS IoT certificado inactivo en la solicitud.

Compruebe que el dispositivo cliente tiene un certificado adjunto y que el certificado está activo. Para obtener más información, consulte Adjuntar un elemento o una política a un certificado de cliente y Activar o desactivar un certificado de cliente en la Guía para AWS IoT Core desarrolladores.

awsiot.greengrass_discovery.DiscoveryException: ('Error during discover call: response_code=403', 403)

Es posible que aparezca este error si el dispositivo cliente no tiene permiso para realizar llamadas `greengrass:Discover` por sí mismo.

Compruebe que el certificado del dispositivo cliente tenga una política que lo permita`greengrass:Discover`. No puedes usar variables de política de cosas (`iot:Connection.Thing.*`) en la `Resource` sección correspondiente a este permiso. Para obtener más información, consulte Descubrimiento, autenticación y autorización.

awsiot.greengrass_discovery.DiscoveryException: ('Error during discover call: response_code=404', 404)

Es posible que veas este error en los siguientes casos:

- El dispositivo cliente no está asociado a ningún dispositivo o AWS IoT Greengrass V1 grupo principal de Greengrass.
- Ninguno de los dispositivos o AWS IoT Greengrass V1 grupos principales de Greengrass asociados al dispositivo cliente tiene un punto final intermediario MQTT.
- Ninguno de los dispositivos principales de Greengrass asociados al dispositivo cliente ejecuta el componente de autenticación del dispositivo cliente.

Compruebe que el dispositivo cliente esté asociado al dispositivo principal al que desea que se conecte. A continuación, compruebe que el dispositivo principal ejecute el componente de autenticación del dispositivo cliente y que tenga al menos un punto final intermediario MQTT. Para obtener más información, consulte los siguientes temas:

- Asociar dispositivos cliente
- Administre los puntos finales de los dispositivos principales
- Configure la detección en la nube (consola)

Problemas de descubrimiento de Greengrass (SDK para dispositivos con AWS IoTversión 2 para C++)

Utilice la siguiente información para solucionar problemas relacionados con la detección de Greengrass en SDK para dispositivos con AWS IoTla versión 2 para C++.

Temas

aws-c-http: AWS_ERROR_HTTP_CONNECTION_CLOSED, The connection has closed or is closing.

Es posible que aparezca este error si especifica un AWS IoT certificado inactivo en la solicitud.

Compruebe que el dispositivo cliente tiene un certificado adjunto y que el certificado está activo. Para obtener más información, consulte Adjuntar un elemento o una política a un certificado de cliente y Activar o desactivar un certificado de cliente en la Guía para AWS IoT Core desarrolladores.

aws-c-common: AWS_ERROR_UNKNOWN, Unknown error. (HTTP 403)

Es posible que aparezca este error si el dispositivo cliente no tiene permiso para realizar llamadas `greengrass:Discover` por sí mismo.

Compruebe que el certificado del dispositivo cliente tenga una política que lo permita`greengrass:Discover`. No puedes usar variables de política de cosas (`iot:Connection.Thing.*`) en la `Resource` sección correspondiente a este permiso. Para obtener más información, consulte Descubrimiento, autenticación y autorización.

aws-c-common: AWS_ERROR_UNKNOWN, Unknown error. (HTTP 404)

Es posible que veas este error en los siguientes casos:

- El dispositivo cliente no está asociado a ningún dispositivo o AWS IoT Greengrass V1 grupo principal de Greengrass.
- Ninguno de los dispositivos o AWS IoT Greengrass V1 grupos principales de Greengrass asociados al dispositivo cliente tiene un punto final intermediario MQTT.
- Ninguno de los dispositivos principales de Greengrass asociados al dispositivo cliente ejecuta el componente de autenticación del dispositivo cliente.

Compruebe que el dispositivo cliente esté asociado al dispositivo principal al que desea que se conecte. A continuación, compruebe que el dispositivo principal ejecute el componente de

autenticación del dispositivo cliente y que tenga al menos un punto final intermediario MQTT. Para obtener más información, consulte los siguientes temas:

- Asociar dispositivos cliente
- Administre los puntos finales de los dispositivos principales
- Configure la detección en la nube (consola)

Problemas de descubrimiento de Greengrass (SDK para dispositivos con AWS IoTversión 2 para) JavaScript

Utilice la siguiente información para solucionar problemas relacionados con el descubrimiento de Greengrass en SDK para dispositivos con AWS IoTla versión 2 para. JavaScript

Temas

- 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.

Es posible que aparezca este error si especifica un AWS IoT certificado inactivo en la solicitud.

Compruebe que el dispositivo cliente tiene un certificado adjunto y que el certificado está activo. Para obtener más información, consulte Adjuntar un elemento o una política a un certificado de cliente y Activar o desactivar un certificado de cliente en la Guía para AWS IoT Core desarrolladores.

Error: Discovery failed (headers: [object Object]) { response_code: 403 }

Es posible que aparezca este error si el dispositivo cliente no tiene permiso para realizar llamadas `greengrass:Discover` por sí mismo.

Compruebe que el certificado del dispositivo cliente tenga una política que lo permita`greengrass:Discover`. No puedes usar variables de política de cosas (`iot:Connection.Thing.*`) en la `Resource` sección correspondiente a este permiso. Para obtener más información, consulte Descubrimiento, autenticación y autorización.

Error: Discovery failed (headers: [object Object]) { response_code: 404 }

Es posible que veas este error en los siguientes casos:

- El dispositivo cliente no está asociado a ningún dispositivo o AWS IoT Greengrass V1 grupo principal de Greengrass.
- Ninguno de los dispositivos o AWS IoT Greengrass V1 grupos principales de Greengrass asociados al dispositivo cliente tiene un punto final intermediario MQTT.
- Ninguno de los dispositivos principales de Greengrass asociados al dispositivo cliente ejecuta el componente de autenticación del dispositivo cliente.

Compruebe que el dispositivo cliente esté asociado al dispositivo principal al que desea que se conecte. A continuación, compruebe que el dispositivo principal ejecute el componente de autenticación del dispositivo cliente y que tenga al menos un punto final intermediario MQTT. Para obtener más información, consulte los siguientes temas:

- Asociar dispositivos cliente
- Administre los puntos finales de los dispositivos principales
- Configure la detección en la nube (consola)

Error: Discovery failed (headers: [object Object])

Es posible que vea este error (sin un código de respuesta HTTP) al ejecutar el ejemplo de descubrimiento de Greengrass. Este error puede producirse por varios motivos.

- Es posible que aparezca este error si el dispositivo cliente no tiene permiso para realizar llamadas `greengrass:Discover` por sí mismo.

  Compruebe que el certificado del dispositivo cliente tenga una política que lo permita`greengrass:Discover`. No puedes usar variables de política de cosas (`iot:Connection.Thing.*`) en la `Resource` sección correspondiente a este permiso. Para obtener más información, consulte Descubrimiento, autenticación y autorización.
- Es posible que veas este error en los siguientes casos:
  - El dispositivo cliente no está asociado a ningún dispositivo o AWS IoT Greengrass V1 grupo principal de Greengrass.
  - Ninguno de los dispositivos o AWS IoT Greengrass V1 grupos principales de Greengrass asociados al dispositivo cliente tiene un punto final intermediario MQTT.

- Ninguno de los dispositivos principales de Greengrass asociados al dispositivo cliente ejecuta el componente de autenticación del dispositivo cliente.

Compruebe que el dispositivo cliente esté asociado al dispositivo principal al que desea que se conecte. A continuación, compruebe que el dispositivo principal ejecute el componente de autenticación del dispositivo cliente y que tenga al menos un punto final intermediario MQTT. Para obtener más información, consulte los siguientes temas:

- Asociar dispositivos cliente
- Administre los puntos finales de los dispositivos principales
- Configure la detección en la nube (consola)

Problemas de descubrimiento de Greengrass (SDK para dispositivos con AWS IoTversión 2 para Java)

Utilice la siguiente información para solucionar problemas relacionados con la detección de Greengrass en SDK para dispositivos con AWS IoTla versión 2 para Java.

Temas

- 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.)

Es posible que aparezca este error si especifica un AWS IoT certificado inactivo en la solicitud.

Compruebe que el dispositivo cliente tiene un certificado adjunto y que el certificado está activo. Para obtener más información, consulte Adjuntar un elemento o una política a un certificado de cliente y Activar o desactivar un certificado de cliente en la Guía para AWS IoT Core desarrolladores.

java.lang.RuntimeException: Error x-amzn-ErrorType(403)

Es posible que aparezca este error si el dispositivo cliente no tiene permiso para realizar llamadas `greengrass:Discover` por sí mismo.

Compruebe que el certificado del dispositivo cliente tenga una política que lo permita`greengrass:Discover`. No puedes usar variables de política de cosas (`iot:Connection.Thing.*`) en la `Resource` sección correspondiente a este permiso. Para obtener más información, consulte Descubrimiento, autenticación y autorización.

java.lang.RuntimeException: Error x-amzn-ErrorType(404)

Es posible que veas este error en los siguientes casos:

- El dispositivo cliente no está asociado a ningún dispositivo o AWS IoT Greengrass V1 grupo principal de Greengrass.
- Ninguno de los dispositivos o AWS IoT Greengrass V1 grupos principales de Greengrass asociados al dispositivo cliente tiene un punto final intermediario MQTT.
- Ninguno de los dispositivos principales de Greengrass asociados al dispositivo cliente ejecuta el componente de autenticación del dispositivo cliente.

Compruebe que el dispositivo cliente esté asociado al dispositivo principal al que desea que se conecte. A continuación, compruebe que el dispositivo principal ejecute el componente de autenticación del dispositivo cliente y que tenga al menos un punto final intermediario MQTT. Para obtener más información, consulte los siguientes temas:

- Asociar dispositivos cliente
- Administre los puntos finales de los dispositivos principales
- Configure la detección en la nube (consola)

Problemas de conexión con MQTT

Utilice la siguiente información para solucionar problemas con las conexiones MQTT de los dispositivos cliente. Estos problemas pueden producirse cuando los dispositivos cliente intentan conectarse a un dispositivo principal a través de MQTT.

Temas

io.moquette.broker.Authorizator: Client does not have read permissions on the topic

Es posible que veas este error en los registros de Greengrass cuando un dispositivo cliente intenta suscribirse a un tema de MQTT para el que no tiene permiso. El mensaje de error incluye el tema.

Compruebe que la configuración del componente de autenticación del dispositivo cliente incluya lo siguiente:

- Un grupo de dispositivos que coincida con el dispositivo cliente.
- Una política de autorización de dispositivos cliente para ese grupo de dispositivos que concede el `mqtt:subscribe` permiso para el tema.

Para obtener más información sobre cómo implementar y configurar el componente de autenticación del dispositivo cliente, consulte lo siguiente:

- Configure la detección en la nube (consola)
- Autenticación del dispositivo cliente
- Crear implementaciones

Problemas de conexión con MQTT (Python)

Utilice la siguiente información para solucionar problemas con las conexiones MQTT de los dispositivos cliente cuando utilice la SDK para dispositivos con AWS IoTversión 2 para Python.

Temas

AWS_ERROR_MQTT_PROTOCOL_ERROR: Protocol error occurred

Es posible que aparezca este error si el componente de autenticación del dispositivo cliente no define una política de autorización del dispositivo cliente que conceda al dispositivo cliente permiso para conectarse.

Compruebe que la configuración del componente de autenticación del dispositivo cliente incluya lo siguiente:

- Un grupo de dispositivos que coincida con el dispositivo cliente.
- Una política de autorización de dispositivos cliente para ese grupo de dispositivos que otorga el `mqtt:connect` permiso para el dispositivo cliente.

Para obtener más información sobre cómo implementar y configurar el componente de autenticación del dispositivo cliente, consulte lo siguiente:

- Configure la detección en la nube (consola)
- Autenticación del dispositivo cliente
- Crear implementaciones

AWS_ERROR_MQTT_UNEXPECTED_HANGUP: Unexpected hangup occurred

Es posible que aparezca este error si el componente de autenticación del dispositivo cliente no define una política de autorización del dispositivo cliente que conceda al dispositivo cliente permiso para conectarse.

Compruebe que la configuración del componente de autenticación del dispositivo cliente incluya lo siguiente:

- Un grupo de dispositivos que coincida con el dispositivo cliente.
- Una política de autorización de dispositivos cliente para ese grupo de dispositivos que otorga el `mqtt:connect` permiso para el dispositivo cliente.

Para obtener más información sobre cómo implementar y configurar el componente de autenticación del dispositivo cliente, consulte lo siguiente:

- Configure la detección en la nube (consola)
- Autenticación del dispositivo cliente

- Crear implementaciones

Problemas de conexión con MQTT (C++)

Utilice la siguiente información para solucionar problemas con las conexiones MQTT de los dispositivos cliente cuando utilice la SDK para dispositivos con AWS IoTversión 2 para C++.

Temas

- AWS_ERROR_MQTT_PROTOCOL_ERROR: Protocol error occurred
- AWS_ERROR_MQTT_UNEXPECTED_HANGUP: Unexpected hangup occurred

AWS_ERROR_MQTT_PROTOCOL_ERROR: Protocol error occurred

Es posible que aparezca este error si el componente de autenticación del dispositivo cliente no define una política de autorización del dispositivo cliente que conceda al dispositivo cliente permiso para conectarse.

Compruebe que la configuración del componente de autenticación del dispositivo cliente incluya lo siguiente:

- Un grupo de dispositivos que coincida con el dispositivo cliente.
- Una política de autorización de dispositivos cliente para ese grupo de dispositivos que otorga el `mqtt:connect` permiso para el dispositivo cliente.

Para obtener más información sobre cómo implementar y configurar el componente de autenticación del dispositivo cliente, consulte lo siguiente:

- Configure la detección en la nube (consola)
- Autenticación del dispositivo cliente
- Crear implementaciones

AWS_ERROR_MQTT_UNEXPECTED_HANGUP: Unexpected hangup occurred

Es posible que aparezca este error si el componente de autenticación del dispositivo cliente no define una política de autorización del dispositivo cliente que conceda al dispositivo cliente permiso para conectarse.

Compruebe que la configuración del componente de autenticación del dispositivo cliente incluya lo siguiente:

- Un grupo de dispositivos que coincida con el dispositivo cliente.
- Una política de autorización de dispositivos cliente para ese grupo de dispositivos que otorga el `mqtt:connect` permiso para el dispositivo cliente.

Para obtener más información sobre cómo implementar y configurar el componente de autenticación del dispositivo cliente, consulte lo siguiente:

- Configure la detección en la nube (consola)
- Autenticación del dispositivo cliente
- Crear implementaciones

Problemas de conexión con MQTT (Java)

Utilice la siguiente información para solucionar problemas con las conexiones MQTT de los dispositivos cliente cuando utilice la SDK para dispositivos con AWS IoTversión 2 para Java.

Temas

- 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

Es posible que aparezca este error si el componente de autenticación del dispositivo cliente no define una política de autorización del dispositivo cliente que conceda al dispositivo cliente permiso para conectarse.

Compruebe que la configuración del componente de autenticación del dispositivo cliente incluya lo siguiente:

- Un grupo de dispositivos que coincida con el dispositivo cliente.
- Una política de autorización de dispositivos cliente para ese grupo de dispositivos que otorga el `mqtt:connect` permiso para el dispositivo cliente.

Para obtener más información sobre cómo implementar y configurar el componente de autenticación del dispositivo cliente, consulte lo siguiente:

- Configure la detección en la nube (consola)
- Autenticación del dispositivo cliente
- Crear implementaciones

AWS_ERROR_MQTT_UNEXPECTED_HANGUP: Unexpected hangup occurred

Es posible que aparezca este error si el componente de autenticación del dispositivo cliente no define una política de autorización del dispositivo cliente que conceda al dispositivo cliente permiso para conectarse.

Compruebe que la configuración del componente de autenticación del dispositivo cliente incluya lo siguiente:

- Un grupo de dispositivos que coincida con el dispositivo cliente.
- Una política de autorización de dispositivos cliente para ese grupo de dispositivos que otorga el `mqtt:connect` permiso para el dispositivo cliente.

Para obtener más información sobre cómo implementar y configurar el componente de autenticación del dispositivo cliente, consulte lo siguiente:

- Configure la detección en la nube (consola)
- Autenticación del dispositivo cliente
- Crear implementaciones

Problemas de conexión con MQTT () JavaScript

Utilice la siguiente información para solucionar problemas con las conexiones MQTT de los dispositivos cliente cuando utilice la SDK para dispositivos con AWS IoT versión 2 para. JavaScript

Temas

AWS_ERROR_MQTT_PROTOCOL_ERROR: Protocol error occurred

Es posible que aparezca este error si el componente de autenticación del dispositivo cliente no define una política de autorización del dispositivo cliente que conceda al dispositivo cliente permiso para conectarse.

Compruebe que la configuración del componente de autenticación del dispositivo cliente incluya lo siguiente:

- Un grupo de dispositivos que coincida con el dispositivo cliente.
- Una política de autorización de dispositivos cliente para ese grupo de dispositivos que otorga el `mqtt:connect` permiso para el dispositivo cliente.

Para obtener más información sobre cómo implementar y configurar el componente de autenticación del dispositivo cliente, consulte lo siguiente:

- Configure la detección en la nube (consola)
- Autenticación del dispositivo cliente
- Crear implementaciones

AWS_ERROR_MQTT_UNEXPECTED_HANGUP: Unexpected hangup occurred

Es posible que aparezca este error si el componente de autenticación del dispositivo cliente no define una política de autorización del dispositivo cliente que conceda al dispositivo cliente permiso para conectarse.

Compruebe que la configuración del componente de autenticación del dispositivo cliente incluya lo siguiente:

- Un grupo de dispositivos que coincida con el dispositivo cliente.
- Una política de autorización de dispositivos cliente para ese grupo de dispositivos que otorga el `mqtt:connect` permiso para el dispositivo cliente.

Para obtener más información sobre cómo implementar y configurar el componente de autenticación del dispositivo cliente, consulte lo siguiente:

- Configure la detección en la nube (consola)
- Autenticación del dispositivo cliente

- Crear implementaciones

Interactúa con las sombras de los dispositivos

Los dispositivos principales de Greengrass pueden interactuar con las sombras de los AWS IoT dispositivos mediante componentes. Una sombra es un documento JSON que almacena la información de estado actual o deseada de una AWS IoT cosa. Las sombras pueden hacer que el estado de un dispositivo esté disponible para otros AWS IoT Greengrass componentes, ya sea que el dispositivo esté conectado AWS IoT o no. Cada AWS IoT dispositivo tiene su propia sombra clásica sin nombre. También puedes crear varias sombras con nombre para cada dispositivo.

Los dispositivos y servicios pueden crear, actualizar y eliminar sombras de nubes mediante MQTT y los temas de sombras de MQTT reservados, HTTP mediante la API REST Device Shadow y for. AWS CLIAWS IoT

El componente administrador de sombras permite a sus componentes de Greengrass crear, actualizar y eliminar sombras locales mediante el servicio de sombras local y los temas de sombra locales de publicación/suscripción. El administrador de sombras también gestiona el almacenamiento de estos documentos ocultos locales en su dispositivo principal y gestiona la sincronización de la información sobre el estado de las sombras con las sombras de las nubes.

También puede utilizar el componente administrador de sombras para gestionar las sombras locales de los dispositivos cliente que se conectan al dispositivo principal. Para permitir que el administrador de sombras gestione las sombras de los dispositivos cliente, debe configurar el componente de puente MQTT para retransmitir mensajes entre el agente MQTT local y el servicio local de publicación/suscripción. Para obtener más información, consulte Interactúa con las sombras de los dispositivos cliente y sincronízalas.

Para obtener más información sobre los conceptos de sombra de AWS IoT dispositivos, consulte AWS IoTel servicio Device Shadow en la Guía para AWS IoT desarrolladores.

Temas

- Interactúa con las sombras de los componentes
- Sincronice las sombras de los dispositivos locales con AWS IoT Core

Interactúa con las sombras de los componentes

Puede desarrollar componentes personalizados, incluidos los componentes de la función Lambda, que utilicen el servicio de sombra local para leer y modificar documentos de sombra locales y documentos de sombra de dispositivos cliente.

Los componentes personalizados interactúan con el servicio paralelo local mediante las bibliotecas AWS IoT Greengrass Core IPC del. SDK para dispositivos con AWS IoT El componente de administrador de sombras habilita el servicio oculto local en su dispositivo principal.

Para implementar el componente shadow manager en un dispositivo principal de Greengrass, cree una implementación que incluya el `aws.greengrass.ShadowManager` componente.

> **Note**
>
> De forma predeterminada, la implementación del componente de administrador de sombras solo permite las operaciones ocultas locales. AWS IoT GreengrassPara poder sincronizar la información sobre el estado de las sombras de los dispositivos principales o de cualquier sombra de los dispositivos cliente con los documentos de sombra de nube correspondientesAWS IoT Core, debe crear una actualización de configuración para el componente del administrador de sombras que incluya el `synchronize` parámetro. Para obtener más información, consulte Sincronice las sombras de los dispositivos locales con AWS IoT Core.

Temas

- Recupere y modifique los estados de sombra
- Reaccione a los cambios en el estado de sombra

Recupere y modifique los estados de sombra

Las operaciones de IPC clandestinas recuperan y actualizan la información sobre el estado de los documentos alternativos locales. El componente de administrador de sombras se encarga del almacenamiento de estos documentos ocultos en el dispositivo principal.

Para modificar los estados ocultos locales

1. Añada políticas de autorización a la receta de su componente personalizado para permitir que el componente reciba mensajes sobre temas alternativos locales.

   Para ver ejemplos de políticas de autorización, consulte Ejemplos de políticas de autorización de IPC ocultas locales.
2. Utilice las operaciones de IPC ocultas para recuperar y modificar la información sobre el estado oculto. Para obtener más información sobre el uso de operaciones de IPC ocultas en el código de los componentes, consulte. Interactúa con las sombras locales

> **Note**
>
> Para permitir que un dispositivo principal interactúe con las sombras de los dispositivos cliente, también debe configurar e implementar el componente de puente MQTT. Para obtener más información, consulte Habilitar el administrador de sombras para que se comunique con los dispositivos cliente.

Reaccione a los cambios en el estado de sombra

Los componentes de Greengrass utilizan la interfaz local de publicación/suscripción para comunicarse en un dispositivo central. Para permitir que un componente personalizado reaccione ante los cambios en el estado de sombra, puede suscribirse a los temas locales de publicación o suscripción. Esto permite que el componente reciba mensajes sobre los temas alternativos locales y, a continuación, actúe en función de esos mensajes.

Los temas ocultos locales utilizan el mismo formato que los temas MQTT ocultos del AWS IoT dispositivo. Para obtener más información sobre temas ocultos, consulte los temas MQTT de Device Shadow en la Guía para AWS IoT desarrolladores.

Para reaccionar ante los cambios en el estado de sombra local

1. Añada políticas de control de acceso a la receta de su componente personalizado para permitir que el componente reciba mensajes sobre temas alternativos locales.

   Para ver ejemplos de políticas de autorización, consulte Ejemplos de políticas de autorización de IPC ocultas locales.

2. Para iniciar una acción personalizada en un componente, utilice las operaciones de `SubscribeToTopic` IPC para suscribirse a los temas paralelos sobre los que desee recibir mensajes. Para obtener más información sobre el uso de las operaciones de IPC locales de publicación o suscripción en el código del componente, consulte. Publicar/suscribir mensajes locales
3. Para invocar una función Lambda, utilice la configuración de la fuente de eventos para proporcionar el nombre del tema paralelo y especificar que se trata de un tema local de publicación/suscripción. Para obtener información sobre la creación de componentes de funciones Lambda, consulte. AWS LambdaFunciones de ejecución

> **Note**
>
> Para permitir que un dispositivo principal interactúe con las sombras de los dispositivos cliente, también debe configurar e implementar el componente puente MQTT. Para obtener más información, consulte Habilitar el administrador de sombras para que se comunique con los dispositivos cliente.

Sincronice las sombras de los dispositivos locales con AWS IoT Core

El componente administrador de sombras permite sincronizar AWS IoT Greengrass los estados de sombra de los dispositivos locales conAWS IoT Core. Debe modificar la configuración del componente administrador de sombras para incluir el parámetro de `synchronization` configuración y AWS IoT especificar los nombres de los dispositivos y las sombras que desea sincronizar.

Al configurar el administrador de sombras para que sincronice las sombras, sincroniza todos los cambios de estado de las sombras especificadas, independientemente de si los cambios se producen en documentos de sombra locales o en documentos de sombra de nubes.

También puede especificar si el componente del administrador de sombras sincroniza las sombras en tiempo real o en intervalos periódicos. De forma predeterminada, el componente administrador de sombras sincroniza las sombras en tiempo real, por lo que el dispositivo principal envía y recibe actualizaciones ocultas desde y hacia el momento en que AWS IoT Core se produce cada

actualización. Puede configurar intervalos periódicos para reducir el uso de ancho de banda y los cargos.

Temas

Requisitos previos

Para sincronizar las sombras localesAWS IoT Core, debe configurar la AWS IoT política del dispositivo principal de Greengrass para permitir las siguientes acciones de política AWS IoT Core clandestina.

- `iot:GetThingShadow`
- `iot:UpdateThingShadow`
- `iot:DeleteThingShadow`

Para obtener más información, consulte los siguientes temas:

- AWS IoT Coreacciones políticas en la Guía AWS IoTpara desarrolladores
- AWS IoTPolítica mínima para los dispositivos AWS IoT Greengrass V2 principales
- Actualice la política de un dispositivo principal AWS IoT

Configure el componente shadow manager

El administrador de sombras necesita una lista de asignaciones de nombres ocultas para sincronizar la información sobre el estado de las sombras en los documentos paralelos locales y ocultarlos. AWS IoT Core

Para sincronizar los estados de sombra, cree una implementación que incluya el `aws.greengrass.ShadowManager` componente y especifique las sombras que desea sincronizar en el parámetro de configuración de la `synchronize` configuración del administrador de sombras de la implementación.

> **Note**
>
> Para permitir que un dispositivo principal interactúe con las sombras de los dispositivos cliente, también debe configurar e implementar el componente MQTT bridge. Para obtener más información, consulte Habilitar el administrador de sombras para que se comunique con los dispositivos cliente.

El siguiente ejemplo de actualización de configuración indica al componente administrador de sombras que sincronice las siguientes sombras conAWS IoT Core:

- La sombra clásica para el dispositivo principal
- El nombre `MyCoreShadow` del dispositivo principal
- La sombra clásica para una cosa del IoT llamada `MyDevice2`
- Las sombras nombradas `MyShadowA` y `MyShadowB` para una cosa del IoT llamada `MyDevice1`

Esta actualización de configuración especifica sincronizar las sombras con AWS IoT Core ellas en tiempo real. Si utiliza la versión 2.1.0 o posterior del administrador de sombras, puede configurar el componente del administrador de sombras para que sincronice las sombras a intervalos periódicos. Para configurar esta función, cambie la estrategia de sincronización a `periodic` y especifique una `delay` en segundos para el intervalo. Para obtener más información, consulte el parámetro de configuración de la estrategia del componente shadow manager.

Esta actualización de configuración especifica que las sombras se sincronicen en ambas direcciones entre AWS IoT Core y el dispositivo principal. Si utiliza la versión 2.2.0 o posterior del administrador de sombras, puede configurar el componente de gestión de sombras para que sincronice las sombras en una sola dirección. Para configurar esta función, cambie la sincronización `direction` a `deviceToCloud` o. `cloudToDevice` Para obtener más información, consulte el parámetro de configuración de dirección del componente Shadow Manager.

```
{
  "strategy": {
    "type": "realTime"
  },
  "synchronize": {
    "coreThing": {
      "classic": true,
      "namedShadows": [
```

```
        "MyCoreShadow"
      ]
    },
    "shadowDocuments": [
      {
        "thingName": "MyDevice1",
        "classic": false,
        "namedShadows": [
          "MyShadowA",
          "MyShadowB"
        ]
      },
      {
        "thingName": "MyDevice2",
        "classic": true,
        "namedShadows": [ ]
      }
    ],
    "direction": "betweenDeviceAndCloud"
  }
}
```

Sincronice las sombras locales

Cuando el dispositivo principal de Greengrass está conectado a la AWS IoT nube, el administrador de sombras realiza las siguientes tareas para las sombras que especifique en la configuración del componente. El comportamiento depende de la opción de configuración de la dirección de sincronización de sombras que especifique. De forma predeterminada, el administrador de sombras usa la `betweenDeviceAndCloud` opción de sincronizar las sombras en ambas direcciones. Si utiliza la versión 2.2.0 o posterior del administrador de sombras, puede configurar el dispositivo principal para que sincronice las sombras en una sola dirección, que puede ser `cloudToDevice` o. `deviceToCloud`

- Si la configuración de dirección de sincronización de sombras es `betweenDeviceAndCloud` o`cloudToDevice`, el administrador de sombras recupera la información de estado notificada del documento de sombra de la nube en el que se encuentra. AWS IoT Core A continuación, actualiza los documentos ocultos almacenados localmente para sincronizar el estado del dispositivo.
- Si la configuración de dirección de sincronización oculta es `betweenDeviceAndCloud` o`deviceToCloud`, el administrador de sombras publica el estado actual del dispositivo en el documento oculto de la nube.

Comportamiento conflictivo de fusión de sombras

En algunos casos, por ejemplo, cuando el dispositivo principal está desconectado de Internet, es posible que una sombra cambie en el servicio virtual local y en la AWS IoT nube antes de que el administrador virtual sincronice los cambios. Como resultado, los estados deseados y notificados difieren entre el servicio paralelo local y la nube AWS IoT

Cuando el administrador de sombras sincroniza la sombra, fusiona los cambios de acuerdo con el siguiente comportamiento:

- Si utiliza una versión del administrador de sombras anterior a la 2.2.0, o si especifica la dirección de sincronización de las `betweenDeviceAndCloud` sombras, se aplica el siguiente comportamiento:
  - Cuando se produce un conflicto de fusión en el estado deseado de una sombra, el administrador de sombras sobrescribe la sección conflictiva del documento paralelo local con el valor de la nube. AWS IoT
  - Cuando se produce un conflicto de fusión en el estado registrado de una sombra, el administrador de la sombra sobrescribe la sección conflictiva de la sombra de la AWS IoT nube con el valor del documento paralelo local.
- Al especificar la dirección de sincronización de las `deviceToCloud` sombras, el administrador de sombras sobrescribe la sección conflictiva de la sombra en la AWS IoT nube con el valor del documento paralelo local.
- Al especificar la dirección de sincronización de las `cloudToDevice` sombras, el administrador de sombras sobrescribe la sección conflictiva del documento paralelo local con el valor de la nube. AWS IoT

Gestione los flujos de datos en los dispositivos principales de Greengrass

AWS IoT Greengrassstream manager hace que sea más eficiente y confiable transferir datos de IoT de gran volumen alNube de AWS. Stream Manager procesa los flujos de datos en el AWS IoT Greengrass Core antes de exportarlos alNube de AWS. Stream Manager se integra en escenarios periféricos comunes, como la inferencia del aprendizaje automático (ML), en la que el dispositivo AWS IoT Greengrass Core procesa y analiza los datos antes de exportarlos a los destinos de almacenamiento locales Nube de AWS o a los destinos de almacenamiento.

Stream Manager proporciona una interfaz común para simplificar el desarrollo de componentes personalizados, de modo que no sea necesario crear una funcionalidad de administración de transmisiones personalizada. Sus componentes pueden usar un mecanismo estandarizado para procesar transmisiones de gran volumen y administrar las políticas locales de retención de datos. Puede definir políticas para el tipo de almacenamiento, el tamaño y la retención de datos para cada transmisión a fin de controlar la forma en que el administrador de transmisiones procesa y exporta los datos.

Stream Manager funciona en entornos con conectividad intermitente o limitada. Puede definir el uso del ancho de banda, el comportamiento de los tiempos de espera y la forma en que el AWS IoT Greengrass Core gestiona los datos de la transmisión cuando está conectado o desconectado. También puedes establecer prioridades para controlar el orden en el que el AWS IoT Greengrass Core exporta las transmisiones a. Nube de AWS Esto le permite gestionar los datos críticos antes que otros datos.

Puede configurar el administrador de transmisiones para que exporte automáticamente los datos a Nube de AWS él para su almacenamiento o posterior procesamiento y análisis. Stream Manager admite las exportaciones a los siguientes Nube de AWS destinos:

- Canales de entradaAWS IoT Analytics. AWS IoT Analyticsle permite realizar análisis avanzados de sus datos para ayudarle a tomar decisiones empresariales y mejorar los modelos de aprendizaje automático. Para obtener más información, consulte ¿Qué es AWS IoT Analytics? en la Guía del usuario de AWS IoT Analytics.
- Transmite en Amazon Kinesis Data Streams. Puede usar Kinesis Data Streams para agregar grandes volúmenes de datos y cargarlos en un almacén MapReduce de datos o un clúster. Para obtener más información, consulte Qué son los Amazon Kinesis Data Streams en la Guía para desarrolladores de Amazon Kinesis Data Streams.

- Propiedades de los activos en. AWS IoT SiteWise AWS IoT SiteWisele permite recopilar, organizar y analizar datos de equipos industriales a escala. Para obtener más información, consulte ¿Qué es AWS IoT SiteWise? en la Guía del usuario de AWS IoT SiteWise.
- Objetos en Amazon Simple Storage Service Amazon S3. Puede utilizar Amazon S3 para almacenar y recuperar grandes cantidades de datos. Para obtener más información, consulte ¿Qué es Amazon S3? en la Guía para desarrolladores de Amazon Simple Storage Service.

Flujo de trabajo de la administración de secuencias

Sus aplicaciones de IoT interactúan con Stream Manager a través del SDK de Stream Manager.

En un flujo de trabajo simple, un componente del AWS IoT Greengrass núcleo consume datos de IoT, como métricas de temperatura y presión de series temporales. El componente puede filtrar o comprimir los datos y, a continuación, llamar al SDK de Stream Manager para escribir los datos en una transmisión en Stream Manager. El administrador de transmisiones puede exportar la transmisión a la transmisión Nube de AWS automáticamente en función de las políticas que definas para la transmisión. Los componentes también pueden enviar datos directamente a las bases de datos locales o a los repositorios de almacenamiento.

Sus aplicaciones de IoT pueden incluir varios componentes personalizados que leen o escriben en las transmisiones. Estos componentes pueden leer y escribir en las transmisiones para filtrar, agregar y analizar los datos del dispositivo AWS IoT Greengrass principal. Esto permite responder rápidamente a los eventos locales y extraer información valiosa antes de que los datos se transfieran del núcleo a los Nube de AWS destinos locales.

Para empezar, implemente el componente administrador de transmisiones en su dispositivo AWS IoT Greengrass principal. En la implementación, configure los parámetros del componente del administrador de transmisiones para definir los ajustes que se apliquen a todas las transmisiones del dispositivo principal de Greengrass. Utilice estos parámetros para controlar la forma en que el administrador de transmisiones almacena, procesa y exporta las transmisiones en función de las necesidades de su empresa y las limitaciones del entorno.

Después de configurar el administrador de secuencias, puede crear e implementar sus aplicaciones de IoT. Por lo general, se trata de componentes personalizados que se utilizan `StreamManagerClient` en el SDK de Stream Manager para crear transmisiones e interactuar con ellas. Al crear una transmisión, puede definir políticas por transmisión, como los destinos de exportación, la prioridad y la persistencia.

Requisitos

Se aplican los siguientes requisitos para el administrador de flujo:

- El administrador de transmisiones requiere un mínimo de 70 MB de RAM además del software AWS IoT Greengrass Core. El requisito total de memoria depende de la carga de trabajo.
- AWS IoT Greengrasslos componentes deben usar el SDK de Stream Manager para interactuar con Stream Manager. El SDK de Stream Manager está disponible en los siguientes idiomas:
  - SDK de Stream Manager para Java (v1.1.0 o posterior)
  - SDK de Stream Manager para Node.js (v1.1.0 o posterior)
  - SDK de Stream Manager para Python (versión 1.1.0 o posterior)
- AWS IoT Greengrasslos componentes deben especificar el componente del administrador de transmisiones (`aws.greengrass.StreamManager`) como una dependencia en su receta para usar el administrador de transmisiones.

  > **Note**
  >
  > Si usa el administrador de transmisiones para exportar datos a la nube, no puede actualizar la versión 2.0.7 del componente de administrador de transmisiones a una versión entre la v2.0.8 y la v2.0.11. Si va a implementar Stream Manager por primera vez, le recomendamos encarecidamente que implemente la última versión del componente Stream Manager.

- Si define los destinos de Nube de AWS exportación para una transmisión, debe crear sus objetivos de exportación y conceder permisos de acceso en la función de dispositivo de Greengrass. Según el destino, es posible que también se apliquen otros requisitos. Para obtener más información, consulte:
  - the section called “Canales de AWS IoT Analytics”
  - the section called “Amazon Kinesis Data Streams”
  - the section called “AWS IoT SiteWise propiedades de activos”
  - the section called “Objetos de Amazon S3”

  Usted es responsable del mantenimiento de estos recursos de Nube de AWS.

Seguridad de los datos

Cuando utilice el administrador de secuencias, tenga en cuenta las siguientes consideraciones de seguridad.

Seguridad de los datos locales

AWS IoT Greengrassno cifra los datos de la transmisión en reposo o en tránsito entre los componentes locales del dispositivo principal.

- Datos en reposo. Los datos de secuencias se almacenan localmente en un directorio de almacenamiento. Para garantizar la seguridad de los datos, AWS IoT Greengrass se basa en los permisos de los archivos y en el cifrado de disco completo, si está activado. Puede utilizar el parámetro opcional STREAM_MANAGER_STORE_ROOT_DIR para especificar el directorio de almacenamiento. Si cambia este parámetro más adelante para utilizar un directorio de almacenamiento diferente, AWS IoT Greengrass no eliminará el directorio de almacenamiento anterior ni su contenido.
- Datos en tránsito local. AWS IoT Greengrassno cifra los datos de transmisión en tránsito local entre las fuentes de datos, AWS IoT Greengrass los componentes, el SDK de Stream Manager y el administrador de transmisiones.
- Datos en tránsito a la Nube de AWS. Las secuencias de datos exportadas por el administrador de secuencias a la Nube de AWS utilizan cifrado de cliente de servicio AWS estándar con seguridad de la capa de transporte (TLS)

Autenticación del cliente

Los clientes de Stream Manager utilizan el SDK de Stream Manager para comunicarse con Stream Manager. Cuando la autenticación del cliente está habilitada, solo los componentes de Greengrass pueden interactuar con las transmisiones en el administrador de transmisiones. Cuando la autenticación del cliente está deshabilitada, cualquier proceso que se ejecute en el dispositivo principal de Greengrass puede interactuar con las transmisiones en el administrador de transmisiones. Debe deshabilitar la autenticación solo si su caso de negocio lo requiere.

Utilice el parámetro STREAM_MANAGER_AUTHENTICATE_CLIENT para establecer el modo de autenticación del cliente. Puede configurar este parámetro al implementar el componente del administrador de transmisiones en los dispositivos principales.

	Habilitado	Deshabilitad
Valor del parámetro	`true` (predeterminado y recomendado)	`false`
Clientes permitidos	Componentes de Greengrass en el dispositivo principal	Componentes de Greengrass en el dispositivo principal Otros procesos que se ejecutan en el dispositivo del núcleo de Greengrass

Véase también

- the section called “Configurar el administrador de secuencias”
- the section called “Se usa StreamManagerClient para trabajar con transmisiones”
- the section called “Exportación de configuraciones para destinos de nube compatibles”

Cree componentes personalizados que usen el administrador de transmisiones

Utilice el administrador de flujos en los componentes personalizados de Greengrass para almacenar, procesar y exportar datos de dispositivos de IoT. Utilice los procedimientos y ejemplos de esta sección para crear recetas de componentes, artefactos y aplicaciones que funcionen con Stream Manager. Para obtener más información sobre cómo desarrollar y probar componentes, consulteCrear AWS IoT Greengrass componentes.

Temas

- Defina las recetas de componentes que utilizan el administrador de flujos
- Conéctese al administrador de transmisiones en el código de la aplicación

Defina las recetas de componentes que utilizan el administrador de flujos

Para usar el administrador de flujos en un componente personalizado, debe definir el `aws.greengrass.StreamManager` componente como una dependencia. También debes proporcionar el SDK de Stream Manager. Complete las siguientes tareas para descargar y usar el SDK de Stream Manager en el idioma que prefiera.

Usa el SDK de Stream Manager para Java

El SDK de Stream Manager para Java está disponible como un archivo JAR que puede usar para compilar su componente. A continuación, puede crear un JAR de aplicación que incluya el SDK de Stream Manager, definir el JAR de la aplicación como un artefacto componente y ejecutar el JAR de la aplicación durante el ciclo de vida del componente.

Para usar el SDK de Stream Manager para Java

1. Descarga el archivo JAR del SDK for Java de Stream Manager.
2. Realice una de las siguientes acciones para crear artefactos componentes a partir de su aplicación Java y el archivo JAR del SDK de Stream Manager:
   - Cree su aplicación como un archivo JAR que incluya el JAR del SDK de Stream Manager y ejecute este archivo JAR en la receta del componente.
   - Defina el JAR del SDK de Stream Manager como un artefacto componente. Agrega ese artefacto a la ruta de clases cuando ejecutes tu aplicación en la receta de tu componente.

   La receta de su componente podría parecerse a la del siguiente ejemplo. Este componente ejecuta una versión modificada del ejemplo de StreamManagerS3.java, que `StreamManagerS3.jar` incluye el JAR del SDK de 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
```

Para obtener más información sobre cómo desarrollar y probar componentes, consulte[Crear AWS IoT Greengrass componentes](#).

Usa el SDK de Stream Manager para Python

El SDK de Stream Manager para Python está disponible como código fuente que puede incluir en su componente. Crea un archivo ZIP del SDK de Stream Manager, define el archivo ZIP como un artefacto de un componente e instala los requisitos del SDK en el ciclo de vida del componente.

Para usar el SDK de Stream Manager para Python

1. Clona o descarga el repositorio aws-greengrass-stream-manager-sdk-python.

   ```
   git clone git@github.com:aws-greengrass/aws-greengrass-stream-manager-sdk-python.git
   ```

2. Cree un archivo ZIP que contenga la `stream_manager` carpeta, que contiene el código fuente del SDK de Stream Manager para Python. Puedes proporcionar este archivo ZIP como un artefacto componente que el software AWS IoT Greengrass principal descomprimirá cuando instale tu componente. Haga lo siguiente:

   a. Abra la carpeta que contiene el repositorio que clonó o descargó en el paso anterior.

      ```
      cd aws-greengrass-stream-manager-sdk-python
      ```

   b. Comprima la `stream_manager` carpeta en un archivo ZIP llamado`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. Compruebe que el `stream_manager_sdk.zip` archivo contiene la `stream_manager` carpeta y su contenido. Ejecute el siguiente comando para mostrar el contenido del archivo ZIP.

Linux or Unix

```
unzip -l stream_manager_sdk.zip
```

Windows Command Prompt (CMD)

```
tar -tf stream_manager_sdk.zip
```

El resultado de debería parecerse al siguiente.

```
Archive:  aws-greengrass-stream-manager-sdk-python/stream_manager.zip
  Length      Date    Time    Name
---------  ---------- -----   ----
        0  02-24-2021 20:45   stream_manager/
      913  02-24-2021 20:45   stream_manager/__init__.py
     9719  02-24-2021 20:45   stream_manager/utilinternal.py
     1412  02-24-2021 20:45   stream_manager/exceptions.py
     1004  02-24-2021 20:45   stream_manager/util.py
        0  02-24-2021 20:45   stream_manager/data/
   254463  02-24-2021 20:45   stream_manager/data/__init__.py
    26515  02-24-2021 20:45   stream_manager/streammanagerclient.py
---------                     -------
   294026                     8 files
```

3. Copia los artefactos del SDK de Stream Manager a la carpeta de artefactos de tu componente. Además del archivo ZIP del SDK de Stream Manager, tu componente utiliza el `requirements.txt` archivo del SDK para instalar las dependencias del SDK de Stream Manager. Sustituye *~/greengrass-components por* la ruta a la carpeta que utilizas para el desarrollo local.

   Linux or Unix

   ```
   cp {stream_manager_sdk.zip,requirements.txt} ~/greengrass-components/artifacts/
   com.example.StreamManagerS3Python/1.0.0/
   ```

   Windows Command Prompt (CMD)

   ```
   robocopy . %USERPROFILE%\greengrass-components\artifacts
   \com.example.StreamManagerS3Python\1.0.0 stream_manager_sdk.zip
   ```

```
robocopy . %USERPROFILE%\greengrass-components\artifacts
\com.example.StreamManagerS3Python\1.0.0 requirements.txt
```

PowerShell

```
cp .\stream_manager_sdk.zip,.\requirements.txt ~\greengrass-components\artifacts
\com.example.StreamManagerS3Python\1.0.0\
```

4. Crea tu receta de componentes. En la receta, haga lo siguiente:

   a. Defina `stream_manager_sdk.zip` y `requirements.txt` como artefactos.

   b. Defina su aplicación de Python como un artefacto.

   c. En el ciclo de vida de la instalación, instala los requisitos del SDK de Stream Manager desde`requirements.txt`.

   d. En el ciclo de vida de ejecución, añada el SDK de Stream Manager a la `PYTHONPATH` aplicación Python y ejecútela.

   La receta de tus componentes podría parecerse a la del siguiente ejemplo. Este componente ejecuta el ejemplo 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
```

Para obtener más información sobre cómo desarrollar y probar componentes, consulte[Crear AWS IoT Greengrass componentes](#).

Utilice el SDK de Stream Manager para JavaScript

El SDK de Stream Manager JavaScript está disponible como código fuente que puede incluir en su componente. Cree un archivo ZIP del SDK de Stream Manager, defina el archivo ZIP como un artefacto de un componente e instale el SDK en el ciclo de vida del componente.

Para usar el SDK de Stream Manager para JavaScript

1. Clona o descarga el repositorio [aws-greengrass-stream-manager-sdk-js](#).

   ```
   git clone git@github.com:aws-greengrass/aws-greengrass-stream-manager-sdk-js.git
   ```

2. Crea un archivo ZIP que contenga la `aws-greengrass-stream-manager-sdk` carpeta, que contiene el código fuente del SDK de Stream Manager para. JavaScript Puedes proporcionar este archivo ZIP como un artefacto componente que el software AWS IoT Greengrass principal descomprimirá al instalar el componente. Haga lo siguiente:

   a. Abra la carpeta que contiene el repositorio que clonó o descargó en el paso anterior.

      ```
      cd aws-greengrass-stream-manager-sdk-js
      ```

   b. Comprima la `aws-greengrass-stream-manager-sdk` carpeta en un archivo ZIP llamado`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. Compruebe que el `stream-manager-sdk.zip` archivo contiene la `aws-greengrass-stream-manager-sdk` carpeta y su contenido. Ejecute el siguiente comando para mostrar el contenido del archivo ZIP.

Linux or Unix

```
unzip -l stream-manager-sdk.zip
```

Windows Command Prompt (CMD)

```
tar -tf stream-manager-sdk.zip
```

El resultado de debería parecerse al siguiente.

```
Archive:  stream-manager-sdk.zip
  Length      Date    Time    Name
---------  ---------- -----   ----
        0  02-24-2021 22:36   aws-greengrass-stream-manager-sdk/
      369  02-24-2021 22:36   aws-greengrass-stream-manager-sdk/package.json
     1017  02-24-2021 22:36   aws-greengrass-stream-manager-sdk/util.js
     8374  02-24-2021 22:36   aws-greengrass-stream-manager-sdk/utilInternal.js
     1937  02-24-2021 22:36   aws-greengrass-stream-manager-sdk/exceptions.js
        0  02-24-2021 22:36   aws-greengrass-stream-manager-sdk/data/
   353343  02-24-2021 22:36   aws-greengrass-stream-manager-sdk/data/index.js
    22599  02-24-2021 22:36   aws-greengrass-stream-manager-sdk/client.js
      216  02-24-2021 22:36   aws-greengrass-stream-manager-sdk/index.js
---------                     -------
   387855                     9 files
```

3. Copia el artefacto del SDK de Stream Manager a la carpeta de artefactos de tu componente. Sustituye *~/greengrass-components por* la ruta a la carpeta que utilizas para el desarrollo local.

   Linux or Unix

   ```
   cp stream-manager-sdk.zip ~/greengrass-components/artifacts/
   com.example.StreamManagerS3JS/1.0.0/
   ```

   Windows Command Prompt (CMD)

   ```
   robocopy . %USERPROFILE%\greengrass-components\artifacts
   \com.example.StreamManagerS3JS\1.0.0 stream-manager-sdk.zip
   ```

   PowerShell

   ```
   cp .\stream-manager-sdk.zip ~\greengrass-components\artifacts
   \com.example.StreamManagerS3JS\1.0.0\
   ```

4. Crea tu receta de componentes. En la receta, haga lo siguiente:

   a. Definir `stream-manager-sdk.zip` como un artefacto.

   b. Defina su JavaScript aplicación como un artefacto.

   c. En el ciclo de vida de la instalación, instala el SDK de Stream Manager desde el `stream-manager-sdk.zip` artefacto. Este `npm install` comando crea una `node_modules` carpeta que contiene el SDK de Stream Manager y sus dependencias.

   d. En el ciclo de vida de ejecución, añada la `node_modules` carpeta a `NODE_PATH` la aplicación y JavaScript ejecútela.

   La receta de sus componentes podría parecerse a la del siguiente ejemplo. Este componente ejecuta el ejemplo de 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
```

Para obtener más información sobre cómo desarrollar y probar componentes, consulteCrear AWS IoT Greengrass componentes.

Conéctese al administrador de transmisiones en el código de la aplicación

Para conectarte al administrador de transmisiones de tu aplicación, crea una instancia `StreamManagerClient` desde el SDK de Stream Manager. Este cliente se conecta al componente del administrador de transmisiones en su puerto predeterminado 8088 o en el puerto que especifiques. Para obtener más información sobre cómo utilizarla `StreamManagerClient` después de crear una instancia, consulteSe usa StreamManagerClient para trabajar con transmisiones.

Example Ejemplo: Conectarse al administrador de transmisiones con el puerto predeterminado

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 Ejemplo: Conectarse al administrador de transmisiones con un puerto no predeterminado

Si configura el administrador de transmisiones con un puerto que no sea el predeterminado, debe utilizar la comunicación entre procesos para recuperar el puerto de la configuración del componente.

> **Note**
>
> El parámetro de `port` configuración contiene el valor que se especifica `STREAM_MANAGER_SERVER_PORT` al implementar Stream Manager.

Java

```
void connectToStreamManagerWithCustomPort() {
    EventStreamRPCConnection eventStreamRpcConnection =
 IPCUtils.getEventStreamRpcConnection();
    GreengrassCoreIPCClient greengrassCoreIPCClient = new
 GreengrassCoreIPCClient(eventStreamRpcConnection);
```

```
    List<String> keyPath = new ArrayList<>();
    keyPath.add("port");

    GetConfigurationRequest request = new GetConfigurationRequest();
    request.setComponentName("aws.greengrass.StreamManager");
    request.setKeyPath(keyPath);
    GetConfigurationResponse response =
            greengrassCoreIPCClient.getConfiguration(request,
 Optional.empty()).getResponse().get();
    String port = response.getValue().get("port").toString();
    System.out.print("Stream Manager is running on port: " + port);

    final StreamManagerClientConfig config = StreamManagerClientConfig.builder()

 .serverInfo(StreamManagerServerInfo.builder().port(Integer.parseInt(port)).build()).build()

    StreamManagerClient client =
 StreamManagerClientFactory.standard().withClientConfig(config).build();

    // Use the client.
}
```

Python

```
import awsiot.greengrasscoreipc
from awsiot.greengrasscoreipc.model import (
    GetConfigurationRequest
)
from stream_manager import (
    StreamManagerClient
)

TIMEOUT = 10

def connect_to_stream_manager_with_custom_port():
    # Use IPC to get the port from the stream manager component configuration.
    ipc_client = awsiot.greengrasscoreipc.connect()
    request = GetConfigurationRequest()
    request.component_name = "aws.greengrass.StreamManager"
    request.key_path = ["port"]
    operation = ipc_client.new_get_configuration()
    operation.activate(request)
    future_response = operation.get_response()
```

```
    response = future_response.result(TIMEOUT)
    stream_manager_port = str(response.value["port"])

    # Use port to create a stream manager client.
    stream_client = StreamManagerClient(port=stream_manager_port)

    # Use the client.
```

Se usa StreamManagerClient para trabajar con transmisiones

Los componentes de Greengrass definidos por el usuario que se ejecutan en el dispositivo principal de Greengrass pueden usar `StreamManagerClient` el objeto del SDK de Stream Manager para crear transmisiones en Stream Manager y, a continuación, interactuar con las transmisiones. Cuando un componente crea una transmisión, define los Nube de AWS destinos, la priorización y otras políticas de exportación y retención de datos para la transmisión. Para enviar datos al administrador de transmisiones, los componentes agregan los datos a la transmisión. Si se define un destino de exportación para el flujo, el administrador de secuencias exporta la secuencia automáticamente.

> **Note**
>
> Normalmente, los clientes de Stream Manager son componentes de Greengrass definidos por el usuario. Si su modelo de negocio lo requiere, también puede permitir que los procesos no componentes que se ejecutan en el núcleo de Greengrass (por ejemplo, un contenedor de Docker) interactúen con Stream Manager. Para obtener más información, consulte the section called “Autenticación del cliente”.

Los fragmentos de este tema muestran cómo los clientes llaman `StreamManagerClient` métodos para trabajar con secuencias. Para obtener detalles sobre la implementación de los métodos y sus argumentos, utilice los vínculos a la referencia del SDK de cada fragmento.

Si usa el administrador de flujos en una función de Lambda, la función de Lambda debería crearse fuera del controlador de funciones. `StreamManagerClient` Si se crea una instancia en el controlador, la función crea un `client` y una conexión al administrador de secuencias cada vez que se invoca.

> **Note**
>
> Si crea una instancia `StreamManagerClient` en el controlador, debe llamar explícitamente al método `close()` cuando el `client` complete su trabajo. De lo contrario, el `client` mantiene la conexión abierta y otro subproceso en ejecución hasta que salga del script.

`StreamManagerClient` admite las siguientes operaciones:

- the section called “Creación de una secuencia de mensajes”
- the section called “Agregar un mensaje”
- the section called “Lectura de mensajes”
- the section called “Lista de secuencias”
- the section called “Descripción de una secuencia de mensajes”
- the section called “Actualizar una secuencia de mensajes”
- the section called “Eliminación de una secuencia de mensajes”

Creación de una secuencia de mensajes

Para crear una transmisión, un componente de Greengrass definido por el usuario llama al método create y pasa un objeto. `MessageStreamDefinition` Este objeto especifica el nombre exclusivo del flujo y define cómo el administrador del flujo debe gestionar los nuevos datos cuando se alcanza el tamaño máximo de flujo. Puede utilizar `MessageStreamDefinition` y sus tipos de datos (como `ExportDefinition`, `StrategyOnFull` y `Persistence`) para definir otras propiedades de secuencias. Entre ellos se incluyen:

- El destino AWS IoT Analytics, Kinesis Data Streams, AWS IoT SiteWise y Amazon S3 de destino para las exportaciones automáticas. Para obtener más información, consulte the section called “Exportación de configuraciones para destinos de nube compatibles”.
- Prioridad de exportación. El administrador de secuencias exporta secuencias de mayor prioridad antes que secuencias de menor prioridad.
- Tamaño e intervalo de lote máximos para AWS IoT Analytics, Kinesis Data Streams y destinos AWS IoT SiteWise. El administrador de secuencias exporta mensajes cuando se cumple cualquiera de las condiciones.

- T ime-to-live (TTL). La cantidad de tiempo para garantizar que los datos de secuencia están disponibles para su procesamiento. Debe asegurarse de que los datos se pueden consumir dentro de este período de tiempo. Esta no es una política de eliminación. Es posible que los datos no se eliminen inmediatamente después del período TTL.
- Persistencia de secuencia. Elija guardar las secuencias en el sistema de archivos para conservar los datos en los reinicios del núcleo o guardar las secuencias en la memoria.
- Starting sequence number. Especifique el número de secuencia del mensaje que se utilizará como mensaje de inicio en la exportación.

Para obtener más información acerca de `MessageStreamDefinition`, consulte la referencia del SDK para el lenguaje de destino:

- MessageStreamDefinitionen el SDK de Java
- MessageStreamDefinitionen el SDK de Node.js
- MessageStreamDefinitionen el SDK de Python

> **Note**
>
> `StreamManagerClient` también proporciona un destino que puede utilizar para exportar secuencias a un servidor HTTP. Este destino está pensado solo con fines de prueba. No es estable y no se admite para su uso en entornos de producción.

Una vez creada una transmisión, sus componentes de Greengrass pueden añadir mensajes a la transmisión para enviar datos para su exportación y leer los mensajes de la transmisión para su procesamiento local. El número de secuencias que cree depende de sus capacidades de hardware y de su caso de negocio. Una estrategia es crear una secuencia para cada canal de destino en AWS IoT Analytics o secuencia de datos de Kinesis, aunque puede definir varios destinos para una secuencia. Una secuencia tiene una vida útil duradera.

Requisitos

Esta operación tiene los siguientes requisitos:

- Versión mínima del SDK de Stream Manager: Python: 1.1.0 | Java: 1.1.0 | Node.js: 1.1.0

Ejemplos

El siguiente fragmento crea una secuencia denominada `StreamName`. Define las propiedades de las secuencias en `MessageStreamDefinition` y los tipos de datos subordinados.

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 Nube de AWS.
            kinesis=None,
            iot_analytics=None,
            iot_sitewise=None,
            s3_task_executor=None
        )
    ))
except StreamManagerException:
    pass
    # Properly handle errors.
except ConnectionError or asyncio.TimeoutError:
    pass
    # Properly handle errors.
```

Referencia del SDK de Python: create_message_stream | MessageStreamDefinition

Java

```
try (final StreamManagerClient client =
 StreamManagerClientFactory.standard().build()) {
    client.createMessageStream(
            new MessageStreamDefinition()
                    .withName("StreamName") // Required.
                    .withMaxSize(268435456L)    // Default is 256 MB.
```

```
                    .withStreamSegmentSize(16777216L)    // Default is 16 MB.
                    .withTimeToLiveMillis(null)    // By default, no TTL is enabled.
                    .withStrategyOnFull(StrategyOnFull.OverwriteOldestData)    //
 Required.
                    .withPersistence(Persistence.File)    // Default is File.
                    .withFlushOnWrite(false)    // Default is false.
                    .withExportDefinition(    // Optional. Choose where/how the
 stream is exported to the Nube de AWS.
                            new ExportDefinition()
                                    .withKinesis(null)
                                    .withIotAnalytics(null)
                                    .withIotSitewise(null)
                                    .withS3(null)
                    )

    );
} catch (StreamManagerException e) {
    // Properly handle exception.
}
```

Referencia del createMessageStreamSDK de Java: | MessageStreamDefinition

Node.js

```
const client = new StreamManagerClient();
client.onConnected(async () => {
  try {
    await client.createMessageStream(
      new MessageStreamDefinition()
        .withName("StreamName") // Required.
        .withMaxSize(268435456)  // Default is 256 MB.
        .withStreamSegmentSize(16777216)  // Default is 16 MB.
        .withTimeToLiveMillis(null)  // By default, no TTL is enabled.
        .withStrategyOnFull(StrategyOnFull.OverwriteOldestData)  // Required.
        .withPersistence(Persistence.File)  // Default is File.
        .withFlushOnWrite(false)  // Default is false.
        .withExportDefinition(  // Optional. Choose where/how the stream is exported
 to the Nube de AWS.
          new ExportDefinition()
            .withKinesis(null)
            .withIotAnalytics(null)
            .withIotSiteWise(null)
            .withS3(null)
        )
```

```
    );
  } catch (e) {
    // Properly handle errors.
  }
});
client.onError((err) => {
  // Properly handle connection errors.
  // This is called only when the connection to the StreamManager server fails.
});
```

Referencia del SDK de Node.js: createMessageStream| MessageStreamDefinition

Para obtener más información acerca de la configuración de destinos de exportación, consulte the section called “Exportación de configuraciones para destinos de nube compatibles”.

Agregar un mensaje

Para enviar datos al administrador de transmisiones para su exportación, sus componentes de Greengrass anexan los datos a la transmisión de destino. El destino de exportación determina el tipo de datos que se van a transferir a este método.

Requisitos

Esta operación tiene los siguientes requisitos:

- Versión mínima del SDK de Stream Manager: Python: 1.1.0 | Java: 1.1.0 | Node.js: 1.1.0

Ejemplos

AWS IoT Analytics o destinos de exportación de Kinesis Data Streams

El siguiente fragmento añade un mensaje a la secuencia denominada `StreamName`. Para los AWS IoT Analytics destinos de Kinesis Data Streams, sus componentes de Greengrass añaden una masa de datos.

Este fragmento tiene los siguientes requisitos:

- Versión mínima del SDK de Stream Manager: Python: 1.1.0 | Java: 1.1.0 | Node.js: 1.1.0

Python

```
client = StreamManagerClient()

try:
    sequence_number = client.append_message(stream_name="StreamName",
 data=b'Arbitrary bytes data')
except StreamManagerException:
    pass
    # Properly handle errors.
except ConnectionError or asyncio.TimeoutError:
    pass
    # Properly handle errors.
```

Referencia del SDK de 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.
}
```

Referencia del Java de Python: 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.
});
```

Referencia del Node.js SDK: appendMessage

AWS IoT SiteWise destinos de exportación

El siguiente fragmento añade un mensaje a la secuencia denominada `StreamName`. Para AWS IoT SiteWise los destinos, sus componentes de Greengrass añaden un objeto serializado. `PutAssetPropertyValueEntry` Para obtener más información, consulte the section called "Exportación a AWS IoT SiteWise".

> **Note**
>
> Cuando envía datos a AWS IoT SiteWise, los datos deben cumplir todos los requisitos de la acción `BatchPutAssetPropertyValue`. Para obtener más información, consulte BatchPutAssetPropertyValue en la Referencia de la API de AWS IoT SiteWise.

Este fragmento tiene los siguientes requisitos:

- Versión mínima del SDK de Stream Manager: Python: 1.1.0 | Java: 1.1.0 | Node.js: 1.1.0

Python

```
client = StreamManagerClient()

try:
    # SiteWise requires unique timestamps in all messages and also needs timestamps
 not earlier
    # than 10 minutes in the past. Add some randomness to time and offset.

    # Note: To create a new asset property data, you should use the classes defined
 in the
    # greengrasssdk.stream_manager module.

    time_in_nanos = TimeInNanos(
        time_in_seconds=calendar.timegm(time.gmtime()) - random.randint(0, 60),
 offset_in_nanos=random.randint(0, 10000)
    )
    variant = Variant(double_value=random.random())
    asset = [AssetPropertyValue(value=variant, quality=Quality.GOOD,
 timestamp=time_in_nanos)]
```

```
    putAssetPropertyValueEntry =
 PutAssetPropertyValueEntry(entry_id=str(uuid.uuid4()),
 property_alias="PropertyAlias", property_values=asset)
    sequence_number = client.append_message(stream_name="StreamName",
 Util.validate_and_serialize_to_json_bytes(putAssetPropertyValueEntry))
except StreamManagerException:
    pass
    # Properly handle errors.
except ConnectionError or asyncio.TimeoutError:
    pass
    # Properly handle errors.
```

Referencia del SDK de Python: append_message | PutAssetPropertyValueEntry

Java

```
try (final StreamManagerClient client =
 GreengrassClientBuilder.streamManagerClient().build()) {
    Random rand = new Random();
    // Note: To create a new asset property data, you should use the classes defined
 in the
    // com.amazonaws.greengrass.streammanager.model.sitewise package.
    List<AssetPropertyValue> entries = new ArrayList<>() ;

    // IoTSiteWise requires unique timestamps in all messages and also needs
 timestamps not earlier
    // than 10 minutes in the past. Add some randomness to time and offset.
    final int maxTimeRandomness = 60;
    final int maxOffsetRandomness = 10000;
    double randomValue = rand.nextDouble();
    TimeInNanos timestamp = new TimeInNanos()
            .withTimeInSeconds(Instant.now().getEpochSecond() -
 rand.nextInt(maxTimeRandomness))
            .withOffsetInNanos((long) (rand.nextInt(maxOffsetRandomness)));
    AssetPropertyValue entry = new AssetPropertyValue()
            .withValue(new Variant().withDoubleValue(randomValue))
            .withQuality(Quality.GOOD)
            .withTimestamp(timestamp);
    entries.add(entry);

    PutAssetPropertyValueEntry putAssetPropertyValueEntry = new
 PutAssetPropertyValueEntry()
            .withEntryId(UUID.randomUUID().toString())
            .withPropertyAlias("PropertyAlias")
```

```
            .withPropertyValues(entries);
    long sequenceNumber = client.appendMessage("StreamName",
 ValidateAndSerialize.validateAndSerializeToJsonBytes(putAssetPropertyValueEntry));
} catch (StreamManagerException e) {
    // Properly handle exception.
}
```

Referencia del SDK de Java: AppendMessage | PutAssetPropertyValueEntry

Node.js

```
const client = new StreamManagerClient();
client.onConnected(async () => {
    try {
        const maxTimeRandomness = 60;
        const maxOffsetRandomness = 10000;
        const randomValue = Math.random();
        // Note: To create a new asset property data, you should use the classes
 defined in the
        // aws-greengrass-core-sdk StreamManager module.
        const timestamp = new TimeInNanos()
            .withTimeInSeconds(Math.round(Date.now() / 1000) -
 Math.floor(Math.random() - maxTimeRandomness))
            .withOffsetInNanos(Math.floor(Math.random() * maxOffsetRandomness));
        const entry = new AssetPropertyValue()
            .withValue(new Variant().withDoubleValue(randomValue))
            .withQuality(Quality.GOOD)
            .withTimestamp(timestamp);

        const putAssetPropertyValueEntry =    new PutAssetPropertyValueEntry()
            .withEntryId(`${ENTRY_ID_PREFIX}${i}`)
            .withPropertyAlias("PropertyAlias")
            .withPropertyValues([entry]);
        const sequenceNumber = await client.appendMessage("StreamName",
 util.validateAndSerializeToJsonBytes(putAssetPropertyValueEntry));
    } catch (e) {
        // Properly handle errors.
    }
});
client.onError((err) => {
    // Properly handle connection errors.
    // This is called only when the connection to the StreamManager server fails.
});
```

Referencia del SDK de Node.js: AppendMessage | PutAssetPropertyValueEntry

Destinos de exportación de Amazon S3

El siguiente fragmento añade una tarea de exportación a la secuencia denominada `StreamName`. Para los destinos de Amazon S3, sus componentes de Greengrass añaden un `S3ExportTaskDefinition` objeto serializado que contiene información sobre el archivo de entrada de origen y el objeto de Amazon S3 de destino. Si el objeto especificado no existe, el administrador de flujos lo crea por usted. Para obtener más información, consulte the section called "Exportar a Amazon S3.".

Este fragmento tiene los siguientes requisitos:

- Versión mínima del SDK de Stream Manager: Python: 1.1.0 | Java: 1.1.0 | Node.js: 1.1.0

Python

```
client = StreamManagerClient()

try:
    # Append an Amazon S3 Task definition and print the sequence number.
    s3_export_task_definition = S3ExportTaskDefinition(input_url="URLToFile",
 bucket="BucketName", key="KeyName")
    sequence_number = client.append_message(stream_name="StreamName",
 Util.validate_and_serialize_to_json_bytes(s3_export_task_definition))
except StreamManagerException:
    pass
    # Properly handle errors.
except ConnectionError or asyncio.TimeoutError:
    pass
    # Properly handle errors.
```

Referencia del SDK de Python: append_message | S3 ExportTaskDefinition

Java

```
try (final StreamManagerClient client =
 GreengrassClientBuilder.streamManagerClient().build()) {
    // Append an Amazon S3 export task definition and print the sequence number.
    S3ExportTaskDefinition s3ExportTaskDefinition = new S3ExportTaskDefinition()
        .withBucket("BucketName")
```

```
        .withKey("KeyName")
        .withInputUrl("URLToFile");
    long sequenceNumber = client.appendMessage("StreamName",
 ValidateAndSerialize.validateAndSerializeToJsonBytes(s3ExportTaskDefinition));
} catch (StreamManagerException e) {
    // Properly handle exception.
}
```

Referencia del SDK de Java: 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.
});
```

Referencia del SDK de Node.js: AppendMessage | S3 ExportTaskDefinition

Lectura de mensajes

Leer mensajes de un flujo.

Requisitos

Esta operación tiene los siguientes requisitos:

- Versión mínima del SDK de Stream Manager: Python: 1.1.0 | Java: 1.1.0 | Node.js: 1.1.0

Ejemplos

El siguiente fragmento lee los mensajes de la secuencia denominada `StreamName`. El método read toma un objeto `ReadMessagesOptions` opcional que especifica el número de secuencia para comenzar a leer, los números mínimo y máximo para leer y un tiempo de espera para leer mensajes.

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.
```

Referencia del SDK de Python: read_messages | ReadMessagesOptions

Java

```
try (final StreamManagerClient client =
 StreamManagerClientFactory.standard().build()) {
```

```
    List<Message> messages = client.readMessages("StreamName",
            // By default, if no options are specified, it tries to read one message
 from the beginning of the stream.
            new ReadMessagesOptions()
                    // Try to read from sequence number 100 or greater. By default
 this is 0.
                    .withDesiredStartSequenceNumber(100L)
                    // Try to read 10 messages. If 10 messages are not available,
 then NotEnoughMessagesException is raised. By default, this is 1.
                    .withMinMessageCount(10L)
                    // Accept up to 100 messages. By default this is 1.
                    .withMaxMessageCount(100L)
                    // Try to wait at most 5 seconds for the min_messsage_count to
 be fulfilled. By default, this is 0, which immediately returns the messages or an
 exception.
                    .withReadTimeoutMillis(Duration.ofSeconds(5L).toMillis())
    );
} catch (StreamManagerException e) {
    // Properly handle exception.
}
```

Referencia del SDK de 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.
});
```

Referencia del SDK de Node.js: readMessages | ReadMessagesOptions

Lista de secuencias

Obtenga la lista de flujos en el administrador de flujos.

Requisitos

Esta operación tiene los siguientes requisitos:

- Versión mínima del SDK de Stream Manager: Python: 1.1.0 | Java: 1.1.0 | Node.js: 1.1.0

Ejemplos

El siguiente fragmento de código obtiene una lista de las secuencias (por nombre) del administrador de secuencias.

Python

```
client = StreamManagerClient()

try:
    stream_names = client.list_streams()
except StreamManagerException:
    pass
    # Properly handle errors.
except ConnectionError or asyncio.TimeoutError:
    pass
    # Properly handle errors.
```

Referencia del SDK de Python: list_streams

Java

```
try (final StreamManagerClient client =
 StreamManagerClientFactory.standard().build()) {
    List<String> streamNames = client.listStreams();
} catch (StreamManagerException e) {
    // Properly handle exception.
}
```

Referencia del Java de Python: 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.
});
```

Referencia del Node.js SDK: listStreams

Descripción de una secuencia de mensajes

Obtenga metadatos sobre un flujo, incluida la definición, el tamaño y el estado de exportación del flujo.

Requisitos

Esta operación tiene los siguientes requisitos:

- Versión mínima del SDK de Stream Manager: Python: 1.1.0 | Java: 1.1.0 | Node.js: 1.1.0

Ejemplos

El siguiente fragmento de código obtiene metadatos de una secuencia llamada `StreamName`, como su definición, su tamaño y los estados del exportador.

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.
```

Referencia del SDK de 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.
}
```

Referencia del SDK de Java: describeMessageStream

Node.js

```
const client = new StreamManagerClient();
client.onConnected(async () => {
    try {
        const description = await client.describeMessageStream("StreamName");
        const lastErrorMessage = description.exportStatuses[0].errorMessage;
        if (lastErrorMessage) {
            // The last export of export destination 0 failed with some error.
            // Here is the last sequence number that was successfully exported.
            description.exportStatuses[0].lastExportedSequenceNumber;
        }

        if (description.storageStatus.newestSequenceNumber >
            description.exportStatuses[0].lastExportedSequenceNumber) {
            // The end of the stream is ahead of the last exported sequence number.
        }
    } catch (e) {
        // Properly handle errors.
    }
});
client.onError((err) => {
    // Properly handle connection errors.
    // This is called only when the connection to the StreamManager server fails.
});
```

Referencia del SDK de Node.js: describeMessageStream

Actualizar una secuencia de mensajes

Actualiza las propiedades de un flujo existente. Es posible que desee actualizar un flujo si sus requisitos cambian después de crearlo. Por ejemplo:

- Agregue una nueva configuración de exportación para un destino Nube de AWS.
- Aumente el tamaño máximo de un flujo para cambiar la forma en que se exportan o conservan los datos. Por ejemplo, el tamaño del flujo, en combinación con su estrategia en la configuración completa, puede provocar que los datos se eliminen o rechacen antes de que el administrador de flujos pueda procesarlos.
- Pausa y reanuda las exportaciones; por ejemplo, si las tareas de exportación son prolongadas y quiere racionar los datos de carga.

Sus componentes de Greengrass siguen este proceso de alto nivel para actualizar una transmisión:

1. Consiga la descripción del flujo.
2. Actualice las propiedades de destino de los objetos correspondientes `MessageStreamDefinition` y subordinados.
3. Transfiera la actualización `MessageStreamDefinition`. Asegúrese de incluir las definiciones de objetos completas para el flujo actualizado. Las propiedades no definidas revierten a los valores predeterminados.

   Puede especificar el número de secuencia del mensaje que se utilizará como mensaje de inicio en la exportación.

Requisitos

Esta operación tiene los siguientes requisitos:

- Versión mínima del SDK de Stream Manager: Python: 1.1.0 | Java: 1.1.0 | Node.js: 1.1.0

Ejemplos

El siguiente fragmento de código actualiza la secuencia llamada `StreamName`. Actualiza varias propiedades de un flujo que se exporta a 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.
```

Referencia del SDK de Python: updateMessageStream| MessageStreamDefinition

Java

```
try (final StreamManagerClient client =
 GreengrassClientBuilder.streamManagerClient().build()) {
    MessageStreamInfo messageStreamInfo = client.describeMessageStream(STREAM_NAME);
    // Update the message stream with new values.
    client.updateMessageStream(
        messageStreamInfo.getDefinition()
            .withStrategyOnFull(StrategyOnFull.RejectNewData) // Required. Updating
 Strategy on full to reject new data.
           // Max Size update should be greater than initial Max Size defined in
 Create Message Stream request
            .withMaxSize(536870912L) // Update Max Size to 512 MB.
            .withStreamSegmentSize(33554432L) // Update Segment Size to 32 MB.
            .withFlushOnWrite(true) // Update flush on write to true.
            .withPersistence(Persistence.Memory) // Update the persistence to
 Memory.
            .withTimeToLiveMillis(3600000L)    // Update TTL to 1 hour.
            .withExportDefinition(
                // Optional. Choose where/how the stream is exported to the Nube de
 AWS.
                messageStreamInfo.getDefinition().getExportDefinition().
                    // Updating Export definition to add a Kinesis Stream
 configuration.
```

```
                    .withKinesis(new ArrayList<KinesisConfig>() {{
                        add(new KinesisConfig()
                            .withIdentifier(EXPORT_IDENTIFIER)
                            .withKinesisStreamName("test"));
                        }})
            );
} catch (StreamManagerException e) {
    // Properly handle exception.
}
```

Referencia del SDK de 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 Nube
 de AWS.
                    messageStreamInfo.definition.exportDefinition
                        // Updating Export definition to add a Kinesis Stream
 configuration.
                        .withKinesis([new
 KinesisConfig().withIdentifier(uuidv4()).withKinesisStreamName(uuidv4())])
                )
        );
    } catch (e) {
```

```
        // Properly handle errors.
    }
});
client.onError((err) => {
    // Properly handle connection errors.
    // This is called only when the connection to the StreamManager server fails.
});
```

Referencia del SDK de Node.js: | updateMessageStreamMessageStreamDefinition

Restricciones para actualizar los flujos

Se aplican las siguientes restricciones al actualizar flujos. A menos que se indique en la siguiente lista, las actualizaciones se aplican de inmediato.

- No puede actualizar la persistencia de un flujo. Para cambiar este comportamiento, elimine el flujo y crea un flujo que defina la nueva política de persistencia.
- Puede actualizar el tamaño máximo de una transmisión solo en las siguientes condiciones:
  - El tamaño máximo debe ser superior o igual al tamaño actual del flujo. Para encontrar esta información, describa la secuencia y, a continuación, compruebe el estado de almacenamiento del objeto `MessageStreamInfo` devuelto.
  - El tamaño máximo debe ser superior o igual al tamaño del segmento del flujo.
- Puede actualizar el tamaño del segmento de la transmisión a un valor inferior al tamaño máximo del flujo. La configuración actualizada se aplica a los segmentos nuevos.
- Las actualizaciones de la propiedad tiempo de vida (TTL) se aplican a las nuevas operaciones de anexión. Si reduce este valor, es posible que el administrador de flujos también elimine los segmentos existentes que superen el TTL.
- Las actualizaciones de la estrategia en toda la propiedad se aplican a las nuevas operaciones de anexión. Si establece la estrategia para sobrescribir los datos más antiguos, es posible que el administrados de flujos también sobrescriba los segmentos existentes en función del nuevo ajuste.
- Las actualizaciones de la propiedad de vaciar al escribir se aplican a los mensajes nuevos.
- Las actualizaciones de las configuraciones de exportación se aplican a las nuevas exportaciones. La solicitud de actualización debe incluir todas las configuraciones de exportación que desee admitir. De lo contrario, el administrador de flujos las elimina.
  - Al actualizar una configuración de exportación, especifique el identificador de la configuración de exportación de destino.

  - Para añadir una configuración de exportación, especifique un identificador único para la nueva configuración de exportación.
  - Para eliminar una configuración de exportación, omita la configuración de exportación.
- Para actualizar el número de secuencia inicial de una configuración de exportación en un flujo, debe especificar un valor inferior al último número de secuencia. Para encontrar esta información, describa la secuencia y, a continuación, compruebe el estado de almacenamiento del objeto `MessageStreamInfo` devuelto.

Eliminación de una secuencia de mensajes

Elimina un flujo. Cuando elimina una secuencia, todos los datos almacenados para la secuencia se eliminan del disco.

Requisitos

Esta operación tiene los siguientes requisitos:

- Versión mínima del SDK de Stream Manager: Python: 1.1.0 | Java: 1.1.0 | Node.js: 1.1.0

Ejemplos

El siguiente fragmento de código elimina la secuencia llamada `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.
```

Referencia del SDK de Python: deleteMessageStream

Java

```
try (final StreamManagerClient client =
 StreamManagerClientFactory.standard().build()) {
    client.deleteMessageStream("StreamName");
} catch (StreamManagerException e) {
    // Properly handle exception.
}
```

Referencia del SDK de 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.
});
```

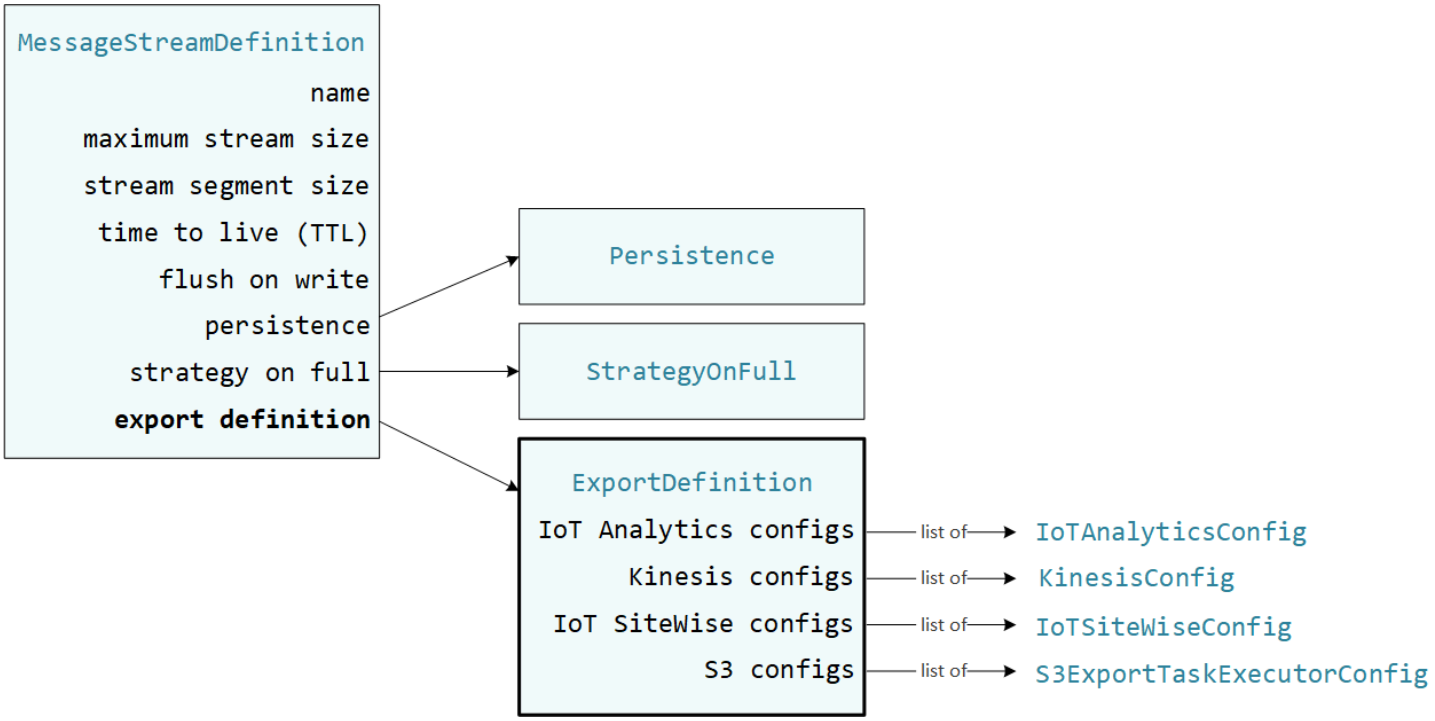
Referencia del SDK de Node.js: deleteMessageStream

Véase también

- Gestione los flujos de datos en los dispositivos principales de Greengrass
- Configurar el administrador de secuencias de AWS IoT Greengrass
- Exportación de configuraciones para Nube de AWS destinos compatibles
- `StreamManagerClient`en la referencia del SDK de Stream Manager:
  - Python
  - Java
  - Node.js

Exportación de configuraciones para Nube de AWS destinos compatibles

Los componentes de Greengrass definidos por el usuario se `StreamManagerClient` utilizan en el SDK de Stream Manager para interactuar con el administrador de transmisiones. Cuando un componente crea una transmisión o actualiza una transmisión, pasa un `MessageStreamDefinition` objeto que representa las propiedades de la transmisión, incluida la definición de exportación. El objeto `ExportDefinition` contiene las configuraciones de exportación definidas para el flujo. El administrador de flujos utiliza estas configuraciones de exportación para determinar dónde y cómo exportar el flujo.



Puede definir cero o más configuraciones de exportación en un flujo, incluidas varias configuraciones de exportación para un único tipo de destino. Por ejemplo, puede exportar un flujo a dos canales AWS IoT Analytics y a un flujo de datos de Kinesis.

En caso de intentos fallidos de exportación, el administrador del flujo vuelve a intentar exportar los datos continuamente al Nube de AWS a intervalos de hasta cinco minutos. La cantidad de reintentos no tiene un límite máximo.

> **Note**
>
> `StreamManagerClient` también proporciona un destino que puede utilizar para exportar secuencias a un servidor HTTP. Este destino está pensado solo con fines de prueba. No es estable y no se admite para su uso en entornos de producción.

Destinos Nube de AWS admitidos

- Canales de AWS IoT Analytics
- Amazon Kinesis Data Streams
- AWS IoT SiteWise propiedades de activos
- Objetos de Amazon S3

Usted es responsable del mantenimiento de estos recursos de Nube de AWS.

Canales de AWS IoT Analytics

El administrador de flujos admite la exportación automática a AWS IoT Analytics. AWS IoT Analytics le permite realizar análisis avanzados de sus datos para ayudarle a tomar decisiones de negocios y mejorar los modelos de machine learning. Para obtener más información, consulte ¿Qué es AWS IoT Analytics? en la AWS IoT Analytics Guía del usuario.

En el SDK de Stream Manager, sus componentes de Greengrass utilizan el `IoTAnalyticsConfig` para definir la configuración de exportación para este tipo de destino. Para obtener más información, consulte la referencia del SDK para el lenguaje de destino.

- IoT AnalyticsConfig en el SDK de Python
- IoT AnalyticsConfig en el SDK de Java
- IoT AnalyticsConfig en el SDK de Node.js

Requisitos

Este destino de exportación tiene los siguientes requisitos:

- Los canales de destino AWS IoT Analytics deben estar en el mismo Cuenta de AWS y Región de AWS igual que el dispositivo principal de Greengrass.

- El Autorizar a los dispositivos principales a interactuar con AWS los servicios debe permitir el permiso `iotanalytics:BatchPutMessage` para segmentar los canales. Por ejemplo:

```
{
  "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"
      ]
    }
  ]
}
```

  Puede conceder acceso granular o condicional a recursos (por ejemplo, utilizando un esquema de nomenclatura con comodín *) Para obtener más información, consulte Adición y eliminación de políticas de IAM en la Guía del usuario de IAM.

Exportación a AWS IoT Analytics

Para crear una transmisión a la que se exporteAWS IoT Analytics, sus componentes de Greengrass crean una transmisión con una definición de exportación que incluye uno o más `IoTAnalyticsConfig` objetos. Este objeto define los ajustes de exportación, como el canal de destino, el tamaño del lote, el intervalo del lote y la prioridad.

Cuando sus componentes de Greengrass reciben datos de los dispositivos, añaden mensajes que contienen una masa de datos a la transmisión de destino.

A continuación, el administrador de flujos exporta los datos en función de los ajustes del lote y la prioridad definidos en las configuraciones de exportación del flujo.

Amazon Kinesis Data Streams

El administrador de flujos permite exportar automáticamente a Amazon Kinesis Data Streams. Kinesis Data Streams se utiliza normalmente para agregar grandes volúmenes de datos y cargarlos

en un almacén MapReduce de datos o un clúster. Para obtener más información, consulte ¿Qué son los Amazon Kinesis Data Streams? en la Guía para desarrolladores de Amazon Kinesis.

En el SDK de Stream Manager, sus componentes de Greengrass utilizan el `KinesisConfig` para definir la configuración de exportación para este tipo de destino. Para obtener más información, consulte la referencia del SDK para el lenguaje de destino.

- KinesisConfigen el SDK de Python
- KinesisConfigen el SDK de Java
- KinesisConfigen el SDK de Node.js

Requisitos

Este destino de exportación tiene los siguientes requisitos:

- Las transmisiones de destino de Kinesis Data Streams deben estar en el Cuenta de AWS mismo dispositivo principal de Greengrass Región de AWS y en el mismo.
- El Autorizar a los dispositivos principales a interactuar con AWS los servicios debe permitir el permiso `kinesis:PutRecords` para destinarse a flujos de datos. Por ejemplo:

```
{
  "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"
      ]
    }
  ]
}
```

Puede conceder acceso granular o condicional a recursos (por ejemplo, utilizando un esquema de nomenclatura con comodín *) Para obtener información, consulte Adición y eliminación de políticas de IAM en la Guía del usuario de IAM.

Exportar a Kinesis Data Streams

Para crear una transmisión que se exporte a Kinesis Data Streams, sus componentes de Greengrass crean una transmisión con una definición de exportación que incluye uno o más objetos. `KinesisConfig` Este objeto define los ajustes de exportación, como el flujo de datos de destino, el tamaño del lote, el intervalo del lote y la prioridad.

Cuando sus componentes de Greengrass reciben datos de los dispositivos, añaden mensajes que contienen una masa de datos a la transmisión de destino. A continuación, el administrador de flujos exporta los datos en función de los ajustes del lote y la prioridad definidos en las configuraciones de exportación del flujo.

El administrador de flujos genera un UUID aleatorio único como clave de partición para cada registro cargado en Amazon Kinesis.

AWS IoT SiteWise propiedades de activos

El administrador de flujos admite la exportación automática a AWS IoT SiteWise. AWS IoT SiteWise le permite recopilar, organizar y analizar datos de equipos industriales a escala. Para obtener más información, consulte ¿Qué es AWS IoT SiteWise? en la AWS IoT SiteWiseGuía del usuario.

En el SDK de Stream Manager, sus componentes de Greengrass utilizan el `IoTSiteWiseConfig` para definir la configuración de exportación para este tipo de destino. Para obtener más información, consulte la referencia del SDK para el lenguaje de destino.

- IoT SiteWiseConfig en el SDK de Python
- IoT SiteWiseConfig en el SDK de Java
- IoT SiteWiseConfig en el SDK de Node.js

> **Note**
>
> AWStambién proporciona AWS IoT SiteWise componentes, que ofrecen una solución prediseñada que puede utilizar para transmitir datos desde fuentes OPC-UA. Para obtener más información, consulte Colector IoT SiteWise OPC-UA.

Requisitos

Este destino de exportación tiene los siguientes requisitos:

- Las propiedades de los activos de destino AWS IoT SiteWise deben estar en el mismo Cuenta de AWS y Región de AWS igual que el dispositivo principal de Greengrass.

> **Note**
>
> Para ver la lista de Región de AWS dispositivos AWS IoT SiteWise compatibles, consulte los AWS IoT SiteWisepuntos finales y las cuotas en la Referencia AWSgeneral.

- Autorizar a los dispositivos principales a interactuar con AWS los serviciosEl `iotsitewise:BatchPutAssetPropertyValue` debe permitir que el permiso se dirija a las propiedades de los activos. El siguiente ejemplo de política utiliza la clave `iotsitewise:assetHierarchyPath` de condición para conceder acceso a un activo raíz de destino y a sus elementos secundarios. Puede quitar `Condition` de la política para permitir el acceso a todos sus recursos AWS IoT SiteWise o especificar ARN de los activos individuales.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
       "Effect": "Allow",
       "Action": "iotsitewise:BatchPutAssetPropertyValue",
       "Resource": "*",
       "Condition": {
         "StringLike": {
           "iotsitewise:assetHierarchyPath": [
             "/root node asset ID",
             "/root node asset ID/*"
           ]
         }
       }
    }
  ]
}
```

Puede conceder acceso granular o condicional a recursos (por ejemplo, utilizando un esquema de nomenclatura con comodín *) Para obtener información, consulte Adición y eliminación de políticas de IAM en la Guía del usuario de IAM.

Para obtener información de seguridad importante, consulte la BatchPutAssetPropertyValue autorización en la Guía del AWS IoT SiteWise usuario.

Exportación a AWS IoT SiteWise

Para crear una transmisión a la que se exporteAWS IoT SiteWise, sus componentes de Greengrass crean una transmisión con una definición de exportación que incluye uno o más `IoTSiteWiseConfig` objetos. Este objeto define los ajustes de exportación, como el tamaño del lote, el intervalo del lote y la prioridad.

Cuando sus componentes de Greengrass reciben datos de propiedades de activos de los dispositivos, añaden mensajes que contienen los datos al flujo de destino. Los mensajes son objetos `PutAssetPropertyValueEntry` serializados en JSON que contienen los valores de propiedad de una o más propiedades de los activos. Para obtener más información, consulte Añadir un mensaje para los AWS IoT SiteWisedestinos de exportación.

> **Note**
>
> Cuando envía datos a AWS IoT SiteWise, los datos deben cumplir todos los requisitos de la acción `BatchPutAssetPropertyValue`. Para obtener más información, consulte BatchPutAssetPropertyValue en la Referencia de la API de AWS IoT SiteWise.

A continuación, el administrador de flujos exporta los datos en función de los ajustes del lote y la prioridad definidos en las configuraciones de exportación del flujo.

Puede ajustar la configuración del administrador de transmisiones y la lógica de los componentes de Greengrass para diseñar su estrategia de exportación. Por ejemplo:

- Para realizar exportaciones prácticamente en tiempo real, establezca ajustes del tamaño de lote e intervalos bajos y añada los datos al flujo cuando lo reciba.
- Para optimizar el procesamiento por lotes, mitigar las restricciones de ancho de banda o minimizar los costos, sus componentes de Greengrass pueden agrupar timestamp-quality-value los puntos de datos (TQV) recibidos para una sola propiedad de activo antes de agregar los datos a la transmisión. Una estrategia consiste en agrupar las entradas de hasta 10 combinaciones diferentes de propiedades y activos (o alias de propiedades) en un mensaje, en lugar de enviar más de una entrada para la misma propiedad. Esto ayuda al Administrador de flujos a mantenerse dentro de las cuotas de AWS IoT SiteWise.

Objetos de Amazon S3

El administrador de flujos permite exportar automáticamente a Amazon S3. Puede utilizar Amazon S3 para almacenar y recuperar grandes cantidades de datos. Para obtener más información, consulte ¿Qué es Amazon S3? en la Guía para desarrolladores de Amazon Simple Storage Service.

En el SDK de Stream Manager, sus componentes de Greengrass utilizan el `S3ExportTaskExecutorConfig` para definir la configuración de exportación para este tipo de destino. Para obtener más información, consulte la referencia del SDK para el lenguaje de destino.

- S3 ExportTaskExecutorConfig en el SDK de Python
- S3 ExportTaskExecutorConfig en el SDK de Java
- S3 ExportTaskExecutorConfig en el SDK de Node.js

Requisitos

Este destino de exportación tiene los siguientes requisitos:

- Los buckets Amazon S3 de destino deben estar en el mismo lugar Cuenta de AWS que el dispositivo principal de Greengrass.
- Si una función Lambda que se ejecuta en el modo contenedor de Greengrass escribe archivos de entrada en un directorio de archivos de entrada, debe montar el directorio como un volumen en el contenedor con permisos de escritura. Esto garantiza que los archivos se escriban en el sistema de archivos raíz y sean visibles para el componente del administrador de flujos, que se ejecuta fuera del contenedor.
- Si un componente de contenedor de Docker escribe archivos de entrada en un directorio de archivos de entrada, debe montar el directorio como un volumen en el contenedor con permisos de escritura. Esto garantiza que los archivos se escriban en el sistema de archivos raíz y sean visibles para el componente del administrador de flujos, que se ejecuta fuera del contenedor.
- El Autorizar a los dispositivos principales a interactuar con AWS los servicios debe permitir los siguientes permisos para los buckets de destino. Por ejemplo:

```
{
  "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/*"
      ]
    }
  ]
}
```

Puede conceder acceso granular o condicional a recursos (por ejemplo, utilizando un esquema de nomenclatura con comodín *) Para obtener información, consulte Adición y eliminación de políticas de IAM en la Guía del usuario de IAM.

Exportar a Amazon S3.

Para crear una transmisión que se exporte a Amazon S3, sus componentes de Greengrass utilizan el `S3ExportTaskExecutorConfig` objeto para configurar la política de exportación. La política define los ajustes de exportación, como el umbral de carga multiparte y la prioridad. Para las exportaciones de Amazon S3, el administrador de flujos carga los datos que lee de los archivos locales en el dispositivo principal. Para iniciar una carga, sus componentes de Greengrass añaden una tarea de exportación a la transmisión de destino. La tarea de exportación contiene información sobre el archivo de entrada y el objeto Amazon S3 de destino. El administrador de transmisiones ejecuta las tareas en la secuencia en que se adjuntan a la transmisión.

> **Note**
>
> El bucket de destino ya debe existir en su Cuenta de AWS. Si no existe un objeto para la clave especificada, el administrador de flujos crea el objeto automáticamente.

El administrador de flujos utiliza la propiedad de umbral de carga multiparte, el ajuste del tamaño mínimo de las piezas y el tamaño del archivo de entrada para determinar cómo cargar los datos. El umbral de carga multiparte debe ser igual o mayor que el tamaño mínimo de la pieza. Si desea cargar datos en paralelo, puede crear varios flujos.

Las claves que especifican los objetos de Amazon S3 de destino pueden incluir DateTimeFormatter cadenas Java válidas en `!{timestamp:value}` los marcadores de posición. Puede utilizar estos marcadores de fecha y hora para particionar los datos en Amazon S3 en función de la hora en la que se cargaron los datos del archivo de entrada. Por ejemplo, el siguiente nombre de clave se resuelve en un valor como `my-key/2020/12/31/data.txt`.

```
my-key/!{timestamp:YYYY}/!{timestamp:MM}/!{timestamp:dd}/data.txt
```

> **Note**
>
> Si desea supervisar el estado de exportación de un flujo, cree primero un flujo de estado y, a continuación, configure el flujo de exportación para utilizarlo. Para obtener más información, consulte the section called “Supervise las tareas de exportación”.

Administrar datos de entrada

Puede crear un código que las aplicaciones de IoT usen para administrar el ciclo de vida de los datos de entrada. El siguiente ejemplo de flujo de trabajo muestra cómo se pueden utilizar los componentes de Greengrass para gestionar estos datos.

1. Un proceso local recibe datos de dispositivos o periféricos y, a continuación, los escribe en los archivos de un directorio del dispositivo principal. Estos son los archivos de entrada del administrador de flujos.
2. Un componente de Greengrass escanea el directorio y añade una tarea de exportación a la secuencia de destino cuando se crea un nuevo archivo. La tarea es un objeto `S3ExportTaskDefinition` serializado en JSON que especifica la URL del archivo de entrada, el bucket y la clave de Amazon S3 de destino y los metadatos de usuario opcionales.
3. El administrador de flujos lee el archivo de entrada y exporta los datos a Amazon S3 en el orden de las tareas anexas. El bucket de destino ya debe existir en su Cuenta de AWS. Si no existe un objeto para la clave especificada, el administrador de flujos crea el objeto automáticamente.
4. El componente Greengrass lee los mensajes de un flujo de estado para supervisar el estado de la exportación. Una vez finalizadas las tareas de exportación, el componente Greengrass puede eliminar los archivos de entrada correspondientes. Para obtener más información, consulte the section called “Supervise las tareas de exportación”.

Supervise las tareas de exportación

Puede crear un código que las aplicaciones de IoT utilizan para monitorear el estado de sus exportaciones de Amazon S3. Sus componentes de Greengrass deben crear un flujo de estado y, a continuación, configurar el flujo de exportación para escribir actualizaciones de estado en el flujo de estado. Una sola transmisión de estado puede recibir actualizaciones de estado de varias transmisiones que se exportan a Amazon S3.

En primer lugar, cree un flujo para utilizarlo como flujo de estado. Puede configurar las políticas de tamaño y retención del flujo para controlar la vida útil de los mensajes de estado. Por ejemplo:

- Configure `Persistence` en `Memory` si no desea guardar los mensajes de estado.
- Configure `StrategyOnFull` en `OverwriteOldestData` para que no se pierdan los nuevos mensajes de estado.

A continuación, cree o actualice el flujo de exportación para usar el flujo de estado.
En concreto, defina la propiedad de configuración de estado de la configuración de `S3ExportTaskExecutorConfig` exportación del flujo. Esta configuración indica al administrador de transmisiones que escriba mensajes de estado sobre las tareas de exportación en la secuencia de estado. En el objeto `StatusConfig`, especifique el nombre del flujo de estado y el nivel de detalle. Los siguientes valores admitidos van desde el menos detallado (`ERROR`) al más detallado (). `TRACE` El valor predeterminado es `INFO`.

- `ERROR`
- `WARN`
- `INFO`
- `DEBUG`
- `TRACE`

El siguiente ejemplo de flujo de trabajo muestra cómo los componentes de Greengrass pueden utilizar un flujo de estado para supervisar el estado de la exportación.

1. Como se describió en el flujo de trabajo anterior, un componente de Greengrass añade una tarea de exportación a un flujo que está configurado para escribir mensajes de estado sobre las tareas de exportación en un flujo de estado. La operación de incorporación devuelve un número de secuencia que representa el ID de la tarea.

2. Un componente de Greengrass lee los mensajes secuencialmente del flujo de estado y, a continuación, filtra los mensajes en función del nombre del flujo y el ID de la tarea o en función de una propiedad de la tarea de exportación del contexto del mensaje. Por ejemplo, el componente Greengrass puede filtrar por la URL del archivo de entrada de la tarea de exportación, que está representada por el `S3ExportTaskDefinition` objeto en el contexto del mensaje.

   Los siguientes códigos de estado indican que una tarea de exportación ha alcanzado un estado completo:

   - `Success`. Se ha completado correctamente la carga.
   - `Failure`. El administrador de flujos detectó un error; por ejemplo, el bucket especificado no existe. Tras resolver el problema, puede volver a añadir la tarea de exportación al flujo.
   - `Canceled`. La tarea se detuvo porque se eliminó la definición de transmisión o exportación o porque el período time-to-live (TTL) de la tarea expiró.

   > **Note**
   >
   > La tarea también puede tener un estado de `InProgress` o `Warning`. El administrador de flujos emite advertencias cuando un evento devuelve un error que no afecta a la ejecución de la tarea. Por ejemplo, si no se limpia una carga parcial, aparecerá una advertencia.

3. Una vez finalizadas las tareas de exportación, el componente Greengrass puede eliminar los archivos de entrada correspondientes.

El siguiente ejemplo muestra cómo un componente de Greengrass puede leer y procesar los mensajes de estado.

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.
```

Referencia del SDK de Python: read_messages | StatusMessage

Java

```
import com.amazonaws.greengrass.streammanager.client.StreamManagerClient;
import com.amazonaws.greengrass.streammanager.client.StreamManagerClientFactory;
import com.amazonaws.greengrass.streammanager.client.utils.ValidateAndSerialize;
import com.amazonaws.greengrass.streammanager.model.ReadMessagesOptions;
import com.amazonaws.greengrass.streammanager.model.Status;
import com.amazonaws.greengrass.streammanager.model.StatusConfig;
import com.amazonaws.greengrass.streammanager.model.StatusLevel;
import com.amazonaws.greengrass.streammanager.model.StatusMessage;

 try (final StreamManagerClient client =
 StreamManagerClientFactory.standard().build()) {
    try {
        boolean isS3UploadComplete = false;
        while (!isS3UploadComplete) {
            try {
                // Read the statuses from the export status stream
                List<Message> messages = client.readMessages("StatusStreamName",
                    new
 ReadMessagesOptions().withMinMessageCount(1L).withReadTimeoutMillis(1000L));
                for (Message message : messages) {
                    // Deserialize the status message first.
                    StatusMessage statusMessage =
 ValidateAndSerialize.deserializeJsonBytesToObj(message.getPayload(),
 StatusMessage.class);
                    // Check the status of the status message. If the status is
 "Success", the file was successfully uploaded to S3.
                    // If the status was either "Failure" or "Canceled", the server
 was unable to upload the file to S3.
                    // We will print the message for why the upload to S3 failed
 from the status message.
                    // If the status was "InProgress", the status indicates that the
 server has started uploading the S3 task.
                    if (Status.Success.equals(statusMessage.getStatus())) {
                        System.out.println("Successfully uploaded file at path " +
 FILE_URL + " to S3.");
                        isS3UploadComplete = true;
```

```
                    } else if (Status.Failure.equals(statusMessage.getStatus()) ||
 Status.Canceled.equals(statusMessage.getStatus())) {
                        System.out.println(String.format("Unable to upload file at
 path %s to S3. Message %s",

 statusMessage.getStatusContext().getS3ExportTaskDefinition().getInputUrl(),
                            statusMessage.getMessage()));
                        sS3UploadComplete = true;
                    }
                }
            } catch (StreamManagerException ignored) {
            } finally {
                // Sleep for sometime for the S3 upload task to complete before
 trying to read the status message.
                Thread.sleep(5000);
            }
        } catch (e) {
        // Properly handle errors.
    }
} catch (StreamManagerException e) {
    // Properly handle exception.
}
```

Referencia del SDK de 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.
});
```

Referencia del SDK de Node.js: readMessages | StatusMessage

Configurar el administrador de secuencias de AWS IoT Greengrass

En los dispositivos principales de Greengrass, el administrador de transmisiones puede almacenar, procesar y exportar datos de dispositivos de IoT. El administrador de transmisiones proporciona los parámetros que se utilizan para configurar los ajustes de tiempo de ejecución. Estos ajustes se aplican a todas las transmisiones del dispositivo principal de Greengrass. Puede usar la AWS IoT Greengrass consola o la API para configurar los ajustes del administrador de transmisiones al implementar el componente. Los cambios se aplican una vez finalizada la implementación.

Parámetros del administrador de secuencias

Stream Manager proporciona los siguientes parámetros que puede configurar al implementar el componente en sus dispositivos principales. Todos los parámetros son opcionales.

Directorio de almacenamiento

Nombre del parámetro: `STREAM_MANAGER_STORE_ROOT_DIR`

La ruta absoluta de la carpeta local utilizada para almacenar las transmisiones. Este valor debe comenzar con una barra inclinada (por ejemplo, `/data`).

Debe especificar una carpeta existente y el usuario del sistema que ejecuta el componente del administrador de transmisiones debe tener permisos para leer y escribir en esta carpeta. Por ejemplo, puede ejecutar los siguientes comandos para crear y configurar una carpeta`/var/greengrass/streams`, que especifique como carpeta raíz del administrador de flujos. Estos comandos permiten al usuario predeterminado del sistema`ggc_user`,, leer y escribir en esta carpeta.

```
sudo mkdir /var/greengrass/streams
sudo chown ggc_user /var/greengrass/streams
sudo chmod 700 /var/greengrass/streams
```

Para obtener información sobre cómo proteger los datos de secuencias, consulte the section called “Seguridad de los datos locales”.

Valor predeterminado: *`/greengrass/v2`*`/work/aws.greengrass.StreamManager`

Puerto del servidor

Nombre del parámetro: `STREAM_MANAGER_SERVER_PORT`

El número de puerto local utilizado para comunicarse con el administrador de secuencias. El valor predeterminado es `8088`.

Puede especificar el uso `0` de un puerto disponible de forma aleatoria.

Autenticar cliente

Nombre del parámetro: `STREAM_MANAGER_AUTHENTICATE_CLIENT`

Indica si los clientes deben autenticarse para interactuar con el administrador de secuencias. El SDK de Stream Manager controla toda la interacción entre los clientes y el administrador de transmisiones. Este parámetro determina qué clientes pueden llamar al SDK de Stream Manager para trabajar con las transmisiones. Para obtener más información, consulte the section called “Autenticación del cliente”.

Los valores válidos son `true` o `false`. El valor predeterminado es `true` (recomendado).

- `true`. Solo permite como clientes los componentes de Greengrass. Los componentes utilizan protocolos AWS IoT Greengrass básicos internos para autenticarse con el SDK de Stream Manager.
- `false`. Permite que cualquier proceso que se ejecute en el AWS IoT Greengrass Core sea un cliente. No establezca el valor en, a `false` menos que su modelo de negocio lo requiera. Por ejemplo, úselo `false` solo si los procesos que no son componentes del dispositivo principal deben comunicarse directamente con el administrador de transmisiones.

Ancho de banda máximo

Nombre del parámetro: `STREAM_MANAGER_EXPORTER_MAX_BANDWIDTH`

El ancho de banda máximo promedio (en kilobits por segundo) que se puede utilizar para exportar datos. El valor predeterminado permite el uso ilimitado del ancho de banda disponible.

Tamaño del grupo de subprocesos

Nombre del parámetro: `STREAM_MANAGER_EXPORTER_THREAD_POOL_SIZE`

Cantidad máxima de subprocesos activos que se pueden utilizar para exportar datos. El valor predeterminado es `5`.

El tamaño óptimo depende del hardware, el volumen de secuencias y la cantidad planificada de secuencias de exportación. Si la velocidad de exportación es lenta, puede ajustar esta configuración para encontrar el tamaño óptimo para su hardware y su caso de negocio. La CPU

y la memoria del hardware del dispositivo principal son factores limitantes. Para comenzar, puede intentar establecer este valor igual a la cantidad de núcleos de procesador en el dispositivo.

Tenga cuidado de no establecer un tamaño superior al que admite el hardware. Cada transmisión consume recursos de hardware, así que intente limitar la cantidad de transmisiones de exportación en dispositivos restringidos.

Argumentos de JVM

Nombre del parámetro: `JVM_ARGS`

Argumentos personalizados de la máquina virtual de Java para pasar al administrador de secuencias al inicio. Varios argumentos deben separarse por espacios.

Utilice este parámetro sólo cuando deba anular la configuración predeterminada utilizada por la JVM. Por ejemplo, puede que necesite aumentar el tamaño predeterminado del montón si planea exportar un gran número de secuencias.

Nivel de registro

Nombre del parámetro: `LOG_LEVEL`

El nivel de registro del componente. Elija uno de los siguientes niveles de registro, que se muestran aquí en orden de niveles:

- `TRACE`
- `DEBUG`
- `INFO`
- `WARN`
- `ERROR`

Valor predeterminado: `INFO`

Tamaño mínimo para la carga de varias partes

Nombre del parámetro: `STREAM_MANAGER_EXPORTER_S3_DESTINATION_MULTIPART_UPLOAD_MIN_PART_SIZE_BYTES`

El tamaño mínimo (en bytes) de una parte en una carga multiparte a Amazon S3. El administrador de secuencias utiliza esta configuración y el tamaño del archivo de entrada para determinar cómo agrupar los datos en una solicitud PUT de varias partes. El valor por defecto y mínimo es de `5242880` bytes (5 MB).

> **Note**
>
> El administrador de secuencias usa la propiedad `sizeThresholdForMultipartUploadBytes` de la transmisión para determinar si se debe exportar a Amazon S3 como una carga única o multiparte. Los componentes de Greengrass definidos por el usuario establecen este umbral cuando crean una transmisión que se exporta a Amazon S3. El umbral de tamaño predeterminado es 5 MB.

Véase también

- Gestione los flujos de datos en los dispositivos principales de Greengrass
- Se usa StreamManagerClient para trabajar con transmisiones
- Exportación de configuraciones para Nube de AWS destinos compatibles

Cómo realizar la inferencia de machine learning

Con élAWS IoT Greengrass, puede realizar inferencias de aprendizaje automático (ML) en sus dispositivos periféricos a partir de datos generados localmente mediante modelos entrenados en la nube. Benefíciese de la baja latencia y el ahorro de costos que supone la ejecución de inferencias locales, aprovechando al mismo tiempo la potencia de cómputo de la nube para el entrenamiento de modelos y el procesamiento complejo.

AWS IoT Greengrasshace que los pasos necesarios para realizar la inferencia sean más eficientes. Puede entrenar sus modelos de inferencia en cualquier lugar e implementarlos localmente como componentes de aprendizaje automático. Por ejemplo, puede crear y entrenar modelos de aprendizaje profundo en Amazon SageMaker o modelos de visión artificial en Amazon Lookout for Vision. A continuación, puede almacenar estos modelos en un bucket de Amazon S3, de forma que pueda utilizarlos como artefactos en sus componentes para realizar inferencias en sus dispositivos principales.

Temas

Funcionamiento de la inferencia de machine learning de AWS IoT Greengrass

AWSproporciona componentes de aprendizaje automático que puede usar para crear implementaciones en un solo paso para realizar inferencias de aprendizaje automático en su

dispositivo. También puede utilizar estos componentes como plantillas para crear componentes personalizados que se adapten a sus requisitos específicos.

AWSproporciona las siguientes categorías de componentes de aprendizaje automático:

- Componente de modelo: contiene modelos de aprendizaje automático como artefactos de Greengrass.
- Componente de tiempo de ejecución: contiene el script que instala el marco de aprendizaje automático y sus dependencias en el dispositivo principal de Greengrass.
- Componente de inferencia: contiene el código de inferencia e incluye las dependencias de los componentes para instalar el marco de aprendizaje automático y descargar modelos de aprendizaje automático previamente entrenados.

Cada implementación que cree para realizar inferencias de aprendizaje automático consta de al menos un componente que ejecuta la aplicación de inferencia, instala el marco de aprendizaje automático y descarga sus modelos de aprendizaje automático. Para realizar una inferencia de ejemplo con los componentes AWS proporcionados, debe implementar un componente de inferencia en el dispositivo principal, que incluye automáticamente los componentes del modelo y del tiempo de ejecución correspondientes como dependencias. Para personalizar las implementaciones, puede conectar o cambiar los componentes del modelo de muestra por componentes de modelo personalizados, o puede utilizar las recetas de componentes de los componentes AWS proporcionados como plantillas para crear sus propios componentes personalizados de inferencia, modelo y tiempo de ejecución.

Para realizar inferencias de aprendizaje automático mediante componentes personalizados:

1. Cree un componente de modelo. Este componente contiene los modelos de aprendizaje automático que desea utilizar para realizar inferencias. AWSproporciona ejemplos de modelos DLR y TensorFlow Lite previamente entrenados. Para usar un modelo personalizado, cree su propio componente de modelo.
2. Cree un componente de tiempo de ejecución. Este componente contiene los scripts necesarios para instalar el tiempo de ejecución de aprendizaje automático en sus modelos. AWSproporciona ejemplos de componentes de tiempo de ejecución para Deep Learning Runtime (DLR) y TensorFlow Lite. Para usar otros tiempos de ejecución con sus modelos personalizados y su código de inferencia, cree sus propios componentes de tiempo de ejecución.

3. Cree un componente de inferencia. Este componente contiene el código de inferencia e incluye los componentes del modelo y del tiempo de ejecución como dependencias. AWSproporciona ejemplos de componentes de inferencia para la clasificación de imágenes y la detección de objetos mediante DLR y Lite. TensorFlow Para realizar otros tipos de inferencias, o para utilizar modelos y tiempos de ejecución personalizados, cree su propio componente de inferencia.
4. Implemente el componente de inferencia. Al implementar este componente, AWS IoT Greengrass también implementa automáticamente las dependencias del modelo y del componente de tiempo de ejecución.

Para empezar con los componentes AWS proporcionados, consulte. the section called “Realice una inferencia de clasificación de imágenes de muestra”

Para obtener información sobre la creación de componentes de aprendizaje automático personalizados, consultePersonalice sus componentes de aprendizaje automático.

¿Qué hay de diferente en AWS IoT Greengrass la versión 2?

AWS IoT Greengrassconsolida las unidades funcionales del aprendizaje automático (como los modelos, los tiempos de ejecución y el código de inferencia) en componentes que permiten utilizar un proceso de un solo paso para instalar el entorno de ejecución del aprendizaje automático, descargar los modelos entrenados y realizar inferencias en el dispositivo.

Al utilizar los componentes de aprendizaje automático AWS proporcionados, tiene la flexibilidad de empezar a realizar inferencias de aprendizaje automático con ejemplos de códigos de inferencia y modelos previamente entrenados. Puede conectar componentes de modelos personalizados para utilizar sus propios modelos personalizados con los componentes de inferencia y tiempo de ejecución que proporcionan. AWS Para obtener una solución de aprendizaje automático completamente personalizada, puede usar los componentes públicos como plantillas para crear componentes personalizados y usar cualquier tiempo de ejecución, modelo o tipo de inferencia que desee.

Requisitos

Para crear y utilizar componentes de aprendizaje automático, debe disponer de lo siguiente:

- Un dispositivo central de Greengrass. Si no dispone de una, consulte Tutorial: Introducción a AWS IoT Greengrass V2.

- Espacio de almacenamiento local mínimo de 500 MB para usar los componentes AWS de aprendizaje automático de muestra proporcionados.

Orígenes de modelos admitidos

AWS IoT Greengrassadmite el uso de modelos de aprendizaje automático personalizados que se almacenan en Amazon S3. También puede utilizar los trabajos de empaquetado SageMaker edge de Amazon para crear directamente componentes de modelo para sus modelos SageMaker compilados en NEO. Para obtener información sobre el uso de SageMaker Edge Manager conAWS IoT Greengrass, consulteUtilice Amazon SageMaker Edge Manager en los dispositivos principales de Greengrass. También puede utilizar los trabajos de empaquetado de modelos de Amazon Lookout for Vision para crear componentes de modelos para sus modelos de Lookout for Vision. Para obtener más información sobre el uso de Lookout for Vision AWS IoT Greengrass con, Amazon Lookout Lookout for Vision out out out Greengrass out out out out consulte.

Los cubos S3 que contienen sus modelos deben cumplir los siguientes requisitos:

- No deben cifrarse mediante SSE-C. En el caso de los buckets que utilizan el cifrado del lado del servidor, la inferencia de aprendizaje AWS IoT Greengrass automático actualmente solo admite las opciones de cifrado SSE-S3 o SSE-KMS. Para obtener más información sobre las opciones de cifrado del lado del servidor, consulte Protección de datos con el cifrado del lado del servidor en la Guía del usuario de Amazon Simple Storage Service.
- Sus nombres no deben incluir puntos (). `.` Para obtener más información, consulte la regla sobre el uso de buckets de estilo alojado virtuales con SSL en Reglas para la nomenclatura de buckets en la Guía del usuario de Amazon Simple Storage Service.
- Los depósitos de S3 que almacenan las fuentes de los modelos deben estar en los mismos componentes de aprendizaje automático Cuenta de AWS y Región de AWS al igual que ellos.
- AWS IoT Greengrassdebe tener `read` permiso para acceder a la fuente del modelo. Para permitir el acceso AWS IoT Greengrass a los buckets de S3, el rol de dispositivo de Greengrass debe permitir `s3:GetObject` la acción. Para obtener más información sobre la función del dispositivo, consulte. Autorizar a los dispositivos principales a interactuar con AWS los servicios

Tiempos de ejecución de aprendizaje automático compatibles

AWS IoT Greengrassle permite crear componentes personalizados para utilizar cualquier entorno de aprendizaje automático que elija para realizar inferencias de aprendizaje automático con sus

modelos entrenados a medida. Para obtener información sobre la creación de componentes de aprendizaje automático personalizados, consulte. Personalice sus componentes de aprendizaje automático

Para que el proceso de introducción al aprendizaje automático sea más eficiente, AWS IoT Greengrass proporciona ejemplos de componentes de inferencia, modelo y tiempo de ejecución que utilizan los siguientes tiempos de ejecución de aprendizaje automático:

- Deep Learning Runtime (DLR) v1.6.0 y v1.3.0
- TensorFlow Lite v2.5.0

AWS-proporcionó componentes de aprendizaje automático

En la siguiente tabla se enumeran los componentes AWS proporcionados que se utilizan para el aprendizaje automático.

> **Note**
>
> Varios AWS de los componentes proporcionados dependen de versiones secundarias específicas del núcleo de Greengrass. Debido a esta dependencia, es necesario actualizar estos componentes al actualizar el núcleo de Greengrass a una nueva versión secundaria. Para obtener información sobre las versiones específicas del núcleo de las que depende cada componente, consulte el tema del componente correspondiente. Para obtener más información sobre la actualización del núcleo, consulteActualice el software AWS IoT Greengrass principal (OTA).

Componente	Descripción	Tipo de componente	Sistema operativo admitido	Código abierto
Lookout for Vision Edge Agent	Implementa el motor de ejecución Amazon Lookout for Vision en el	Genérico	Linux	No

Componente	Descripción	Tipo de componente	Sistema operativo admitido	Código abierto
	dispositivo principal de Greengrass, de forma que pueda utilizar la visión artificial para detectar defectos en productos industriales.			
SageMaker Administrador perimetral	Implementa el agente Amazon SageMaker Edge Manager en el dispositivo principal de Greengrass.	Genérico	Linux, Windows	No

Componente	Descripción	Tipo de componente	Sistema operativo admitido	Código abierto
Clasificación de imágenes DLR	Componente de inferencia que utiliza el almacén de modelos de clasificación de imágenes del DLR y el componente de tiempo de ejecución del DLR como dependencias para instalar el DLR, descargar modelos de clasificación de imágenes de muestra y realizar inferencias de clasificación de imágenes en los dispositivos compatibles.	Genérico	Linux, Windows	No

Componente	Descripción	Tipo de componente	Sistema operativo admitido	Código abierto
Detección de objetos DLR	Componente de inferencia que utiliza el almacén de modelos de detección de objetos del DLR y el componente de tiempo de ejecución del DLR como dependencias para instalar el DLR, descargar modelos de detección de objetos de muestra y realizar inferencias de detección de objetos en los dispositivos compatibles.	Genérico	Linux, Windows	No

Componente	Descripción	Tipo de componente	Sistema operativo admitido	Código abierto
Tienda de modelos de clasificación de imágenes DLR	Componente de modelo que contiene ejemplos de ResNet -50 modelos de clasificación de imágenes como artefactos de Greengrass.	Genérico	Linux, Windows	No
Tienda de modelos de detección de objetos DLR	Componente de modelo que contiene ejemplos de modelos de detección de objetos de YoloV3 como artefactos de Greengrass.	Genérico	Linux, Windows	No

Componente	Descripción	Tipo de componente	Sistema operativo admitido	Código abierto
Tiempo de ejecución de DLR	Componente de tiempo de ejecución que contiene un script de instalación que se utiliza para instalar el DLR y sus dependencias en el dispositivo principal de Greengrass.	Genérico	Linux, Windows	No

Componente	Descripción	Tipo de componente	Sistema operativo admitido	Código abierto
TensorFlow Clasificación de imágenes Lite	Componente de inferencia que utiliza el TensorFlow almacén de modelos de clasificación de imágenes de TensorFlow Lite y el componente de tiempo de ejecución de Lite como dependencias para instalar TensorFlow Lite, descargar modelos de clasificación de imágenes de muestra y realizar inferencias de clasificación de imágenes en dispositivos compatibles.	Genérico	Linux, Windows	No

Componente	Descripción	Tipo de componente	Sistema operativo admitido	Código abierto
TensorFlow Detección de objetos Lite	Componente de inferencia que utiliza el TensorFlow almacén de modelos de detección de objetos de TensorFlow Lite y el componente de tiempo de ejecución de Lite como dependencias para instalar TensorFlow Lite, descargar modelos de detección de objetos de muestra y realizar inferencias de detección de objetos en dispositivos compatibles.	Genérico	Linux, Windows	No

Componente	Descripción	Tipo de componente	Sistema operativo admitido	Código abierto
TensorFlow Tienda de modelos de clasificación de imágenes Lite	Componente de modelo que contiene un modelo MobileNet v1 de muestra como artefacto de Greengrass.	Genérico	Linux, Windows	No
TensorFlow Tienda de modelos de detección de objetos Lite	Componente de modelo que contiene un MobileNet modelo de muestra de detección de disparo único (SSD) como un artefacto de Greengrass.	Genérico	Linux, Windows	No

Componente	Descripción	Tipo de componente	Sistema operativo admitido	Código abierto
TensorFlow Tiempo de ejecución Lite	Componente de tiempo de ejecución que contiene un script de instalación que se utiliza para instalar TensorFlow Lite y sus dependencias en el dispositivo principal de Greengrass.	Genérico	Linux, Windows	No

Utilice Amazon SageMaker Edge Manager en los dispositivos principales de Greengrass

> **Important**
>
> SageMaker Edge Manager dejará de fabricarse el 26 de abril de 2024. Para obtener más información sobre cómo seguir implementando sus modelos en los dispositivos periféricos, consulte el final del ciclo de vida de SageMaker Edge Manager.

Amazon SageMaker Edge Manager es un agente de software que se ejecuta en dispositivos periféricos. SageMaker Edge Manager proporciona administración de modelos para dispositivos periféricos para que pueda empaquetar y usar modelos SageMaker compilados por Amazon NEO directamente en los dispositivos principales de Greengrass. Al usar SageMaker Edge Manager, también puede muestrear los datos de entrada y salida del modelo de sus dispositivos principales y enviar esos datos a ellos Nube de AWS para su monitoreo y análisis. Como SageMaker Edge

Manager utiliza SageMaker Neo para optimizar los modelos para el hardware de destino, no es necesario que instales el DLR Runtime directamente en el dispositivo. En los dispositivos Greengrass, SageMaker Edge Manager no carga los AWS IoT certificados locales ni llama directamente al punto final del proveedor de AWS IoT credenciales. En su lugar, SageMaker Edge Manager utiliza el servicio de intercambio de fichas para obtener una credencial temporal de un punto final de TES.

En esta sección se describe cómo funciona SageMaker Edge Manager en los dispositivos principales de Greengrass.

Cómo funciona SageMaker Edge Manager en los dispositivos Greengrass

Para implementar el agente SageMaker Edge Manager en sus dispositivos principales, cree una implementación que incluya el `aws.greengrass.SageMakerEdgeManager` componente. AWS IoT Greengrassgestiona la instalación y el ciclo de vida del agente Edge Manager en sus dispositivos. Cuando haya disponible una nueva versión del binario del agente, implemente la versión actualizada del `aws.greengrass.SageMakerEdgeManager` componente para actualizar la versión del agente que está instalada en el dispositivo.

Cuando utilizas SageMaker Edge Manager conAWS IoT Greengrass, tu flujo de trabajo incluye los siguientes pasos de alto nivel:

1. Compila modelos con SageMaker Neo.
2. Empaque sus modelos SageMaker compilados en NEO mediante trabajos de embalaje SageMaker perimetral. Cuando ejecuta un trabajo de empaquetado perimetral para su modelo, puede optar por crear un componente del modelo con el modelo empaquetado como un artefacto que se puede implementar en su dispositivo principal de Greengrass.
3. Cree un componente de inferencia personalizado. Este componente de inferencia se utiliza para interactuar con el agente de Edge Manager y realizar la inferencia en el dispositivo principal. Estas operaciones incluyen cargar modelos, invocar solicitudes de predicción para ejecutar la inferencia y descargar modelos cuando el componente se apaga.
4. Implemente el componente SageMaker Edge Manager, el componente de modelo empaquetado y el componente de inferencia para ejecutar el modelo en el motor de SageMaker inferencia (agente de Edge Manager) de su dispositivo.

Para obtener más información sobre la creación de trabajos de empaquetado perimetral y componentes de inferencia que funcionen con SageMaker Edge Manager, consulte Deploy

Model Package y Edge Manager Agent con AWS IoT Greengrass en la Guía para SageMaker desarrolladores de Amazon.

El Tutorial: Cómo empezar a usar SageMaker Edge Manager tutorial le muestra cómo configurar y usar el agente SageMaker Edge Manager en un dispositivo principal de Greengrass existente, mediante el código de ejemplo AWS proporcionado que puede usar para crear ejemplos de componentes de inferencia y modelo.

Cuando usa SageMaker Edge Manager en los dispositivos principales de Greengrass, también puede usar la función de captura de datos para cargar datos de muestra en. Nube de AWS La captura de datos es una SageMaker función que se utiliza para cargar entradas de inferencia, resultados de inferencias y datos de inferencia adicionales a un bucket de S3 o a un directorio local para futuros análisis. Para obtener más información sobre el uso de datos de captura con SageMaker Edge Manager, consulte Manage Model en la Guía para SageMaker desarrolladores de Amazon.

Requisitos

Debe cumplir los siguientes requisitos para utilizar el agente SageMaker Edge Manager en los dispositivos principales de Greengrass.

- Un dispositivo principal de Greengrass que se ejecuta en Amazon Linux 2, una plataforma Linux basada en Debian (x86_64 o Armv8) o Windows (x86_64). Si no dispone de una, consulte Tutorial: Introducción a AWS IoT Greengrass V2.
- Python 3.6 o posterior, incluida `pip` la versión de Python, instalada en el dispositivo principal.
- El rol del dispositivo Greengrass se configuró con lo siguiente:
  - Una relación de confianza que `sagemaker.amazonaws.com` permite `credentials.iot.amazonaws.com` y asume el rol, como se muestra en el siguiente ejemplo de política de IAM.

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

```
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "sagemaker.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

- La política gestionada por AmazonSageMakerEdgeDeviceFleetPolicyIAM.
- La `s3:PutObject` acción, tal como se muestra en el siguiente ejemplo de política de IAM.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "s3:PutObject"
      ],
      "Resource": [
        "*"
      ],
      "Effect": "Allow"
    }
  ]
}
```

- Un bucket de Amazon S3 creado en el mismo dispositivo principal de Greengrass Cuenta de AWS y Región de AWS en el mismo que él. SageMaker Edge Manager necesita un bucket S3 para crear una flota de dispositivos perimetrales y almacenar datos de muestra derivados de la ejecución de inferencias en su dispositivo. Para obtener información sobre la creación de buckets de S3, consulte Introducción a Amazon S3.
- Una flota de dispositivos SageMaker periféricos que usa el mismo alias de AWS IoT rol que su dispositivo principal de Greengrass. Para obtener más información, consulte Cree una flota de dispositivos periféricos.
- Su dispositivo principal Greengrass está registrado como dispositivo perimetral en su flota de dispositivos SageMaker Edge. El nombre del dispositivo perimetral debe coincidir con el AWS IoT nombre del dispositivo principal. Para obtener más información, consulte Registra tu dispositivo principal de Greengrass.

Comience a usar SageMaker Edge Manager

Puedes completar un tutorial para empezar a usar SageMaker Edge Manager. El tutorial le muestra cómo empezar a usar SageMaker Edge Manager con los componentes AWS de muestra proporcionados en un dispositivo principal existente. Estos componentes de ejemplo utilizan el componente SageMaker Edge Manager como dependencia para implementar el agente de Edge Manager y realizar inferencias mediante modelos previamente entrenados que se compilaron con Neo. SageMaker Para obtener más información, consulte Tutorial: Cómo empezar a usar SageMaker Edge Manager.

Amazon Lookout Lookout for Vision out out out Greengrass out out out out

> **Note**
>
> AWS IoT Greengrassactualmente no admite esta función en los dispositivos principales de Windows.

Amazon Lookout for Vision es un programaServicio de AWS que puede utilizar para encontrar defectos visuales en productos industriales. Utiliza la visión artificial para identificar los componentes que faltan en un producto industrial, los daños a vehículos o estructuras, las irregularidades en las líneas de producción, la falta de condensadores en las placas de circuito impreso y los defectos en las obleas de silicio o cualquier otro elemento físico en el que la calidad sea importante. Para obtener más información, consulte ¿Qué es Amazon Lookout for Vision? en la Guía para desarrolladores de Amazon Lookout for Vision.

Puede crear aplicaciones de Greengrass que utilicen la inferencia de Lookout for Vision para encontrar defectos visuales en los dispositivos principales de Greengrass. Tras implementar un flujo de trabajo de Lookout for Vision en un dispositivo principal de Greengrass, puede realizar la visión artificial sin necesidad de conectarse al servicio Lookout for Vision delNube de AWS. Para crear una aplicación de Greengrass que utilice Lookout for Vision, configure e implemente los siguientes componentes de Greengrass:

- Componentes del modelo Lookout for Vision: contiene los modelos de aprendizaje automático de Lookout for Vision como artefactos de Greengrass. Puede utilizar la consola y la API de Lookout for Vision para generar componentes de modelo que empaqueten sus modelos de

aprendizaje automático previamente entrenados. Estos componentes son componentes privados de Greengrass en suCuenta de AWS. Para obtener más información, consulte Crear un modelo de Lookout for Vision y Empaquetar un modelo Lookout for Vision en la Guía para desarrolladores de Amazon Lookout for Vision.

- Componente Lookout for Vision Edge Agent: proporciona un servidor de ejecución local de Lookout for Vision que utiliza la visión artificial para detectar anomalías mediante los modelos de aprendizaje automático que usted proporciona. Este componente es un componenteAWS proporcionado por. Para obtener más información, consulte el componente Lookout for Vision Edge Agent.
- Componente de aplicación cliente Lookout for Vision: interactúa con el componente Lookout for Vision Edge Agent para procesar las imágenes en busca de anomalías. Puede desarrollar componentes de aplicaciones cliente personalizados que envíen imágenes y transmisiones de vídeo al agente local de Lookout for Vision Edge e informen de cualquier anomalía que detecten los modelos de aprendizaje automático. Para obtener más información, consulte Cómo escribir un componente de aplicación cliente y la referencia a la API de agentes de Lookout for Vision en la Guía para desarrolladores de Amazon Lookout for Vision.

Para obtener más información sobre cómo crear, configurar y utilizar estos componentes, consulte Uso de un modelo de Lookout for Vision en un dispositivo periférico en la Guía para desarrolladores de Amazon Lookout for Vision.

Personalice sus componentes de aprendizaje automático

EnAWS IoT Greengrass, puede configurar ejemplos de componentes de aprendizaje automático para personalizar la forma en que realiza la inferencia de aprendizaje automático en sus dispositivos, con los componentes de inferencia, modelo y tiempo de ejecución como componentes básicos. AWS IoT Greengrasstambién le proporciona la flexibilidad de utilizar los componentes de muestra como plantillas y crear sus propios componentes personalizados según sea necesario. Puede combinar este enfoque modular para personalizar los componentes de inferencia del aprendizaje automático de las siguientes maneras:

Uso de ejemplos de componentes de inferencia

- Modifique la configuración de los componentes de inferencia al implementarlos.
- Utilice un modelo personalizado con el componente de inferencia de muestra sustituyendo el componente de tienda de modelos de muestra por un componente de modelo personalizado.

El modelo personalizado debe entrenarse con el mismo tiempo de ejecución que el modelo de muestra.

Uso de componentes de inferencia personalizados

- Use código de inferencia personalizado con los modelos y tiempos de ejecución de muestra agregando componentes de modelos públicos y componentes de tiempo de ejecución como dependencias de los componentes de inferencia personalizados.
- Cree y añada componentes de modelo personalizados o componentes de tiempo de ejecución como dependencias de componentes de inferencia personalizados. Debe usar componentes personalizados si quiere usar un código de inferencia personalizado o un tiempo de ejecución para el que AWS IoT Greengrass no se proporcione un componente de muestra.

Temas

Modifique la configuración de un componente de inferencia público

En la AWS IoT Greengrassconsola, la página del componente muestra la configuración predeterminada de ese componente. Por ejemplo, la configuración predeterminada del componente de clasificación de imágenes de TensorFlow Lite es la siguiente:

```
{
  "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"
  }
}
```

Al implementar un componente de inferencia público, puede modificar la configuración predeterminada para personalizar su implementación. Para obtener información sobre los parámetros de configuración disponibles para cada componente de inferencia pública, consulte el tema del componente en. AWS-proporcionó componentes de aprendizaje automático

En esta sección se describe cómo implementar un componente modificado desde la AWS IoT Greengrass consola. Para obtener información sobre la implementación de componentes mediante elAWS CLI, consulteCrear implementaciones.

Para implementar un componente de inferencia pública modificado (consola)

1. Inicie sesión en la consola de AWS IoT Greengrass.
2. En el menú de navegación, elija Componentes.
3. En la página Componentes, en la pestaña Componentes públicos, elija el componente que desee implementar.
4. En la página de componentes, elija Implementar.
5. En Añadir a la implementación, elija una de las siguientes opciones:
   a. Para combinar este componente con una implementación existente en el dispositivo de destino, elija Agregar a la implementación existente y, a continuación, seleccione la implementación que desee revisar.
   b. Para crear una nueva implementación en el dispositivo de destino, elija Crear nueva implementación. Si tiene una implementación existente en su dispositivo, al elegir este paso se reemplaza la implementación existente.
6. En la página Especificar detalles, haga lo siguiente:
   a. En Información de implementación, introduzca o modifique el nombre descriptivo de su implementación.

b. En Objetivos de implementación, seleccione un objetivo para su implementación y elija Siguiente. No puede cambiar el objetivo de implementación si está revisando una implementación existente.

7. En la página Seleccionar componentes, en Componentes públicos, compruebe que está seleccionado el componente de inferencia con la configuración modificada y elija Siguiente.
8. En la página Configurar componentes, haga lo siguiente:

   a. Seleccione el componente de inferencia y elija Configurar componente.

   b. En Actualización de la configuración, introduzca los valores de configuración que desee actualizar. Por ejemplo, introduzca la siguiente actualización de configuración en el cuadro Configuración para fusionar para cambiar el intervalo de inferencia a 15 segundos e indique al componente que busque la imagen nombrada `custom.jpg` en la `/custom-ml-inference/images/` carpeta.

   ```
   {
     "InferenceInterval": "15",
     "ImageName": "custom.jpg",
     "ImageDirectory": "/custom-ml-inference/images/"
   }
   ```

   Para restablecer toda la configuración de un componente a sus valores predeterminados, especifique una sola cadena vacía `""` en el cuadro Restablecer rutas.

   c. Seleccione Confirmar y, a continuación, elija Siguiente.

9. En la página Configurar los ajustes avanzados, conserve los valores de configuración predeterminados y seleccione Siguiente.
10. En la página de revisión, elija Implementar

Utilice un modelo personalizado con el componente de inferencia de muestra

Si desea utilizar el componente de inferencia de ejemplo con sus propios modelos de aprendizaje automático para un tiempo de ejecución que AWS IoT Greengrass proporcione un componente de tiempo de ejecución de muestra, debe anular los componentes del modelo público por componentes que utilicen esos modelos como artefactos. En un nivel alto, debe completar los siguientes pasos para usar un modelo personalizado con el componente de inferencia de muestra:

1. Cree un componente de modelo que utilice un modelo personalizado en un depósito de S3 como artefacto. Su modelo personalizado debe entrenarse con el mismo tiempo de ejecución que el modelo que desea reemplazar.
2. Modifique el parámetro de `ModelResourceKey` configuración en el componente de inferencia para usar el modelo personalizado. Para obtener información sobre la actualización de la configuración del componente de inferencia, consulte Modifique la configuración de un componente de inferencia público

Al implementar el componente de inferencia, AWS IoT Greengrass busca la versión más reciente de las dependencias de sus componentes. Anula el componente del modelo público dependiente si existe una versión personalizada posterior del componente en la misma banda. Cuenta de AWS Región de AWS

Cree un componente de modelo personalizado (consola)

1. Cargue su modelo en un bucket de S3. Para obtener información sobre cómo cargar sus modelos en un bucket de S3, consulte Cómo trabajar con buckets de Amazon S3 en la Guía del usuario de Amazon Simple Storage Service.

   > **Note**
   >
   > Debe almacenar sus artefactos en depósitos S3 que estén en el mismo lugar Cuenta de AWS y Región de AWS como los componentes. Para permitir el acceso AWS IoT Greengrass a estos artefactos, el rol de dispositivo de Greengrass debe permitir la `s3:GetObject` acción. Para obtener más información sobre la función del dispositivo, consulteAutorizar a los dispositivos principales a interactuar con AWS los servicios.

2. En el menú de navegación de la AWS IoT Greengrassconsola, elija Componentes.
3. Recupere la receta del componente de la tienda de modelos pública.

   a. En la página Componentes, en la pestaña Componentes públicos, busque y elija el componente del modelo público para el que desee crear una nueva versión. Por ejemplo, `variant.DLR.ImageClassification.ModelStore`.

   b. En la página de componentes, elija Ver receta y copie la receta JSON que se muestra.
4. En la página Componentes, en la pestaña Mis componentes, elija Crear componente.
5. En la página Crear componente, en Información del componente, seleccione Introducir la receta como JSON como fuente del componente.

6. En el cuadro Receta, pegue la receta del componente que copió anteriormente.
7. En la receta, actualice los siguientes valores:

   - `ComponentVersion`: Aumente la versión secundaria del componente.

     Al crear un componente personalizado para anular un componente de modelo público, debe actualizar solo la versión secundaria de la versión del componente existente. Por ejemplo, si la versión del componente público es`2.1.0`, puede crear un componente personalizado con la versión`2.1.1`.

   - `Manifests.Artifacts.Uri`: actualice cada valor de URI al URI de Amazon S3 del modelo que desee utilizar.

   > **Note**
   >
   > No cambie el nombre del componente.

8. Selecciona Crear componente.

Cree un componente de modelo personalizado (AWS CLI)

1. Cargue su modelo en un bucket de S3. Para obtener información sobre cómo cargar sus modelos en un bucket de S3, consulte Cómo trabajar con buckets de Amazon S3 en la Guía del usuario de Amazon Simple Storage Service.

   > **Note**
   >
   > Debe almacenar sus artefactos en depósitos S3 que estén en el mismo lugar Cuenta de AWS y Región de AWS como los componentes. Para permitir el acceso AWS IoT Greengrass a estos artefactos, el rol de dispositivo de Greengrass debe permitir la `s3:GetObject` acción. Para obtener más información sobre la función del dispositivo, consulteAutorizar a los dispositivos principales a interactuar con AWS los servicios.

2. Ejecute el siguiente comando para recuperar la receta del componente público. Este comando escribe la receta del componente en el archivo de salida que usted proporciona en el comando. Convierte la cadena codificada en base64 recuperada a JSON o YAML, según sea necesario.

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. Actualice el nombre del archivo de recetas para que *<component-name>*-*<component-version>* la versión del componente sea la versión de destino del nuevo componente. Por ejemplo, `variant.DLR.ImageClassification.ModelStore-2.1.1.yaml`.
4. En la receta, actualice los siguientes valores:

   - `ComponentVersion`: Aumente la versión secundaria del componente.

     Al crear un componente personalizado para anular un componente de modelo público, debe actualizar solo la versión secundaria de la versión del componente existente. Por ejemplo, si la versión del componente público es`2.1.0`, puede crear un componente personalizado con la versión`2.1.1`.

- `Manifests.Artifacts.Uri`: actualice cada valor de URI al URI de Amazon S3 del modelo que desee utilizar.

> **Note**
>
> No cambie el nombre del componente.

5. Ejecute el siguiente comando para crear un nuevo componente con la receta que recuperó y modificó.

```
aws greengrassv2 create-component-version \
    --inline-recipe fileb://path/to/component/recipe
```

> **Note**
>
> Este paso crea el componente en el AWS IoT Greengrass servicio enNube de AWS. Puede usar la CLI de Greengrass para desarrollar, probar e implementar su componente localmente antes de subirlo a la nube. Para obtener más información, consulte Desarrolle AWS IoT Greengrass componentes.

Para obtener más información sobre la creación de componentes, consulteDesarrolle AWS IoT Greengrass componentes.

Cree componentes de aprendizaje automático personalizados

Debe crear componentes personalizados si desea utilizar un código de inferencia personalizado o un tiempo de ejecución para el que AWS IoT Greengrass no se proporcione un componente de muestra. Puede usar su código de inferencia personalizado con los modelos de aprendizaje automático y tiempos de ejecución de muestra AWS proporcionados, o puede desarrollar una solución de inferencia de aprendizaje automático completamente personalizada con sus propios modelos y tiempo de ejecución. Si sus modelos utilizan un entorno de ejecución para el que se AWS IoT Greengrass proporciona un ejemplo de componente de tiempo de ejecución, puede utilizar ese componente de tiempo de ejecución y tendrá que crear componentes personalizados únicamente para el código de inferencia y los modelos que desee utilizar.

Temas

- Recupera la receta de un componente público
- Recupere ejemplos de artefactos de los componentes
- Cargue los artefactos de los componentes en un bucket de S3
- Cree componentes personalizados

Recupera la receta de un componente público

Puede utilizar la receta de un componente público de aprendizaje automático existente como plantilla para crear un componente personalizado. Para ver la receta de componentes de la última versión de un componente público, utilice la consola o utilice las AWS CLI siguientes opciones:

- Uso de la consola

  1. En la página Componentes, en la pestaña Componentes públicos, busque y elija el componente público.
  2. En la página del componente, elija Ver receta.

- Uso de AWS CLI

  Ejecute el siguiente comando para recuperar la receta del componente de la variante pública. Este comando escribe la receta del componente en el archivo de recetas JSON o YAML que proporciones en el comando.

  Linux, macOS, or Unix

  ```
  aws greengrassv2 get-component \
      --arn <arn> \
      --recipe-output-format <recipe-format> \
      --query recipe \
      --output text | base64 --decode > <recipe-file>
  ```

  Windows Command Prompt (CMD)

  ```
  aws greengrassv2 get-component ^
      --arn <arn> ^
      --recipe-output-format <recipe-format> ^
      --query recipe ^
      --output text > <recipe-file>.base64

  certutil -decode <recipe-file>.base64 <recipe-file>
  ```

PowerShell

```
aws greengrassv2 get-component `
    --arn <arn> `
    --recipe-output-format <recipe-format> `
    --query recipe `
    --output text > <recipe-file>.base64

certutil -decode <recipe-file>.base64 <recipe-file>
```

Sustituya los valores del comando de la siguiente manera:

- *<arn>*. El nombre de recurso de Amazon (ARN) del componente público.
- *<recipe-format>*. El formato en el que desea crear el archivo de recetas. Los valores admitidos son `JSON` y `YAML`.
- *<recipe-file>*. El nombre de la receta en el formato*<component-name>*-*<component-version>*.

Recupere ejemplos de artefactos de los componentes

Puede utilizar los artefactos que utilizan los componentes públicos de aprendizaje automático como plantillas para crear sus artefactos de componentes personalizados, como códigos de inferencia o scripts de instalación en tiempo de ejecución.

Para ver los artefactos de muestra que se incluyen en los componentes públicos de aprendizaje automático, implemente el componente de inferencia público y, a continuación, visualice los artefactos del dispositivo en la */greengrass/v2*`/packages/artifacts-unarchived/`*component-name*/*component-version*/ carpeta.

Cargue los artefactos de los componentes en un bucket de S3

Antes de poder crear un componente personalizado, debe cargar los artefactos del componente en un bucket de S3 y utilizar los URI de S3 en la receta del componente. Por ejemplo, para usar un código de inferencia personalizado en su componente de inferencia, cargue el código en un bucket de S3. A continuación, puede utilizar el URI de Amazon S3 de su código de inferencia como un artefacto en su componente.

Para obtener información sobre cómo cargar contenido en un bucket de S3, consulte Trabajar con buckets de Amazon S3 en la Guía del usuario de Amazon Simple Storage Service.

> **Note**
>
> Debe almacenar sus artefactos en depósitos de S3 que estén en el mismo lugar Cuenta de AWS y Región de AWS como los componentes. Para permitir el acceso AWS IoT Greengrass a estos artefactos, el rol de dispositivo de Greengrass debe permitir la `s3:GetObject` acción. Para obtener más información sobre la función del dispositivo, consulteAutorizar a los dispositivos principales a interactuar con AWS los servicios.

Cree componentes personalizados

Puede utilizar los artefactos y las recetas que ha recuperado para crear sus componentes de aprendizaje automático personalizados. Para ver un ejemplo, consulte Cree un componente de inferencia personalizado.

Para obtener información detallada sobre la creación e implementación de componentes en los dispositivos Greengrass, consulte Desarrolle AWS IoT Greengrass componentes y. Implemente AWS IoT Greengrass componentes en los dispositivos

Cree un componente de inferencia personalizado

En esta sección, se muestra cómo crear un componente de inferencia personalizado utilizando el componente de clasificación de imágenes del DLR como plantilla.

Temas

- Cargue su código de inferencia a un bucket de Amazon S3
- Cree una receta para su componente de inferencia
- Cree el componente de inferencia

Cargue su código de inferencia a un bucket de Amazon S3

Cree su código de inferencia y, a continuación, cárguelo en un bucket de S3. Para obtener información sobre cómo cargar contenido en un bucket de S3, consulte Trabajar con buckets de Amazon S3 en la Guía del usuario de Amazon Simple Storage Service.

> **Note**
>
> Debe almacenar sus artefactos en depósitos de S3 que estén en el mismo lugar Cuenta de AWS y Región de AWS como los componentes. Para permitir el acceso AWS IoT Greengrass a estos artefactos, el rol de dispositivo de Greengrass debe permitir la `s3:GetObject` acción. Para obtener más información sobre la función del dispositivo, consulteAutorizar a los dispositivos principales a interactuar con AWS los servicios.

Cree una receta para su componente de inferencia

1. Ejecute el siguiente comando para recuperar la receta del componente de clasificación de imágenes del DLR. Este comando escribe la receta del componente en el archivo de recetas JSON o YAML que usted proporciona en el comando.

   Linux, macOS, or Unix

   ```
   aws greengrassv2 get-component \
       --arn
    arn:aws:greengrass:region:aws:components:aws.greengrass.DLRImageClassification:versions
    \
       --recipe-output-format JSON | YAML \
       --query recipe \
       --output text | base64 --decode > <recipe-file>
   ```

   Windows Command Prompt (CMD)

   ```
   aws greengrassv2 get-component ^
       --arn
    arn:aws:greengrass:region:aws:components:aws.greengrass.DLRImageClassification:versions
    ^
       --recipe-output-format JSON | YAML ^
       --query recipe ^
       --output text > <recipe-file>.base64

   certutil -decode <recipe-file>.base64 <recipe-file>
   ```

   PowerShell

   ```
   aws greengrassv2 get-component `
   ```

```
    --arn
 arn:aws:greengrass:region:aws:components:aws.greengrass.DLRImageClassification:versions
 `
    --recipe-output-format JSON | YAML `
    --query recipe `
    --output text > <recipe-file>.base64

certutil -decode <recipe-file>.base64 <recipe-file>
```

<recipe-file>Sustitúyala por el nombre de la receta en el formato*<component-name>*-*<component-version>*.

2. En el `ComponentDependencies` objeto de la receta, realice una o varias de las siguientes acciones en función del modelo y los componentes del tiempo de ejecución que desee utilizar:

   - Mantenga la dependencia de los componentes del DLR si desea utilizar modelos compilados por el DLR. También puede reemplazarlo por una dependencia de un componente de tiempo de ejecución personalizado, como se muestra en el siguiente ejemplo.

     Componente de ejecución

     JSON

     ```
     {
         "<runtime-component>": {
             "VersionRequirement": "<version>",
             "DependencyType": "HARD"
         }
     }
     ```

     YAML

     ```
     <runtime-component>:
         VersionRequirement: "<version>"
         DependencyType: HARD
     ```

   - Mantenga la dependencia del almacén de modelos de clasificación de imágenes del DLR para utilizar los ResNet -50 modelos previamente entrenados que AWS proporciona, o modifíquelo para utilizar un componente de modelo personalizado. Al incluir una dependencia para un componente de modelo público, si existe una versión personalizada posterior del componente en el mismo Cuenta de AWS yRegión de AWS, entonces, el componente de inferencia utiliza

ese componente personalizado. Especifique la dependencia del componente del modelo como se muestra en los siguientes ejemplos.

Componente de modelo público

JSON

```
{
    "variant.DLR.ImageClassification.ModelStore": {
        "VersionRequirement": "<version>",
        "DependencyType": "HARD"
    }
}
```

YAML

```
variant.DLR.ImageClassification.ModelStore:
    VersionRequirement: "<version>"
    DependencyType: HARD
```

Componente de modelo personalizado

JSON

```
{
    "<custom-model-component>": {
        "VersionRequirement": "<version>",
        "DependencyType": "HARD"
    }
}
```

YAML

```
<custom-model-component>:
    VersionRequirement: "<version>"
    DependencyType: HARD
```

3. En el `ComponentConfiguration` objeto, añada la configuración por defecto para este componente. Más adelante, podrá modificar esta configuración al implementar el componente. El siguiente extracto muestra la configuración del componente de clasificación de imágenes del DLR.

Por ejemplo, si usa un componente de modelo personalizado como dependencia para su componente de inferencia personalizado, modifíquelo `ModelResourceKey` para proporcionar los nombres de los modelos que está utilizando.

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. En el `Manifests` objeto, proporcione información sobre los artefactos y la configuración de este componente que se utilizan cuando el componente se implementa en diferentes plataformas y cualquier otra información necesaria para ejecutar correctamente el componente. El siguiente extracto muestra la configuración del `Manifests` objeto para la plataforma Linux en el componente de clasificación de imágenes del 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
```

Para obtener información detallada sobre la creación de recetas de componentes, consulte. AWS IoT Greengrass referencia de recetas de componentes

Cree el componente de inferencia

Utilice la AWS IoT Greengrass consola o la AWS CLI para crear un componente con la receta que acaba de definir. Después de crear el componente, puede implementarlo para realizar inferencias

en su dispositivo. Para ver un ejemplo de cómo implementar un componente de inferencia, consulte. Tutorial: Realice una inferencia de clasificación de imágenes de muestra con Lite TensorFlow

Crear un componente de inferencia personalizado (consola)

1. Inicie sesión en la consola de AWS IoT Greengrass.
2. En el menú de navegación, elija Componentes.
3. En la página Componentes, en la pestaña Mis componentes, elija Crear componente.
4. En la página Crear componente, en Información del componente, seleccione Introducir la receta como JSON o Introducir la receta como YAML como fuente del componente.
5. En el cuadro Receta, introduce la receta personalizada que has creado.
6. Haga clic en Crear componente.

Cree un componente de inferencia personalizado () AWS CLI

Ejecute el siguiente comando para crear un nuevo componente personalizado con la receta que ha creado.

```
aws greengrassv2 create-component-version \
    --inline-recipe fileb://path/to/recipe/file
```

> **Note**
>
> Este paso crea el componente en el AWS IoT Greengrass servicio enNube de AWS. Puede usar la CLI de Greengrass para desarrollar, probar e implementar su componente localmente antes de subirlo a la nube. Para obtener más información, consulte Desarrolle AWS IoT Greengrass componentes.

Solución de problemas de inferencia de aprendizaje automático

Utilice la información y las soluciones de solución de problemas de esta sección para ayudar a resolver los problemas con los componentes de aprendizaje automático. Para ver los componentes públicos de inferencia del aprendizaje automático, consulta los mensajes de error en los siguientes registros de componentes:

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

Si un componente está instalado correctamente, el registro del componente contiene la ubicación de la biblioteca que utiliza para la inferencia.

Problemas

- No se pudo recuperar la biblioteca
- Cannot open shared object file
- Error: ModuleNotFoundError: No module named '<library>'
- No se ha detectado ningún dispositivo compatible con CUDA
- No existe ese archivo o directorio
- RuntimeError: module compiled against API version 0xf but this version of NumPy is <version>
- picamera.exc.PiCameraError: Camera is not enabled
- Errores de memoria
- Errores de espacio en disco
- Errores de tiempo de espera

No se pudo recuperar la biblioteca

El siguiente error se produce cuando el script de instalación no puede descargar una biblioteca necesaria durante la implementación en un dispositivo Raspberry Pi.

```
Err:2 http://raspbian.raspberrypi.org/raspbian buster/main armhf python3.7-dev armhf
 3.7.3-2+deb10u1
404 Not Found [IP: 93.93.128.193 80]
E: Failed to fetch http://raspbian.raspberrypi.org/raspbian/pool/main/p/python3.7/
libpython3.7-dev_3.7.3-2+deb10u1_armhf.deb 404 Not Found [IP: 93.93.128.193 80]
```

Ejecute `sudo apt-get update` y vuelva a implementar el componente.

Cannot open shared object file

Es posible que se produzcan errores similares a los siguientes cuando el script del instalador no pueda descargar una dependencia necesaria `opencv-python` durante la implementación en un dispositivo Raspberry Pi.

```
ImportError: libopenjp2.so.7: cannot open shared object file: No such file or directory
```

Ejecute el siguiente comando para instalar manualmente las dependencias de`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>'

Es posible que veas este error en los registros de los componentes de tiempo de ejecución de ML (`variant.DLR.log`o`variant.TensorFlowLite.log`) cuando la biblioteca de tiempo de ejecución de ML o sus dependencias no estén instaladas correctamente. Este error puede producirse en los siguientes casos:

- Si utiliza la `UseInstaller` opción, que está habilitada de forma predeterminada, este error indica que el componente de tiempo de ejecución de ML no pudo instalar el tiempo de ejecución o sus dependencias. Haga lo siguiente:
  1. Configure el componente de tiempo de ejecución de ML para deshabilitar la `UseInstaller` opción.

2. Instale el motor de ejecución de ML y sus dependencias y póngalos a disposición del usuario del sistema que ejecuta los componentes de ML. Para obtener más información, consulte los siguientes temas:
   - Opción de tiempo de ejecución de DLR UseInstaller
   - TensorFlowOpción de tiempo de ejecución UseInstaller Lite

- Si no utiliza la `UseInstaller` opción, este error indica que el motor de ejecución de ML o sus dependencias no están instalados para el usuario del sistema que ejecuta los componentes de ML. Haga lo siguiente:

  1. Compruebe que la biblioteca esté instalada para el usuario del sistema que ejecuta los componentes de ML. Sustituya *`ggc_user`* por el nombre del usuario del sistema y sustituya *`tflite_runtime`* por el nombre de la biblioteca para comprobarlo.

     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. Si la biblioteca no está instalada, instálala para ese usuario. Sustituya *`ggc_user`* por el nombre del usuario del sistema y sustituya *`tflite_runtime`* por el nombre de la biblioteca.

     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"
     ```

     Para obtener más información sobre las dependencias de cada tiempo de ejecución de ML, consulte lo siguiente:
     - Opción de tiempo de ejecución de DLR UseInstaller
     - TensorFlowOpción de tiempo de ejecución UseInstaller Lite

  3. Si el problema persiste, instala la biblioteca para que otro usuario confirme si este dispositivo puede instalarla. El usuario puede ser, por ejemplo, su usuario, el usuario root o un usuario

administrador. Si no puedes instalar la biblioteca correctamente para ningún usuario, es posible que tu dispositivo no sea compatible con la biblioteca. Consulta la documentación de la biblioteca para revisar los requisitos y solucionar los problemas de instalación.

No se ha detectado ningún dispositivo compatible con CUDA

Es posible que aparezca el siguiente error al utilizar la aceleración de la GPU. Ejecute el siguiente comando para habilitar el acceso a la GPU para el usuario de Greengrass.

```
sudo usermod -a -G video ggc_user
```

No existe ese archivo o directorio

Los siguientes errores indican que el componente de tiempo de ejecución no pudo configurar el entorno virtual correctamente:

- *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`

Compruebe los registros para asegurarse de que todas las dependencias del tiempo de ejecución se instalaron correctamente. Para obtener más información sobre las bibliotecas que instala el script del instalador, consulte los siguientes temas:

- Tiempo de ejecución de DLR
- TensorFlow Tiempo de ejecución Lite

De forma predeterminada, *ML RootPath* está establecido en*/greengrass/v2*/`work/`*component-name*`/greengrass_ml`. Para cambiar esta ubicación, incluya el componente Tiempo de ejecución de DLR o el componente de tiempo de TensorFlow Tiempo de ejecución Lite ejecución directamente en la implementación y especifique un valor modificado para el `MLRootPath` parámetro en una actualización de combinación de configuraciones. Para obtener más información sobre la configuración del componente, consulteActualizar las configuraciones de los componentes.

> **Note**
>
> Para el componente DLR v1.3.x, se establece el `MLRootPath` parámetro en la configuración del componente de inferencia y el valor predeterminado es. `$HOME/greengrass_ml`

RuntimeError: module compiled against API version 0xf but this version of NumPy is <version>

Es posible que aparezcan los siguientes errores al ejecutar la inferencia de aprendizaje automático en una Raspberry Pi con el sistema operativo Bullseye de Raspberry Pi.

```
RuntimeError: module compiled against API version 0xf but this version of numpy is 0xd
ImportError: numpy.core.multiarray failed to import
```

Este error se produce porque Raspberry Pi OS Bullseye incluye una versión anterior a la NumPy que requiere OpenCV. Para solucionar este problema, ejecuta el siguiente comando para actualizar NumPy a la versión más reciente.

```
pip3 install --upgrade numpy
```

picamera.exc.PiCameraError: Camera is not enabled

Es posible que aparezca el siguiente error al ejecutar una inferencia de aprendizaje automático en una Raspberry Pi que ejecute el sistema operativo Bullseye de Raspberry Pi.

```
picamera.exc.PiCameraError: Camera is not enabled. Try running 'sudo raspi-config' and
 ensure that the camera has been enabled.
```

Este error se produce porque el sistema operativo Bullseye de Raspberry Pi incluye una nueva pila de cámaras que no es compatible con los componentes del aprendizaje automático. Para solucionar este problema, habilita la pila de cámaras antigua.

Para activar la pila de cámaras antigua

1. Ejecute el siguiente comando para abrir la herramienta de configuración de Raspberry Pi.

   ```
   sudo raspi-config
   ```

2. Seleccione Opciones de interfaz.
3. Seleccione Cámara antigua para activar la pila de cámaras antiguas.
4. Reinicie el Raspberry Pi.

Errores de memoria

Los siguientes errores suelen producirse cuando el dispositivo no tiene suficiente memoria y se interrumpe el proceso de los componentes.

- `stderr. Killed.`
- `exitCode=137`

Recomendamos un mínimo de 500 MB de memoria para implementar un componente de inferencia de aprendizaje automático público.

Errores de espacio en disco

El `no space left on device` error suele producirse cuando un dispositivo no tiene suficiente espacio de almacenamiento. Asegúrese de que haya suficiente espacio en disco disponible en el dispositivo antes de volver a implementar el componente. Recomendamos un mínimo de 500 MB de espacio libre en disco para implementar un componente de inferencia de aprendizaje automático público.

Errores de tiempo de espera

Los componentes públicos de aprendizaje automático descargan archivos de modelos de aprendizaje automático de gran tamaño que superan los 200 MB. Si se agota el tiempo de espera de la descarga durante la implementación, compruebe la velocidad de la conexión a Internet y vuelva a intentar la implementación.

Gestione los dispositivos principales de Greengrass con AWS Systems Manager

> **Note**
>
> AWS IoT Greengrassactualmente no admite esta función en los dispositivos principales de Windows.

Systems Manager es un AWS servicio que puede utilizar para ver y controlar su infraestructuraAWS, incluidas las instancias de Amazon EC2, los servidores y máquinas virtuales (VM) locales y los dispositivos periféricos. Systems Manager le permite ver los datos operativos, automatizar las tareas operativas y mantener la seguridad y el cumplimiento. Cuando se registra una máquina en Systems Manager, se denomina nodo gestionado. Para obtener más información, consulte ¿Qué es AWS Systems Manager? en la Guía del usuario de AWS Systems Manager.

El AWS Systems Manager agente (System Manager Agent) es un software que se puede instalar en los dispositivos para permitir que Systems Manager los actualice, gestione y configure. Para instalar el agente de Systems Manager en los dispositivos principales de Greengrass, despliegue el componente Systems Manager Agent. Al implementar el Agente de Systems Manager por primera vez, este registra el dispositivo principal como un nodo gestionado por Systems Manager. El agente de Systems Manager se ejecuta en el dispositivo para permitir la comunicación con el servicio Systems Manager delNube de AWS. Para obtener más información acerca de cómo instalar y configurar el componente Agente de Systems Manager, consulteInstalación del agente de AWS Systems Manager.

Las herramientas y funciones de Systems Manager se denominan capacidades. Los dispositivos principales de Greengrass admiten todas las funciones de Systems Manager. Para obtener más información sobre estas capacidades y sobre cómo usar Systems Manager para administrar los dispositivos principales, consulte Capacidades de Systems Manager en la Guía del AWS Systems Manager usuario.

AWS Systems Managerofrece un nivel de instancias estándar y un nivel de instancias avanzadas para los nodos gestionados por Systems Manager. Si usa Systems Manager por primera vez, comienza en el nivel de instancias estándar. En el nivel de instancias estándar, puede registrar hasta 1000 nodos gestionados por cada uno de ellos. Región de AWS Cuenta de AWS Si necesita registrar

más de 1000 nodos gestionados en una sola cuenta y región, o si necesita utilizar la función de administrador de sesiones, utilice el nivel de instancias avanzadas. Para obtener más información, consulte Configuración de niveles de instancias en la Guía del AWS Systems Manager usuario.

Temas

- Instalación del agente de AWS Systems Manager
- Desinstalación del agente de AWS Systems Manager

Instalación del agente de AWS Systems Manager

El AWS Systems Manager agente (Systems Manager Agent) es un software de Amazon que se instala para permitir que Systems Manager actualice, gestione y configure los dispositivos principales de Greengrass, las instancias de Amazon EC2 y otros recursos. El agente procesa y ejecuta las solicitudes del servicio Systems Manager enNube de AWS. A continuación, el agente devuelve la información de estado y tiempo de ejecución al servicio Systems Manager. Para obtener más información, consulte Acerca del agente de Systems Manager en la Guía del AWS Systems Manager usuario.

AWSproporciona el agente de Systems Manager como un componente de Greengrass que puede implementar en sus dispositivos principales de Greengrass para administrarlos con Systems Manager. El componente Systems Manager Agent instala el software Systems Manager Agent y registra el dispositivo principal como nodo gestionado en Systems Manager. Siga los pasos de esta página para completar los requisitos previos e implementar el componente Systems Manager Agent en un dispositivo principal o en un grupo de dispositivos principales.

Temas

- Paso 1: completar los pasos generales de configuración de Systems Manager
- Paso 2: Crear un rol de servicio de IAM para Systems Manager
- Paso 3: Añadir permisos a la función de intercambio de tokens
- Paso 4: Implementar el componente Systems Manager Agent
- Paso 5: Verificar el registro del dispositivo principal con Systems Manager

Paso 1: completar los pasos generales de configuración de Systems Manager

Si aún no lo ha hecho, complete los pasos generales AWS Systems Manager de configuración de. Para obtener más información, consulte los pasos generales completos de configuración de Systems Manager en la Guía del AWS Systems Manager usuario.

Paso 2: Crear un rol de servicio de IAM para Systems Manager

El agente de Systems Manager utiliza un rol de servicio AWS Identity and Access Management (IAM) para comunicarse con AWS Systems Manager él. Systems Manager asume esta función para habilitar las capacidades de Systems Manager en cada dispositivo principal. El componente Systems Manager Agent también utiliza esta función para registrar el dispositivo principal como nodo gestionado por Systems Manager al implementar el componente. Si aún no lo ha hecho, cree un rol de servicio de Systems Manager para que lo utilice el componente Systems Manager Agent. Para obtener más información, consulte Crear una función de servicio de IAM para dispositivos perimetrales en la Guía del AWS Systems Manager usuario.

Paso 3: Añadir permisos a la función de intercambio de tokens

Los dispositivos principales de Greengrass utilizan una función de servicio de IAM, denominada función de intercambio de fichas, para interactuar con los servicios. AWS Cada dispositivo principal tiene una función de intercambio de fichas que se crea al instalar el software AWS IoT Greengrass Core. Muchos componentes de Greengrass, como el Agente de Systems Manager, requieren permisos adicionales para esta función. El componente del agente Systems Manager requiere los siguientes permisos, que incluyen el permiso para usar el rol en el que creóPaso 2: Crear un rol de servicio de IAM para 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": "*"
    }
  ]
}
```

Si aún no lo ha hecho, añada estos permisos a la función de intercambio de tokens del dispositivo principal para permitir que funcione el Agente de Systems Manager. Puede añadir una nueva política a la función de intercambio de fichas para conceder este permiso.

Para añadir permisos a la función de intercambio de fichas (consola)

1. En el menú de navegación de la consola de IAM, seleccione Roles.
2. Elija la función de IAM que configuró como función de intercambio de fichas al instalar el software AWS IoT Greengrass principal. Si no especificó un nombre para la función de intercambio de fichas al instalar el software AWS IoT Greengrass Core, se creó una función denominada`GreengrassV2TokenExchangeRole`.
3. En Permisos, selecciona Añadir permisos y, a continuación, selecciona Adjuntar políticas.
4. Elija Crear política. La página Crear política se abre en una nueva pestaña del navegador.
5. En la página Create policy (Crear política), haga lo siguiente:

   a. Elija JSON para abrir el editor de JSON.

   b. Pegue la siguiente política de en el editor JSON. *`Sustituya SSM ServiceRole`* por el nombre del rol de servicio en Paso 2: Crear un rol de servicio de IAM para Systems Manager el que lo creó.

   ```
   {
     "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. Elija Siguiente: Etiquetas.

d. Elija Siguiente: Revisar.

e. Introduzca un Nombre para la política, como **`GreengrassSSMAgentComponentPolicy`**.

f. Elija Crear política.

g. Cambie a la pestaña anterior del navegador en la que tenía abierta la función de intercambio de fichas.

6. En la página Añadir permisos, pulse el botón de actualización y, a continuación, seleccione la política de agente de Greengrass Systems Manager que creó en el paso anterior.

7. Seleccione Asociar políticas.

   Los dispositivos principales que utilizan esta función de intercambio de fichas ahora tienen permiso para interactuar con el servicio Systems Manager.

Para añadir permisos a la función de intercambio de fichas (AWS CLI)

Para agregar una política que conceda permiso para usar Systems Manager

1. Cree un archivo llamado `ssm-agent-component-policy.json` y copie el siguiente JSON en el archivo. *`Sustituya SSM ServiceRole`* por el nombre del rol de servicio en Paso 2: Crear un rol de servicio de IAM para Systems Manager el que lo creó.

   ```
   {
     "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. Ejecute el siguiente comando para crear la política a partir del documento de política incluido en `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
   ```

Copie la política Amazon Resource Name (ARN) de los metadatos de la política en la salida. Utilice este ARN para adjuntar esta política a la función de dispositivo principal en el siguiente paso.

3. Ejecute el siguiente comando para adjuntar la política a la función de intercambio de tokens.

   - Sustituya *GreengrassV2 TokenExchangeRole* por el nombre de la función de intercambio de fichas que especificó al instalar el software AWS IoT Greengrass principal. Si no especificó un nombre para la función de intercambio de fichas al instalar el software AWS IoT Greengrass Core, se creó una función denominada. `GreengrassV2TokenExchangeRole`
   - Sustituya el ARN de la política por el ARN del paso anterior.

   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
   ```

   Si el comando no tiene salida, se ha realizado correctamente. Los dispositivos principales que utilizan esta función de intercambio de fichas ahora tienen permiso para interactuar con el servicio Systems Manager.

Paso 4: Implementar el componente Systems Manager Agent

Complete los siguientes pasos para implementar y configurar el componente System Manager Agent. Puede implementar el componente en un dispositivo de un solo núcleo o en un grupo de dispositivos principales.

Para implementar el componente Systems Manager Agent (consola)

1. En el menú de navegación de la AWS IoT Greengrassconsola, elija Componentes.
2. En la página Componentes, seleccione la pestaña Componentes públicos y, a continuación, elija aws.greengrass.SystemsManagerAgent.
3. En la página aws.greengrass.SystemsManagerAgent, elija Implementar.
4. En Agregar a la implementación, elija una implementación existente para revisarla o cree una nueva y, a continuación, elija Siguiente.
5. Si opta por crear una nueva implementación, elija el dispositivo principal o el grupo de cosas de destino para la implementación. En la página Especificar el destino, en Destino del despliegue, elija un dispositivo principal o un grupo de cosas y, a continuación, elija Siguiente.
6. En la página Seleccionar componentes, compruebe que el aws.greengrass.SystemsManagerAgentcomponente esté seleccionado y seleccione Siguiente.
7. En la página Configurar componentes, seleccione y aws.greengrass.SystemsManagerAgent, a continuación, haga lo siguiente:

   a. Seleccione Configurar componente.

   b. En el aws.greengrass.SystemsManagerAgent modal Configurar, en Actualización de configuración, en Configuración para fusionar, introduzca la siguiente actualización de configuración. *`Sustituya SSM ServiceRole`* por el nombre del rol de servicio en Paso 2: Crear un rol de servicio de IAM para Systems Manager el que lo creó.

```
{
  "SSMRegistrationRole": "SSMServiceRole",
  "SSMOverrideExistingRegistration": false
}
```

> **Note**
>
> Si el dispositivo principal ya ejecuta el Agente de Systems Manager registrado con una activación híbrida, cámbielo `SSMOverrideExistingRegistration` a `true`.

Este parámetro especifica si el componente Systems Manager Agent registra el dispositivo principal cuando el Agente Systems Manager ya se está ejecutando en el dispositivo con una activación híbrida.
También puede especificar etiquetas (`SSMResourceTags`) para añadirlas al nodo gestionado por Systems Manager que el componente Systems Manager Agent crea para el dispositivo principal. Para obtener más información, consulte Configuración de componentes de Systems Manager Agent.

c. Elija Confirmar para cerrar el modal y, a continuación, elija Siguiente.

8. En la página Configurar ajustes avanzados, mantenga los ajustes de configuración predeterminados y seleccione Siguiente.
9. En la página Revisar, elija Implementar.

   La implementación puede tardar hasta un minuto en completarse.

Para implementar el componente Systems Manager Agent (AWS CLI)

Para implementar el componente Systems Manager Agent, cree un documento de despliegue que lo incluya `aws.greengrass.SystemsManagerAgent` en el `components` objeto y especifique la actualización de configuración del componente. Siga las instrucciones Crear implementaciones para crear una nueva implementación o revisar una implementación existente.

El siguiente ejemplo de documento de despliegue parcial especifica el uso de un rol de servicio denominado`SSMServiceRole`. *`Sustituya SSM ServiceRole`* por el nombre del rol de servicio en Paso 2: Crear un rol de servicio de IAM para Systems Manager el que lo creó.

```
{
  ...,
  "components": {
    ...,
    "aws.greengrass.SystemsManagerAgent": {
      "componentVersion": "1.0.0",
      "configurationUpdate": {
        "merge": "{\"SSMRegistrationRole\":\"SSMServiceRole\",
\"SSMOverrideExistingRegistration\":false}"
      }
    }
  }
}
```

> **Note**
>
> Si el dispositivo principal ya ejecuta el Agente de Systems Manager registrado con una activación híbrida, cámbielo `SSMOverrideExistingRegistration` a `true`. Este parámetro especifica si el componente Systems Manager Agent registra el dispositivo principal cuando el Agente Systems Manager ya se está ejecutando en el dispositivo con una activación híbrida.
> También puede especificar etiquetas (`SSMResourceTags`) para añadirlas al nodo gestionado por Systems Manager que el componente Systems Manager Agent crea para el dispositivo principal. Para obtener más información, consulte Configuración de componentes de Systems Manager Agent.

La implementación puede tardar varios minutos en completarse. Puede utilizar el AWS IoT Greengrass servicio para comprobar el estado de la implementación y comprobar los registros del software AWS IoT Greengrass principal y los registros de los componentes del Agente de Systems Manager para comprobar que el Agente de Systems Manager se ejecuta correctamente. Para obtener más información, consulte los siguientes temas:

- Verificar el estado de implementación
- Supervisar AWS IoT Greengrass registros
- Visualización de los registros del agente de Systems Manager en la Guía AWS Systems Manager del usuario

Si la implementación falla o el Agente de Systems Manager no se ejecuta, puede solucionar los problemas de la implementación en cada dispositivo principal. Para obtener más información, consulte los siguientes temas:

- Solución de problemas AWS IoT Greengrass V2
- Solución de problemas del agente Systems Manager en la guía AWS Systems Manager del usuario

Paso 5: Verificar el registro del dispositivo principal con Systems Manager

Cuando se ejecuta el componente Systems Manager Agent, registra el dispositivo principal como un nodo gestionado en Systems Manager. Puede usar la AWS IoT Greengrass consola, la consola

de Systems Manager y la API de Systems Manager para comprobar que un dispositivo principal esté registrado como nodo gestionado. Los nodos gestionados también se denominan instancias en partes de la AWS consola y la API.

Para verificar el registro del dispositivo principal (AWS IoT Greengrassconsola)

1. En el menú de navegación de la AWS IoT Greengrassconsola, selecciona Dispositivos principales.
2. Elige el dispositivo principal que deseas verificar.
3. En la página de detalles del dispositivo principal, busca la propiedad de la AWS Systems Managerinstancia. Si esta propiedad está presente y muestra un enlace a la consola de Systems Manager, el dispositivo principal se registra como nodo gestionado.

   También puede encontrar la propiedad AWS Systems Managerping status para comprobar el estado del Agente de Systems Manager en el dispositivo principal. Cuando el estado es En línea, puede administrar el dispositivo principal con Systems Manager.

Para verificar el registro del dispositivo principal (consola de Systems Manager)

1. En el menú de navegación de la consola de Systems Manager, seleccione Fleet Manager.
2. En Nodos gestionados, haga lo siguiente:

   a. Agregue un filtro donde esté el tipo de fuente AWS::IoT::Thing.

   b. Agregue un filtro en el que el ID de origen sea el nombre del dispositivo principal que se va a verificar.
3. Busque el dispositivo principal en la tabla de nodos gestionados. Si el dispositivo principal está en la tabla, está registrado como nodo gestionado.

   También puede encontrar la propiedad de estado de ping del Agente de Systems Manager para comprobar el estado del Agente de Systems Manager en el dispositivo principal. Cuando el estado es En línea, puede administrar el dispositivo principal con Systems Manager.

Para verificar el registro del dispositivo principal (AWS CLI)

- Utilice la DescribeInstanceInformationoperación para obtener la lista de nodos gestionados que coinciden con el filtro que especifique. Ejecute el siguiente comando para comprobar si un

dispositivo principal está registrado como nodo gestionado. *MyGreengrassCore*Sustitúyalo por el nombre del dispositivo principal para verificarlo.

```
aws ssm describe-instance-information --filter
 Key=SourceIds,Values=MyGreengrassCore Key=SourceTypes,Values=AWS::IoT::Thing
```

La respuesta contiene la lista de nodos gestionados que coinciden con el filtro. Si la lista contiene un nodo gestionado, el dispositivo principal se registra como nodo gestionado. También puede encontrar más información sobre el nodo gestionado del dispositivo principal en la respuesta. Si la `PingStatus` propiedad es`Online`, puede administrar el dispositivo principal con Systems Manager.

Tras comprobar que un dispositivo principal está registrado como nodo gestionado en Systems Manager, puede utilizar la consola y la API de Systems Manager para gestionar ese dispositivo principal. Para obtener más información sobre las capacidades de Systems Manager que puede usar para administrar los dispositivos principales de Greengrass, consulte las capacidades de Systems Manager en la Guía del AWS Systems Managerusuario.

Desinstalación del agente de AWS Systems Manager

Si ya no desea administrar un dispositivo principal de Greengrass conAWS Systems Manager, puede anular el registro del dispositivo principal en Systems Manager y desinstalar elAWS Systems Manager agente (agente de Systems Manager) del dispositivo.

Puede volver a registrar un dispositivo principal en cualquier momento. Para ello, vuelva a implementar el componente Systems Manager Agent, que registra el dispositivo principal en Systems Manager cuando se instala. Systems Manager almacena el historial de comandos de un dispositivo principal con el registro anulado durante 30 días.

Temas

- Paso 1: Anule el registro del dispositivo principal de Systems Manager
- Paso 2: Desinstalar el componente Systems Manager Agent
- Paso 3: Desinstalar el software de Systems Manager

Paso 1: Anule el registro del dispositivo principal de Systems Manager

Puede utilizar la consola o la API de Systems Manager para anular el registro del dispositivo principal. Para obtener más información, consulte Anular el registro de nodos gestionados en la Guía delAWS Systems Manager usuario.

Paso 2: Desinstalar el componente Systems Manager Agent

Tras anular el registro del dispositivo principal, desinstale el componente Systems Manager Agent del dispositivo. Para eliminar un componente de un dispositivo principal de Greengrass, revise la implementación en la que se instaló el componente y quítelo de la implementación. El softwareAWS IoT Greengrass Core desinstala un componente cuando ninguna de las implementaciones de un dispositivo principal especifica ese componente. Para obtener más información, consulte Implemente AWS IoT Greengrass componentes en los dispositivos.

Para desinstalar el componente Systems Manager Agent (consola)

1. En el menú de navegación de la AWS IoT Greengrassconsola, selecciona Dispositivos principales.
2. Elija el dispositivo principal en el que desee desinstalar el componente Systems Manager Agent.
3. En la página de detalles del dispositivo principal, elija la pestaña Implementaciones.
4. Elija la implementación que implementa el componente Systems Manager Agent en el dispositivo principal.
5. En la página de detalles de la implementación, seleccione Revisar.
6. En el modo Revisar implementación, elija Revisar implementación.
7. En el paso 1: especifique el objetivo, elija Siguiente.
8. En el paso 2: seleccione los componentes, borre la selección del aws.greengrass.SystemsManagerAgentcomponente y, a continuación, elija Siguiente.
9. En el paso 3: Configurar los componentes, elija Siguiente.
10. En el paso 4: Configurar los ajustes avanzados, selecciona Siguiente.
11. En el paso 5: Revisar, elija Implementar.

Para desinstalar el componente Systems Manager Agent (CLI)

Para desinstalar el componente Systems Manager Agent, revise la implementación que lo implementa y elimínelo de la implementación. Para obtener más información, consulte Revisar las implementaciones.

La implementación puede tardar varios minutos en completarse. Puede utilizar elAWS IoT Greengrass servicio para comprobar el estado de la implementación. Para obtener más información, consulte Verificar el estado de implementación.

Paso 3: Desinstalar el software de Systems Manager

El software Systems Manager Agent sigue ejecutándose en el dispositivo principal después de eliminar el componente Systems Manager Agent. Para eliminar el software Systems Manager Agent, puede ejecutar comandos en el dispositivo principal. Para obtener más información, consulte Desinstalar el agente de Systems Manager de las instancias de Linux en la Guía delAWS Systems Manager usuario.

Seguridad en AWS IoT Greengrass

La seguridad en la nube de AWS es la máxima prioridad. Como cliente de AWS, se beneficia de una arquitectura de red y un centro de datos diseñados para satisfacer los requisitos de seguridad de las organizaciones más exigentes.

La seguridad es una responsabilidad compartida entre AWS y usted. El modelo de responsabilidad compartida la describe como seguridad de la nube y seguridad en la nube:

- Seguridad de la nube: AWS es responsable de proteger la infraestructura que ejecuta servicios de AWS en la Nube de AWS. AWS también le proporciona servicios que puede utilizar de forma segura. Los auditores externos prueban y verifican periódicamente la eficacia de nuestra seguridad como parte de los Programas de conformidad de AWS . Para obtener información sobre los programas de conformidad que se aplican a AWS IoT Greengrass, consulte Servicios de AWS en el ámbito del programa de conformidad.
- Seguridad en la nube: su responsabilidad viene determinada por el servicio de AWS que utilice. También es responsable de otros factores, incluida la confidencialidad de los datos, los requisitos de la empresa y la legislación y los reglamentos aplicables.

Cuando utiliza AWS IoT Greengrass, también es responsable de proteger los dispositivos, la conexión de red local y las claves privadas.

Esta documentación le ayuda a comprender cómo aplicar el modelo de responsabilidad compartida a la hora de utilizar AWS IoT Greengrass. Los siguientes temas le mostrarán cómo configurar AWS IoT Greengrass para satisfacer sus objetivos de seguridad y de conformidad. También puede aprender a utilizar otros servicios de AWS que lo ayuden a supervisar y proteger los recursos de AWS IoT Greengrass.

Temas

- Seguridad de la infraestructura en AWS IoT Greengrass
- Configuración y análisis de vulnerabilidades en AWS IoT Greengrass
- Integridad de código enAWS IoT Greengrass V2
- AWS IoT Greengrass y puntos de conexión de VPC de interfaz (AWS PrivateLink)
- Prácticas recomendadas de seguridad para AWS IoT Greengrass

Protección de los datos en AWS IoT Greengrass

El modelo de responsabilidad compartida de AWS se aplica a la protección de datos de AWS IoT Greengrass. Como se describe en este modelo, AWS es responsable de proteger la infraestructura global que ejecuta toda la Nube de AWS. Usted es responsable de mantener el control sobre el contenido alojado en esta infraestructura. Usted también es responsable de las tareas de administración y configuración de seguridad para los Servicios de AWS que utiliza. Para obtener más información sobre la privacidad de los datos, consulte las Preguntas frecuentes sobre la privacidad de datos. Para obtener información sobre la protección de datos en Europa, consulte la publicación de blog Modelo de responsabilidad compartida y GDPR de AWS en el Blog de seguridad de AWS.

Con fines de protección de datos, recomendamos proteger las credenciales de la cuenta de Cuenta de AWS y configurar cuentas de usuario individuales con AWS IAM Identity Center o AWS Identity and Access Management (IAM). De esta manera, solo se otorgan a cada usuario los permisos necesarios para cumplir sus obligaciones laborales. También recomendamos proteger sus datos de la siguiente manera:

- Utilice autenticación multifactor (MFA) en cada cuenta.
- Utilice SSL/TLS para comunicarse con los recursos de AWS. reSe recomienda el uso de TLS 1.2 y recomendamos TLS 1.3.
- Configure la API y el registro de actividad del usuario con AWS CloudTrail.
- Utilice las soluciones de cifrado de AWS, junto con todos los controles de seguridad predeterminados dentro de los Servicios de AWS.
- Utilice servicios de seguridad gestionados avanzados, como Amazon Macie, que lo ayuden a detectar y proteger los datos confidenciales almacenados en Amazon S3.
- Si necesita módulos criptográficos validados FIPS 140-2 al acceder a AWS a través de una interfaz de la línea de comandos o una API, utilice un punto de conexión de FIPS. Para obtener más información sobre los puntos de conexión de FIPS disponibles, consulte Estándar de procesamiento de la información federal (FIPS) 140-2.

Se recomienda encarecidamente no ingresar nunca información confidencial o sensible, como, por ejemplo, direcciones de correo electrónico de clientes, en etiquetas o campos de formato libre, tales como el campo Nombre. Esto incluye las situaciones en las que debe trabajar con la AWS IoT Greengrass u otros Servicios de AWS a través de la consola, la API, la AWS CLI o los SDK de AWS. Cualquier dato que ingrese en etiquetas o campos de formato libre utilizados para nombres se puede emplear para los registros de facturación o diagnóstico. Si proporciona una URL a un servidor externo, recomendamos encarecidamente que no incluya información de credenciales en la URL a fin de validar la solicitud para ese servidor.

Para obtener más información acerca de la protección de información confidencial en AWS IoT Greengrass, consulte the section called “No registre información confidencial”.

Para obtener más información sobre la protección de datos, consulte la entrada de blog relativa al modelo de responsabilidad compartida de AWS y GDPR en el blog de seguridad de AWS.

Temas

Cifrado de datos

AWS IoT Greengrass utiliza el cifrado para proteger los datos mientras están en tránsito (a través de Internet o red local) y en reposo (almacenados en la Nube de AWS).

Los dispositivos de un entorno de AWS IoT Greengrass suelen recopilar datos que se envían a los servicios de AWS para su posterior procesamiento. Para obtener más información acerca del cifrado de datos en otros servicios de AWS, consulte la documentación de seguridad de ese servicio.

Temas

Cifrado en tránsito

AWS IoT Greengrasstiene dos modos de comunicación donde los datos están en tránsito:

- the section called “Datos en tránsito a través de Internet”. Comunicación entre un núcleo de Greengrass yAWS IoT Greengrassa través de Internet está cifrada.
- the section called “Datos del dispositivo central”. La comunicación entre componentes en el dispositivo del núcleo de Greengrass no está cifrada.

Datos en tránsito a través de Internet

AWS IoT Greengrass utiliza Transport Layer Security (TLS) para cifrar toda la comunicación a través de Internet. Todos los datos enviados alNube de AWSse envía a través de una conexión TLS con protocolos MQTT o HTTPS, por lo que es segura de forma predeterminada.AWS IoT Greengrassutiliza elAWS IoTmodelo de seguridad de transporte de. Para obtener más información, consulte Seguridad de transporte en la Guía del desarrollador de AWS IoT Core.

Datos del dispositivo central

AWS IoT Greengrass no cifra los datos intercambiados localmente en el dispositivo del núcleo de Greengrass porque los datos no salen del dispositivo. Esto incluye la comunicación entre los componentes definidos por el usuario, elAWS IoTSDK de dispositivos y componentes públicos, como el administrador de flujos.

Cifrado en reposo

AWS IoT Greengrass almacena los datos:

- the section called “Datos en reposo delNube de AWS”. Estos datos se cifran.
- the section called “Datos en reposo en el núcleo de Greengrass”. Estos datos no están cifrados (excepto las copias locales de sus secretos).

Datos en reposo delNube de AWS

AWS IoT Greengrasscifra los datos de los clientes almacenados en elNube de AWS. Estos datos están protegidos mediante claves de AWS KMS administradas por AWS IoT Greengrass.

Datos en reposo en el núcleo de Greengrass

AWS IoT Greengrass se basa en permisos de archivos Unix y cifrado de disco completo (si está habilitado) para proteger los datos en reposo en el núcleo. Tiene la responsabilidad de proteger el sistema de archivos y el dispositivo.

Sin embargo, AWS IoT Greengrass cifra las copias locales de sus secretos recuperados de AWS Secrets Manager. Para obtener más información, consulte laSecrets Managercomponente.

Administración de claves en el dispositivo del núcleo de Greengrass

Es responsabilidad del cliente garantizar el almacenamiento seguro de claves criptográficas (públicas y privadas) en el dispositivo del núcleo de Greengrass.AWS IoT Greengrassutiliza claves públicas y privadas para el siguiente escenario:

- La clave de cliente de IoT se utiliza con el certificado IoT para autenticar el protocolo de enlace Transport Layer Security (TLS) cuando un núcleo de Greengrass se conecta a AWS IoT Core. Para obtener más información, consulte the section called “Autenticación y autorización de dispositivos”.

  > **Note**
  >
  > La clave y el certificado también se conocen como clave privada del núcleo y el certificado de dispositivo del núcleo.

Un dispositivo del núcleo de Greengrass admite el almacenamiento de claves privadas mediante permisos del sistema de archivos omódulo de seguridad de hardware. Si utiliza claves privadas basadas en el sistema de archivos, es responsable de su almacenamiento seguro en el dispositivo del núcleo.

Integración de la seguridad de hardware

> **Note**
>
> Esta función está disponible para la versión 2.5.3 y versiones posteriores del componente núcleo de Greengrass. AWS IoT Greengrassactualmente no admite esta función en los dispositivos principales de Windows.

Puede configurar el software AWS IoT Greengrass Core para que utilice un módulo de seguridad de hardware (HSM) a través de la interfaz PKCS #11. Esta función le permite almacenar de forma segura la clave privada y el certificado del dispositivo para que no queden expuestos ni duplicados en el software. Puede almacenar la clave privada y el certificado en un módulo de hardware, como un HSM o un módulo de plataforma segura (TPM).

El software AWS IoT Greengrass Core utiliza una clave privada y un certificado X.509 para autenticar las conexiones a los servicios y. AWS IoT AWS IoT Greengrass El componente de administrador de secretos utiliza esta clave privada para cifrar y descifrar de forma segura los secretos que se despliegan en un dispositivo central de Greengrass. Al configurar un dispositivo principal para usar un HSM, estos componentes utilizan la clave privada y el certificado que se almacenan en el HSM.

El componente broker MQTT de Moquette también almacena una clave privada para su certificado de servidor MQTT local. Este componente almacena la clave privada en el sistema de archivos del dispositivo, en la carpeta de trabajo del componente. Actualmente, AWS IoT Greengrass no admite el almacenamiento de esta clave privada o certificado en un HSM.

> **Tip**
>
> Busque los dispositivos que admitan esta función en el catálogo de dispositivos AWS asociados.

Temas

Requisitos

Debe cumplir los siguientes requisitos para utilizar un HSM en un dispositivo principal de Greengrass:

- Greengrass nucleus v2.5.3 o posterior instalado en el dispositivo principal. Puede elegir una versión compatible al instalar el software AWS IoT Greengrass Core en un dispositivo principal.
- El componente del proveedor PKCS #11 está instalado en el dispositivo principal. Puede descargar e instalar este componente al instalar el software AWS IoT Greengrass Core en un dispositivo principal.
- Módulo de seguridad de hardware que admite el esquema de firmas PKCS #1 v1.5 y claves RSA con un tamaño de clave RSA-2048 (o mayor) o claves ECC.

> **Note**
>
> Para usar un módulo de seguridad de hardware con claves ECC, debe usar Greengrass nucleus v2.5.6 o posterior.
> Para usar un módulo de seguridad de hardware y un administrador de secretos, debe usar un módulo de seguridad de hardware con claves RSA.

- Una biblioteca de proveedores PKCS #11 que el software AWS IoT Greengrass principal puede cargar en tiempo de ejecución (mediante libdl) para invocar las funciones PKCS #11. La biblioteca de proveedores PKCS #11 debe implementar las siguientes operaciones de la API PKCS #11:
  - `C_Initialize`
  - `C_Finalize`
  - `C_GetSlotList`
  - `C_GetSlotInfo`
  - `C_GetTokenInfo`
  - `C_OpenSession`
  - `C_GetSessionInfo`
  - `C_CloseSession`
  - `C_Login`
  - `C_Logout`
  - `C_GetAttributeValue`
  - `C_FindObjectsInit`
  - `C_FindObjects`
  - `C_FindObjectsFinal`
  - `C_DecryptInit`
  - `C_Decrypt`
  - `C_DecryptUpdate`
  - `C_DecryptFinal`
  - `C_SignInit`
  - `C_Sign`
  - `C_SignUpdate`

  - `C_SignFinal`
  - `C_GetMechanismList`
  - `C_GetMechanismInfo`
  - `C_GetInfo`
  - `C_GetFunctionList`
- El módulo de hardware debe resolverlo la etiqueta de ranura, tal y como se define en la especificación de PKCS#11.
- Debe almacenar la clave privada y el certificado en el HSM, en la misma ranura, y deben usar la misma etiqueta de objeto e ID de objeto, si el HSM admite los ID de objeto.
- El certificado y la clave privada deben poder resolverse mediante etiquetas de objetos.
- La clave privada debe tener los siguientes permisos:
  - `sign`
  - `decrypt`
- (Opcional) Para usar el componente de administrador de secretos, debe usar la versión 2.1.0 o posterior, y la clave privada debe tener los siguientes permisos:
  - `unwrap`
  - `wrap`

Prácticas recomendadas de seguridad de hardware

Tenga en cuenta las siguientes prácticas recomendadas al configurar la seguridad del hardware en los dispositivos principales de Greengrass.

- Genere claves privadas directamente en el HSM utilizando el generador de números aleatorios de hardware interno. Este enfoque es más seguro que importar una clave privada que se genere en otro lugar, ya que la clave privada permanece dentro del HSM.
- Configure las claves privadas para que sean inmutables y prohíba la exportación.
- Utilice la herramienta de aprovisionamiento que el proveedor de hardware de HSM recomienda para generar una solicitud de firma de certificado (CSR) con la clave privada protegida por hardware y, a continuación, utilice la consola o la API para generar un certificado de cliente. AWS IoT

> **Note**
>
> La práctica recomendada de seguridad consistente en rotar las claves no se aplica cuando se generan claves privadas en un HSM.

Instale el software AWS IoT Greengrass principal con seguridad de hardware

Al instalar el software AWS IoT Greengrass Core, puede configurarlo para que utilice una clave privada que genere en un HSM. Este enfoque sigue las prácticas recomendadas de seguridad para generar la clave privada en el HSM, de modo que la clave privada permanezca dentro del HSM.

Para instalar el software AWS IoT Greengrass principal con seguridad de hardware, haga lo siguiente:

1. Genere una clave privada en el HSM.
2. Cree una solicitud de firma de certificado (CSR) a partir de la clave privada.
3. Cree un certificado a partir de la CSR. Puede crear un certificado firmado por AWS IoT o por otra entidad emisora de certificados (CA) raíz. Para obtener más información sobre cómo usar otra CA raíz, consulte Crear sus propios certificados de cliente en la Guía para AWS IoT Core desarrolladores.
4. Descargue el AWS IoT certificado e impórtelo al HSM.
5. Instale el software AWS IoT Greengrass principal desde un archivo de configuración que especifique el uso del componente proveedor PKCS #11 y la clave privada y el certificado en el HSM.

Puede elegir una de las siguientes opciones de instalación para instalar el software AWS IoT Greengrass principal con seguridad de hardware:

- Instalación manual

  Seleccione esta opción para crear manualmente los AWS recursos necesarios y configurar la seguridad del hardware. Para obtener más información, consulte Instale el software AWS IoT Greengrass principal con aprovisionamiento manual de recursos.
- Instalación con aprovisionamiento personalizado

Elija esta opción para desarrollar una aplicación Java personalizada que cree automáticamente los AWS recursos necesarios y configure la seguridad del hardware. Para obtener más información, consulte Instale el software AWS IoT Greengrass principal con aprovisionamiento de recursos personalizado.

Actualmente, AWS IoT Greengrass no admite la instalación del software AWS IoT Greengrass principal con seguridad de hardware cuando se instala con el aprovisionamiento automático de recursos o el aprovisionamiento de AWS IoTflotas.

Configura la seguridad del hardware en un dispositivo principal existente

Puede importar la clave privada y el certificado de un dispositivo principal a un HSM para configurar la seguridad del hardware.

Consideraciones

- Debe tener acceso root al sistema de archivos del dispositivo principal.
- En este procedimiento, se apaga el software AWS IoT Greengrass principal para que el dispositivo principal quede desconectado y no esté disponible mientras se configura la seguridad del hardware.

Para configurar la seguridad del hardware en un dispositivo principal existente, haga lo siguiente:

1. Inicialice el HSM.
2. Implemente el componente del proveedor PKCS #11 en el dispositivo principal.
3. Detenga el software AWS IoT Greengrass principal.
4. Importe la clave privada y el certificado del dispositivo principal al HSM.
5. Actualice el archivo de configuración del software AWS IoT Greengrass principal para usar la clave privada y el certificado del HSM.
6. Inicie el software AWS IoT Greengrass principal.

Paso 1: inicialice el módulo de seguridad de hardware

Complete el siguiente paso para inicializar el HSM en su dispositivo principal.

Para inicializar el módulo de seguridad de hardware

- Inicialice un token PKCS #11 en el HSM y guarde el ID de ranura y el PIN de usuario correspondientes al token. Consulta la documentación de tu HSM para obtener información sobre cómo inicializar un token. El ID de ranura y el PIN de usuario se utilizan más adelante al implementar y configurar el componente de proveedor PKCS #11.

Paso 2: Implemente el componente de proveedor PKCS #11

Complete los siguientes pasos para implementar y configurar el componente de proveedor PKCS #11. Puede implementar el componente en uno o más dispositivos principales.

Para implementar el componente proveedor PKCS #11 (consola)

1. En el menú de navegación de la AWS IoT Greengrassconsola, elija Componentes.
2. En la página Componentes, seleccione la pestaña Componentes públicos y, a continuación, elija aws.greengrass.crypto.Pkcs11Provider.
3. En la página aws.greengrass.crypto.Pkcs11Provider, elija Implementar.
4. En Añadir a la implementación, elija una implementación existente para revisarla o cree una nueva y, a continuación, elija Siguiente.
5. Si opta por crear una nueva implementación, elija el dispositivo principal o el grupo de cosas de destino para la implementación. En la página Especificar el destino, en Destino del despliegue, elija un dispositivo principal o un grupo de cosas y, a continuación, elija Siguiente.
6. En la página Seleccionar componentes, en Componentes públicos, seleccione y aws.greengrass.crypto.Pkcs11Provider, a continuación, elija Siguiente.
7. En la página Configurar componentes, seleccione y aws.greengrass.crypto.Pkcs11Provider, a continuación, haga lo siguiente:

   a. Seleccione Configurar componente.

   b. En el aws.greengrass.crypto.Pkcs11Provider modal Configurar, en Actualización de configuración, en Configuración para fusionar, introduzca la siguiente actualización de configuración. Actualice los siguientes parámetros de configuración con los valores de los dispositivos principales de destino. Especifique el ID de ranura y el PIN de usuario en los que inicializó anteriormente el token PKCS #11. Más adelante, importará la clave privada y el certificado a esta ranura del HSM.

`name`

Un nombre para la configuración PKCS #11.

`library`

La ruta absoluta del archivo a la biblioteca de la implementación de PKCS #11 que el software AWS IoT Greengrass Core puede cargar con libdl.

`slot`

El ID de la ranura que contiene la clave privada y el certificado del dispositivo. Este valor es diferente del índice o la etiqueta de la ranura.

`userPin`

El PIN del usuario que se utilizará para acceder a la ranura.

```
{
  "name": "softhsm_pkcs11",
  "library": "/usr/lib/softhsm/libsofthsm2.so",
  "slot": 1,
  "userPin": "1234"
}
```

c. Selecciona Confirmar para cerrar el modal y, a continuación, selecciona Siguiente.

8. En la página Configurar ajustes avanzados, mantenga los ajustes de configuración predeterminados y seleccione Siguiente.
9. En la página Revisar, elija Implementar.

   La implementación puede tardar hasta un minuto en completarse.

Para implementar el componente de proveedor PKCS #11 () AWS CLI

Para implementar el componente proveedor PKCS #11, cree un documento de despliegue que lo incluya `aws.greengrass.crypto.Pkcs11Provider` en el `components` objeto y especifique la actualización de configuración del componente. Siga las instrucciones Crear implementaciones para crear una nueva implementación o revisar una implementación existente.

El siguiente ejemplo de documento de despliegue parcial especifica el despliegue y la configuración del componente de proveedor PKCS #11. Actualice los siguientes parámetros de configuración con

los valores de los dispositivos principales de destino. Guarde el ID de ranura y el PIN de usuario para usarlos más adelante cuando importe la clave privada y el certificado al HSM.

`name`

Nombre para la configuración PKCS #11.

`library`

La ruta absoluta del archivo a la biblioteca de la implementación de PKCS #11 que el software AWS IoT Greengrass Core puede cargar con libdl.

`slot`

El ID de la ranura que contiene la clave privada y el certificado del dispositivo. Este valor es diferente del índice o la etiqueta de la ranura.

`userPin`

El PIN del usuario que se utilizará para acceder a la ranura.

```
{
  "name": "softhsm_pkcs11",
  "library": "/usr/lib/softhsm/libsofthsm2.so",
  "slot": 1,
  "userPin": "1234"
}
```

```
{
  ...,
  "components": {
    ...,
    "aws.greengrass.crypto.Pkcs11Provider": {
      "componentVersion": "2.0.0",
      "configurationUpdate": {
        "merge": "{\"name\":\"softhsm_pkcs11\",\"library\":\"/usr/lib/softhsm/
libsofthsm2.so\",\"slot\":1,\"userPin\":\"1234\"}"
      }
    }
  }
}
```

La implementación puede tardar varios minutos en completarse. Puede utilizar el AWS IoT Greengrass servicio para comprobar el estado de la implementación. Puede consultar los registros del software AWS IoT Greengrass principal para comprobar que el componente del proveedor PKCS #11 se implementa correctamente. Para obtener más información, consulte los siguientes temas:

- Verificar el estado de implementación
- Supervisar AWS IoT Greengrass registros

Si la implementación falla, puede solucionar el problema de la implementación en cada dispositivo principal. Para obtener más información, consulte Solución de problemas AWS IoT Greengrass V2.

Paso 3: Actualice la configuración del dispositivo principal

El software AWS IoT Greengrass Core utiliza un archivo de configuración que especifica cómo funciona el dispositivo. Este archivo de configuración incluye dónde encontrar la clave privada y el certificado que utiliza el dispositivo para conectarse alNube de AWS. Complete los siguientes pasos para importar la clave privada y el certificado del dispositivo principal al HSM y actualice el archivo de configuración para usar el HSM.

Para actualizar la configuración del dispositivo principal para utilizar la seguridad del hardware

1. Detenga el software AWS IoT Greengrass principal. Si configuró el software AWS IoT Greengrass Core como un servicio del sistema con systemd, puede ejecutar el siguiente comando para detener el software.

   ```
   sudo systemctl stop greengrass.service
   ```

2. Busque la clave privada y los archivos de certificado del dispositivo principal.

   - Si instaló el software AWS IoT Greengrass Core con el aprovisionamiento automático o el aprovisionamiento de flota, la clave privada está en */greengrass/v2*`/privKey.key` y el certificado está en. */greengrass/v2*`/thingCert.crt`
   - Si instaló el software AWS IoT Greengrass Core con el aprovisionamiento manual, la clave privada existe de forma */greengrass/v2*`/private.pem.key` predeterminada y el certificado existe de forma predeterminada. */greengrass/v2*`/device.pem.crt`

También puede comprobar las `system.certificateFilePath` propiedades `system.privateKeyPath` y `/greengrass/v2/config/effectiveConfig.yaml` para encontrar la ubicación de estos archivos.

3. Importe la clave privada y el certificado al HSM. Consulte la documentación de su HSM para obtener información sobre cómo importarle claves privadas y certificados. Importe la clave privada y el certificado utilizando el ID de ranura y el PIN de usuario en los que inicializó anteriormente el token PKCS #11. Debe usar la misma etiqueta de objeto e ID de objeto para la clave privada y el certificado. Guarde la etiqueta de objeto que especifique al importar cada archivo. Esta etiqueta se utiliza más adelante cuando se actualiza la configuración del software AWS IoT Greengrass principal para utilizar la clave privada y el certificado en el HSM.
4. Actualice la configuración AWS IoT Greengrass principal para usar la clave privada y el certificado en el HSM. Para actualizar la configuración, modifique el archivo de configuración AWS IoT Greengrass principal y ejecute el software AWS IoT Greengrass principal con el archivo de configuración actualizado para aplicar la nueva configuración.

   Haga lo siguiente:

   a. Cree una copia de seguridad del archivo de configuración AWS IoT Greengrass principal. Puede utilizar esta copia de seguridad para restaurar el dispositivo principal si tiene problemas al configurar la seguridad del hardware.

      ```
      sudo cp /greengrass/v2/config/effectiveConfig.yaml ~/ggc-config-backup.yaml
      ```

   b. Abra el archivo de configuración del AWS IoT Greengrass núcleo en un editor de texto. Por ejemplo, puede ejecutar el siguiente comando para usar GNU nano para editar el archivo. */greengrass/v2*Sustitúyala por la ruta a la carpeta raíz de Greengrass.

      ```
      sudo nano /greengrass/v2/config/effectiveConfig.yaml
      ```

   c. Sustituya el valor de por el URI PKCS #11 para la clave privada del HSM. `system.privateKeyPath` Sustituya *`iotdevicekey`* por la etiqueta del objeto donde importó anteriormente la clave privada y el certificado.

      ```
      pkcs11:object=iotdevicekey;type=private
      ```

d. Sustituya el valor de por el `system.certificateFilePath` URI PKCS #11 del certificado en el HSM. Sustituya *`iotdevicekey`* por la etiqueta de objeto en la que importó anteriormente la clave privada y el certificado.

```
pkcs11:object=iotdevicekey;type=cert
```

Una vez finalizados estos pasos, la `system` propiedad del archivo de configuración AWS IoT Greengrass principal debería tener un aspecto similar al del ejemplo siguiente.

```
system:
  certificateFilePath: "pkcs11:object=iotdevicekey;type=cert"
  privateKeyPath: "pkcs11:object=iotdevicekey;type=private"
  rootCaPath: "/greengrass/v2/rootCA.pem"
  rootpath: "/greengrass/v2"
  thingName: "MyGreengrassCore"
```

5. Aplique la configuración en el `effectiveConfig.yaml` archivo actualizado. Ejecute `Greengrass.jar` con el `--init-config` parámetro en el que desee aplicar la configuración`effectiveConfig.yaml`. */greengrass/v2*Sustitúyala por la ruta a la carpeta raíz de 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. Inicie el software AWS IoT Greengrass principal. Si configuró el software AWS IoT Greengrass Core como un servicio del sistema con systemd, puede ejecutar el siguiente comando para iniciar el software.

```
sudo systemctl start greengrass.service
```

Para obtener más información, consulte Ejecute el software AWS IoT Greengrass principal.

7. Compruebe los registros del software AWS IoT Greengrass principal para comprobar que el software se inicia y se conecta alNube de AWS. El software AWS IoT Greengrass principal utiliza la clave privada y el certificado para conectarse a los AWS IoT Greengrass servicios AWS IoT y.

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

Los siguientes mensajes de registro a nivel de información indican que el software AWS IoT Greengrass principal se ha conectado correctamente a los servicios AWS IoT andAWS IoT Greengrass.

```
2021-12-06T22:47:53.702Z [INFO] (Thread-3)
 com.aws.greengrass.mqttclient.AwsIotMqttClient: Successfully connected to AWS IoT
 Core. {clientId=MyGreengrassCore5, sessionPresent=false}
```

8. (Opcional) Tras comprobar que el software AWS IoT Greengrass principal funciona con la clave privada y el certificado del HSM, elimine la clave privada y los archivos de certificado del sistema de archivos del dispositivo. Ejecute el siguiente comando y sustituya las rutas de los archivos por las rutas a los archivos de clave privada y certificado.

```
sudo rm /greengrass/v2/privKey.key
sudo rm /greengrass/v2/thingCert.crt
```

Utilice hardware que no sea compatible con PKCS #11

La biblioteca de PKCS#11 suele ser proporcionada por el proveedor de hardware o es de código abierto. Por ejemplo, con hardware que cumplan los estándares (como TPM1.2), se puede utilizar software de código abierto. Sin embargo, si su hardware no tiene una implementación de biblioteca PKCS #11 correspondiente o si desea crear un proveedor PKCS #11 personalizado, póngase en contacto con su representante de Amazon Web Services Enterprise Support si tiene preguntas relacionadas con la integración.

Véase también

- Guía de uso de la interfaz de token criptográfico PKCS #11, versión 2.4.0
- RFC 7512
- PKCS #1: Cifrado RSA versión 1.5

Autenticación y autorización de dispositivos en AWS IoT Greengrass

Los dispositivos en entornos de AWS IoT Greengrass utilizan certificados X.509 para la autenticación y políticas de AWS IoT para la autorización. Los certificados y las políticas permiten que los dispositivos se conecten de forma segura entre sí, con AWS IoT Core y AWS IoT Greengrass.

Los certificados X.509 son certificados digitales que utilizan el estándar de infraestructura de clave pública X.509 para asociar una clave pública a una identidad contenida en un certificado. Una entidad de confianza conocida como entidad de certificación (CA) emite los certificados X.509. La CA administra uno o varios certificados especiales llamados certificados de CA, que utiliza para generar certificados X.509. Solo la entidad de certificación tiene acceso a los certificados de entidad de certificación.

Las políticas de AWS IoT definen el conjunto de operaciones permitidas para los dispositivos de AWS IoT. Específicamente, permiten y deniegan el acceso a operaciones de plano de datos de AWS IoT Core y AWS IoT Greengrass, como la publicación de mensajes MQTT y la recuperación de sombras de dispositivos.

Todos los dispositivos requieren una entrada en el registro de AWS IoT Core y un certificado X.509 activado con una política de AWS IoT asociada. Los dispositivos se dividen en dos categorías:

- Dispositivos principales de Greengrass

  Los dispositivos principales de Greengrass utilizan certificados y AWS IoT políticas para conectarse AWS IoT Core y. AWS IoT Greengrass Los certificados y las políticas también permiten AWS IoT Greengrass implementar componentes y configuraciones en los dispositivos principales.

- Dispositivos cliente

  Los dispositivos cliente de MQTT utilizan certificados y políticas para conectarse al AWS IoT Greengrass servicio AWS IoT Core y al mismo. Esto permite a los dispositivos cliente utilizar la detección AWS IoT Greengrass en la nube para buscar y conectarse a un dispositivo principal de Greengrass. Un dispositivo cliente utiliza el mismo certificado para conectarse al servicio AWS IoT Core en la nube y a los dispositivos principales. Los dispositivos de cliente también utilizan información de detección para la autenticación mutua con el dispositivo principal. Para obtener más información, consulte Interactúa con dispositivos IoT locales.

Certificados X.509

La comunicación entre los dispositivos principales y los dispositivos cliente y entre los dispositivos AWS IoT Core y/o AWS IoT Greengrass debe autenticarse. Esta autenticación mutua se basa en certificados de dispositivo X.509 registrados y claves criptográficas.

En un entorno de AWS IoT Greengrass, los dispositivos utilizan certificados con claves públicas y privadas para las siguientes conexiones de Transport Layer Security (TLS):

- El componente de AWS IoT cliente del dispositivo principal de Greengrass que se conecta a Internet AWS IoT Core y a AWS IoT Greengrass través de ella.
- Dispositivos cliente que se conectan a AWS IoT Greengrass través de Internet para descubrir los dispositivos principales.
- El componente intermediario MQTT del núcleo de Greengrass que se conecta a los dispositivos Greengrass del grupo a través de la red local.

AWS IoT Greengrasslos dispositivos principales almacenan los certificados en la carpeta raíz de Greengrass.

Certificados de entidad de certificación (CA)

Los dispositivos principales y los dispositivos cliente de Greengrass descargan un certificado de CA raíz que se utiliza para la autenticación con los servicios AWS IoT Core yAWS IoT Greengrass. Le recomendamos que utilice un certificado de entidad de certificación raíz de Amazon Trust Services (ATS), como Amazon Root CA 1. Para obtener más información, consulte Certificados de CA para autenticación de servidor en la Guía del desarrollador de AWS IoT Core.

Los dispositivos cliente también descargan un certificado de CA para dispositivos principales de Greengrass. Utilizan este certificado para validar el certificado del servidor MQTT en el dispositivo principal durante la autenticación mutua.

Rotación de certificados en el broker MQTT local

Al habilitar la compatibilidad con dispositivos cliente, los dispositivos principales de Greengrass generan un certificado de servidor MQTT local que los dispositivos cliente utilizan para la autenticación mutua. Este certificado está firmado por el certificado CA del dispositivo principal, que el dispositivo principal almacena en la AWS IoT Greengrass nube. Los dispositivos cliente recuperan el certificado de CA del dispositivo principal cuando descubren el dispositivo principal. Utilizan el certificado de CA del dispositivo principal para verificar el certificado del servidor MQTT del

dispositivo principal cuando se conectan al dispositivo principal. El certificado de CA del dispositivo principal caduca a los 5 años.

El certificado del servidor MQTT caduca cada 7 días de forma predeterminada, y puede configurar esta duración entre 2 y 10 días. Este período limitado se basa en las prácticas recomendadas de seguridad. Esta rotación ayuda a mitigar la amenaza de que un atacante robe el certificado del servidor MQTT y la clave privada para hacerse pasar por el dispositivo principal de Greengrass.

El dispositivo principal de Greengrass rota el certificado del servidor MQTT 24 horas antes de que caduque. El dispositivo principal de Greengrass genera un nuevo certificado y reinicia el broker MQTT local. Cuando esto ocurre, todos los dispositivos cliente conectados al dispositivo principal de Greengrass se desconectan. Los dispositivos cliente pueden volver a conectarse al dispositivo principal de Greengrass después de un breve período de tiempo.

Políticas de AWS IoT para operaciones de plano de datos

Utilice AWS IoT políticas para autorizar el acceso a los planos de AWS IoT Greengrass datos AWS IoT Core y a los mismos. El plano de AWS IoT Core datos proporciona operaciones para dispositivos, usuarios y aplicaciones. Estas operaciones incluyen la posibilidad de conectarse a los temas AWS IoT Core y suscribirse a ellos. El plano AWS IoT Greengrass de datos proporciona operaciones para los dispositivos Greengrass. Para obtener más información, consulte Acciones de política de AWS IoT Greengrass V2. Estas operaciones incluyen la capacidad de resolver las dependencias de los componentes y descargar artefactos de componentes públicos.

Una política de AWS IoT es un documento JSON que es similar a una política de IAM. Contiene una o varias instrucciones de política que especifican las siguientes propiedades:

- `Effect`. El modo de acceso, que puede ser `Allow` o `Deny`.
- `Action`. La lista de acciones permitidas o denegadas por la política.
- `Resource`. La lista de recursos en los que se permite o deniega la acción.

Las políticas de AWS IoT admiten * como caracteres comodín y tratan los caracteres comodín MQTT (+ y #) como cadenas literales. Para obtener más información sobre el comodín *, consulte Uso del comodín en los ARN de los recursos en la Guía del usuario de AWS Identity and Access Management.

Para obtener más información, consulte Políticas de AWS IoT y Acciones de política de AWS IoT en la Guía del desarrollador de AWS IoT Core.

Important

Las variables de política de cosas (`iot:Connection.Thing.*`) no se admiten en las AWS IoT políticas de dispositivos principales ni en las operaciones del plano de datos de Greengrass. En su lugar, puede utilizar un comodín que haga coincidir varios dispositivos con nombres similares. Por ejemplo, puede `MyGreengrassDevice*` especificar que coincida `MyGreengrassDevice1MyGreengrassDevice2`, etc.

Note

AWS IoT Core permite adjuntar políticas de AWS IoT a grupos de objetos a fin de definir los permisos para grupos de dispositivos. Las políticas de grupos de objetos no permiten el acceso a las operaciones del plano de datos de AWS IoT Greengrass. Para permitir que una objeto acceda a una operación del plano de datos de AWS IoT Greengrass, añada el permiso a una política de AWS IoT que adjunta al certificado del objeto.

Acciones de política de AWS IoT Greengrass V2

AWS IoT Greengrass V2define las siguientes acciones políticas que los dispositivos principales y los dispositivos cliente de Greengrass pueden usar en AWS IoT las políticas. Para especificar un recurso para una acción política, utilice el nombre de recurso de Amazon (ARN) del recurso.

Acciones principales del dispositivo

`greengrass:GetComponentVersionArtifact`

Otorga permiso para obtener una URL prefirmada para descargar un artefacto de componente público o un artefacto de componente Lambda.

Este permiso se evalúa cuando un dispositivo principal recibe una implementación que especifica un componente público o una Lambda que tiene artefactos. Si el dispositivo principal ya tiene el artefacto, no lo volverá a descargar.

Tipo de recurso: `componentVersion`

Formato ARN del recurso: `arn:aws:greengrass:`*`region`*`:`*`account-id`*`:components:`*`component-name`*`:versions:`*`component-version`*

`greengrass:ResolveComponentCandidates`

Otorga permiso para identificar una lista de componentes que cumplen con los requisitos de componentes, versiones y plataformas para una implementación. Si los requisitos entran en conflicto o no existe ningún componente que los cumpla, esta operación devuelve un error y la implementación falla en el dispositivo.

Este permiso se evalúa cuando un dispositivo principal recibe una implementación que especifica los componentes.

Tipo de recurso: ninguno

Formato ARN del recurso: *

`greengrass:GetDeploymentConfiguration`

Otorga permiso para obtener una URL prefirmada para descargar un documento de implementación de gran tamaño.

Este permiso se evalúa cuando un dispositivo principal recibe una implementación que especifica un documento de implementación de más de 7 KB (si la implementación se dirige a un grupo de cosas) o 31 KB (si la implementación está dirigida a un grupo de cosas). El documento de despliegue incluye las configuraciones de los componentes, las políticas de despliegue y los metadatos de despliegue. Para obtener más información, consulte Implemente AWS IoT Greengrass componentes en los dispositivos.

Esta función está disponible para la versión 2.3.0 y versiones posteriores del componente núcleo de Greengrass.

Tipo de recurso: ninguno

Formato ARN del recurso: *

`greengrass:ListThingGroupsForCoreDevice`

Otorga permiso para obtener la jerarquía de grupos de cosas de un dispositivo principal.

Este permiso se comprueba cuando un dispositivo principal recibe una implementación desdeAWS IoT Greengrass. El dispositivo principal utiliza esta acción para identificar si se ha eliminado de un grupo de cosas desde la última implementación. Si el dispositivo principal se eliminó de un grupo de cosas y ese grupo de cosas es el objetivo de una implementación en el dispositivo principal, el dispositivo principal elimina los componentes instalados por esa implementación.

Esta función la utilizan la versión 2.5.0 y versiones posteriores del componente núcleo de Greengrass.

Tipo de recurso: `thing` (dispositivo principal)

Formato ARN del recurso: `arn:aws:iot:`*`region`*`:`*`account-id`*`:thing/`*`core-device-thing-name`*

`greengrass:VerifyClientDeviceIdentity`

Otorga permiso para verificar la identidad de un dispositivo cliente que se conecta a un dispositivo principal.

Este permiso se evalúa cuando un dispositivo principal ejecuta el componente de autenticación del dispositivo cliente y recibe una conexión MQTT desde un dispositivo cliente. El dispositivo cliente presenta su certificado de AWS IoT dispositivo. A continuación, el dispositivo principal envía el certificado del dispositivo al servicio AWS IoT Greengrass en la nube para verificar la identidad del dispositivo cliente. Para obtener más información, consulte Interactúa con dispositivos IoT locales.

Tipo de recurso: ninguno

Formato ARN del recurso: *

`greengrass:VerifyClientDeviceIoTCertificateAssociation`

Otorga permiso para comprobar si un dispositivo cliente está asociado a un AWS IoT certificado.

Este permiso se evalúa cuando un dispositivo principal ejecuta el componente de autenticación del dispositivo cliente y autoriza a un dispositivo cliente a conectarse a través de MQTT. Para obtener más información, consulte Interactúa con dispositivos IoT locales.

> **Note**
>
> Para que un dispositivo principal utilice esta operación, la función de servicio Greengrass debe estar asociada a usted Cuenta de AWS y permitir el `iot:DescribeCertificate` permiso.

Tipo de recurso: `thing` (dispositivo cliente)

Formato ARN del recurso: `arn:aws:iot:`*`region`*`:`*`account-id`*`:thing/`*`client-device-thing-name`*

`greengrass:PutCertificateAuthorities`

Otorga permiso para cargar certificados de la autoridad de certificación (CA) que los dispositivos cliente pueden descargar para verificar el dispositivo principal.

Este permiso se evalúa cuando un dispositivo principal instala y ejecuta el componente de autenticación del dispositivo cliente. Este componente crea una entidad emisora de certificados local y utiliza esta operación para cargar sus certificados de CA. Los dispositivos cliente descargan estos certificados de CA cuando utilizan la operación Discover para encontrar los dispositivos principales a los que pueden conectarse. Cuando los dispositivos cliente se conectan a un intermediario MQTT en un dispositivo principal, utilizan estos certificados de CA para verificar la identidad del dispositivo principal. Para obtener más información, consulte Interactúa con dispositivos IoT locales.

Tipo de recurso: ninguno

Formato ARN: *

`greengrass:GetConnectivityInfo`

Otorga permiso para obtener información de conectividad de un dispositivo principal. Esta información describe cómo los dispositivos cliente se pueden conectar al dispositivo principal.

Este permiso se evalúa cuando un dispositivo principal instala y ejecuta el componente de autenticación del dispositivo cliente. Este componente utiliza la información de conectividad para generar certificados de CA válidos para cargarlos en el servicio AWS IoT Greengrass en la nube junto con la PutCertificateAuthoriesoperación. Los dispositivos cliente utilizan estos certificados de CA para verificar la identidad del dispositivo principal. Para obtener más información, consulte Interactúa con dispositivos IoT locales.

También puede utilizar esta operación en el plano de AWS IoT Greengrass control para ver la información de conectividad de un dispositivo principal. Para obtener más información, consulte GetConnectivityInfo en la Referencia de la API de AWS IoT Greengrass V1.

Tipo de recurso: `thing` (dispositivo principal)

Formato ARN del recurso: `arn:aws:iot:`*`region`*`:`*`account-id`*`:thing/`*`core-device-thing-name`*

`greengrass:UpdateConnectivityInfo`

Otorga permiso para actualizar la información de conectividad de un dispositivo principal. Esta información describe cómo los dispositivos cliente se pueden conectar al dispositivo principal.

Este permiso se evalúa cuando un dispositivo principal ejecuta el componente detector de IP. Este componente identifica la información que los dispositivos cliente necesitan para conectarse al dispositivo principal de la red local. A continuación, este componente utiliza esta operación para cargar la información de conectividad en el servicio AWS IoT Greengrass en la nube, de modo que los dispositivos cliente puedan recuperar esta información con la operación Discover. Para obtener más información, consulte Interactúa con dispositivos IoT locales.

También puede utilizar esta operación en el plano de AWS IoT Greengrass control para actualizar manualmente la información de conectividad de un dispositivo principal. Para obtener más información, consulte UpdateConnectivityInfo en la Referencia de la API de AWS IoT Greengrass V1.

Tipo de recurso: `thing` (dispositivo principal)

Formato ARN del recurso: `arn:aws:iot:`*`region`*`:`*`account-id`*`:thing/`*`core-device-thing-name`*

Acciones del dispositivo cliente

`greengrass:Discover`

Otorga permiso para descubrir la información de conectividad de los dispositivos principales a los que se puede conectar un dispositivo cliente. Esta información describe cómo el dispositivo cliente puede conectarse a los dispositivos principales. Un dispositivo cliente solo puede detectar los dispositivos principales a los que esté asociado mediante esta BatchAssociateClientDeviceWithCoreDeviceoperación. Para obtener más información, consulte Interactúa con dispositivos IoT locales.

Tipo de recurso: `thing` (dispositivo cliente)

Formato ARN del recurso: `arn:aws:iot:`*`region`*`:`*`account-id`*`:thing/`*`client-device-thing-name`*

Actualice la política de un dispositivo principal AWS IoT

Puedes usar las AWS IoT consolas AWS IoT Greengrass y la AWS IoT API para ver y actualizar la AWS IoT política de un dispositivo principal.

> **Note**
>
> Si has utilizado el instalador de software AWS IoT Greengrass principal para aprovisionar recursos, tu dispositivo principal tiene una AWS IoT política que permite el acceso a todas AWS IoT Greengrass las acciones (`greengrass:*`). Puedes seguir estos pasos para restringir el acceso únicamente a las acciones que utiliza un dispositivo principal.

Revisa y actualiza la AWS IoT política de un dispositivo principal (consola)

1. En el menú de navegación de la AWS IoT Greengrassconsola, selecciona Dispositivos principales.
2. En la página de dispositivos principales, selecciona el dispositivo principal que deseas actualizar.
3. En la página de detalles del dispositivo principal, selecciona el enlace al dispositivo principal. Este enlace abre la página de detalles del dispositivo en la AWS IoT consola.
4. En la página de detalles de la cosa, selecciona Certificados.
5. En la pestaña Certificados, selecciona el certificado activo del objeto.
6. En la página de detalles del certificado, selecciona Políticas.
7. En la pestaña Políticas, elija la AWS IoT política que desee revisar y actualizar. Puede añadir los permisos necesarios a cualquier política que esté asociada al certificado activo del dispositivo principal.

   > **Note**
   >
   > Si usó el instalador de software AWS IoT Greengrass Core para aprovisionar recursos, tiene dos AWS IoT políticas. Le recomendamos que elija la política nombrada GreengrassV2IoTThingPolicy, si existe. Los dispositivos principales que cree con el instalador rápido utilizan este nombre de política de forma predeterminada. Si agrega permisos a esta política, también los otorga a otros dispositivos principales que usan esta política.

8. En la descripción general de la política, selecciona Editar la versión activa.
9. Revise la política y agregue, elimine o edite los permisos según sea necesario.
10. Para establecer una nueva versión de la política como la versión activa, en Estado de la versión de la política, seleccione Establecer la versión editada como la versión activa de esta política.
11. Seleccione Guardar como versión nueva.

Revisa y actualiza la AWS IoT política de un dispositivo principal (AWS CLI)

1. Enumere los principios del AWS IoT dispositivo principal. Los elementos principales pueden ser certificados de dispositivo X.509 u otros identificadores. Ejecute el siguiente comando y *MyGreengrassCore*sustitúyalo por el nombre del dispositivo principal.

   ```
   aws iot list-thing-principals --thing-name MyGreengrassCore
   ```

   La operación devuelve una respuesta que enumera los elementos principales del dispositivo principal.

   ```
   {
       "principals": [
           "arn:aws:iot:us-west-2:123456789012:cert/certificateId"
       ]
   }
   ```

2. Identifique el certificado activo del dispositivo principal. Ejecute el siguiente comando y sustituya *CertificateID* por el ID de cada certificado del paso anterior hasta que encuentre el certificado activo. El identificador del certificado es la cadena hexadecimal situada al final del ARN del certificado. El `--query` argumento especifica que solo se muestre el estado del certificado.

   ```
   aws iot describe-certificate --certificate-id certificateId --query
    'certificateDescription.status'
   ```

   La operación devuelve el estado del certificado en forma de cadena. Por ejemplo, si el certificado está activo, se genera esta operación`"ACTIVE"`.

3. Enumere las AWS IoT políticas que se adjuntan al certificado. Ejecute el siguiente comando y sustituya el ARN del certificado por el ARN del certificado.

   ```
   aws iot list-principal-policies --principal arn:aws:iot:us-
   west-2:123456789012:cert/certificateId
   ```

   La operación devuelve una respuesta que enumera las AWS IoT políticas adjuntas al certificado.

   ```
   {
       "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. Elija la política que desee ver y actualizar.

> **Note**
>
> Si ha utilizado el instalador de software AWS IoT Greengrass principal para aprovisionar recursos, tiene dos AWS IoT políticas. Le recomendamos que elija la política nombrada GreengrassV2IoTThingPolicy, si existe. Los dispositivos principales que cree con el instalador rápido utilizan este nombre de política de forma predeterminada. Si agrega permisos a esta política, también los otorga a otros dispositivos principales que usan esta política.

5. Obtenga el documento de la política. Ejecute el siguiente comando y sustituya *GreengrassV2IoT* por el nombre ThingPolicy de la política.

```
aws iot get-policy --policy-name GreengrassV2IoTThingPolicy
```

La operación devuelve una respuesta que contiene el documento de la política y otra información sobre la política. El documento de política es un objeto JSON serializado como una cadena.

```
{
    "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. Utilice un conversor en línea u otra herramienta para convertir la cadena del documento de política en un objeto JSON y, a continuación, guárdela en un archivo denominado`iot-policy.json`.

   Por ejemplo, si tiene instalada la herramienta jq, puede ejecutar el siguiente comando para obtener el documento de política, convertirlo en un objeto JSON y guardar el documento de política como un objeto JSON.

   ```
   aws iot get-policy --policy-name GreengrassV2IoTThingPolicy --query
    'policyDocument' | jq fromjson >> iot-policy.json
   ```

7. Revise el documento de política y añada, elimine o edite los permisos según sea necesario.

   Por ejemplo, en un sistema basado en Linux, puede ejecutar el siguiente comando para usar GNU nano y abrir el archivo.

   ```
   nano iot-policy.json
   ```

   Cuando haya terminado, el documento de política podría tener un aspecto similar a la AWS IoTpolítica mínima para los dispositivos principales.

8. Guarda los cambios como una nueva versión de la política. Ejecute el siguiente comando y sustituya *GreengrassV2IoT* por el nombre ThingPolicy de la política.

```
aws iot create-policy-version --policy-name GreengrassV2IoTThingPolicy --policy-
document file://iot-policy.json --set-as-default
```

La operación devuelve una respuesta similar a la del siguiente ejemplo si tiene éxito.

```
{
    "policyArn": "arn:aws:iot:us-west-2:123456789012:policy/
GreengrassV2IoTThingPolicy",
    "policyDocument": "{\
  \\"Version\\": \\"2012-10-17\\",\
  \\"Statement\\": [\
    {\
      \\"Effect\\": \\"Allow\\",\
      \\"Action\\": [\
\\t\\t\\"iot:Connect\\",\
\\t\\t\\"iot:Publish\\",\
\\t\\t\\"iot:Subscribe\\",\
\\t\\t\\"iot:Receive\\",\
\\t\\t\\"greengrass:*\\"\
      ],\
      \\"Resource\\": \\"*\\"\
    }\
  ]\
}",
    "policyVersionId": "2",
    "isDefaultVersion": true
}
```

AWS IoTPolítica mínima para los dispositivos AWS IoT Greengrass V2 principales

> **Important**
>
> Las versiones posteriores del componente núcleo de Greengrass requieren permisos adicionales en la política mínimaAWS IoT. Es posible que tengas que actualizar las AWS IoT políticas de tus dispositivos principales para conceder permisos adicionales.

- Los dispositivos principales que ejecutan Greengrass nucleus v2.5.0 y versiones posteriores utilizan el `greengrass:ListThingGroupsForCoreDevice` permiso para desinstalar componentes al eliminar un dispositivo principal de un grupo de cosas.
- Los dispositivos principales que ejecutan Greengrass nucleus v2.3.0 y versiones posteriores utilizan el `greengrass:GetDeploymentConfiguration` permiso para admitir documentos de configuración de despliegue de gran tamaño.

La siguiente política de ejemplo incluye un conjunto mínimo de acciones necesario para respaldar la funcionalidad básica de Greengrass para su dispositivo del núcleo.

- La `Connect` política incluye un `*` comodín después del nombre del dispositivo principal (por ejemplo,). *`core-device-thing-name`*`*` El dispositivo principal utiliza el mismo certificado de dispositivo para realizar varias suscripciones simultáneasAWS IoT Core, pero es posible que el ID de cliente de una conexión no coincida exactamente con el nombre del dispositivo principal. Después de las primeras 50 suscripciones, el dispositivo principal se utiliza *`core-device-thing-name`*`#`*`number`* como identificador de cliente, que se *`number`* incrementa por cada 50 suscripciones adicionales. Por ejemplo, cuando un dispositivo principal denominado `MyCoreDevice` crea 150 suscripciones simultáneas, utiliza los siguientes ID de cliente:
  - Suscripciones del 1 al 50: `MyCoreDevice`
  - Suscripciones del 51 al 100: `MyCoreDevice#2`
  - Suscripciones 101 a 150: `MyCoreDevice#3`

  El comodín permite que el dispositivo principal se conecte cuando utiliza estos ID de cliente que tienen un sufijo.
- La política muestra los temas de MQTT y los filtros de tema en los que el dispositivo del núcleo puede publicar mensajes, suscribirse y recibir mensajes, incluidos temas utilizados para estado de sombra. Para permitir el intercambio de mensajes entre AWS IoT Core los componentes de Greengrass y los dispositivos cliente, especifique los temas y los filtros de temas que desee permitir. Para obtener más información, consulte Ejemplos de política de publicación/suscripción en la Guía del desarrollador de AWS IoT Core.
- La política concede permiso para publicar datos de telemetría en el siguiente tema.

```
$aws/things/core-device-thing-name/greengrass/health/json
```

Puede eliminar este permiso en los dispositivos principales en los que se deshabilita la telemetría. Para obtener más información, consulte Recopile datos de telemetría del estado del sistema de los dispositivos principales AWS IoT Greengrass.

- La política concede permiso para asumir una función de IAM mediante un AWS IoT alias de función. El dispositivo principal utiliza esta función, denominada función de intercambio de fichas, para adquirir AWS credenciales que puede utilizar para autenticar las solicitudes. AWS Para obtener más información, consulte Autorizar a los dispositivos principales a interactuar con AWS los servicios.

  Al instalar el software AWS IoT Greengrass principal, se crea y se adjunta una segunda AWS IoT política que incluye solo este permiso. Si incluye este permiso en la AWS IoT política principal de su dispositivo principal, puede separar y eliminar la otra AWS IoT política.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": "arn:aws:iot:region:account-id:client/core-device-thing-name*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive",
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:region:account-id:topic/$aws/things/core-device-thing-name/greengrass/health/json",
                "arn:aws:iot:region:account-id:topic/$aws/things/core-device-thing-name/greengrassv2/health/json",
                "arn:aws:iot:region:account-id:topic/$aws/things/core-device-thing-name/jobs/*",
                "arn:aws:iot:region:account-id:topic/$aws/things/core-device-thing-name/shadow/*"
            ]
        },
```

```
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:region:account-id:topicfilter/$aws/things/core-device-
thing-name/jobs/*",
                "arn:aws:iot:region:account-id:topicfilter/$aws/things/core-device-
thing-name/shadow/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "iot:AssumeRoleWithCertificate",
            "Resource": "arn:aws:iot:region:account-id:rolealias/token-exchange-role-
alias-name"
        },
        {
            "Effect": "Allow",
            "Action": [
                "greengrass:GetComponentVersionArtifact",
                "greengrass:ResolveComponentCandidates",
                "greengrass:GetDeploymentConfiguration",
                "greengrass:ListThingGroupsForCoreDevice"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS IoTPolítica mínima de compatibilidad con los dispositivos cliente

El siguiente ejemplo de política incluye el conjunto mínimo de acciones necesarias para permitir la interacción con los dispositivos cliente en un dispositivo principal. Para ser compatible con los dispositivos cliente, un dispositivo principal debe tener los permisos de esta AWS IoT política además de la AWS IoTpolítica mínima para su funcionamiento básico.

- La política permite que el dispositivo principal actualice su propia información de conectividad. Este permiso (`greengrass:UpdateConnectivityInfo`) solo es necesario si se implementa el componente del detector de IP en el dispositivo principal.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:region:account-id:topic/$aws/things/core-device-thing-name-gci/shadow/get"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:region:account-id:topicfilter/$aws/things/core-device-thing-name-gci/shadow/update/delta",
                "arn:aws:iot:region:account-id:topicfilter/$aws/things/core-device-thing-name-gci/shadow/get/accepted"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:region:account-id:topic/$aws/things/core-device-thing-name-gci/shadow/update/delta",
                "arn:aws:iot:region:account-id:topic/$aws/things/core-device-thing-name-gci/shadow/get/accepted"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "greengrass:PutCertificateAuthorities",
                "greengrass:VerifyClientDeviceIdentity"
            ],
```

```
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "greengrass:VerifyClientDeviceIoTCertificateAssociation"
            ],
            "Resource": "arn:aws:iot:region:account-id:thing/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "greengrass:GetConnectivityInfo",
                "greengrass:UpdateConnectivityInfo"
            ],
            "Resource": [
                "arn:aws:iot:region:account-id:thing/core-device-thing-name"
            ]
        }
    ]
}
```

AWS IoTPolítica mínima para los dispositivos cliente

El siguiente ejemplo de política incluye el conjunto mínimo de acciones necesarias para que un dispositivo cliente detecte los dispositivos principales a los que se conecta y se comunica a través de MQTT. La AWS IoT política del dispositivo cliente debe incluir la `greengrass:Discover` acción que permita al dispositivo descubrir la información de conectividad de sus dispositivos principales Greengrass asociados. En la `Resource` sección, especifique el nombre de recurso de Amazon (ARN) del dispositivo cliente, no el ARN del dispositivo principal de Greengrass.

- La política permite la comunicación sobre todos los temas de MQTT. Para seguir las mejores prácticas de seguridad, restrinja los `iot:Publish` `iot:Receive` permisos y los permisos al conjunto mínimo de temas que un dispositivo cliente requiera para su caso de uso. `iot:Subscribe`
- La política permite que el dispositivo descubra los dispositivos principales para AWS IoT todo tipo de aplicaciones. Para seguir las mejores prácticas de seguridad, restrinja el `greengrass:Discover` permiso al dispositivo cliente o AWS IoT a un comodín que coincida con un conjunto de AWS IoT elementos.

> **Important**
>
> Las variables de política de cosas (`iot:Connection.Thing.*`) no se admiten en las AWS IoT políticas de dispositivos principales ni en las operaciones del plano de datos de Greengrass. En su lugar, puede utilizar un comodín que haga coincidir varios dispositivos con nombres similares. Por ejemplo, puede `MyGreengrassDevice*` especificar que coincida `MyGreengrassDevice1MyGreengrassDevice2`, etc.

- La AWS IoT política de un dispositivo cliente no suele requerir permisos ni `iot:DeleteThingShadow` acciones `iot:GetThingShadowiot:UpdateThingShadow`, ya que el dispositivo principal de Greengrass gestiona las operaciones de sincronización oculta para los dispositivos cliente. Para permitir que el dispositivo principal gestione las sombras de los dispositivos cliente, compruebe que la AWS IoT política del dispositivo principal permita estas acciones y que la `Resource` sección incluya los ARN de los dispositivos cliente.

```
{
    "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/*"
            ]
        }
    ]
}
```

Administración de identidades y accesos en AWS IoT Greengrass

AWS Identity and Access Management (IAM) es un Servicio de AWS que ayuda a los administradores a controlar de forma segura el acceso a los recursos de AWS. Los administradores de IAM controlan quién está autenticado (ha iniciado sesión) y autorizado (tiene permisos) para utilizar recursos de AWS IoT Greengrass. IAM es un servicio de Servicio de AWS que se puede utilizar sin cargo adicional.

> **Note**
>
> En este tema se describen conceptos y características de IAM. Para obtener información acerca de las características de AWS IoT Greengrass admitidas por IAM, consulte the section called “Cómo AWS IoT Greengrass funciona con IAM”.

Público

La forma en que utilice AWS Identity and Access Management (IAM) difiere en función del trabajo que realice en AWS IoT Greengrass.

Usuario de servicio: si utiliza el servicio de AWS IoT Greengrass para realizar su trabajo, su administrador le proporciona las credenciales y los permisos que necesita. A medida que utilice más características de AWS IoT Greengrass para realizar su trabajo, es posible que necesite permisos adicionales. Entender cómo se administra el acceso puede ayudarlo a solicitar los permisos correctos al administrador. Si no puede acceder a una característica en AWS IoT Greengrass, consulte Solución de problemas de administración de identidades y accesos en AWS IoT Greengrass.

Administrador de servicio: si está a cargo de los recursos de AWS IoT Greengrass en su empresa, probablemente tenga acceso completo a AWS IoT Greengrass. Su trabajo consiste en determinar a qué características y recursos de AWS IoT Greengrass deben acceder los usuarios del servicio. A continuación, debe enviar solicitudes a su administrador de IAM para cambiar los permisos de los usuarios de sus servicios. Revise la información de esta página para conocer los conceptos básicos de IAM. Para obtener más información sobre cómo su empresa puede utilizar IAM con AWS IoT Greengrass, consulte Cómo AWS IoT Greengrass funciona con IAM.

Administrador de IAM: si es un administrador de IAM, es posible que quiera conocer más detalles sobre cómo escribir políticas para administrar el acceso a AWS IoT Greengrass. Para consultar ejemplos de políticas basadas en identidad de AWS IoT Greengrass que puede utilizar en IAM, consulte Ejemplos de políticas basadas en identidades de AWS IoT Greengrass.

Autenticación con identidades

La autenticación es la manera de iniciar sesión en AWS mediante credenciales de identidad. Debe estar autenticado (haber iniciado sesión en AWS) como Usuario raíz de la cuenta de AWS, como un usuario de IAM o asumiendo un rol de IAM.

Puede iniciar sesión en AWS como una identidad federada mediante las credenciales proporcionadas a través de una fuente de identidad de AWS IAM Identity Center. Los usuarios (del IAM Identity Center), la autenticación de inicio de sesión único de su empresa y sus credenciales de Google o Facebook son ejemplos de identidades federadas. Al iniciar sesión como una identidad federada, su administrador habrá configurado previamente la federación de identidades mediante roles de IAM. Cuando accede a AWS mediante la federación, está asumiendo un rol de forma indirecta.

Según el tipo de usuario que sea, puede iniciar sesión en la AWS Management Console o en el portal de acceso a AWS. Para obtener más información sobre el inicio de sesión en AWS, consulte Cómo iniciar sesión en su Cuenta de AWS en la Guía del usuario de AWS Sign-In.

Si accede a AWS mediante programación, AWS proporciona un kit de desarrollo de software (SDK) y una interfaz de la línea de comandos (CLI) para firmar criptográficamente las solicitudes mediante el uso de las credenciales. Si no usa las herramientas de AWS, debe firmar usted mismo las solicitudes. Para obtener más información sobre el método recomendado para la firma de solicitudes, consulte Firma de solicitudes API de AWS en la Guía del usuario de IAM.

Independientemente del método de autenticación que use, es posible que deba proporcionar información de seguridad adicional. Por ejemplo, AWS le recomienda el uso de la autenticación multifactor (MFA) para aumentar la seguridad de su cuenta. Para obtener más información, consulte Autenticación multifactor en la Guía del usuario de AWS IAM Identity Center y Uso de la autenticación multifactor (MFA) en AWS en la Guía del usuario de IAM.

Usuario raíz de Cuenta de AWS

Cuando se crea una Cuenta de AWS, se comienza con una identidad de inicio de sesión que tiene acceso completo a todos los recursos y Servicios de AWS de la cuenta. Esta identidad recibe el nombre de usuario raíz de la Cuenta de AWS y se accede a ella iniciando sesión con el email y la contraseña que utilizó para crear la cuenta. Recomendamos encarecidamente que no utilice el usuario raíz para sus tareas diarias. Proteja las credenciales del usuario raíz y utilícelas solo para las tareas que solo el usuario raíz pueda realizar. Para ver la lista completa de las tareas que requieren que inicie sesión como usuario raíz, consulte Tareas que requieren credenciales de usuario raíz en la Guía del usuario de IAM.

Usuarios y grupos de IAM

Un usuario de IAM es una identidad en su Cuenta de AWS que dispone de permisos específicos para una sola persona o aplicación. Siempre que sea posible, recomendamos emplear credenciales temporales, en lugar de crear usuarios de IAM que tengan credenciales de larga duración como contraseñas y claves de acceso. No obstante, si tiene casos de uso específicos que requieran credenciales de larga duración con usuarios de IAM, recomendamos rotar las claves de acceso. Para más información, consulte Rotar las claves de acceso periódicamente para casos de uso que requieran credenciales de larga duración en la Guía del Usuario de IAM.

Un grupo de IAM es una identidad que especifica un conjunto de usuarios de IAM. No puede iniciar sesión como grupo. Puede usar los grupos para especificar permisos para varios usuarios a la

vez. Los grupos facilitan la administración de los permisos de grandes conjuntos de usuarios. Por ejemplo, podría tener un grupo cuyo nombre fuese IAMAdmins y conceder permisos a dicho grupo para administrar los recursos de IAM.

Los usuarios son diferentes de los roles. Un usuario se asocia exclusivamente a una persona o aplicación, pero la intención es que cualquier usuario pueda asumir un rol que necesite. Los usuarios tienen credenciales permanentes a largo plazo y los roles proporcionan credenciales temporales. Para más información, consulte Cuándo crear un usuario de IAM (en lugar de un rol) en la Guía del Usuario de IAM.

Roles de IAM

Un rol de IAM es una identidad de tu Cuenta de AWS que dispone de permisos específicos. Es similar a un usuario de IAM, pero no está asociado a una determinada persona. Puede asumir temporalmente un rol de IAM en la AWS Management Consolecambiando de roles. Puede asumir un rol llamando a una operación de AWS CLI o de la API de AWS, o utilizando una URL personalizada. Para más información sobre los métodos para el uso de roles, consulte Uso de roles de IAM en la Guía del Usuario de IAM.

Los roles de IAM con credenciales temporales son útiles en las siguientes situaciones:

- Acceso de usuario federado: para asignar permisos a una identidad federada, puede crear un rol y definir sus permisos. Cuando se autentica una identidad federada, se asocia la identidad al rol y se le conceden los permisos define el rol. Para obtener información acerca de roles para federación, consulte Creación de un rol para un proveedor de identidades de terceros en la Guía del Usuario de IAM. Si utiliza el IAM Identity Center, debe configurar un conjunto de permisos. El IAM Identity Center correlaciona el conjunto de permisos con un rol en IAM para controlar a qué pueden acceder las identidades después de autenticarse. Para obtener información acerca de los conjuntos de permisos, consulte Conjuntos de permisos en la Guía del usuario de AWS IAM Identity Center.
- Permisos de usuario de IAM temporales: un usuario de IAM puede asumir un rol de IAM para recibir temporalmente permisos distintos que le permitan realizar una tarea concreta.
- Acceso entre cuentas: puede utilizar un rol de IAM para permitir que alguien (una entidad principal de confianza) de otra cuenta acceda a los recursos de la cuenta. Los roles son la forma principal de conceder acceso entre cuentas. No obstante, con algunos Servicios de AWS se puede asociar una política directamente a un recurso (en lugar de utilizar un rol como representante). Para obtener información sobre la diferencia entre los roles y las políticas basadas en recursos para

el acceso entre cuentas, consulte Cómo los roles de IAM difieren de las políticas basadas en recursos en la Guía del usuario de IAM.

- Acceso entre servicios: algunos Servicios de AWS utilizan características de otros Servicios de AWS. Por ejemplo, cuando realiza una llamada en un servicio, es común que ese servicio ejecute aplicaciones en Amazon EC2 o almacene objetos en Amazon S3. Es posible que un servicio haga esto usando los permisos de la entidad principal, usando un rol de servicio o usando un rol vinculado a servicios.
  - Reenviar sesiones de acceso (FAS): cuando utiliza un rol o un usuario de IAM para llevar a cabo acciones en AWS, se le considera una entidad principal. Cuando utiliza algunos servicios, es posible que realice una acción que desencadene otra acción en un servicio diferente. FAS utiliza los permisos de la entidad principal para llamar a un Servicio de AWS, combinados con el Servicio de AWS solicitante para realizar solicitudes a servicios posteriores. Las solicitudes de FAS solo se realizan cuando un servicio recibe una solicitud que requiere interacciones con otros Servicios de AWS o recursos para completarse. En este caso, debe tener permisos para realizar ambas acciones. Para obtener información sobre las políticas a la hora de realizar solicitudes de FAS, consulte Reenviar sesiones de acceso.
  - Rol de servicio: un rol de servicio es un rol de IAM que adopta un servicio para realizar acciones en su nombre. Un administrador de IAM puede crear, modificar y eliminar un rol de servicio desde IAM. Para obtener más información, consulte Creación de un rol para delegar permisos a un Servicio de AWS en la Guía del usuario de IAM.
  - Rol vinculado a servicios: un rol vinculado a servicios es un tipo de rol de servicio que está vinculado a un Servicio de AWS. El servicio puede asumir el rol para realizar una acción en su nombre. Los roles vinculados a servicios aparecen en la Cuenta de AWS y son propiedad del servicio. Un administrador de IAM puede ver, pero no editar, los permisos de los roles vinculados a servicios.
- Aplicaciones que se ejecutan en Amazon EC2: puede utilizar un rol de IAM que le permita administrar credenciales temporales para las aplicaciones que se ejecutan en una instancia de EC2 y realizan solicitudes a la AWS CLI o a la API de AWS. Es preferible hacerlo de este modo a almacenar claves de acceso en la instancia EC2. Para asignar un rol de AWS a una instancia de EC2 y ponerla a disposición de todas las aplicaciones, cree un perfil de instancia asociado a la instancia. Un perfil de instancia contiene el rol y permite a los programas que se ejecutan en la instancia EC2 obtener credenciales temporales. Para obtener más información, consulte Uso de un rol de IAM para conceder permisos a aplicaciones que se ejecutan en instancias de Amazon EC2 en la Guía del usuario de IAM.

Para obtener información sobre el uso de los roles de IAM, consulte Cuándo crear un rol de IAM (en lugar de un usuario) en la Guía del Usuario de IAM.

Administración de acceso mediante políticas

Para controlar el acceso en AWS, se crean políticas y se adjuntan a identidades o recursos de AWS. Una política es un objeto de AWS que, cuando se asocia a una identidad o un recurso, define sus permisos. AWS evalúa estas políticas cuando una entidad principal (sesión de rol, usuario o usuario raíz) realiza una solicitud. Los permisos en las políticas determinan si la solicitud se permite o se deniega. La mayoría de las políticas se almacenan en AWS como documentos JSON. Para obtener más información sobre la estructura y el contenido de los documentos de política JSON, consulte Información general de las políticas JSON en la Guía del Usuario de IAM.

Los administradores pueden utilizar las políticas JSON de AWS para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

De forma predeterminada, los usuarios y los roles no tienen permisos. Para conceder permiso a los usuarios para realizar acciones en los recursos que necesiten, un administrador de IAM puede crear políticas de IAM. A continuación, el administrador puede añadir las políticas de IAM a roles y los usuarios pueden asumirlos.

Las políticas de IAM definen permisos para una acción independientemente del método que se utilice para realizar la operación. Por ejemplo, suponga que dispone de una política que permite la acción `iam:GetRole`. Un usuario con dicha política puede obtener información del rol de la AWS Management Console, la AWS CLI o la API de AWS.

Políticas basadas en identidades

Las políticas basadas en identidades son documentos de políticas de permisos JSON que puede adjuntar a una identidad, como un usuario, un grupo de usuarios o un rol de IAM. Estas políticas controlan qué acciones pueden realizar los usuarios y los roles, en qué recursos y en qué condiciones. Para obtener más información sobre cómo crear una política basada en identidad, consulte Creación de políticas de IAM en la Guía del usuario de IAM.

Las políticas basadas en identidades pueden clasificarse además como políticas insertadas o políticas administradas. Las políticas insertadas se integran directamente en un único usuario, grupo o rol. Las políticas administradas son políticas independientes que puede asociar a varios usuarios, grupos y roles de su Cuenta de AWS. Las políticas administradas incluyen las políticas administradas de AWS y las políticas administradas por el cliente. Para obtener más información sobre cómo elegir

una política administrada o una política insertada, consulte Elegir entre políticas administradas y políticas insertadas en la Guía del usuario de IAM.

Políticas basadas en recursos

Las políticas basadas en recursos son documentos de política JSON que se asocian a un recurso. Ejemplos de políticas basadas en recursos son las políticas de confianza de roles de IAM y las políticas de bucket de Amazon S3. En los servicios que admiten políticas basadas en recursos, los administradores de servicios pueden utilizarlos para controlar el acceso a un recurso específico. Para el recurso al que se asocia la política, la política define qué acciones puede realizar una entidad principal especificada en ese recurso y en qué condiciones. Debe especificar una entidad principal en una política en función de recursos. Las entidades principales pueden incluir cuentas, usuarios, roles, usuarios federados o Servicios de AWS.

Las políticas basadas en recursos son políticas insertadas que se encuentran en ese servicio. No se puede utilizar políticas de IAM administradas de AWS en una política basada en recursos.

Listas de control de acceso (ACL)

Las listas de control de acceso (ACL) controlan qué entidades principales (miembros de cuentas, usuarios o roles) tienen permisos para acceder a un recurso. Las ACL son similares a las políticas basadas en recursos, aunque no utilizan el formato de documento de políticas JSON.

Amazon S3, AWS WAF y Amazon VPC son ejemplos de servicios que admiten las ACL. Para obtener más información sobre las ACL, consulte Información general de Lista de control de acceso (ACL) en la Guía para Desarrolladores de Amazon Simple Storage Service.

Otros tipos de políticas

AWS admite otros tipos de políticas adicionales menos frecuentes. Estos tipos de políticas pueden establecer el máximo de permisos que los tipos de políticas más frecuentes le conceden.

- Límites de permisos: un límite de permisos es una característica avanzada que le permite establecer los permisos máximos que una política basada en identidad puede conceder a una entidad de IAM (usuario o rol de IAM). Puede establecer un límite de permisos para una entidad. Los permisos resultantes son la intersección de las políticas basadas en la identidad de la entidad y los límites de permisos. Las políticas basadas en recursos que especifiquen el usuario o rol en el campo `Principal` no estarán restringidas por el límite de permisos. Una denegación explícita en cualquiera de estas políticas anulará el permiso. Para obtener más información sobre los límites

de los permisos, consulte Límites de permisos para las entidades de IAM en la Guía del Usuario de IAM.

- Políticas de control de servicio (SCP): las SCP son políticas de JSON que especifican los permisos máximos de una organización o una unidad organizativa en AWS Organizations. AWS Organizations es un servicio que le permite agrupar y administrar de manera centralizada varias Cuentas de AWS que posea su empresa. Si habilita todas las características en una empresa, entonces podrá aplicar políticas de control de servicio (SCP) a una o todas sus cuentas. Una SCP limita los permisos para las entidades de las cuentas de miembros, incluido cada Usuario raíz de la cuenta de AWS. Para obtener más información acerca de Organizations y las SCP, consulte Funcionamiento de las SCP en la Guía del usuario de AWS Organizations.
- Políticas de sesión: las políticas de sesión son políticas avanzadas que se pasan como parámetro cuando se crea una sesión temporal mediante programación para un rol o un usuario federado. Los permisos de la sesión resultantes son la intersección de las políticas basadas en identidades del rol y las políticas de la sesión. Los permisos también pueden proceder de una política en función de recursos. Una denegación explícita en cualquiera de estas políticas anulará el permiso. Para más información, consulte Políticas de sesión en la Guía del Usuario de IAM.

Varios tipos de políticas

Cuando se aplican varios tipos de políticas a una solicitud, los permisos resultantes son más complicados de entender. Para obtener información sobre cómo AWS decide si permite o no una solicitud cuando hay varios tipos de políticas implicados, consulte Lógica de evaluación de políticas en la Guía del usuario de IAM.

Véase también

- the section called “Cómo AWS IoT Greengrass funciona con IAM”
- the section called “Ejemplos de políticas basadas en identidad”
- the section called “Solución de problemas de identidades y accesos”

Cómo AWS IoT Greengrass funciona con IAM

Antes de utilizar IAM para administrar el acceso aAWS IoT Greengrass, debe comprender las características de IAM con las que puede utilizarAWS IoT Greengrass.

Características de IAM	¿Compatible con Greengrass?
Políticas basadas en identidad con permisos de nivel de recursos	Sí
Políticas basadas en recursos	No
Listas de control de acceso (ACL)	No
Autorización basada en etiquetas	Sí
Credenciales temporales	Sí
Roles vinculados a servicios	No
Roles de servicio	Sí

Para obtener una perspectiva general de cómo funcionan otrosAWS servicios con IAM, consulte AWSServicios de que funcionan con IAM en la Guía del usuario de IAM.

Políticas de AWS IoT Greengrass basadas en identidades

Con las políticas basadas en identidades de IAM, puede especificar las acciones y los recursos permitidos o denegados y las condiciones en las que se permiten o deniegan las acciones. AWS IoT Greengrass admite acciones, claves de condiciones y recursos específicos. Para obtener más información acerca de los elementos que utiliza en una política, consulte Referencia de los elementos de las políticas de JSON de IAM en la Guía del usuario de IAM.

Acciones

Los administradores pueden utilizar las políticas JSON de AWS para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y bajo qué condiciones.

El elemento `Action` de una política JSON describe las acciones que puede utilizar para permitir o denegar el acceso en una política. Las acciones de la política generalmente tienen el mismo nombre que la operación de API de AWS asociada. Hay algunas excepciones, como acciones de solo permiso que no tienen una operación de API coincidente. También hay algunas operaciones que requieren varias acciones en una política. Estas acciones adicionales se denominan acciones dependientes.

Incluya acciones en una política para conceder permisos y así llevar a cabo la operación asociada.

Las acciones de políticas en AWS IoT Greengrass utilizan el prefijo `greengrass:` antes de la acción. Por ejemplo, para permitir que alguien utilice la operación de la`ListCoreDevices` API para incluir los dispositivos principales de suCuenta de AWS, incluya la`greengrass:ListCoreDevices` acción en su política. Las instrucciones de política deben incluir un elemento `Action` o `NotAction`. AWS IoT Greengrass define su propio conjunto de acciones que describen las tareas que se pueden realizar con este servicio.

Para especificar varias acciones en una única instrucción, sepárelas entre corchetes (`[]`) y sepárelas con comas del siguiente modo:

```
"Action": [
  "greengrass:action1",
  "greengrass:action2",
  "greengrass:action3"
]
```

Puede utilizar comodines (`*`) para especificar varias acciones. Por ejemplo, para especificar todas las acciones que comiencen con la palabra `List`, incluya la siguiente acción:

```
"Action": "greengrass:List*"
```

> **Note**
>
> Se recomienda evitar el uso de comodines para especificar todas las acciones disponibles para un servicio. Como práctica recomendada, debe conceder permisos de mínimo privilegio y acotar el alcance de los permisos en una política. Para obtener más información, consulte the section called “Conceda los mínimos permisos posibles”.

Para obtener la lista completa deAWS IoT Greengrass acciones, consulte Acciones definidas porAWS IoT Greengrass en la Guía del usuario de IAM.

Recursos

Los administradores pueden utilizar las políticas JSON de AWS para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y bajo qué condiciones.

El elemento `Resource` de la política JSON especifica el objeto u objetos a los que se aplica la acción. Las instrucciones deben contener un elemento `Resource` o `NotResource`. Como práctica recomendada, especifique un recurso utilizando el Nombre de recurso de Amazon (ARN). Puede hacerlo para acciones que admitan un tipo de recurso específico, conocido como permisos de nivel de recurso.

Para las acciones que no admiten permisos de nivel de recurso, como las operaciones de descripción, utilice un carácter comodín (*) para indicar que la instrucción se aplica a todos los recursos.

```
"Resource": "*"
```

La tabla siguiente contiene los ARN de recursos de AWS IoT Greengrass que se pueden utilizar en el elemento `Resource` de una declaración de política. Para ver un mapeo de los permisos de nivel de recursos admitidos paraAWS IoT Greengrass las acciones, consulte Acciones definidas porAWS IoT Greengrass en la Guía del usuario de IAM.

Algunas acciones de AWS IoT Greengrass (por ejemplo, algunas operaciones de lista) no se pueden realizar en un recurso específico. En dichos casos, debe utilizar solo el carácter comodín.

```
"Resource": "*"
```

Para especificar varios ARN de recursos en una sentencia, póngalos entre corchetes (`[]`) y sepárelos con comas, de la siguiente manera:

```
"Resource": [
  "resource-arn1",
  "resource-arn2",
  "resource-arn3"
]
```

Para obtener más información sobre los formatos de ARN, consulte este artículo sobre los nombres de recursos de Amazon (ARN) y los espacios de nombres de servicio de AWS en la Referencia general de Amazon Web Services.

Claves de condición

Los administradores pueden utilizar las políticas JSON de AWS para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y bajo qué condiciones.

El elemento `Condition` (o bloque de `Condition`) permite especificar condiciones en las que entra en vigor una instrucción. El elemento `Condition` es opcional. Puede crear expresiones condicionales que utilicen operadores de condición, tales como igual o menor que, para que la condición de la política coincida con los valores de la solicitud.

Si especifica varios elementos de `Condition` en una instrucción o varias claves en un único elemento de `Condition`, AWS las evalúa mediante una operación `AND` lógica. Si especifica varios valores para una única clave de condición, AWS evalúa la condición con una operación lógica `OR`. Se deben cumplir todas las condiciones antes de que se concedan los permisos de la instrucción.

También puede utilizar variables de marcador de posición al especificar condiciones. Por ejemplo, puede conceder un permiso de usuario de IAM para acceder a un recurso solo si está etiquetado con su nombre de usuario de IAM. Para obtener más información, consulte Elementos de la política de IAM: variables y etiquetas en la Guía del usuario de IAM.

AWS admite claves de condición globales y claves de condición específicas del servicio. Para ver todas las claves de condición globales de AWS, consulte Claves de contexto de condición globales de AWS en la Guía del usuario de IAM.

Ejemplos

Para ver ejemplos de políticas basadas en identidad de AWS IoT Greengrass, consulte the section called “Ejemplos de políticas basadas en identidad”.

Políticas de AWS IoT Greengrass basadas en recursos

AWS IoT Greengrass no admite políticas basadas en recursos.

Listas de control de acceso (ACL)

AWS IoT Greengrass no es compatible con los ACL.

Autorización basada en etiquetas de AWS IoT Greengrass

Puede asociar etiquetas a los recursos de AWS IoT Greengrass admitidos o transferirlas en una solicitud a AWS IoT Greengrass. Para controlar el acceso utilizando etiquetas, debe proporcionar información de las etiquetas en el elemento de condición de una política utilizando las claves de condición `aws:ResourceTag/${TagKey}`, `aws:RequestTag/${TagKey}` o `aws:TagKeys`. Para obtener más información, consulte Etiquetar los recursos.

Funciones de IAM paraAWS IoT Greengrass

Un rol de IAM es una entidad de la Cuenta de AWS que dispone de permisos específicos.

Uso de credenciales temporales con AWS IoT Greengrass

Las credenciales temporales se utilizan para iniciar sesión con identidad federada, asumir un rol de IAM o asumir un rol de acceso entre cuentas. Las credenciales de seguridad temporales se obtienen mediante una llamada a operacionesAWS STS de la API de, como AssumeRoleo GetFederationToken.

En el núcleo de Greengrass, las credenciales temporales para la función de dispositivo están disponibles para los componentes de Greengrass. Si sus componentes utilizan elAWS SDK, no necesita añadir lógica para obtener las credenciales, ya que elAWS SDK lo hace por usted.

Roles vinculados a servicios

AWS IoT Greengrass no admite roles vinculados a servicios.

Roles de servicio

Esta característica permite que un servicio asuma un rol de servicio en su nombre. Este rol permite que el servicio obtenga acceso a los recursos de otros servicios para completar una acción en su nombre. Los roles de servicio aparecen en su cuenta de IAM y son propiedad de la cuenta. Esto significa que un administrador de IAM puede cambiar los permisos de este rol. Sin embargo, hacerlo podría deteriorar la funcionalidad del servicio.

AWS IoT Greengrasslos dispositivos principales utilizan una función de servicio para permitir que los componentes de Greengrass y las funciones de Lambda accedan a algunos de susAWS recursos en su nombre. Para obtener más información, consulte the section called “Autorizar a los dispositivos principales a interactuar con AWS los servicios”.

AWS IoT Greengrass utiliza un rol de servicio para acceder a algunos de los recursos de AWS en su nombre. Para obtener más información, consulte Rol de servicio de Greengrass.

Ejemplos de políticas basadas en identidades de AWS IoT Greengrass

De forma predeterminada, los usuarios y los roles de IAM no tienen permiso para crear, ver ni modificar recursos de AWS IoT Greengrass. Tampoco pueden realizar tareas mediante la AWS Management Console, la AWS CLI, o la API de AWS. Un administrador de IAM debe crear políticas

de IAM que concedan permisos a los usuarios y a los roles para realizar operaciones de la API concretas en los recursos especificados que necesiten. El administrador debe adjuntar esas políticas a los usuarios o grupos de IAM que necesiten esos permisos.

Prácticas recomendadas relativas a políticas

Las políticas basadas en identidades determinan si alguien puede crear, acceder o eliminar los recursos de AWS IoT Greengrass de la cuenta. Estas acciones pueden generar costes adicionales para su Cuenta de AWS. Siga estas directrices y recomendaciones al crear o editar políticas basadas en identidad:

- Comience con las políticas administradas por AWS y continúe con los permisos de privilegio mínimo: a fin de comenzar a conceder permisos a los usuarios y las cargas de trabajo, utilice las políticas administradas por AWS, que conceden permisos para muchos casos de uso comunes. Están disponibles en la Cuenta de AWS. Se recomienda definir políticas administradas por el cliente de AWS específicas para sus casos de uso a fin de reducir aún más los permisos. Con el fin de obtener más información, consulte las políticas administradas por AWS o las políticas administradas por AWS para funciones de trabajo en la Guía de usuario de IAM.
- Aplique permisos de privilegio mínimo: cuando establezca permisos con políticas de IAM, conceda solo los permisos necesarios para realizar una tarea. Para ello, debe definir las acciones que se pueden llevar a cabo en determinados recursos en condiciones específicas, también conocidos como permisos de privilegios mínimos. Con el fin de obtener más información sobre el uso de IAM para aplicar permisos, consulte Políticas y permisos en IAM en la Guía de usuario de IAM.
- Use condiciones en las políticas de IAM para restringir aún más el acceso: puede agregar una condición a sus políticas para limitar el acceso a las acciones y los recursos. Por ejemplo, puede escribir una condición de política para especificar que todas las solicitudes deben enviarse utilizando SSL. También puede usar condiciones para conceder acceso a acciones de servicios si se emplean a través de un Servicio de AWS determinado, como por ejemplo AWS CloudFormation. Para obtener más información, consulte Elementos de la política JSON de IAM: condición en la Guía del usuario de IAM.
- Use el Analizador de acceso de IAM para validar las políticas de IAM con el fin de garantizar la seguridad y funcionalidad de los permisos: el Analizador de acceso de IAM valida políticas nuevas y existentes para que respeten el lenguaje (JSON) de las políticas de IAM y las prácticas recomendadas de IAM. IAM Access Analyzer proporciona más de 100 verificaciones de políticas y recomendaciones procesables para ayudar a crear políticas seguras y funcionales. Para obtener más información, consulte la política de validación del Analizador de acceso de IAM en la Guía de usuario de IAM.

- Solicite la autenticación multifactor (MFA): si se encuentra en una situación en la que necesita usuarios raíz o de IAM en su Cuenta de AWS, active la MFA para mayor seguridad. Para solicitar la MFA cuando se invocan las operaciones de la API, agregue las condiciones de MFA a sus políticas. Para obtener más información, consulte Configuración de acceso a una API protegida por MFA en la Guía de usuario de IAM.

Para obtener más información sobre las prácticas recomendadas de IAM, consulte las Prácticas recomendadas de seguridad en IAM en la Guía de usuario de IAM.

Ejemplos de políticas

En el ejemplo siguiente, las políticas definidas por el cliente conceden permisos para situaciones comunes.

Ejemplos

- Permitir a los usuarios consultar sus propios permisos

Para obtener más información acerca de cómo crear una política basada en identidad de IAM con estos documentos de políticas de JSON de ejemplo, consulte Creación de políticas en la pestaña JSON en la Guía del usuario de IAM.

Permitir a los usuarios consultar sus propios permisos

En este ejemplo, se muestra cómo podría crear una política que permita a los usuarios de IAM ver las políticas administradas e insertadas que se adjuntan a la identidad de sus usuarios. Esta política incluye permisos para llevar a cabo esta acción en la consola o mediante programación con la AWS CLI o la API de 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": "*"
        }
    ]
}
```

Autorizar a los dispositivos principales a interactuar con AWS los servicios

AWS IoT Greengrass los dispositivos principales utilizan el proveedor de AWS IoT Core credenciales para autorizar las llamadas a AWS los servicios. El proveedor de AWS IoT Core credenciales permite a los dispositivos utilizar sus certificados X.509 como identidad única del dispositivo para autenticar las solicitudes. AWS Esto elimina la necesidad de almacenar un identificador de clave de AWS acceso y una clave de acceso secreta en los dispositivos AWS IoT Greengrass principales. Para obtener más información, consulte Autorizar llamadas directas a AWS los servicios en la Guía para AWS IoT Core desarrolladores.

Al ejecutar el software AWS IoT Greengrass principal, puede optar por aprovisionar los AWS recursos que requiere el dispositivo principal. Esto incluye la función AWS Identity and Access Management (IAM) que asume su dispositivo principal a través del proveedor de AWS IoT Core credenciales. Utilice el `--provision true` argumento para configurar una función y políticas que permitan al dispositivo principal obtener AWS credenciales temporales. Este argumento también configura un alias de AWS IoT rol que apunta a este rol de IAM. Puede especificar el nombre del rol de IAM y el alias del rol que se van a AWS IoT utilizar. Si lo especifica `--provision true` sin estos otros parámetros de nombre, el dispositivo principal de Greengrass crea y utiliza los siguientes recursos predeterminados:

- Función de IAM: `GreengrassV2TokenExchangeRole`

  Este rol tiene un nombre de política `GreengrassV2TokenExchangeRoleAccess` y una relación de confianza que `credentials.iot.amazonaws.com` permite asumir el rol. La política incluye los permisos mínimos para el dispositivo principal.

  > **Important**
  >
  > Esta política no incluye el acceso a los archivos de los depósitos de S3. Debe añadir permisos a la función para permitir que los dispositivos principales recuperen los artefactos de los componentes de los depósitos de S3. Para obtener más información, consulte Permita el acceso a los depósitos de S3 para los artefactos de los componentes.

- AWS IoT alias del rol: `GreengrassV2TokenExchangeRoleAlias`

  Este alias de rol hace referencia al rol de IAM.

Para obtener más información, consulte Paso 3: instale el softwareAWS IoT Greengrass principal.

También puede establecer el alias de rol para un dispositivo principal existente. Para ello, configure el parámetro de `iotRoleAlias` configuración del componente núcleo de Greengrass.

Puede adquirir AWS credenciales temporales para esta función de IAM a fin de realizar AWS operaciones en sus componentes personalizados. Para obtener más información, consulte Interactúa con AWS los servicios.

Temas

- Permisos de rol de servicio para los dispositivos principales
- Permita el acceso a los depósitos de S3 para los artefactos de los componentes

Permisos de rol de servicio para los dispositivos principales

El rol permite que el siguiente servicio asuma el rol:

- `credentials.iot.amazonaws.com`

Si utilizas el software AWS IoT Greengrass principal para crear este rol, este utilizará la siguiente política de permisos para permitir que los dispositivos principales se conecten y envíen registros

a ellos AWS. De forma predeterminada, el nombre de la política es el nombre del rol de IAM que termina en. `Access` Por ejemplo, si usa el nombre de rol de IAM predeterminado, el nombre de esta política es. `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": "*"
    }
  ]
}
```

Permita el acceso a los depósitos de S3 para los artefactos de los componentes

La función de dispositivo principal predeterminada no permite que los dispositivos principales accedan a los depósitos de S3. Para implementar componentes que tienen artefactos en los depósitos de S3, debe añadir el `s3:GetObject` permiso que permita a los dispositivos principales descargar artefactos de los componentes. Puede añadir una nueva política a la función de dispositivo principal para conceder este permiso.

Para añadir una política que permita el acceso a los artefactos de los componentes en Amazon S3

1. Cree un archivo llamado `component-artifact-policy.json` y copie el siguiente JSON en el archivo. Esta política permite el acceso a todos los archivos de un bucket de S3. Sustituya DOC-EXAMPLE-BUCKET por el nombre del depósito de S3 para permitir el acceso del dispositivo principal.

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

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

2. Ejecute el siguiente comando para crear la política a partir del documento de política incluido en. `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
   ```

   Copie la política Amazon Resource Name (ARN) de los metadatos de la política en la salida. Utilice este ARN para adjuntar esta política a la función de dispositivo principal en el siguiente paso.

3. Ejecute el siguiente comando para asociar la política a la función de dispositivo principal. Sustituya *GreengrassV2 TokenExchangeRole* por el nombre de la función que especificó al ejecutar el software AWS IoT Greengrass Core. A continuación, sustituya el ARN de la política por el ARN del paso anterior.

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
```

Si el comando no tiene ningún resultado, se ha realizado correctamente y su dispositivo principal puede acceder a los artefactos que cargue en este bucket de S3.

Política de IAM mínima para que el instalador aprovisione recursos

Al instalar el software AWS IoT Greengrass Core, puede aprovisionar AWS los recursos necesarios, como una AWS IoT cosa y una función de IAM para su dispositivo. También puede implementar herramientas de desarrollo local en el dispositivo. El instalador necesita AWS credenciales para poder realizar estas acciones en su Cuenta de AWS. Para obtener más información, consulte Instalación del software AWS IoT Greengrass Core.

El siguiente ejemplo de política incluye el conjunto mínimo de acciones que el instalador necesita para aprovisionar estos recursos. Estos permisos son necesarios si se especifica el `--provision` argumento del instalador. Sustituya *`account-id`* por su Cuenta de AWS ID y sustituya *`GreengrassV2 TokenExchangeRole`* por el nombre de la función de intercambio de fichas que especifique con el argumento del instalador. `--tes-role-name`

> **Note**
>
> La declaración `DeployDevTools` de política solo es necesaria si se especifica el `--deploy-dev-tools` argumento del instalador.

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": "*"
        }
    ]
}
```

Rol de servicio de Greengrass

El rol de servicio de Greengrass es un rol de servicio de AWS Identity and Access Management (IAM) que autoriza a AWS IoT Greengrass a acceder a recursos de servicios de AWS en su nombre. Esta función permite verificar la identidad de AWS IoT Greengrass los dispositivos cliente y administrar la información de conectividad de los dispositivos principales.

> **Note**
>
> AWS IoT Greengrass V1también utiliza esta función para realizar tareas esenciales. Para obtener más información, consulte la función de servicio de Greengrass en la Guía AWS IoT Greengrass V1para desarrolladores.

Para permitir a AWS IoT Greengrass acceder a sus recursos, el rol de servicio de Greengrass debe estar asociado a su Cuenta de AWS y especificar AWS IoT Greengrass como entidad de confianza. El rol debe incluir la política AWSGreengrassResourceAccessRolePolicyadministrada o una política personalizada que defina permisos equivalentes para las AWS IoT Greengrass funciones que utilice. AWSmantiene esta política, que define el conjunto de permisos que se AWS IoT Greengrass utilizan para acceder a AWS los recursos. Para obtener más información, consulte Política administrada por AWS: AWSGreengrassResourceAccessRolePolicy.

Puedes reutilizar la misma función de servicio de Greengrass en todas partesRegiones de AWS, pero debes asociarla a tu cuenta en todos los Región de AWS lugares donde la utilices. AWS IoT Greengrass Si la función de servicio no está configurada en la versión actualRegión de AWS, los dispositivos principales no pueden verificar los dispositivos cliente ni actualizar la información de conectividad.

En las siguientes secciones se describe cómo crear y administrar el rol de servicio de Greengrass con o. AWS Management Console AWS CLI

Temas

> **Note**
>
> Además de la función de servicio que autoriza el acceso a nivel de servicio, se asigna una función de intercambio de fichas a los dispositivos principales de Greengrass. La función de intercambio de fichas es una función de IAM independiente que controla la forma en que los componentes de Greengrass y las funciones de Lambda del dispositivo principal pueden acceder a los servicios. AWS Para obtener más información, consulte Autorizar a los dispositivos principales a interactuar con AWS los servicios.

Administrar el rol de servicio de Greengrass (consola)

La consola de AWS IoT facilita la administración del rol de servicio de Greengrass. Por ejemplo, cuando configura la detección de dispositivos cliente para un dispositivo principal, la consola comprueba si Cuenta de AWS está vinculado a un rol de servicio de Greengrass en el momento actual. Región de AWS De lo contrario, la consola puede crear y configurar un rol de servicio por usted. Para obtener más información, consulte the section called “Creación del rol de servicio de Greengrass”.

Puede utilizar la consola para las siguientes tareas de administración de roles:

Temas

- Buscar el rol de servicio de Greengrass (consola)
- Creación del rol de servicio de Greengrass (consola)
- Cambiar el rol de servicio de Greengrass (consola)
- Desasociar el rol de servicio de Greengrass (consola)

> **Note**
>
> El usuario que ha iniciado sesión en la consola debe tener permisos para ver, crear o cambiar el rol de servicio.

Buscar el rol de servicio de Greengrass (consola)

Siga los siguientes pasos para encontrar el rol de servicio que AWS IoT Greengrass utiliza en el actualRegión de AWS.

1. Vaya a la consola de AWS IoT.
2. En el panel de navegación, seleccione Configuración.
3. Desplácese hasta la sección Greengrass service role (Rol de servicio de Greengrass) para ver el rol de servicio y sus políticas.

   Si no ve ningún rol de servicio, la consola puede crear o configurar uno automáticamente. Para obtener más información, consulte Creación del rol de servicio de Greengrass.

Creación del rol de servicio de Greengrass (consola)

La consola puede crear y configurar un rol de servicio de Greengrass predeterminado por usted. Este rol incluye las siguientes propiedades.

Propiedad	Valor
Nombre	`Greengrass_ServiceRole`
Entidad de confianza	`AWS service: greengrass`
Política	AWSGreengrassResourceAccessRolePolicy

> **Note**
>
> Si crea este rol con el script de configuración del AWS IoT Greengrass V1 dispositivo, el nombre del rol es`GreengrassServiceRole_`*`random-string`*.

Al configurar la detección de dispositivos cliente para un dispositivo principal, la consola comprueba si una función de servicio de Greengrass está asociada a la suya Cuenta de AWS en la versión actual. Región de AWS Si no lo hay, la consola le pedirá que permita a AWS IoT Greengrass leer y escribir en los servicios de AWS en su nombre.

Si concede permiso, la consola comprueba si existe un rol denominado `Greengrass_ServiceRole` en su Cuenta de AWS.

- Si el rol existe, la consola asocia el rol de servicio a su Cuenta de AWS en la Región de AWS actual.

- Si el rol no existe, la consola crea un rol de servicio de Greengrass predeterminado y lo asocia a su Cuenta de AWS en la Región de AWS actual.

> **Note**
>
> Si desea crear un rol de servicio con políticas de rol personalizadas, utilice la consola de IAM para crear o modificar el rol. Para obtener más información, consulte Creación de un rol para delegar permisos a un servicio AWS o Modificación de un rol en la Guía del usuario de IAM. Asegúrese de que el rol concede permisos equivalentes a la política administrada de `AWSGreengrassResourceAccessRolePolicy` para las características y recursos que utiliza. Le recomendamos que incluya también las claves de contexto de condición global `aws:SourceArn` y `aws:SourceAccount` en su política de confianza para ayudar a prevenir el problema de seguridad del suplente confuso. Las claves de contexto de condición restringen el acceso para permitir solo las solicitudes que provienen de la cuenta especificada y del espacio de trabajo de Greengrass. Para obtener más información sobre el problema del suplente confuso, consulte Prevención del suplente confuso entre servicios. Si crea un rol de servicio, regrese a la AWS IoT consola y asocie el rol al suyoCuenta de AWS. Puede hacerlo en la función de servicio de Greengrass en la página de configuración.

Cambiar el rol de servicio de Greengrass (consola)

Utilice el siguiente procedimiento para seleccionar un rol de servicio de Greengrass diferente y asociarlo a su Cuenta de AWS en la Región de AWS seleccionada actualmente en la consola.

1. Vaya a la consola de AWS IoT.
2. En el panel de navegación, seleccione Configuración.
3. En Rol de servicio de Greengrass, seleccione Elegir un rol diferente.

   Se abre el cuadro de diálogo Actualizar el rol de servicio de Greengrass y muestra los roles de IAM Cuenta de AWS que se definen AWS IoT Greengrass como una entidad de confianza.
4. Elija el rol de servicio de Greengrass que desee asignar.
5. Elija Adjuntar rol.

Desasociar el rol de servicio de Greengrass (consola)

Utilice el siguiente procedimiento para separar el rol de servicio de Greengrass de AWS su cuenta actual. Región de AWS Este revoca los permisos de AWS IoT Greengrass para acceder a los servicios de AWS en la Región de AWS actual.

> **Important**
>
> La desasociación del rol de servicio podría interrumpir las operaciones activas.

1. Vaya a la consola de AWS IoT.
2. En el panel de navegación, seleccione Configuración.
3. En Rol de servicio de Greengras, seleccione Desasociar rol.
4. En el cuadro de diálogo de confirmación, elija Desconectar.

> **Note**
>
> Si ya no necesita el rol, puede eliminarlo en la consola de IAM. Para obtener más información, consulte Eliminación de roles o perfiles de instancia en la Guía del usuario de IAM.
> Otros roles podrían permitir que AWS IoT Greengrass obtenga acceso a los recursos. Para buscar todos los roles que permiten que AWS IoT Greengrass asuma los permisos en su nombre, en la consola de IAM, en la página Roles, busque los roles que incluyan AWS service: greengrass en la columna Entidades de confianza.

Gestione el rol de servicio de Greengrass (CLI)

En los siguientes procedimientos, asumimos que AWS Command Line Interface está instalado y configurado para usar suCuenta de AWS. Para obtener más información, consulte Instalación, actualización y desinstalación AWS CLI y configuración del AWS CLI en la Guía del AWS Command Line Interface usuario.

Puede utilizar la AWS CLI para las siguientes tareas de administración de roles:

Temas

Obtener el rol de servicio de Greengrass (CLI)

Utilice el procedimiento siguiente para descubrir si un rol de servicio de Greengrass está asociado a su Cuenta de AWS en una Región de AWS.

- Obtenga el rol de servicio. Sustituya *región* por su Región de AWS (por ejemplo, `us-west-2`).

  ```
  aws greengrassv2 get-service-role-for-account --region region
  ```

  Si un rol de servicio de Greengrass ya está asociado a su cuenta, la solicitud devuelve los siguientes metadatos del rol.

  ```
  {
    "associatedAt": "timestamp",
    "roleArn": "arn:aws:iam::account-id:role/path/role-name"
  }
  ```

  Si la solicitud no devuelve los metadatos del rol, debe crear el rol de servicio (si no existe) y asociarlo a su cuenta en. Región de AWS

Creación del rol de servicio de Greengrass (CLI)

Siga los pasos que se indican a continuación para crear un rol y asociarlo a su Cuenta de AWS.

Para crear el rol de servicio mediante IAM

1. Cree el rol con una política de confianza que permita a AWS IoT Greengrass asumir el rol. Este ejemplo crea un rol denominado `Greengrass_ServiceRole`, pero puede utilizar un nombre distinto. Le recomendamos que incluya también las claves de contexto de condición global `aws:SourceArn` y `aws:SourceAccount` en su política de confianza para ayudar a prevenir el problema de seguridad del suplente confuso. Las claves de contexto de condición restringen el acceso para permitir solo las solicitudes que provienen de la cuenta especificada y del espacio de trabajo de Greengrass. Para obtener más información sobre el problema del suplente confuso, consulte Prevención del suplente confuso entre servicios.

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. Copie el ARN del rol de los metadatos del rol en la salida. Puede utilizar el ARN para asociar el rol a su cuenta.
3. Asocie la política de `AWSGreengrassResourceAccessRolePolicy` al rol.

```
aws iam attach-role-policy --role-name Greengrass_ServiceRole --policy-arn
 arn:aws:iam::aws:policy/service-role/AWSGreengrassResourceAccessRolePolicy
```

Para asociar el rol de servicio con su Cuenta de AWS

- Asocie el rol a su cuenta. Reemplace *`arn-rol`* por el ARN del rol de servicio y *`región`* por su Región de AWS (por ejemplo, `us-west-2`).

```
aws greengrassv2 associate-service-role-to-account --role-arn role-arn --
region region
```

Si se ejecuta correctamente, la solicitud devuelve la siguiente respuesta.

```
{
  "associatedAt": "timestamp"
}
```

Eliminar el rol de servicio de Greengrass (CLI)

Utilice los pasos siguientes para desasociar el rol de servicio de Greengrass de su Cuenta de AWS.

- Desasocie el rol de servicio de su cuenta. Sustituya *región* por su Región de AWS (por ejemplo, `us-west-2`).

  ```
  aws greengrassv2 disassociate-service-role-from-account --region region
  ```

  Si se ejecuta correctamente, se devuelve la siguiente respuesta.

  ```
  {
    "disassociatedAt": "timestamp"
  }
  ```

  > **Note**
  >
  > Debe eliminar el rol de servicio si no lo está utilizando en ninguna Región de AWS. Use primero delete-role-policy para desasociar la política administrada `AWSGreengrassResourceAccessRolePolicy` del rol y, a continuación, utilice delete-role para eliminar el rol. Para obtener más información, consulte Eliminación de roles o perfiles de instancia en la Guía del usuario de IAM.

Véase también

- Creación de un rol para delegar permisos a un servicio AWSen la Guía del usuario de IAM
- Modificación de un rol en la Guía del usuario de IAM
- Eliminación de roles o perfiles de instancia en la Guía del usuario de IAM
- comandos de AWS IoT Greengrass en la Referencia de los comandos de AWS CLI
  - associate-service-role-to-cuenta
  - disassociate-service-role-from-cuenta
  - get-service-role-for-cuenta
- Comandos de IAM en la Referencia de los comandos de AWS CLI
  - attach-role-policy
  - create-role

- delete-role
- delete-role-policy

Políticas administradas de AWS para AWS IoT Greengrass

Una política administrada de AWS es una política independiente que AWS crea y administra. Las políticas administradas de AWS se diseñan para ofrecer permisos para muchos casos de uso comunes, por lo que puede empezar a asignar permisos a los usuarios, grupos y roles.

Tenga presente que es posible que las políticas administradas de AWS no concedan permisos de privilegio mínimo para los casos de uso concretos, ya que están disponibles para que las utilicen todos los clientes de AWS. Se recomienda definir políticas administradas por el cliente para los casos de uso a fin de reducir aún más los permisos.

No puede cambiar los permisos definidos en las políticas administradas por AWS. Si AWS actualiza los permisos definidos en un política administrada de AWS, la actualización afecta a todas las identidades de entidades principales (usuarios, grupos y roles) a las que está adjunta la política. Lo más probable es que AWS actualice una política administrada de AWS cuando se lance un nuevo Servicio de AWS o las operaciones de la API nuevas estén disponibles para los servicios existentes.

Para obtener más información, consulte Políticas administradas de AWS en la Guía del usuario de IAM.

Temas

- Política administrada por AWS: AWSGreengrassFullAccess
- Política administrada por AWS: AWSGreengrassReadOnlyAccess
- Política administrada por AWS: AWSGreengrassResourceAccessRolePolicy
- Actualizaciones de AWS IoT Greengrass en las políticas administradas de AWS

Política administrada por AWS: AWSGreengrassFullAccess

Puede adjuntar la política `AWSGreengrassFullAccess` a las identidades de IAM.

Esta política otorga permisos administrativos que permiten al director acceder plenamente a todosAWS IoT Greengrassacciones.

Detalles sobre los permisos

Esta política incluye los siguientes permisos:

- `greengrass`— Permite a los directores el acceso total a todosAWS IoT Greengrassacciones.

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

Política administrada por AWS: AWSGreengrassReadOnlyAccess

Puede adjuntar la política `AWSGreengrassReadOnlyAccess` a las identidades de IAM.

Esta política otorga permisos de solo lectura que permiten al director ver, pero no modificar, la información enAWS IoT Greengrass. Por ejemplo, los directores con estos permisos pueden ver la lista de componentes implementados en un dispositivo principal de Greengrass, pero no pueden crear una implementación para cambiar los componentes que se ejecutan en ese dispositivo.

Detalles sobre los permisos

Esta política incluye los siguientes permisos:

- `greengrass`— Permite a los directores realizar acciones que devuelvan una lista de elementos o detalles sobre un elemento. Esto incluye las operaciones de API que comienzan con`ListoGet`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "greengrass:List*",
```

```
                "greengrass:Get*"
            ],
            "Resource": "*"
        }
    ]
}
```

Política administrada por AWS: AWSGreengrassResourceAccessRolePolicy

Puede adjuntar el`AWSGreengrassResourceAccessRolePolicy`política para sus entidades de IAM.AWS IoT Greengrasstambién adjunta esta política a una función de servicio que permiteAWS IoT Greengrasspara realizar acciones en su nombre. Para obtener más información, consulte Rol de servicio de Greengrass.

Esta política otorga permisos administrativos que permitenAWS IoT Greengrasspara realizar tareas esenciales, como recuperar las funciones de Lambda, gestionarAWS IoTsombras de dispositivos y verificación de los dispositivos cliente de Greengrass.

Detalles sobre los permisos

Esta política incluye los siguientes permisos.

- `greengrass`— Administrar los recursos de Greengrass.
- `iot(*Shadow)` — AdministrarAWS IoTsombras que tienen los siguientes identificadores especiales en sus nombres. Estos permisos son necesarios para queAWS IoT Greengrasspuede comunicarse con los dispositivos principales.
  - `*-gci`—AWS IoT Greengrassutiliza esta sombra para almacenar la información de conectividad de los dispositivos principales, de modo que los dispositivos cliente puedan descubrir los dispositivos principales y conectarse a ellos.
  - `*-gcm`—AWS IoT Greengrass V1utiliza esta sombra para notificar al dispositivo principal que el certificado de autoridad certificadora (CA) del grupo Greengrass ha rotado.
  - `*-gda`—AWS IoT Greengrass V1usa esta sombra para notificar una implementación al dispositivo principal.
  - `GG_*`— Sin usar.
- `iot(DescribeThingyDescribeCertificate)` — Recuperar información sobreAWS IoTcosas y certificados. Estos permisos son necesarios para queAWS IoT Greengrasspuede verificar los dispositivos cliente que se conectan a un dispositivo principal. Para obtener más información, consulte Interactúa con dispositivos IoT locales.

- `lambda`— Recuperar información sobreAWS Lambdafunciones. Este permiso es necesario para queAWS IoT Greengrass V1puede implementar funciones de Lambda en los núcleos de Greengrass. Para obtener más información, consulteEjecute la función Lambda enAWS IoT Greengrassnúcleoen elAWS IoT Greengrass V1Guía para desarrolladores.
- `secretsmanager`— Recupera el valor deAWS Secrets Managersecretos cuyos nombres comienzan por`greengrass-`. Este permiso es necesario para queAWS IoT Greengrass V1puede implementar los secretos de Secrets Manager en los núcleos de Greengrass. Para obtener más información, consulteDespliegue secretos en elAWS IoT Greengrassnúcleoen elAWS IoT Greengrass V1Guía para desarrolladores.
- `s3`— Recupera archivos y objetos de cubos de S3 cuyos nombres contienen`greengrassosagemaker`. Estos permisos son necesarios para queAWS IoT Greengrass V1puede implementar los recursos de aprendizaje automático que se almacenan en buckets de S3. Para obtener más información, consulteRecursos de aprendizaje automáticoen elAWS IoT Greengrass V1Guía para desarrolladores.
- `sagemaker`— Recuperar información sobre AmazonSageMakermodelos de inferencia de aprendizaje automático. Este permiso es necesario para queAWS IoT Greengrass V1puede implementar modelos de aprendizaje automático en los núcleos de Greengrass. Para obtener más información, consulteRealizar inferencias de aprendizaje automáticoen elAWS IoT Greengrass V1Guía para desarrolladores.

```
{
    "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/*"
            ]
        }
    ]
}
```

Actualizaciones de AWS IoT Greengrass en las políticas administradas de AWS

Puede ver los detalles sobre las actualizaciones deAWSpolíticas gestionadas paraAWS IoT Greengrassdesde el momento en que este servicio comenzó a rastrear estos cambios. Para recibir alertas automáticas sobre los cambios en esta página, suscríbase a la fuente RSS delAWS IoT Greengrass V2página de historial del documento.

Cambio	Descripción	Fecha
AWS IoT Greengrass comenzó el seguimiento de los cambios.	AWS IoT Greengrass comenzó el seguimiento de los cambios de las políticas administradas de AWS.	2 de julio de 2021

Prevención del suplente confuso entre servicios

El problema de la sustitución confusa es una cuestión de seguridad en la que una entidad que no tiene permiso para realizar una acción puede obligar a una entidad con más privilegios a realizar la acción. En AWS, la suplantación entre servicios puede dar lugar al problema del suplente confuso. La suplantación entre servicios puede producirse cuando un servicio (el servicio que lleva a cabo las llamadas) llama a otro servicio (el servicio al que se llama). El servicio que lleva a cabo las llamadas se puede manipular para utilizar sus permisos a fin de actuar en función de los recursos de otro cliente de una manera en la que no debe tener permiso para acceder. Para evitarlo, AWS proporciona herramientas que lo ayudan a proteger sus datos para todos los servicios con entidades principales de servicio a las que se les ha dado acceso a los recursos de su cuenta.

Se recomienda utilizar las claves de contexto de condición global `aws:SourceArn` y `aws:SourceAccount` en las políticas de recursos para limitar los permisos que AWS IoT Greengrass concede a otro servicio para el recurso. Si se utilizan ambas claves de contexto de condición global, el valor `aws:SourceAccount` y la cuenta del valor `aws:SourceArn` deben utilizar el mismo ID de cuenta cuando se utilicen en la misma declaración de política.

El valor de`aws:SourceArn`debe ser el recurso de cliente de Greengrass asociado al`sts:AssumeRole`request.

La forma más eficaz de protegerse contra el problema del suplente confuso es utilizar la clave de contexto de condición global de `aws:SourceArn` con el ARN completo del recurso. Si no conoce el ARN completo del recurso o si especifica varios recursos, utilice la clave de condición de contexto global `aws:SourceArn` con comodines (`*`) para las partes desconocidas del ARN. Por ejemplo, `arn:aws:greengrass::`*`account-id`*`:*`.

Para ver un ejemplo de una política que utiliza el`aws:SourceArn`y`aws:SourceAccount`claves de contexto de condición globales deCreación del rol de servicio de Greengrass.

Solución de problemas de administración de identidades y accesos en AWS IoT Greengrass

Utilice la siguiente información para diagnosticar y solucionar los problemas comunes que puedan surgir cuando trabaje con AWS IoT Greengrass e IAM.

Problemas

- No tengo autorización para realizar una acción en AWS IoT Greengrass
- No tengo autorización para realizar la operación:PassRole
- Soy administrador y deseo permitir que otros obtengan acceso a AWS IoT Greengrass
- Quiero permitir a personas externas a mi Cuenta de AWS el acceso a mis recursos de AWS IoT Greengrass

Para obtener ayuda general de solución de problemas, consulte Solución de problemas.

No tengo autorización para realizar una acción en AWS IoT Greengrass

Si recibe un error que indica que no está autorizado para llevar a cabo una acción, debe ponerse en contacto con su administrador para recibir ayuda. Su administrador es la persona que le facilitó su nombre de usuario y contraseña.

El siguiente ejemplo de error se produce cuando`mateojackson`El usuario de IAM intenta ver los detalles de un dispositivo principal, pero no tiene`greengrass:GetCoreDevice`permisos.

```
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
```

En este caso, Mateo pide a su administrador que actualice sus políticas de forma que pueda obtener acceso al recurso `arn:aws:greengrass:us-west-2:123456789012:coreDevices/MyGreengrassCore` mediante la acción `greengrass:GetCoreDevice`.

A continuación, se indican los problemas generales de IAM que pueden surgir al trabajar conAWS IoT Greengrass.

No tengo autorización para realizar la operación:PassRole

Si recibe un error que indica que no está autorizado a realizar la`iam:PassRole`acción, sus políticas deben actualizarse para que pueda transferir un rol aAWS IoT Greengrass.

Algunos servicios de Servicios de AWS le permiten transferir un rol existente a dicho servicio en lugar de crear un nuevo rol de servicio o uno vinculado al servicio. Para ello, debe tener permisos para transferir el rol al servicio.

En el siguiente ejemplo, el error se produce cuando un usuario de IAM denominado `marymajor` intenta utilizar la consola para realizar una acción en AWS IoT Greengrass. Sin embargo, la acción requiere que el servicio cuente con permisos que otorga un rol de servicio. Mary no tiene permisos para transferir el rol al servicio.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

En este caso, las políticas de Mary se deben actualizar para permitirle realizar la acción `iam:PassRole`.

Si necesita ayuda, póngase en contacto con su administrador de AWS. El administrador es la persona que le proporcionó las credenciales de inicio de sesión.

Soy administrador y deseo permitir que otros obtengan acceso a AWS IoT Greengrass

Para permitir que otros obtengan acceso a AWS IoT Greengrass, debe crear una entidad de IAM (usuario o rol) para la persona o la aplicación que necesita acceso. Esta persona utilizará las credenciales de la entidad para acceder a AWS. A continuación, debe asociar una política a la entidad que le conceda los permisos correctos en AWS IoT Greengrass.

Para comenzar de inmediato, consulte Creación del primer grupo y usuario delegado de IAM en la Guía del usuario de IAM.

Quiero permitir a personas externas a mi Cuenta de AWS el acceso a mis recursos de AWS IoT Greengrass

Puede crear un rol de IAM que los usuarios de otras cuentas o las personas externas a la organización puedan utilizar para acceder aAWSRecursos. Puede especificar una persona de confianza para que asuma el rol. Para obtener más información, consulteProporcionar acceso a un

usuario de IAM a otroCuenta de AWSque poseayProporcionar acceso aCuenta de AWSes propiedad de tercerosen elIAM User Guide.

AWS IoT Greengrass no admite el acceso entre cuentas basado en políticas basadas en recursos o listas de control de acceso (ACL).

Permitir el tráfico del dispositivo a través de un proxy o firewall

Los dispositivos principales y los componentes de Greengrass realizan solicitudes salientes a AWS servicios y otros sitios web. Como medida de seguridad, puede limitar el tráfico saliente a una pequeña variedad de puntos finales y puertos. Puede utilizar la siguiente información sobre los puntos de conexión y los puertos para limitar el tráfico de los dispositivos a través de un proxy, un firewall o un grupo de seguridad de Amazon VPC. Para obtener más información sobre cómo configurar un dispositivo principal para usar un proxy, consulte. Realizar la conexión en el puerto 443 o a través de un proxy de red

Temas

- Puntos finales para el funcionamiento básico
- Puntos finales para la instalación con aprovisionamiento automático
- Puntos finales para los componentes proporcionados AWS

Puntos finales para el funcionamiento básico

Los dispositivos principales de Greengrass utilizan los siguientes puntos finales y puertos para su funcionamiento básico.

Recupere los puntos finales AWS IoT

Obtenga los AWS IoT puntos finales que desee y guárdelos para usarlos más adelante. Cuenta de AWS El dispositivo utiliza estos puntos finales para conectarse a ellos. AWS IoT Haga lo siguiente:

1. Obtenga el punto final AWS IoT de datos para su. Cuenta de AWS

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

   La respuesta es similar a la del siguiente ejemplo, si la solicitud se realiza correctamente.

   ```
   {
   ```

```
  "endpointAddress": "device-data-prefix-ats.iot.us-west-2.amazonaws.com"
}
```

2. Obtenga el punto final de AWS IoT credenciales para su. Cuenta de AWS

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

La respuesta es similar a la del siguiente ejemplo, si la solicitud se realiza correctamente.

```
{
  "endpointAddress": "device-credentials-prefix.credentials.iot.us-
west-2.amazonaws.com"
}
```

punto de enlace	Puerto	Obligatoria	Descripción
`greengrass-ats.iot. region.amazonaws.com`	8443 o 443	Sí	Se utiliza para las operaciones del plano de datos, como la instalación de implementaciones y el trabajo con dispositivos cliente.
`device-data-prefix -ats.iot. region.amazonaws.com`	MQTT: 8883 o 443 HTTPS: 8443 o 443	Sí	Se utiliza para las operaciones del

punto de enlace	Puerto	Obligatoria	Descripción
			plano de datos para la administración de dispositivos, como la comunicación MQTT y la sincronización oculta con. AWS IoT Core

punto de enlace	Puerto	Obligatoria	Descripción
device-credentials-prefix .credentials.iot. *region*.amazonaws.com	443	Sí	Se utiliza para adquirir AWS credenciales, que el dispositivo principal utiliza para descargar artefactos de componentes de Amazon S3 y realizar otras operaciones. Para obtener más información, consulte Autorizar a los dispositivos principales a interactuar con

punto de enlace	Puerto	Obligatoria	Descripción
			AWS los servicios.

`*.s3.amazonaws.com`
 	
`*.s3.region.amazonaws.com`
	443	Sí	Se utiliza para las implementaciones. Este formato incluye el * carácter, ya que los prefijos de los puntos finales se controlan internamente y pueden cambiar en cualquier momento.

punto de enlace	Puerto	Obligatoria	Descripción
`data.iot.` *`region`*`.amazonaws.com`	443	No	Necesario si el dispositivo principal ejecuta una versión del núcleo de Greengrass anterior a la v2.4.0 y está configurado para usar un proxy de red. El dispositivo principal utiliza este punto final para la comunicación MQTT con él AWS IoT Core cuando se encuentra detrás de un proxy. Para obtener

punto de enlace	Puerto	Obligatoria	Descripción
			más información, consulte Configure un proxy de red.

Puntos finales para la instalación con aprovisionamiento automático

Los dispositivos principales de Greengrass utilizan los siguientes puntos finales y puertos al instalar el software AWS IoT Greengrass Core con aprovisionamiento automático de recursos.

punto de enlace	Puerto	Obligatoria	Descripción
`iot.region.amazonaws.com`	443	Sí	Se utiliza para crear AWS IoT recursos y recuperar información sobre los recursos existentes. AWS IoT
`iam.amazonaws.com`	443	Sí	Se utiliza para crear recursos de IAM y recuperar informaci

punto de enlace	Puerto	Obligatoria	Descripción
			ón sobre los recursos de IAM existentes.
`sts.region.amazonaws.com`	443	Sí	Se utiliza para obtener el ID de su. Cuenta de AWS
`greengrass. region.amazonaws.com`	443	No	Obligatorio si utiliza el `--deploy-dev-tools` argumento para implementar el componente CLI de Greengrass en el dispositivo principal.

Puntos finales para los componentes proporcionados AWS

Los dispositivos principales de Greengrass utilizan puntos finales adicionales en función de los componentes de software que ejecuten. Puede encontrar los puntos finales que requiere cada

componente AWS proporcionado en la sección de requisitos de la página de cada componente de esta guía para desarrolladores. Para obtener más información, consulte AWS-componentes proporcionados.

Validación de conformidad para AWS IoT Greengrass

Para saber si uno Servicio de AWS está dentro del ámbito de aplicación de programas de cumplimiento específicos, consulte Servicios de AWS Alcance por programa de cumplimiento Servicios de AWS de cumplimiento y elija el programa de cumplimiento que le interese. Para obtener información general, consulte Programas de AWS cumplimiento > Programas AWS .

Puede descargar informes de auditoría de terceros utilizando AWS Artifact. Para obtener más información, consulte Descarga de informes en AWS Artifact .

Su responsabilidad de cumplimiento al Servicios de AWS utilizarlos viene determinada por la confidencialidad de sus datos, los objetivos de cumplimiento de su empresa y las leyes y reglamentos aplicables. AWS proporciona los siguientes recursos para ayudar con el cumplimiento:

- Guías de inicio rápido sobre seguridad y cumplimiento: estas guías de implementación analizan las consideraciones arquitectónicas y proporcionan los pasos para implementar entornos básicos centrados en AWS la seguridad y el cumplimiento.
- Diseño de arquitectura para garantizar la seguridad y el cumplimiento de la HIPAA en Amazon Web Services: en este documento técnico se describe cómo pueden utilizar AWS las empresas para crear aplicaciones aptas para la HIPAA.

  > **Note**
  >
  > No Servicios de AWS todas cumplen con los requisitos de la HIPAA. Para más información, consulte la Referencia de servicios compatibles con HIPAA.

- AWS Recursos de de cumplimiento: esta colección de libros de trabajo y guías puede aplicarse a su industria y ubicación.
- AWS Guías de cumplimiento para clientes: comprenda el modelo de responsabilidad compartida desde el punto de vista del cumplimiento. Las guías resumen las mejores prácticas para garantizar la seguridad Servicios de AWS y orientan los controles de seguridad en varios marcos (incluidos el Instituto Nacional de Estándares y Tecnología (NIST), el Consejo de Normas de Seguridad del Sector de Tarjetas de Pago (PCI) y la Organización Internacional de Normalización (ISO)).

- Evaluación de los recursos con reglas en la guía para AWS Config desarrolladores: el AWS Config servicio evalúa en qué medida las configuraciones de los recursos cumplen con las prácticas internas, las directrices del sector y las normas.
- AWS Security Hub— Esto Servicio de AWS proporciona una visión completa del estado de su seguridad interior AWS. Security Hub utiliza controles de seguridad para evaluar sus recursos de AWS y comprobar su cumplimiento con los estándares y las prácticas recomendadas del sector de la seguridad. Para obtener una lista de los servicios y controles compatibles, consulte la Referencia de controles de Security Hub.
- Amazon GuardDuty: Servicio de AWS detecta posibles amenazas para sus cargas de trabajo Cuentas de AWS, contenedores y datos mediante la supervisión de su entorno para detectar actividades sospechosas y maliciosas. GuardDuty puede ayudarlo a cumplir con varios requisitos de conformidad, como el PCI DSS, al cumplir con los requisitos de detección de intrusiones exigidos por ciertos marcos de cumplimiento.
- AWS Audit Manager— Esto le Servicio de AWS ayuda a auditar continuamente su AWS uso para simplificar la gestión del riesgo y el cumplimiento de las normativas y los estándares del sector.

Resiliencia en AWS IoT Greengrass

LaAWSLa infraestructura global de se basa en regiones y zonas de disponibilidad de Amazon Web Services. CadaRegión de AWSproporciona varias zonas de disponibilidad físicamente independientes y aisladas que se encuentran conectadas mediante redes con un alto nivel de rendimiento y redundancia, además de baja latencia. Con las zonas de disponibilidad, puede diseñar y utilizar aplicaciones y bases de datos que realizan una conmutación por error automática entre las zonas sin interrupciones. Las zonas de disponibilidad tienen una mayor disponibilidad, tolerancia a errores y escalabilidad que las infraestructuras tradicionales de centros de datos únicos o múltiples.

Para obtener más información, consulte Infraestructura global de AWS.

Además de la infraestructura global de AWS, AWS IoT Greengrass ofrece varias características que le ayudan con sus necesidades de resiliencia y copia de seguridad de los datos.

- Puede configurar un dispositivo de núcleo de Greengrass para escribir registros en el sistema de archivos local y en CloudWatch Registros. Si el dispositivo de núcleo pierde conectividad, puede seguir registrando mensajes en el sistema de archivos. Cuando se vuelve a conectar, escribe los mensajes de registro en CloudWatch Registros. Para obtener más información, consulte Supervisar AWS IoT Greengrass registros.

- Si un dispositivo principal pierde energía durante una implementación, reanuda la implementación después de laAWS IoT GreengrassEl software principal se inicia de nuevo.
- Si un dispositivo de núcleo pierde conectividad a Internet, los dispositivos cliente de Greengrass pueden seguir comunicándose a través de la red local.
- Puede crear componentes de Greengrass que leenadministrador de transmisionestransmite y envía los datos a destinos de almacenamiento locales.

Seguridad de la infraestructura en AWS IoT Greengrass

Al tratarse de un servicio administrado, AWS IoT Greengrass está protegido por los procedimientos de seguridad de red globales de AWS que se describen en el documento técnico Amazon Web Services: Información general sobre los procesos de seguridad.

Puede utilizar llamadas a la API publicadas en AWS para obtener acceso a AWS IoT Greengrass a través de la red. Los clientes deben ser compatibles con Transport Layer Security (TLS) 1.2 o una versión posterior. Recomendamos TLS 1.3 o una versión posterior. Los clientes también deben ser compatibles con conjuntos de cifrado con confidencialidad directa total (PFS), como Ephemeral Diffie-Hellman (DHE) o Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). La mayoría de los sistemas modernos como Java 7 y posteriores son compatibles con estos modos.

Las solicitudes deben estar firmadas mediante un ID de clave de acceso y una clave de acceso secreta que esté asociada a una entidad principal de IAM. También puede utilizar AWS Security Token Service (AWS STS) para generar credenciales de seguridad temporales para firmar solicitudes.

En un AWS IoT Greengrass entorno, los dispositivos utilizan certificados X.509 y claves criptográficas para conectarse y autenticarse en el. Nube de AWS Para obtener más información, consulte the section called “Autenticación y autorización de dispositivos”.

Configuración y análisis de vulnerabilidades en AWS IoT Greengrass

Los entornos de IoT pueden constar de un gran número de dispositivos que tienen diversas capacidades, son de larga duración y están distribuidos geográficamente. Estas características hacen que la configuración del dispositivo sea compleja y propensa a errores. Y dado que los dispositivos a menudo están limitados en potencia informática, memoria y capacidades

de almacenamiento, esto limita el uso del cifrado y otras formas de seguridad en los propios dispositivos. Además, los dispositivos a menudo usan software con vulnerabilidades conocidas. Estos factores hacen que los dispositivos de IoT sean un objetivo atractivo para los piratas informáticos y dificultan la protección continuada de los mismos.

Para afrontar estos desafíos, AWS IoT Device Defender proporciona herramientas para identificar los problemas de seguridad y las desviaciones de las prácticas recomendadas. Puede utilizar AWS IoT Device Defender para analizar, auditar y monitorear los dispositivos conectados para detectar comportamientos anómalos y mitigar riesgos de seguridad. AWS IoT Device Defender puede auditar dispositivos para asegurarse de que cumplen las prácticas de seguridad recomendadas y detectan comportamientos anómalos en los dispositivos. Esto permite aplicar políticas de seguridad coherentes en todos los dispositivos y responder rápidamente cuando los dispositivos sufren ataques. Para obtener más información, consulte los siguientes temas:

- Lacomponente Device Defender de
- AWS IoT Device Defender en la Guía para desarrolladores de AWS IoT Core.

En los entornos de AWS IoT Greengrass, debe tener en cuenta las siguientes consideraciones:

- Es su responsabilidad proteger los dispositivos físicos, el sistema de archivos en sus dispositivos y la red local.
- AWS IoT Greengrassno aplica el aislamiento de red para los componentes de Greengrass definidos por el usuario, tanto si se ejecutan en un contenedor de Greengrass como si no. Por lo tanto, es posible que los componentes de Greengrass se comuniquen con cualquier otro proceso que se ejecute en el sistema o fuera a través de la red.

Integridad de código enAWS IoT Greengrass V2

AWS IoT Greengrassimplementa componentes de software desde elNube de AWSa dispositivos que ejecutan elAWS IoT GreengrassSoftware Core. Estos componentes de software incluyenAWS-componentes proporcionadosycomponentes personalizadosque subes a tuCuenta de AWS. Cada componente se compone de una receta. La receta define los metadatos del componente y cualquier número de artefactos, que son binarios de componentes, como código compilado y recursos estáticos. Los artefactos de componentes se almacenan en Amazon S3.

A medida que desarrolla e implementa componentes de Greengrass, sigue estos pasos básicos que funcionan con artefactos de componentes en suCuenta de AWSy en tus dispositivos:

1. Crea y carga artefactos en depósitos de S3.
2. Cree un componente a partir de una receta y artefactos en elAWS IoT Greengrassservicio, que calcula unhash criptográficode cada artefacto.
3. Implemente un componente en los dispositivos principales de Greengrass, que descargan y verifican la integridad de cada artefacto.

AWSes responsable de mantener la integridad de los artefactos después de cargar artefactos en depósitos de S3, incluso cuando implementa componentes en los dispositivos principales de Greengrass. Usted es responsable de proteger los artefactos de software antes de cargar los artefactos en los depósitos de S3. También es responsable de garantizar el acceso a los recursos de suCuenta de AWS, incluidos los depósitos de S3 en los que se cargan artefactos de componentes.

> **Note**
>
> Amazon S3 proporciona una función denominada Bloqueo de objetos S3 que puede utilizar para proteger contra cambios en los artefactos de componentes en los depósitos de S3Cuenta de AWS. Puede usar el bloqueo de objetos de S3 para evitar que se elimine o se sobrescriban los artefactos de los componentes. Para obtener más información, consulteUsar Bloqueo de objetos de S3en laAmazon Simple Storage Service Guide.

CuandoAWSpublica un componente público y, al cargar un componente personalizado,AWS IoT Greengrasscalcula un resumen criptográfico para cada artefacto de componente.AWS IoT Greengrassactualiza la receta de componentes para incluir el resumen de cada artefacto y el algoritmo hash utilizado para calcular ese resumen. Este resumen garantiza la integridad del artefacto, porque si el artefacto cambia enNube de AWSo durante la descarga, su resumen de archivos no coincide con el resumen queAWS IoT Greengrassalmacena en la receta de componentes. Para obtener más información, consulteArtefactos de la referencia de la receta de componentes.

Cuando implementa un componente en un dispositivo principal, elAWS IoT GreengrassEl software principal descarga la receta de componentes y cada artefacto de componente que define la receta. LaAWS IoT GreengrassEl software principal calcula el resumen de cada archivo de artefacto descargado y lo compara con el resumen de ese artefacto en la receta. Si los resúmenes no coinciden, se producirá un error en la implementación y seAWS IoT GreengrassEl software principal elimina los artefactos descargados del sistema de archivos del dispositivo. Para obtener

más información acerca de cómo las conexiones entre los dispositivos de Core yAWS IoT Greengrassestán protegidos, consulteCifrado en tránsito.

Usted es responsable de proteger los archivos de artefactos de componentes en los sistemas de archivos de sus dispositivos principales. LaAWS IoT GreengrassEl software principal guarda artefactos en el`packages`carpeta de la carpeta raíz de Greengrass. Puede usarAWS IoT Device Defenderpara analizar, auditar y supervisar los dispositivos principales. Para obtener más información, consulte Configuración y análisis de vulnerabilidades en AWS IoT Greengrass.

AWS IoT Greengrass y puntos de conexión de VPC de interfaz (AWS PrivateLink)

Puede establecer una conexión privada entre la VPC y el plano de control deAWS IoT Greengrass mediante la creación de un punto de conexión de VPC de interfaz. Puede usar este punto final para administrar los componentes, las implementaciones y los dispositivos principales del AWS IoT Greengrass servicio. Los puntos de conexión de interfaz cuentan con tecnología de AWS PrivateLink que le permite acceder de forma privada a las API de AWS IoT Greengrass sin una puerta de enlace de Internet, un dispositivo NAT, una conexión de VPN o una conexión de AWS Direct Connect. Las instancias de su VPC no necesitan direcciones IP públicas para comunicar con las API de AWS IoT Greengrass. El tráfico entre la VPC y AWS IoT Greengrass no sale de la red de Amazon.

Cada punto de conexión de la interfaz está representado por una o más interfaces de red elásticas en las subredes.

Para obtener más información, consulte Puntos de conexión de la VPC de interfaz (AWS PrivateLink) en la Guía del usuario de Amazon VPC.

Temas

- Consideraciones para los puntos de conexión de VPC de AWS IoT Greengrass
- Crear un punto de conexión de VPC de interfaz para operaciones del plano de control AWS IoT Greengrass
- Creación de una política de puntos de conexión de VPC para AWS IoT Greengrass
- Opere un dispositivo AWS IoT Greengrass central en VPC

Consideraciones para los puntos de conexión de VPC de AWS IoT Greengrass

Antes de configurar un punto de conexión de VPC de interfaz para AWS IoT Greengrass, revise las Propiedades y limitaciones de los puntos de conexión de interfaz en la Guía del usuario de Amazon VPC. Además, tenga en cuenta las siguientes consideraciones:

- AWS IoT Greengrass admite realizar llamadas a todas sus acciones de la API de plano de control desde su VPC. El plano de control incluye operaciones como CreateDeployment y ListEffectiveDeployments. El plano de control no incluye operaciones como ResolveComponentCandidatesDiscover, que son operaciones del plano de datos.
- Los puntos de conexión de VPC para AWS IoT Greengrass actualmente no se admiten en las regiones de AWS de China.

Crear un punto de conexión de VPC de interfaz para operaciones del plano de control AWS IoT Greengrass

Puede crear un punto de conexión de VPC para el plano de control AWS IoT Greengrass mediante la consola de Amazon VPC o la AWS Command Line Interface (AWS CLI). Para obtener más información, consulte Creación de un punto de conexión de interfaz en la Guía del usuario de Amazon VPC.

Cree un punto de enlace de la VPC para AWS IoT Greengrass, mediante el siguiente nombre de servicio:

- com.amazonaws.*region*.greengrass

Si habilita DNS privado para el punto de enlace, puede realizar solicitudes a la API para AWS IoT Greengrass usando su nombre de DNS predeterminado para la región, por ejemplo `greengrass.us-east-1.amazonaws.com`. El DNS privado está habilitado de forma predeterminada.

Para obtener más información, consulte Acceso a un servicio a través de un punto de conexión de interfaz en la Guía del usuario de Amazon VPC.

Creación de una política de puntos de conexión de VPC para AWS IoT Greengrass

Puede adjuntar una política de punto de conexión a su punto de conexión de VPC que controle el acceso a las operaciones del plano de control AWS IoT Greengrass. La política especifica la siguiente información:

- La entidad principal que puede realizar acciones.
- Acciones que la entidad principal puede realizar.
- Los recursos sobre los que la entidad principal puede realizar acciones.

Para obtener más información, consulte Control del acceso a los servicios con puntos de enlace de la VPC en la Guía del usuario de Amazon VPC.

Example Ejemplo: política de punto de conexión de VPC para acciones de AWS IoT Greengrass

A continuación, se muestra un ejemplo de una política de puntos de conexión de AWS IoT Greengrass. Cuando se asocia con un punto de conexión, esta política concede acceso a las acciones de AWS IoT Greengrass mostradas para todas las entidades principales en todos los recursos.

```
{
    "Statement": [
        {
            "Principal": "*",
            "Effect": "Allow",
            "Action": [
                "greengrass:CreateDeployment",
                "greengrass:ListEffectiveDeployments"
            ],
            "Resource": "*"
        }
    ]
}
```

Opere un dispositivo AWS IoT Greengrass central en VPC

Puede operar un dispositivo principal de Greengrass y realizar despliegues en una VPC sin acceso público a Internet. Como mínimo, debe configurar los siguientes puntos de enlace de VPC con los

alias de DNS correspondientes. Para obtener más información sobre cómo crear y usar puntos de enlace de VPC, consulte Crear un punto de enlace de VPC en la Guía del usuario de Amazon VPC.

> **Note**
>
> La función de VPC para crear automáticamente un registro DNS está deshabilitada para las credenciales AWS IoT data yAWS IoT. Para conectar estos puntos de conexión, debe crear manualmente un registro de DNS privado. Para obtener más información, consulte DNS privado para puntos finales de interfaz. Para obtener más información sobre las limitaciones de la AWS IoT Core VPC, consulte Limitaciones de los puntos finales de la VPC.

Requisitos previos

- Debe instalar el software AWS IoT Greengrass principal siguiendo los pasos de aprovisionamiento manual. Para obtener más información, consulte Instale el software AWS IoT Greengrass principal con aprovisionamiento manual de recursos.

Limitaciones

- El funcionamiento de un dispositivo principal de Greengrass en VPC no es compatible en las regiones de China y. AWS GovCloud (US) Regions
- Para obtener más información sobre las limitaciones de los puntos de AWS IoT data enlace de VPC del proveedor de AWS IoT credenciales, consulte Limitaciones.

Configure su dispositivo principal de Greengrass para que funcione en VPC

1. Obtenga los AWS IoT puntos finales que desee y guárdelos para usarlos más adelante. Cuenta de AWS El dispositivo utiliza estos puntos finales para conectarse a ellos. AWS IoT Haga lo siguiente:

   a. Obtenga el punto final AWS IoT de datos para su. Cuenta de AWS

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

   La respuesta es similar a la del siguiente ejemplo, si la solicitud se realiza correctamente.

```
{
  "endpointAddress": "device-data-prefix-ats.iot.us-west-2.amazonaws.com"
}
```

b. Obtenga el punto final de AWS IoT credenciales para su. Cuenta de AWS

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

La respuesta es similar a la del siguiente ejemplo, si la solicitud se realiza correctamente.

```
{
  "endpointAddress": "device-credentials-prefix.credentials.iot.us-
west-2.amazonaws.com"
}
```

2. Cree una interfaz de Amazon VPC para los puntos de enlace AWS IoT data y AWS IoT las credenciales:

a. Vaya a la consola de puntos de conexión de VPC, en Nube virtual privada en el menú de la izquierda, elija Puntos de enlace y después Crear punto de conexión.

b. En la página Crear punto de conexión, especifique la siguiente información.

- Elija Servicio de AWS en Categoría de servicio.
- En Nombre del servicio, busque introduciendo la palabra clave `iot`. En la lista de servicios de `iot` que se muestra, elija el punto de conexión.

  Si crea un punto de conexión de VPC para el plano de datos de AWS IoT Core, elija el punto de conexión de la API del plano de datos de AWS IoT Core para su región. El punto de conexión tendrá el formato `com.amazonaws.region.iot.data`.

  Si crea un punto de conexión de VPC para el proveedor de credenciales de AWS IoT Core, elija el punto de conexión del proveedor de credenciales de AWS IoT Core para su región. El punto de conexión tendrá el formato `com.amazonaws.region.iot.credentials`.

  > **Note**
  >
  > El nombre del servicio para el plano de datos de AWS IoT Core en la región de China tendrá el formato `cn.com.amazonaws.region.iot.data`. La región de

China no admite la creación de puntos de conexión de VPC para el proveedor de credenciales de AWS IoT Core.

- En VPC y Subredes, elija la VPC en la que desee crear el punto de conexión, así como las zonas de disponibilidad (AZ) en las que desee crear la red de puntos de conexión.
- En Habilitar nombre de DNS, asegúrese de que Habilitar para este punto de conexiónno está seleccionado. Ni el plano de datos de AWS IoT Core ni el proveedor de credenciales de AWS IoT Core admiten todavía nombres DNS privados.
- En Grupo de seguridad, elija los grupos de seguridad que deban asociarse a las interfaces de red de punto de conexión.
- Puede agregar o eliminar etiquetas, si lo desea. Las etiquetas son pares de nombre-valor que se utilizan para asociar al punto de conexión.

c. Para crear su punto de conexión de VPC, elija Crear punto de conexión.

3. Después de crear el punto de conexión de AWS PrivateLink, en la pestaña Detalles del punto de conexión, verá una lista de nombres de DNS. Puede utilizar uno de estos nombres DNS que ha creado en esta sección para configurar su zona alojada privada.
4. Cree un punto de conexión Amazon S3. Para obtener más información, consulte Crear un punto de enlace de VPC para Amazon S3.
5. Si utiliza los componentes AWSde Greengrass proporcionados por Greengrass, es posible que se necesiten configuraciones y puntos finales adicionales. Para ver los requisitos de los puntos finales, seleccione el componente de la lista de componentes AWS proporcionados y consulte la sección de requisitos. Por ejemplo, los requisitos del componente del administrador de registros indican que este componente debe poder realizar las solicitudes salientes al punto final. `logs.region.amazonaws.com`

   Si utiliza su propio componente, es posible que deba revisar las dependencias y realizar pruebas adicionales para determinar si se requieren puntos finales adicionales.
6. En la configuración del núcleo de Greengrass, `greengrassDataPlaneEndpoint` debe estar configurado en. **`iotdata`** Para obtener más información, consulte Configuración del núcleo de Greengrass.
7. Si se encuentra en la `us-east-1` región, `s3EndpointType` defina el parámetro de configuración **`REGIONAL`** en la configuración del núcleo de Greengrass. Esta función está disponible para las versiones 2.11.3 o posteriores del núcleo de Greengrass.

Example Ejemplo: configuración de componentes

```
{
"aws.greengrass.Nucleus": {
   "configuration": {
      "awsRegion": "us-east-1",
      "iotCredEndpoint": "xxxxxx.credentials.iot.region.amazonaws.com",
      "iotDataEndpoint": "xxxxxx-ats.iot.region.amazonaws.com",
      "greengrassDataPlaneEndpoint": "iotdata",
      "s3EndpointType": "REGIONAL"
      ...
     }
   }
}
```

La siguiente tabla proporciona información sobre los alias de DNS privados personalizados correspondientes.

Servicio	Nombre del servicio de punto de conexión de VPC	Tipo de punto final de VPC	Alias de DNS privado personalizado	Notas
AWS IoT data	`com.amazonaws.` *`region`*`.iot.d`	Interfaz	*`prefix`*`-ats.iot.` *`region`*`.a` `s.com`	El registro DNS privado debe coincidir con el AWS IoT data punto final de su cuenta:`aws iot describe-`

Servicio	Nombre del servicio de punto de conexión de VPC	Tipo de punto final de VPC	Alias de DNS privado personalizado	Notas
				`endpoint --endpoint-type iot:Data-ATS .`
Credenciales de AWS IoT	`com.amazonaws. region .iot.credentials`	Interfaz	`prefix.credentials.iot.s.com`	El registro DNS privado debe coincidir con el punto final de AWS IoT las credenciales de su cuenta:`aws iot describe-endpoint --endpoint-type iot:CredentialProvider .`

Servicio	Nombre del servicio de punto de conexión de VPC	Tipo de punto final de VPC	Alias de DNS privado personalizado	Notas
Amazon S3	`com.amazonaws.` *`region`*`.s3`	Interfaz		El registro DNS se crea automáticamente.

Prácticas recomendadas de seguridad para AWS IoT Greengrass

Este tema contiene las prácticas de seguridad recomendadas para AWS IoT Greengrass.

Conceda los mínimos permisos posibles

Siga el principio de privilegios mínimos para sus componentes ejecutándolos como usuarios sin privilegios. Los componentes no deben ejecutarse como root a menos que sea absolutamente necesario.

Utilice el conjunto mínimo de permisos en las funciones de IAM. Limite el uso de`*`comodín para el`Action`y`Resource`propiedades en sus políticas de IAM. En su lugar, declare un conjunto finito de acciones y recursos cuando sea posible. Para obtener más información acerca de los privilegios mínimos y otras prácticas recomendadas de política, consulte the section called “Prácticas recomendadas relativas a políticas”.

La mejor práctica de privilegios mínimos también se aplica aAWS IoTpolíticas que adjunte a su núcleo de Greengrass.

No codifique las credenciales en los componentes de Greengrass

No codifique las credenciales en sus componentes de Greengrass definidos por el usuario. Para proteger mejor sus credenciales:

- Para interactuar conAWSservicios, defina los permisos para acciones y recursos específicos en elFunción principal de servicio de dispositivos de Greengrass.
- Usa elcomponente de administrador secretopara almacenar sus credenciales. O bien, si la función usaAWSSDK, utilice las credenciales de la cadena de proveedores de credenciales predeterminada.

No registre información confidencial

Debe evitar el registro de credenciales y otra información de identificación personal (PII). Le recomendamos que implemente las siguientes medidas de seguridad, aunque el acceso a los registros locales de un dispositivo principal requiera privilegios de root y el acceso a CloudWatch Los registros requieren permisos de IAM.

- No utilice información confidencial en las rutas de temas de MQTT.
- No utilice información confidencial en nombres, tipos y atributos de dispositivo (objeto) en el registro de AWS IoT Core.
- No registre información confidencial en los componentes de Greengrass definidos por el usuario ni en las funciones de Lambda.
- No utilice información confidencial en los nombres e identificadores de los recursos de Greengrass:
  - Dispositivos principales
  - Componentes
  - Implementaciones
  - Loggers

Mantenga sincronizado el reloj del dispositivo

Es importante que la hora del dispositivo sea precisa. Los certificados X.509 tienen una fecha y una hora de caducidad. El reloj del dispositivo se utiliza para comprobar que un certificado de servidor sigue siendo válido. Los relojes de dispositivos pueden variar con el tiempo o las baterías pueden descargarse.

Para obtener más información, consulte las prácticas recomendadas Mantener sincronizado el reloj del dispositivo en la Guía del desarrollador de AWS IoT Core.

Recomendaciones de Cipher Suite

De forma predeterminada, Greengrass selecciona los conjuntos de cifrado TLS más recientes disponibles en el dispositivo. Considere la posibilidad de deshabilitar el uso de conjuntos de cifrado antiguos en el dispositivo. Por ejemplo, los conjuntos de cifrado CBC.

Para obtener más información, consulte laConfiguración de criptografía de Java.

Véase también

- Mejores prácticas de seguridad enAWS IoT Coreen elAWS IoTGuía para desarrolladores
- Diez reglas de oro de seguridad para las soluciones de IoT industrialen elInternet de las cosas enAWSBlog oficial

Uso AWS IoT Device Tester para AWS IoT Greengrass V2

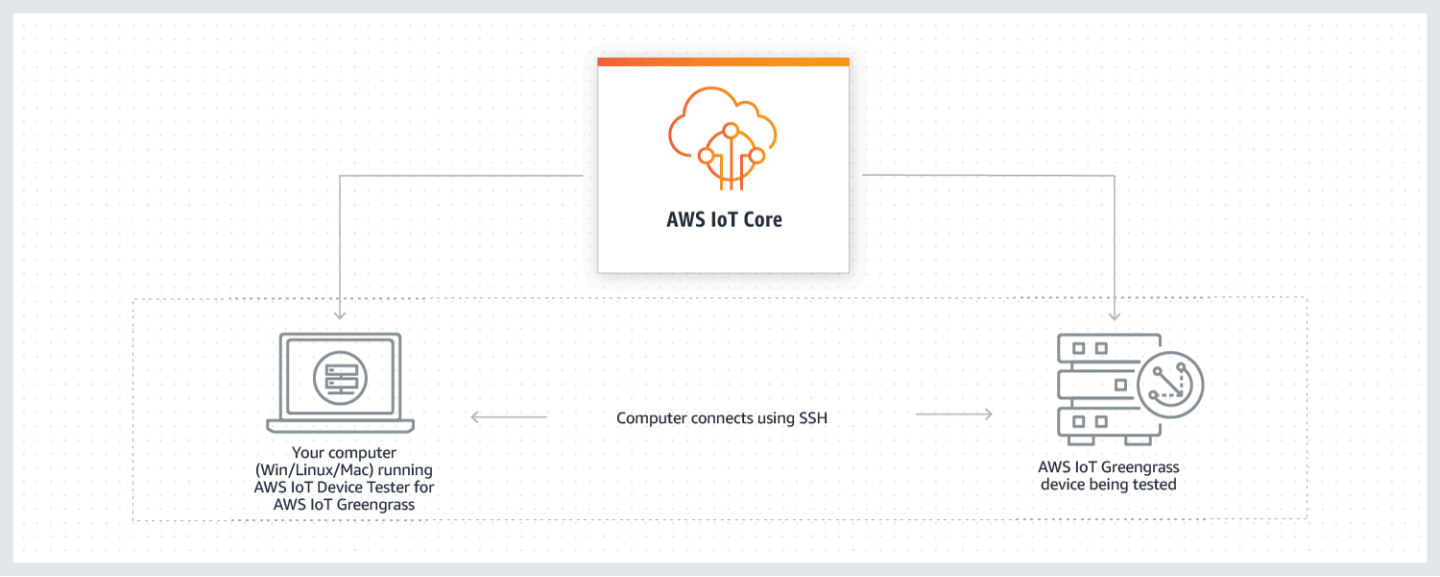
AWS IoT Device Tester (IDT) es un marco de pruebas descargable que le permite validar dispositivos de IoT. Puede usar IDT AWS IoT Greengrass para ejecutar el paquete de AWS IoT Greengrass calificación y crear y ejecutar conjuntos de pruebas personalizados para sus dispositivos.

IDT for AWS IoT Greengrass se ejecuta en el ordenador anfitrión (Windows, macOS o Linux) conectado al dispositivo que se va a probar. Ejecuta pruebas y agrega resultados. También proporciona una interfaz de línea de comandos para administrar el proceso de pruebas.

AWS IoT Greengrass paquete de cualificación

Úselo AWS IoT Device Tester para AWS IoT Greengrass la versión 2 para comprobar que el software AWS IoT Greengrass principal se ejecuta en su hardware y puede comunicarse con el Nube de AWS. También realiza end-to-end pruebas con AWS IoT Core. Por ejemplo, verifica que el dispositivo pueda implementar componentes y actualizarlos.

Si desea añadir su hardware al catálogo de AWS Partner dispositivos, ejecute el conjunto de AWS IoT Greengrass requisitos para generar informes de pruebas a los que pueda enviarlos AWS IoT. Para obtener más información, consulte el Programa de Calificación de Dispositivos de AWS.



IDT for AWS IoT Greengrass V2 organiza las pruebas utilizando los conceptos de conjuntos de pruebas y grupos de pruebas.

- Un conjunto de pruebas es el conjunto de grupos de pruebas que se utiliza para verificar que un dispositivo funciona con versiones particulares de AWS IoT Greengrass.
- Un grupo de pruebas es el conjunto de pruebas individuales relacionadas con una característica en particular, como las implementaciones de componentes.

Para obtener más información, consulte Utilice IDT para ejecutar el conjunto de AWS IoT Greengrass cualificaciones.

Compatibilidad con los conjuntos de prueba

A partir de la versión 4.0.1 de IDT, IDT para AWS IoT Greengrass V2 combina una configuración y un formato de resultados estandarizados con un entorno de conjuntos de pruebas que le permite desarrollar conjuntos de pruebas personalizados para sus dispositivos y el software de los dispositivos. Puede añadir pruebas personalizadas para su propia validación interna o proporcionárselas a sus clientes para la verificación de los dispositivos.

La forma en que un escritor de pruebas configura un conjunto de pruebas personalizado determina las configuraciones de configuración necesarias para ejecutar conjuntos de pruebas personalizados. Para obtener más información, consulte Uso de IDT para desarrollar y ejecutar sus propios conjuntos de pruebas.

Versiones compatibles de AWS IoT Device Tester para AWS IoT Greengrass V2

En este tema se enumeran las versiones compatibles de IDT para V2. AWS IoT Greengrass Como práctica recomendada, le recomendamos que utilice la última versión de IDT para AWS IoT Greengrass V2 que sea compatible con la versión de destino de la AWS IoT Greengrass V2. Las nuevas versiones de AWS IoT Greengrass pueden requerir la descarga de una nueva versión de IDT para AWS IoT Greengrass V2. Cuando inicie una prueba, recibirá una notificación si IDT para AWS IoT Greengrass V2 no es compatible con la versión AWS IoT Greengrass que está utilizando.

Al descargar el software, usted acepta el contrato de AWS IoT Device Tester licencia.

> **Note**
>
> IDT no admite la ejecución por parte de varios usuarios desde una ubicación compartida, como un directorio NFS o una carpeta compartida de red de Windows. Le recomendamos

que extraiga el paquete IDT en una unidad local y ejecute el binario IDT en su estación de trabajo local.

Última versión de IDT para V2 AWS IoT Greengrass

Puede usar esta versión de IDT para AWS IoT Greengrass V2 con la AWS IoT Greengrass versión que se indica aquí.

IDT v4.9.4 para AWS IoT Greengrass

Versiones compatibles: AWS IoT Greengrass

- Núcleo Greengrass v2.12.0, v2.11.0, v2.10.0 y v2.9.5

Descargas del software IDT:

- IDT v4.9.4 con el conjunto de pruebas GGV2Q_2.5.4 para Linux
- IDT v4.9.4 con el conjunto de pruebas GGV2Q_2.5.4 para macOS
- IDT v4.9.4 con el conjunto de pruebas GGV2Q_2.5.4 para Windows

Notas de la versión:

- Permite la validación y calificación de dispositivos que ejecutan las versiones de software AWS IoT Greengrass Core 2.12.0, 2.11.0, 2.10.0 y 2.9.5.
- Elimina los grupos de pruebas del administrador de transmisiones y el aprendizaje automático.

Notas adicionales:

- Si su dispositivo usa un HSM y usted usa nucleus 2.10.x, migre a Greengrass nucleus versión 2.11.0 o posterior.

Versión del conjunto de pruebas:

`GGV2Q_2.5.4`

- Publicado el 03 de mayo de 2022

Versiones anteriores de IDT para AWS IoT Greengrass

También se admiten las siguientes versiones anteriores de IDT para AWS IoT Greengrass V2.

IDT v4.9.3 para AWS IoT Greengrass

Versiones compatibles: AWS IoT Greengrass

- Núcleo Greengrass v2.12.0, v2.11.0, v2.10.0 y v2.9.5

Descargas del software IDT:

- IDT v4.9.3 con el conjunto de pruebas GGV2Q_2.5.3 para Linux
- IDT v4.9.3 con el conjunto de pruebas GGV2Q_2.5.3 para macOS
- IDT v4.9.3 con el conjunto de pruebas GGV2Q_2.5.3 para Windows

Notas de la versión:

- Corrige un problema en las pruebas de componentes al probar un dispositivo Linux desde un host de Windows o viceversa.
- Elimina el caso de `localcomponent component` prueba del grupo de pruebas. Este caso de prueba ya no es necesario para la calificación.

Notas adicionales:

- Si su dispositivo usa un HSM y usted usa nucleus 2.10.x, migre a Greengrass nucleus versión 2.11.0 o posterior.

Versión del conjunto de pruebas:

`GGV2Q_2.5.3`

- Publicado el 04.05 de abril de 2020

Versiones no compatibles de para la versión 2 AWS IoT Device TesterAWS IoT Greengrass

En este tema se enumeran las versiones no compatibles de IDT para la versión 2. AWS IoT Greengrass Las versiones que no son compatibles no reciben actualizaciones ni correcciones de errores. Para obtener más información, consulte the section called “Política de soporte AWS IoT Device Tester para AWS IoT Greengrass”.

IDT v4.9.2 para AWS IoT Greengrass

Notas de la versión:

- Soluciona un problema por el que el conjunto de pruebas Lambda fallaba debido a la obsolescencia de Java 8.

Versión del conjunto de pruebas:

`GGV2Q_2.5.2`

- Publicado el 18 de marzo de 2020

IDT v4.9.1 para AWS IoT Greengrass

Notas de la versión:

- Le permite validar y calificar los dispositivos que ejecutan las versiones 2.12.0, 2.11.0, 2.10.0 y 2.9.5 del software AWS IoT Greengrass principal.
- Correcciones de errores menores.

Versión del conjunto de pruebas:

`GGV2Q_2.5.1`

- Publicado el 05 de octubre de 2020

IDT v4.7.0 para AWS IoT Greengrass

Versiones compatibles: AWS IoT Greengrass

- Núcleo Greengrass v2.11.0, v2.10.0 y v2.9.5

Notas de la versión:

- Le permite validar y calificar los dispositivos que ejecutan las versiones 2.11.0, 2.10.0 y 2.9.5 del software AWS IoT Greengrass Core.
- Añade soporte para almacenar los valores de los datos de usuario de IDT en el almacén de AWS Systems Manager parámetros y recuperarlos en la configuración mediante la sintaxis de marcadores de posición.
- Correcciones de errores menores.

Versión del conjunto de pruebas:

`GGV2Q_2.5.0`

- Publicado el 13 de diciembre de 2022

IDT v4.5.11 para AWS IoT Greengrass

Notas de la versión:

- Le permite validar y calificar los dispositivos que ejecutan las versiones de software AWS IoT Greengrass Core 2.9.1, 2.9.0, 2.8.1, 2.8.0, 2.7.0 y 2.6.0.
- Añade compatibilidad para probar PreInstalled Greengrass en un dispositivo central.
- Correcciones de errores menores.

Versión del conjunto de pruebas:

`GGV2Q_2.4.1`

- Publicado el 13 de octubre de 2022

IDT v4.5.8 para AWS IoT Greengrass

Notas de la versión:

- Le permite validar y calificar los dispositivos que ejecutan las versiones 2.7.0, 2.6.0 y 2.5.6 del software AWS IoT Greengrass Core.
- Le permite realizar pruebas con PreInstalled Greengrass en un dispositivo central.
- Correcciones de errores menores.

Versión del conjunto de pruebas:

`GGV2Q_2.4.0`

- Publicado el 12 de agosto de 2022

IDT v4.5.3 para AWS IoT Greengrass

Notas de la versión:

- Le permite validar y calificar los dispositivos que ejecutan las versiones de software AWS IoT Greengrass Core 2.7.0, 2.6.0, 2.5.6, 2.5.5, 2.5.4 y 2.5.3.
- Actualiza la DockerApplicationManager prueba para usar una imagen de docker basada en ECR.
- Correcciones de errores menores.

Versión del conjunto de pruebas:

`GGV2Q_2.3.1`

- Publicado el 15 de abril de 2022

IDT v4.5.1 para AWS IoT Greengrass

Notas de la versión:

- Le permite validar y calificar los dispositivos que ejecutan el software AWS IoT Greengrass Core v2.5.3.
- Añade soporte para validar y calificar los dispositivos basados en Linux que utilizan un módulo de seguridad de hardware (HSM) para almacenar la clave privada y el certificado que utiliza el software Core. AWS IoT Greengrass
- Implementa el nuevo orquestador de pruebas de IDT para configurar conjuntos de pruebas personalizados. Para obtener más información, consulte Configurar el orquestrador de pruebas IDT.

- Correcciones de errores menores adicionales.

Versión del conjunto de pruebas:

`GGV2Q_2.3.0`

- Publicado el 11 de enero de 2022

IDT v4.4.1 para AWS IoT Greengrass

Notas de la versión:

- Le permite validar y calificar los dispositivos que ejecutan el software AWS IoT Greengrass Core v2.5.2.
- Añade compatibilidad con el uso de una función de IAM definida por el usuario como función de intercambio de fichas que el dispositivo objeto de prueba asume al interactuar con los recursos. AWS

  Puede especificar la función de IAM en el archivo. `userdata.json` Si especificas una función personalizada, IDT utilizará esa función en lugar de crear la función de intercambio de fichas predeterminada durante la ejecución de la prueba.
- Correcciones de errores menores adicionales.

Versión del conjunto de pruebas:

`GGV2Q_2.2.1`

- Publicado el 12 de diciembre de 2021

IDT v4.4.0 para AWS IoT Greengrass

Notas de la versión:

- Le permite validar y calificar los dispositivos que ejecutan el software AWS IoT Greengrass Core v2.5.0.
- Añade soporte para validar y calificar los dispositivos que ejecutan el software AWS IoT Greengrass Core en Windows.
- Admite el uso de la validación de claves públicas para las conexiones de dispositivos Secure Shell (SSH).
- Mejora la política de IAM de permisos de IDT con las mejores prácticas de seguridad.
- Correcciones de errores menores adicionales.

Versión del conjunto de pruebas:

`GGV2Q_2.1.0`

- Publicado el 19 de noviembre de 2021

IDT v4.2.0 para AWS IoT Greengrass

Notas de la versión:

- Incluye soporte para la calificación de las siguientes funciones en dispositivos que ejecutan el software AWS IoT Greengrass Core v2.2.0 y versiones posteriores:
  - Docker: valida que los dispositivos puedan descargar una imagen de contenedor de Docker desde Amazon Elastic Container Registry (Amazon ECR).
  - Aprendizaje automático: valida que los dispositivos puedan realizar inferencias de aprendizaje automático (ML) mediante los marcos Deep Learning Runtime o Lite ML. TensorFlow
  - Stream Manager: valida que los dispositivos puedan descargar, instalar y ejecutar el administrador de transmisiones. AWS IoT Greengrass
- Le permite validar y calificar los dispositivos que ejecutan el software AWS IoT Greengrass Core v2.4.0, v2.3.0, v2.2.0 y v2.1.0.
- *Agrupa los registros de prueba de cada caso de prueba en una carpeta < > independiente dentro del directorio. test-case-id <device-tester-extract-location>*/results/*<execution-id>*/logs/*<test-group-id>*
- Correcciones de errores menores adicionales.

Versión del conjunto de pruebas:

`GGV2Q_2.0.1`

- Publicado el 31 de agosto de 2021

IDT v4.1.0 para AWS IoT Greengrass

Notas de la versión:

- Le permite validar y calificar los dispositivos que ejecutan el software AWS IoT Greengrass Core v2.3.0, v2.2.0, v2.1.0 y v2.0.5.
- Mejora la `userdata.json` configuración al eliminar el requisito de especificar las propiedades y. `GreengrassNucleusVersion` `GreengrassCLIVersion`
- Incluye compatibilidad con la calificación de funciones Lambda y MQTT para el software AWS IoT Greengrass Core v2.1.0 y versiones posteriores. Ahora puede usar IDT para AWS IoT Greengrass V2 para validar que su dispositivo principal puede ejecutar funciones de Lambda y que el dispositivo puede publicar temas de MQTT y suscribirse AWS IoT Core a ellos.
- Mejora las capacidades de registro.

- Correcciones de errores menores adicionales.

Versión del conjunto de pruebas:

`GGV2Q_1.1.1`

- Publicado el 18 de junio de 2021

IDT v4.0.2 para AWS IoT Greengrass

Notas de la versión:

- Le permite validar y calificar los dispositivos que ejecutan el software AWS IoT Greengrass Core v2.1.0.
- Añade compatibilidad con la calificación de funciones Lambda y MQTT para el software AWS IoT Greengrass Core v2.1.0 y versiones posteriores. Ahora puede usar IDT para AWS IoT Greengrass V2 para validar que su dispositivo principal puede ejecutar funciones de Lambda y que el dispositivo puede publicar temas de MQTT y suscribirse AWS IoT Core a ellos.
- Mejora las capacidades de registro.
- Correcciones de errores menores adicionales.

Versión del conjunto de pruebas:

`GGV2Q_1.1.1`

- Publicado el 05 de mayo de 2021

IDT v4.0.1 para AWS IoT Greengrass

Notas de la versión:

- Le permite validar y calificar los dispositivos que ejecutan el software de la AWS IoT Greengrass versión 2.
- Le permite desarrollar y ejecutar sus conjuntos de pruebas personalizados utilizando AWS IoT Device Tester for AWS IoT Greengrass. Para obtener más información, consulte Uso de IDT para desarrollar y ejecutar sus propios conjuntos de pruebas.
- Proporciona aplicaciones IDT con firma de código para macOS y Windows. En macOS, es posible que tengas que conceder una excepción de seguridad para IDT. Para obtener más información, consulte Excepción de seguridad en macOS.

Versión del conjunto de pruebas:

`GGV2Q_1.0.0`

- Publicado el 22 de diciembre de 2020

- El conjunto de pruebas solo ejecuta las pruebas obligatorias para la calificación, a menos que se establezca la correspondiente `value` en la `features` matriz en. `yes`

Descargar IDT para V2 AWS IoT Greengrass

En este tema se describen las opciones de descarga AWS IoT Device Tester para la AWS IoT Greengrass versión 2. Puede utilizar uno de los siguientes enlaces de descarga de software o seguir las instrucciones para descargar IDT mediante programación.

Temas

Al descargar el software, aceptas el contrato AWS IoT Device Tester de licencia.

> **Note**
>
> IDT no admite la ejecución por parte de varios usuarios desde una ubicación compartida, como un directorio NFS o una carpeta compartida de red de Windows. Le recomendamos que extraiga el paquete IDT en una unidad local y ejecute el binario IDT en su estación de trabajo local.

Descarga de IDT manualmente

En este tema se enumeran las versiones compatibles de IDT para AWS IoT Greengrass V2. Como práctica recomendada, le recomendamos que utilice la última versión de IDT para AWS IoT Greengrass V2 que sea compatible con la versión de destino de la AWS IoT Greengrass V2. Las nuevas versiones de AWS IoT Greengrass pueden requerir la descarga de una nueva versión de IDT para AWS IoT Greengrass V2. Cuando inicie una prueba, recibirá una notificación si IDT para AWS IoT Greengrass V2 no es compatible con la versión AWS IoT Greengrass que está utilizando.

IDT v4.9.4 para AWS IoT Greengrass

Versiones compatibles: AWS IoT Greengrass

- Núcleo Greengrass v2.12.0, v2.11.0, v2.10.0 y v2.9.5

Descargas del software IDT:

- IDT v4.9.4 con el conjunto de pruebas GGV2Q_2.5.4 para Linux
- IDT v4.9.4 con el conjunto de pruebas GGV2Q_2.5.4 para macOS
- IDT v4.9.4 con el conjunto de pruebas GGV2Q_2.5.4 para Windows

Notas de la versión:

- Permite la validación y calificación de dispositivos que ejecutan las versiones de software AWS IoT Greengrass Core 2.12.0, 2.11.0, 2.10.0 y 2.9.5.
- Elimina los grupos de pruebas del administrador de transmisiones y el aprendizaje automático.

Notas adicionales:

- Si su dispositivo usa un HSM y usted usa nucleus 2.10.x, migre a Greengrass nucleus versión 2.11.0 o posterior.

Versión del conjunto de pruebas:

`GGV2Q_2.5.4`

- Publicado el 03 de mayo de 2022

Descarga de IDT mediante programación

IDT proporciona una operación de API que puede utilizar para recuperar una URL desde la que descargar IDT mediante programación. También puede usar esta operación de API para comprobar si tiene la última versión de IDT. Esta operación de API tiene el siguiente punto de conexión.

```
https://download.devicetester.iotdevicesecosystem.amazonaws.com/latestidt
```

Para llamar a esta operación de API, debe tener el permiso para realizar la acción **`iot-device-tester:LatestIdt`**. Incluya su AWS firma y utilícela `iot-device-tester` como nombre del servicio.

Solicitud de API

HostOs — El sistema operativo de la máquina host. Puede elegir entre las siguientes opciones:

- `mac`
- `linux`
- `windows`

TestSuiteType — El tipo de conjunto de pruebas. Elija la opción siguiente:

`GGV2`— IDT para V2 AWS IoT Greengrass

ProductVersion

(Opcional) La versión del núcleo de Greengrass. El servicio devuelve la última versión compatible de IDT para esa versión del núcleo de Greengrass. Si no especifica esta opción, el servicio devuelve la última versión de IDT.

Respuesta de la API

La respuesta de la API tiene el siguiente formato. `DownloadURL` incluye un archivo 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
    }
 }
```

Ejemplos

Puede hacer referencia a los siguientes ejemplos para descargar IDT mediante programación. En estos ejemplos se utilizan las credenciales que almacena en las variables de entorno `AWS_ACCESS_KEY_ID` y `AWS_SECRET_ACCESS_KEY`. Para seguir las mejores prácticas recomendadas, no almacene las credenciales en el código.

Example Ejemplo: descarga con cURL 7.75.0 o posterior (Mac y Linux)

Si tiene la versión 7.75.0 o posterior de cURL, puede usar el indicador `aws-sigv4` para firmar la solicitud de API. En este ejemplo se usa jq para analizar la URL de descarga de la respuesta.

> **Warning**
>
> El `aws-sigv4` indicador requiere que los parámetros de consulta de la solicitud CURL GET estén en el orden de o. HostOs/ProductVersion/TestSuiteType HostOs/TestSuiteType Los

órdenes que no se ajusten, provocarán un error al obtener firmas no coincidentes para la cadena canónica de la puerta de enlace de la API.
Si ProductVersion se incluye el parámetro opcional, debe utilizar una versión de producto compatible, tal como se indica en la sección Versiones compatibles de la AWS IoT Device Tester versión AWS IoT Greengrass 2.

- Sustituya *us-west-2* por su. Región de AWS Para obtener la lista de códigos de región, consulte Puntos de conexión regionales.
- Sustituya *linux* por el sistema operativo de su máquina host.
- Sustituya la *2.5.3* por su versión de nucleus. 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 Ejemplo: descarga con una versión anterior de cURL (Mac y Linux)

Puede usar el siguiente comando cURL con una AWS firma que firme y calcule. Para obtener más información sobre cómo firmar y calcular una AWS firma, consulta Firmar solicitudes de AWS API.

- Sustituya *linux* por el sistema operativo de su máquina host.
- Sustituya *Timestamp* por la fecha y la hora, por ejemplo, **20220210T004606Z**.
- Sustituya *Date* por la fecha, por ejemplo, **20220210**.
- *AWSRegion*Sustitúyala por tu Región de AWS. Para obtener la lista de códigos de región, consulte Puntos de conexión regionales.
- *AWSSignature*Sustitúyala por la AWS firma que generes.

```
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 Ejemplo: descarga mediante un script de Python

En este ejemplo se utiliza la biblioteca de solicitudes de Python. Este ejemplo está adaptado del ejemplo de Python para firmar una solicitud de AWS API en la Referencia AWS general.

- Sustituya *us-west-2* por su región. Para obtener la lista de códigos de región, consulte Puntos de conexión regionales.
- Sustituya *linux* por el sistema operativo de su máquina host.

```
# Copyright 2010-2022 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
#License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

# See: http://docs.aws.amazon.com/general/latest/gr/sigv4_signing.html
# This version makes a GET request and passes the signature
# in the Authorization header.
import sys, os, base64, datetime, hashlib, hmac
import requests # pip install requests
# ************* REQUEST VALUES *************
method = 'GET'
service = 'iot-device-tester'
host = 'download.devicetester.iotdevicesecosystem.amazonaws.com'
region = 'us-west-2'
endpoint = 'https://download.devicetester.iotdevicesecosystem.amazonaws.com/latestidt'
request_parameters = 'HostOs=linux&TestSuiteType=GGV2'

# Key derivation functions. See:
# http://docs.aws.amazon.com/general/latest/gr/signature-v4-examples.html#signature-v4-
examples-python
def sign(key, msg):
```

```
    return hmac.new(key, msg.encode('utf-8'), hashlib.sha256).digest()

def getSignatureKey(key, dateStamp, regionName, serviceName):
    kDate = sign(('AWS4' + key).encode('utf-8'), dateStamp)
    kRegion = sign(kDate, regionName)
    kService = sign(kRegion, serviceName)
    kSigning = sign(kService, 'aws4_request')
    return kSigning

# Read AWS access key from env. variables or configuration file. Best practice is NOT
# to embed credentials in code.
access_key = os.environ.get('AWS_ACCESS_KEY_ID')
secret_key = os.environ.get('AWS_SECRET_ACCESS_KEY')
if access_key is None or secret_key is None:
    print('No access key is available.')
    sys.exit()

# Create a date for headers and the credential string
t = datetime.datetime.utcnow()
amzdate = t.strftime('%Y%m%dT%H%M%SZ')
datestamp = t.strftime('%Y%m%d') # Date w/o time, used in credential scope

# ************* TASK 1: CREATE A CANONICAL REQUEST *************
# http://docs.aws.amazon.com/general/latest/gr/sigv4-create-canonical-request.html
# Step 1 is to define the verb (GET, POST, etc.)--already done.
# Step 2: Create canonical URI--the part of the URI from domain to query
# string (use '/' if no path)
canonical_uri = '/latestidt'
# Step 3: Create the canonical query string. In this example (a GET request),
# request parameters are in the query string. Query string values must
# be URL-encoded (space=%20). The parameters must be sorted by name.
# For this example, the query string is pre-formatted in the request_parameters
 variable.
canonical_querystring = request_parameters
# Step 4: Create the canonical headers and signed headers. Header names
# must be trimmed and lowercase, and sorted in code point order from
# low to high. Note that there is a trailing \n.
canonical_headers = 'host:' + host + '\n' + 'x-amz-date:' + amzdate + '\n'
# Step 5: Create the list of signed headers. This lists the headers
# in the canonical_headers list, delimited with ";" and in alpha order.
# Note: The request can include any headers; canonical_headers and
# signed_headers lists those that you want to be included in the
# hash of the request. "Host" and "x-amz-date" are always required.
signed_headers = 'host;x-amz-date'
```

```
# Step 6: Create payload hash (hash of the request body content). For GET
# requests, the payload is an empty string ("").
payload_hash = hashlib.sha256(('').encode('utf-8')).hexdigest()
# Step 7: Combine elements to create canonical request
canonical_request = method + '\n' + canonical_uri + '\n' + canonical_querystring + '\n'
 + canonical_headers + '\n' + signed_headers + '\n' + payload_hash

# ************* TASK 2: CREATE THE STRING TO SIGN*************
# Match the algorithm to the hashing algorithm you use, either SHA-1 or
# SHA-256 (recommended)
algorithm = 'AWS4-HMAC-SHA256'
credential_scope = datestamp + '/' + region + '/' + service + '/' + 'aws4_request'
string_to_sign = algorithm + '\n' +  amzdate + '\n' +  credential_scope + '\n' +
 hashlib.sha256(canonical_request.encode('utf-8')).hexdigest()
# ************* TASK 3: CALCULATE THE SIGNATURE *************
# Create the signing key using the function defined above.
signing_key = getSignatureKey(secret_key, datestamp, region, service)
# Sign the string_to_sign using the signing_key
signature = hmac.new(signing_key, (string_to_sign).encode('utf-8'),
 hashlib.sha256).hexdigest()

# ************* TASK 4: ADD SIGNING INFORMATION TO THE REQUEST *************
# The signing information can be either in a query string value or in
# a header named Authorization. This code shows how to use a header.
# Create authorization header and add to request headers
authorization_header = algorithm + ' ' + 'Credential=' + access_key + '/' +
 credential_scope + ', ' +  'SignedHeaders=' + signed_headers + ', ' + 'Signature=' +
 signature
# The request can include any headers, but MUST include "host", "x-amz-date",
# and (for this scenario) "Authorization". "host" and "x-amz-date" must
# be included in the canonical_headers and signed_headers, as noted
# earlier. Order here is not significant.
# Python note: The 'host' header is added automatically by the Python 'requests'
 library.
headers = {'x-amz-date':amzdate, 'Authorization':authorization_header}

# ************* SEND THE REQUEST *************
request_url = endpoint + '?' + canonical_querystring
print('\nBEGIN REQUEST++++++++++++++++++++++++++++++++++++')
print('Request URL = ' + request_url)
response = requests.get(request_url, headers=headers)
print('\nRESPONSE++++++++++++++++++++++++++++++++++++')
print('Response code: %d\n' % response.status_code)
print(response.text)
```

```
download_url = response.json()["LatestBk"]["DownloadURL"]
r = requests.get(download_url)
open('devicetester.zip', 'wb').write(r.content)
```

Utilice IDT para ejecutar el conjunto de AWS IoT Greengrass cualificaciones

Puede utilizar la AWS IoT Greengrass versión 2 AWS IoT Device Tester para comprobar que el software AWS IoT Greengrass principal se ejecuta en su hardware y se puede comunicar con ella. Nube de AWS También realiza end-to-end pruebas con AWS IoT Core. Por ejemplo, verifica que el dispositivo pueda implementar componentes y actualizarlos.

Además de probar los dispositivos, IDT for AWS IoT Greengrass V2 crea recursos (por ejemplo, AWS IoT cosas, grupos, etc.) Cuenta de AWS para facilitar el proceso de calificación.

Para crear estos recursos, IDT for AWS IoT Greengrass V2 utiliza las AWS credenciales configuradas en el `config.json` archivo para realizar llamadas a la API en su nombre. Estos recursos se aprovisionarán en distintos momentos durante una prueba.

Cuando se utiliza IDT para AWS IoT Greengrass V2 para ejecutar el paquete de AWS IoT Greengrass cualificación, éste lleva a cabo los siguientes pasos:

1. Carga y valida su dispositivo y la configuración de credenciales.
2. Realiza pruebas seleccionadas con los recursos locales y de la nube necesarios.
3. Depura los recursos locales y de la nube.
4. Genera informes de pruebas que indican si la placa supera las pruebas necesarias para la cualificación.

Versiones del conjunto de pruebas

IDT para AWS IoT Greengrass V2 organiza las pruebas en conjuntos de pruebas y grupos de pruebas.

- Un conjunto de pruebas es el conjunto de grupos de pruebas que se utiliza para verificar que un dispositivo funciona con versiones particulares de AWS IoT Greengrass.

- Un grupo de pruebas es el conjunto de pruebas individuales relacionadas con una característica en particular, como la implementación de componentes.

Los conjuntos de pruebas se versionan mediante un *major*.*minor*.*patch* formato, por ejemplo. `GGV2Q_1.0.0` Al descargar IDT, el paquete incluye la última versión del paquete de calificación de Greengrass.

> **Important**
>
> Las pruebas de versiones del conjunto de pruebas no compatibles no son válidas para la cualificación del dispositivo. IDT no imprime informes de cualificación para versiones no compatibles. Para obtener más información, consulte the section called “Política de soporte AWS IoT Device Tester para AWS IoT Greengrass”.
> Puede ejecutar `list-supported-products` una lista de las versiones AWS IoT Greengrass y los conjuntos de pruebas compatibles con su versión actual de IDT.

Descripciones de los grupos de pruebas

Grupos de pruebas necesarias para la cualificación del núcleo

Estos grupos de prueba son necesarios para que su dispositivo AWS IoT Greengrass V2 pueda incluirse en el catálogo de AWS Partner dispositivos.

Dependencias principales

Valida que el dispositivo cumple con todos los requisitos de software y hardware del software AWS IoT Greengrass principal. Este grupo de pruebas incluye el siguiente caso de prueba:

Versión Java

Comprueba que la versión de Java requerida esté instalada en el dispositivo que se está probando. AWS IoT Greengrass requiere Java 8 o una versión posterior.

PreTest Validación

Comprueba que el dispositivo cumple los requisitos de software para ejecutar las pruebas.

- En el caso de los dispositivos basados en Linux, esta prueba comprueba si el dispositivo puede ejecutar los siguientes comandos de Linux:

  `chmod`, `cp`, `echo`, `grep`, `kill`, `ln`, `mkinfo`, `ps`, `rm`, `sh`, `uname`

- En el caso de los dispositivos basados en Windows, esta prueba comprueba si el dispositivo tiene instalado el siguiente software de Microsoft:

  Utilidad Powershell v5.1 o posterior, .NET v4.6.1 o posterior, Visual C++ 2017 o posterior PsExec

Comprobador de versiones

Comprueba que la versión AWS IoT Greengrass proporcionada es compatible con la versión de AWS IoT Device Tester que está utilizando.

Componente

Valida que el dispositivo pueda implementar componentes y actualizarlos. Este grupo de pruebas incluye las siguientes pruebas:

Componente de nube

Valida la capacidad del dispositivo para los componentes de la nube.

Componente local

Valida la capacidad del dispositivo para los componentes locales.

Lambda

Esta prueba no se aplica a los dispositivos basados en Windows.

Valida que el dispositivo pueda implementar componentes de funciones de Lambda que usen el tiempo de ejecución de Java y que las funciones de Lambda puedan AWS IoT Core usar temas de MQTT como fuentes de eventos para los mensajes de trabajo.

MQTT

Valida que el dispositivo pueda suscribirse a temas de MQTT y publicarlos en ellos. AWS IoT Core

Grupos de pruebas opcionales

> **Note**
>
> Estos grupos de prueba son opcionales y se utilizan únicamente para los dispositivos principales Greengrass basados en Linux que reúnan los requisitos. Si decide optar a las pruebas opcionales, su dispositivo aparece con capacidades adicionales en el catálogo de dispositivos. AWS Partner

Dependencias de Docker

Valida que el dispositivo cumpla con todas las dependencias técnicas necesarias para utilizar el componente Docker application AWS manager () proporcionado. `aws.greengrass.DockerApplicationManager`

Cualificación de Docker Application Manager

Valida que el dispositivo pueda descargar una imagen de contenedor de Docker desde Amazon ECR.

Dependencias de Machine Learning

> **Note**
>
> El grupo de prueba opcional de aprendizaje automático solo es compatible con la versión 4.9.3 de IDT.

Valida que el dispositivo cumpla con todas las dependencias técnicas necesarias para utilizar los componentes de aprendizaje automático AWS(ML) proporcionados.

Pruebas de inferencia de Machine Learning

> **Note**
>
> El grupo de pruebas opcional de aprendizaje automático solo es compatible con la versión 4.9.3 de IDT.

Valida que el dispositivo pueda realizar inferencias de aprendizaje automático mediante los marcos Deep Learning Runtime y Lite ML. TensorFlow

Dependencias de Stream Manager

> **Note**
>
> El grupo de pruebas opcional del administrador de transmisiones solo es compatible con la versión 4.9.3 de IDT.

Valida que el dispositivo pueda descargar, instalar y ejecutar el administrador de transmisiones.AWS IoT Greengrass

Integración de la seguridad por hardware (HSI)

> **Note**
>
> Esta prueba está disponible en la versión 4.9.3 de IDT y versiones posteriores únicamente para dispositivos basados en Linux. AWS IoT Greengrass actualmente no admite la integración de seguridad de hardware para dispositivos Windows.

Valida que el dispositivo pueda autenticar las conexiones a los AWS IoT AWS IoT Greengrass servicios mediante una clave privada y un certificado almacenados en un módulo de seguridad de hardware (HSM). Esta prueba también verifica que el componente del proveedor PKCS #11 AWS proporcionado pueda interactuar con el HSM mediante una biblioteca PKCS #11 proporcionada por el proveedor. Para obtener más información, consulte Integración de la seguridad de hardware.

Requisitos previos para ejecutar el paquete de AWS IoT Greengrass calificación

En esta sección se describen los requisitos previos para usar AWS IoT Device Tester (IDT) para. AWS IoT Greengrass

Descargue la última versión de para AWS IoT Device TesterAWS IoT Greengrass

Descargue la última versión de IDT y extraiga el software en una ubicación (*< device-tester-extract-location >*) del sistema de archivos en la que tenga permisos de lectura/escritura.

> **Note**
>
> IDT no admite la ejecución por parte de varios usuarios desde una ubicación compartida, como un directorio NFS o una carpeta compartida de red de Windows. Le recomendamos que extraiga el paquete IDT en una unidad local y ejecute el binario IDT en su estación de trabajo local.

Windows tiene una limitación de longitud de ruta de 260 caracteres. Si utiliza Windows, extraiga IDT en un directorio raíz como `C:\` o `D:\` para mantener las rutas por debajo del límite de 260 caracteres.

Descargue el software AWS IoT Greengrass

IDT for AWS IoT Greengrass V2 comprueba la compatibilidad de su dispositivo con una versión específica de AWS IoT Greengrass. Ejecute el siguiente comando para descargar el software AWS IoT Greengrass principal a un archivo denominado`aws.greengrass.nucleus.zip`. Sustituya *la versión* de IDT por una versión compatible con el componente Nucleus.

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
```

Coloque el `aws.greengrass.nucleus.zip` archivo descargado en la *<device-tester-extract-location>*`/products/` carpeta.

> **Note**
>
> No coloque varios archivos en este directorio para el mismo sistema operativo y arquitectura.

Cree y configure un Cuenta de AWS

Antes de poder utilizarla AWS IoT Device Tester para la AWS IoT Greengrass V2, debe realizar los siguientes pasos:

1. Configure un Cuenta de AWS. Si ya tiene una Cuenta de AWS, vaya al paso 2.
2. Configurar permisos de IDT.

Estos permisos de cuenta permiten a IDT acceder a los AWS servicios y crear AWS recursos, como AWS IoT cosas y AWS IoT Greengrass componentes, en su nombre.

Para crear estos recursos, IDT para AWS IoT Greengrass V2 utiliza las AWS credenciales configuradas en el `config.json` archivo para realizar llamadas a la API en su nombre. Estos recursos se aprovisionarán en distintos momentos durante una prueba.

> **Note**
>
> Si bien la mayoría de las pruebas cumplen los requisitos para el nivel AWS gratuito, debes proporcionar una tarjeta de crédito al inscribirte en un Cuenta de AWS. Para obtener más información, consulte ¿Por qué necesito un método de pago si mi cuenta está cubierta por la capa gratuita?.

Paso 1: Configura un Cuenta de AWS

En este paso, cree y configure una Cuenta de AWS. Si ya tiene una Cuenta de AWS, vaya directamente a the section called “Paso 2: Configurar los permisos de IDT”.

Si no tiene uno Cuenta de AWS, complete los siguientes pasos para crearlo.

Para suscribirte a una Cuenta de AWS

1. Abra https://portal.aws.amazon.com/billing/signup.
2. Siga las instrucciones que se le indiquen.

   Parte del procedimiento de registro consiste en recibir una llamada telefónica e indicar un código de verificación en el teclado del teléfono.

   Cuando te registras en un Cuenta de AWS, Usuario raíz de la cuenta de AWSse crea un. El usuario raíz tendrá acceso a todos los Servicios de AWS y recursos de esa cuenta. Como práctica recomendada de seguridad, asigne acceso administrativo a un usuario y utilice únicamente el usuario raíz para realizar tareas que requieren acceso de usuario raíz.

Para crear un usuario administrador, elija una de las siguientes opciones.

Elegir una forma de administrar el administrador	Para	Haga esto	También puede
En IAM Identity Center (recomendado)	Usar credenciales a corto plazo para acceder a AWS. Esto se ajusta a las prácticas recomendadas de seguridad. Para obtener información sobre las prácticas recomendadas, consulte Prácticas recomendadas de seguridad en IAM en la Guía del usuario de IAM.	Siga las instrucciones en Introducción en la Guía del usuario de AWS IAM Identity Center .	Configure el acceso programático configurando el AWS CLI que se utilizará AWS IAM Identity Center en la Guía del AWS Command Line Interface usuario.
En IAM (no recomendado)	Usar credenciales a largo plazo para acceder a AWS.	Siga las instrucciones en Creación del primer grupo de usuarios y usuario de administrador de IAM en la Guía del usuario de IAM.	Configurar el acceso programático mediante Administración de las claves de acceso de los usuarios de IAM en la Guía del usuario de IAM.

Paso 2: Configurar los permisos de IDT

En este paso, configure los permisos que IDT for AWS IoT Greengrass V2 utiliza para ejecutar pruebas y recopilar datos de uso de IDT. Puede usar AWS Management Consoleo AWS Command

Line Interface (AWS CLI) para crear una política de IAM y un usuario de prueba para IDT y, a continuación, adjuntar políticas al usuario. Si ya ha creado un usuario de prueba para IDT, vaya a. Configuración de su dispositivo para ejecutar pruebas de IDT

Configuración de permisos de IDT (consola)

1. Inicie sesión en la consola de IAM.
2. Crear una política administrada que conceda permisos para crear roles con permisos específicos.

   a. En el panel de navegación, seleccione Políticas y, a continuación, Crear política.

   b. Si no lo está utilizando PreInstalled, en la pestaña JSON, sustituya el contenido del marcador de posición por la siguiente política. Si lo está utilizando PreInstalled, continúe con el siguiente paso.

```
{
    "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. Si lo está utilizando PreInstalled, en la pestaña JSON, sustituya el contenido del marcador de posición por la siguiente política. Asegúrese de:

- Sustituya *ThingName* y *ThingGroup* en la `iotResources` declaración por el nombre y el grupo de cosas que se crearon durante la instalación de Greengrass en el dispositivo en prueba (DUT) para añadir permisos.
- Sustituya los *PassRole* y *RoleAlias* de `roleAliasResources` la declaración y la declaración por `passRoleForResources` los roles que se crearon durante la instalación de Greengrass en su DUT.

```
{
    "Version":"2012-10-17",
    "Statement":[
    {
      "Sid":"passRoleForResources",
      "Effect":"Allow",
      "Action":"iam:PassRole",
      "Resource":"arn:aws:iam::*:role/passRole",
      "Condition":{
        "StringEquals":{
          "iam:PassedToService":[
            "iot.amazonaws.com",
            "lambda.amazonaws.com",
            "greengrass.amazonaws.com"
          ]
        }
      }
    },
    {
      "Sid":"lambdaResources",
      "Effect":"Allow",
      "Action":[
        "lambda:CreateFunction",
        "lambda:PublishVersion",
        "lambda:DeleteFunction",
        "lambda:GetFunction"
      ],
      "Resource":[
        "arn:aws:lambda:*:*:function:idt-*"
      ]
    },
    {
      "Sid":"iotResources",
```

```
      "Effect":"Allow",
      "Action":[
        "iot:CreateThing",
        "iot:DeleteThing",
        "iot:DescribeThing",
        "iot:CreateThingGroup",
        "iot:DeleteThingGroup",
        "iot:DescribeThingGroup",
        "iot:AddThingToThingGroup",
        "iot:RemoveThingFromThingGroup",
        "iot:AttachThingPrincipal",
        "iot:DetachThingPrincipal",
        "iot:UpdateCertificate",
        "iot:DeleteCertificate",
        "iot:CreatePolicy",
        "iot:AttachPolicy",
        "iot:DetachPolicy",
        "iot:DeletePolicy",
        "iot:GetPolicy",
        "iot:Publish",
        "iot:TagResource",
        "iot:ListThingPrincipals",
        "iot:ListAttachedPolicies",
        "iot:ListTargetsForPolicy",
        "iot:ListThingGroupsForThing",
        "iot:ListThingsInThingGroup",
        "iot:CreateJob",
        "iot:DescribeJob",
        "iot:DescribeJobExecution",
        "iot:CancelJob"
      ],
      "Resource":[
        "arn:aws:iot:*:*:thing/thingName",
        "arn:aws:iot:*:*:thinggroup/thingGroup",
        "arn:aws:iot:*:*:policy/idt-*",
        "arn:aws:iot:*:*:cert/*",
        "arn:aws:iot:*:*:topic/idt-*",
        "arn:aws:iot:*:*:job/*"
      ]
    },
    {
      "Sid":"s3Resources",
      "Effect":"Allow",
      "Action":[
```

```
        "s3:GetObject",
        "s3:PutObject",
        "s3:DeleteObjectVersion",
        "s3:DeleteObject",
        "s3:CreateBucket",
        "s3:ListBucket",
        "s3:ListBucketVersions",
        "s3:DeleteBucket",
        "s3:PutObjectTagging",
        "s3:PutBucketTagging"
      ],
      "Resource":"arn:aws:s3::*:idt-*"
    },
    {
      "Sid":"roleAliasResources",
      "Effect":"Allow",
      "Action":[
        "iot:CreateRoleAlias",
        "iot:DescribeRoleAlias",
        "iot:DeleteRoleAlias",
        "iot:TagResource",
        "iam:GetRole"
      ],
      "Resource":[
        "arn:aws:iot:*:*:rolealias/roleAlias",
        "arn:aws:iam::*:role/idt-*"
      ]
    },
    {
      "Sid":"idtExecuteAndCollectMetrics",
      "Effect":"Allow",
      "Action":[
        "iot-device-tester:SendMetrics",
        "iot-device-tester:SupportedVersion",
        "iot-device-tester:LatestIdt",
        "iot-device-tester:CheckVersion",
        "iot-device-tester:DownloadTestSuite"
      ],
      "Resource":"*"
    },
    {
      "Sid":"genericResources",
      "Effect":"Allow",
      "Action":[
```

```
        "greengrass:*",
        "iot:GetThingShadow",
        "iot:UpdateThingShadow",
        "iot:ListThings",
        "iot:DescribeEndpoint",
        "iot:CreateKeysAndCertificate"
      ],
      "Resource":"*"
    },
    {
      "Sid":"iamResourcesUpdate",
      "Effect":"Allow",
      "Action":[
        "iam:CreateRole",
        "iam:DeleteRole",
        "iam:CreatePolicy",
        "iam:DeletePolicy",
        "iam:AttachRolePolicy",
        "iam:DetachRolePolicy",
        "iam:TagRole",
        "iam:TagPolicy",
        "iam:GetPolicy",
        "iam:ListAttachedRolePolicies",
        "iam:ListEntitiesForPolicy"
      ],
      "Resource":[
        "arn:aws:iam::*:role/idt-*",
        "arn:aws:iam::*:policy/idt-*"
      ]
    }
  ]
}
```

> **Note**
>
> Si desea utilizar una función de IAM personalizada como función de intercambio de fichas para el dispositivo que se está probando, asegúrese de actualizar la `roleAliasResources` declaración y la declaración de su política para permitir el recurso `passRoleForResources` de su función de IAM personalizada.

d. Elija Revisar política.

e. En Nombre, ingrese **`IDTGreengrassIAMPermissions`**. En Summary (Resumen), revise los permisos concedidos por la política.

f. Elija Crear política.

3. Cree un usuario de IAM y adjunte los permisos requeridos por IDT para AWS IoT Greengrass.

   a. Cree un usuario de IAM. Siga los pasos del 1 al 5 en Creación de usuarios de IAM (consola) en la Guía del usuario de IAM.

   b. Adjunte los permisos a su usuario de IAM:

      i. En la página Set permissions (Establecer permisos), elija Attach existing policies to user directly (Adjuntar políticas existentes al usuario directamente).

      ii. Busque la política IDTGreengrassIAMPermissions que ha creado en el paso anterior. Seleccione la casilla de verificación.

   c. Elija Siguiente: etiquetas.

   d. Elija Next: Review (Siguiente: revisar) para ver un resumen de sus opciones.

   e. Seleccione la opción Crear un usuario.

   f. Para ver las claves de acceso del usuario (ID de clave de acceso y claves de acceso secretas), elija Show (Mostrar) junto a la contraseña y la clave de acceso. Para guardar las claves de acceso, elija Download.csv (Descargar archivo .csv) y, a continuación, guarde el archivo en un lugar seguro. Utilice esta información más adelante para configurar su archivo de credenciales de AWS .

4. Siguiente paso: Configure su dispositivo físico.

Configuración de permisos de IDT (AWS CLI)

1. En tu ordenador, instala y configura el AWS CLI si aún no está instalado. Siga los pasos que se indican en Instalación de la AWS CLI en la Guía del usuario de la AWS Command Line Interface .

   > **Note**
   >
   > AWS CLI Se trata de una herramienta de código abierto que puede utilizar para interactuar con los AWS servicios desde el shell de la línea de comandos.

2. Cree una política administrada por el cliente que conceda permisos para administrar IDT y roles de AWS IoT Greengrass .

a. Si no la está utilizando PreInstalled, abra un editor de texto y guarde el siguiente contenido de la política en un archivo JSON. Si lo está utilizando PreInstalled, continúe con el siguiente paso.

```
{
    "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. Si lo está utilizando PreInstalled, abra un editor de texto y guarde el siguiente contenido de la política en un archivo JSON. Asegúrese de:

- Sustituya *`ThingName`* y *`ThingGroup`* en la `iotResources` declaración que se creó durante la instalación de Greengrass en su dispositivo bajo prueba (DUT) para añadir permisos.
- Sustituya los *`PassRole`* y *`RoleAlias`* de `roleAliasResources` la declaración y la declaración por `passRoleForResources` los roles que se crearon durante la instalación de Greengrass en su DUT.

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

```
    {
      "Sid":"passRoleForResources",
      "Effect":"Allow",
      "Action":"iam:PassRole",
      "Resource":"arn:aws:iam::*:role/passRole",
      "Condition":{
        "StringEquals":{
          "iam:PassedToService":[
            "iot.amazonaws.com",
            "lambda.amazonaws.com",
            "greengrass.amazonaws.com"
          ]
        }
      }
    },
    {
      "Sid":"lambdaResources",
      "Effect":"Allow",
      "Action":[
        "lambda:CreateFunction",
        "lambda:PublishVersion",
        "lambda:DeleteFunction",
        "lambda:GetFunction"
      ],
      "Resource":[
        "arn:aws:lambda:*:*:function:idt-*"
      ]
    },
    {
      "Sid":"iotResources",
      "Effect":"Allow",
      "Action":[
        "iot:CreateThing",
        "iot:DeleteThing",
        "iot:DescribeThing",
        "iot:CreateThingGroup",
        "iot:DeleteThingGroup",
        "iot:DescribeThingGroup",
        "iot:AddThingToThingGroup",
        "iot:RemoveThingFromThingGroup",
        "iot:AttachThingPrincipal",
        "iot:DetachThingPrincipal",
        "iot:UpdateCertificate",
        "iot:DeleteCertificate",
```

```
        "iot:CreatePolicy",
        "iot:AttachPolicy",
        "iot:DetachPolicy",
        "iot:DeletePolicy",
        "iot:GetPolicy",
        "iot:Publish",
        "iot:TagResource",
        "iot:ListThingPrincipals",
        "iot:ListAttachedPolicies",
        "iot:ListTargetsForPolicy",
        "iot:ListThingGroupsForThing",
        "iot:ListThingsInThingGroup",
        "iot:CreateJob",
        "iot:DescribeJob",
        "iot:DescribeJobExecution",
        "iot:CancelJob"
      ],
      "Resource":[
        "arn:aws:iot:*:*:thing/thingName",
        "arn:aws:iot:*:*:thinggroup/thingGroup",
        "arn:aws:iot:*:*:policy/idt-*",
        "arn:aws:iot:*:*:cert/*",
        "arn:aws:iot:*:*:topic/idt-*",
        "arn:aws:iot:*:*:job/*"
      ]
    },
    {
      "Sid":"s3Resources",
      "Effect":"Allow",
      "Action":[
        "s3:GetObject",
        "s3:PutObject",
        "s3:DeleteObjectVersion",
        "s3:DeleteObject",
        "s3:CreateBucket",
        "s3:ListBucket",
        "s3:ListBucketVersions",
        "s3:DeleteBucket",
        "s3:PutObjectTagging",
        "s3:PutBucketTagging"
      ],
      "Resource":"arn:aws:s3::*:idt-*"
    },
    {
```

```
      "Sid":"roleAliasResources",
      "Effect":"Allow",
      "Action":[
        "iot:CreateRoleAlias",
        "iot:DescribeRoleAlias",
        "iot:DeleteRoleAlias",
        "iot:TagResource",
        "iam:GetRole"
      ],
      "Resource":[
        "arn:aws:iot:*:*:rolealias/roleAlias",
        "arn:aws:iam::*:role/idt-*"
      ]
    },
    {
      "Sid":"idtExecuteAndCollectMetrics",
      "Effect":"Allow",
      "Action":[
        "iot-device-tester:SendMetrics",
        "iot-device-tester:SupportedVersion",
        "iot-device-tester:LatestIdt",
        "iot-device-tester:CheckVersion",
        "iot-device-tester:DownloadTestSuite"
      ],
      "Resource":"*"
    },
    {
      "Sid":"genericResources",
      "Effect":"Allow",
      "Action":[
        "greengrass:*",
        "iot:GetThingShadow",
        "iot:UpdateThingShadow",
        "iot:ListThings",
        "iot:DescribeEndpoint",
        "iot:CreateKeysAndCertificate"
      ],
      "Resource":"*"
    },
    {
      "Sid":"iamResourcesUpdate",
      "Effect":"Allow",
      "Action":[
        "iam:CreateRole",
```

```
        "iam:DeleteRole",
        "iam:CreatePolicy",
        "iam:DeletePolicy",
        "iam:AttachRolePolicy",
        "iam:DetachRolePolicy",
        "iam:TagRole",
        "iam:TagPolicy",
        "iam:GetPolicy",
        "iam:ListAttachedRolePolicies",
        "iam:ListEntitiesForPolicy"
      ],
      "Resource":[
        "arn:aws:iam::*:role/idt-*",
        "arn:aws:iam::*:policy/idt-*"
      ]
    }
  ]
}
```

> **Note**
>
> Si desea utilizar una función de IAM personalizada como función de intercambio de fichas para el dispositivo que se está probando, asegúrese de actualizar la `roleAliasResources` declaración y la declaración de su política para permitir el recurso `passRoleForResources` de su función de IAM personalizada.

c. Ejecuta el siguiente comando para crear una política gestionada por el cliente llamada. `IDTGreengrassIAMPermissions` *`policy.json`*Sustitúyala por la ruta completa al archivo JSON que creaste en el paso anterior.

```
aws iam create-policy --policy-name IDTGreengrassIAMPermissions --policy-
document file://policy.json
```

3. Cree un usuario de IAM y adjunte los permisos requeridos por IDT para AWS IoT Greengrass.

   a. Cree un usuario de IAM. En esta configuración de ejemplo, el usuario se denomina `IDTGreengrassUser`.

```
aws iam create-user --user-name IDTGreengrassUser
```

b. Asocie la política `IDTGreengrassIAMPermissions` que creó en el paso 2 a su usuario de IAM. Sustituya <account-id>el comando por el ID de su Cuenta de AWS.

```
aws iam attach-user-policy --user-name IDTGreengrassUser --policy-arn
 arn:aws:iam::<account-id>:policy/IDTGreengrassIAMPermissions
```

4. Cree una clave de acceso secreta para el usuario.

```
aws iam create-access-key --user-name IDTGreengrassUser
```

Almacene la salida en una ubicación segura. Utilizará esta información más adelante para configurar el archivo de AWS credenciales.

5. Siguiente paso: Configure su dispositivo físico.

AWS IoT Device Tester permisos

Las siguientes políticas describen AWS IoT Device Tester los permisos.

AWS IoT Device Tester requiere estos permisos para las funciones de comprobación de versiones y actualización automática.

- `iot-device-tester:SupportedVersion`

  Otorga AWS IoT Device Tester permiso para obtener la lista de productos, conjuntos de pruebas y versiones de IDT compatibles.

- `iot-device-tester:LatestIdt`

  Otorga AWS IoT Device Tester permiso para obtener la última versión de IDT disponible para su descarga.

- `iot-device-tester:CheckVersion`

  Otorga AWS IoT Device Tester permiso para comprobar la compatibilidad de las versiones de IDT, los conjuntos de pruebas y los productos.

- `iot-device-tester:DownloadTestSuite`

  Otorga AWS IoT Device Tester permiso para descargar las actualizaciones de los conjuntos de pruebas.

AWS IoT Device Tester también utiliza el siguiente permiso para la elaboración de informes de métricas opcionales:

- `iot-device-tester:SendMetrics`

  Otorga permiso AWS para recopilar métricas sobre el uso AWS IoT Device Tester interno. Si se omite este permiso, no se recopilarán estas métricas.

Configuración de su dispositivo para ejecutar pruebas de IDT

Para permitir que IDT ejecute pruebas de calificación del dispositivo, debe configurar el ordenador host para acceder al dispositivo y configurar los permisos de usuario en el dispositivo.

Instale Java en la computadora host

A partir de la versión 4.2.0 de IDT, las pruebas de calificación opcionales AWS IoT Greengrass requieren la ejecución de Java.

Puede utilizar la versión 8 o superior de Java. Le recomendamos que utilice las versiones de soporte a largo plazo de Amazon Corretto u OpenJDK. Se requiere la versión 8 o superior.

Configuración del equipo host para acceder a un dispositivo en pruebas

IDT se ejecuta en su equipo host y debe poder utilizar SSH para conectarse a su dispositivo. Existen dos opciones para permitir que IDT obtenga acceso SSH a los dispositivos sometidos a la prueba:

1. Siga las instrucciones que se indican aquí para crear un par de claves SSH y autorizar su clave para iniciar sesión en su dispositivo en proceso de prueba sin especificar una contraseña.
2. Proporcione un nombre de usuario y una contraseña para cada dispositivo en el archivo `device.json`. Para obtener más información, consulte Configurar device.json.

Puede utilizar cualquier implementación SSL para crear una clave SSH. Las siguientes instrucciones muestran cómo utilizar SSH-KEYGEN o PuTTYgen (para Windows). Si utiliza otra implementación de SSL, consulte la documentación de dicha aplicación.

IDT utiliza claves SSH para autenticar con su dispositivo bajo prueba.

Para crear una clave SSH con SSH-KEYGEN, realice el siguiente procedimiento:

1. Cree una clave de SSH.

Puede utilizar el comando ssh-keygen de Open SSH para crear un par de claves SSH. Si ya tiene un par de claves SSH en su equipo host, es una práctica recomendada crear un par de claves SSH específicamente para IDT. De esta forma, una vez completadas las pruebas, el equipo host ya no podrá conectarse a su dispositivo sin introducir una contraseña. También le permite restringir el acceso al dispositivo remoto únicamente a aquellos que lo necesiten.

> **Note**
>
> Windows no tiene instalado un cliente SSH. Para obtener más información sobre la instalación de un cliente SSH en Windows, consulte Download SSH Client Software.

El comando ssh-keygen le solicita un nombre y la ruta para almacenar el par de claves. De forma predeterminada, los archivos de par de claves se denominan `id_rsa` (clave privada) y `id_rsa.pub` (clave pública). En macOS y Linux, la ubicación predeterminada de estos archivos es `~/.ssh/`. En Windows, la ubicación predeterminada es `C:\Users\`*`<user-name>`*`\.ssh`.

Cuando se le solicite, introduzca una frase clave para proteger la clave SSH. Para obtener más información, consulte la sección acerca de cómo generar una nueva clave SSH.

2. Añada claves SSH autorizadas a su dispositivo en proceso de prueba.

   IDT debe utilizar su clave privada de SSH para iniciar sesión en el dispositivo a prueba. Para autorizar que su clave privada de SSH inicie sesión en el dispositivo a prueba, use el comando ssh-copy-id en su equipo host. Este comando añade su clave pública al archivo `~/.ssh/authorized_keys` que se encuentra en su dispositivo a prueba. Por ejemplo:

   $ ssh-copy-id ***<remote-ssh-user>***@***<remote-device-ip>***

   ¿Dónde *`remote-ssh-user`*está el nombre de usuario utilizado para iniciar sesión en el dispositivo que *`remote-device-ip`*se está probando y la dirección IP del dispositivo que se está probando? Por ejemplo:

   ssh-copy-id pi@192.168.1.5

   Cuando se le solicite, introduzca la contraseña para el nombre de usuario que ha especificado en el comando ssh-copy-id.

ssh-copy-id presupone que la clave pública se denomina `id_rsa.pub` y se almacena en la ubicación predeterminada (en macOS y Linux, `~/.ssh/` y en Windows, `C:\Users\<user-name>\.ssh`). Si asignó a la clave pública un nombre diferente o la almacenó en otra ubicación, debe especificar la ruta completa a su clave pública SSH utilizando la opción -i para ssh-copy-id (por ejemplo: ssh-copy-id -i ~/my/path/myKey.pub). Para obtener más información acerca de la creación de claves de SSH y la copia de las claves públicas, consulte SSH-COPY-ID.

Para crear una clave SSH con PuTTYgen (solo Windows), realice el siguiente procedimiento:

1. Asegúrese de que tiene el servidor y el cliente de OpenSSH instalados en su dispositivo en proceso de prueba. Para obtener más información, consulte OpenSSH.
2. Instale PuTTYgen en su dispositivo en proceso de prueba.
3. Abra PuTTYgen.
4. Elija Generate (Generar) y mueva el cursor del ratón dentro del cuadro para generar una clave privada.
5. En el menú Conversions (Conversiones), elija Export OpenSSH key (Exportar clave OpenSSH) y guarde la clave privada con una extensión de archivo `.pem`.
6. Añada la clave pública al archivo `/home/<user>/.ssh/authorized_keys` en el dispositivo en proceso de prueba.

    a. Copie el texto de la clave pública de la ventana PuTTYgen.

    b. Utilice PuTTY para crear una sesión en su dispositivo en proceso de prueba.

        i. En un símbolo del sistema o en una ventana de Windows PowerShell, ejecute el siguiente comando:

        C:/***<path-to-putty>***/putty.exe -ssh ***<user>***@***<dut-ip-address>***

        ii. Cuando se le solicite, escriba la contraseña de su dispositivo.

        iii. Utilice vi u otro editor de texto para añadir la clave pública al archivo `/home/<user>/.ssh/authorized_keys` en su dispositivo en proceso de prueba.
7. Actualice el archivo `device.json` con su nombre de usuario, la dirección IP y la ruta al archivo de clave privada que acaba de guardar en el equipo host para cada dispositivo en proceso de prueba. Para obtener más información, consulte the section called “Configurar device.json”. Asegúrese de proporcionar la ruta completa y el nombre de archivo a la clave privada y utilizar

barras diagonales ("/"). Por ejemplo, para la ruta de Windows `C:\DT\privatekey.pem`, utilice `C:/DT/privatekey.pem` en el archivo `device.json`.

Configura las credenciales de usuario para los dispositivos Windows

Para calificar un dispositivo basado en Windows, debe configurar las credenciales de usuario en la LocalSystem cuenta del dispositivo que se está probando para los siguientes usuarios:

- El usuario predeterminado de Greengrass ()`ggc_user`.
- El usuario que utilizas para conectarte al dispositivo que se está probando. Este usuario se configura en el `device.json`archivo.

Debe crear cada usuario en la LocalSystem cuenta del dispositivo que se está probando y, a continuación, almacenar el nombre de usuario y la contraseña del usuario en la instancia de Credential Manager de la LocalSystem cuenta.

Para configurar los usuarios en dispositivos Windows

1. Abra la línea de comandos de Windows (`cmd.exe`) como administrador.
2. Cree los usuarios en la LocalSystem cuenta del dispositivo Windows. Ejecute el siguiente comando para cada usuario que desee crear. Para el usuario predeterminado de Greengrass, sustituya el nombre de *`usuario por`*. `ggc_user` Sustituya *`la contraseña`* por una contraseña segura.

   ```
   net user /add user-name password
   ```

3. Descargue e instale la PsExecutilidad de Microsoft en el dispositivo.
4. Utilice la PsExec utilidad para almacenar el nombre de usuario y la contraseña del usuario predeterminado en la instancia de Credential Manager de la LocalSystem cuenta.

   Ejecute el siguiente comando para cada usuario que desee configurar en Credential Manager. Para el usuario predeterminado de Greengrass, sustituya el nombre de *`usuario por`*. `ggc_user` Sustituya la *`contraseña`* por la contraseña del usuario que configuró anteriormente.

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

   Si se PsExec License Agreementabre, Acceptacepte la licencia y ejecute el comando.

> **Note**
>
> En los dispositivos Windows, la LocalSystem cuenta ejecuta el núcleo de Greengrass y debe usar la PsExec utilidad para almacenar la información del usuario en la LocalSystem cuenta. El uso de la aplicación Credential Manager almacena esta información en la cuenta de Windows del usuario que ha iniciado sesión actualmente, en lugar de en la LocalSystem cuenta.

Configuración de los permisos de usuario en el dispositivo

IDT realiza operaciones en diversos directorios y archivos en un dispositivo que se está probando. Algunas de estas operaciones requieren permisos elevados (usando sudo). Para automatizar estas operaciones, IDT para AWS IoT Greengrass V2 debe poder ejecutar comandos con sudo sin que se le pida una contraseña.

Siga estos pasos en el dispositivo a prueba para permitir acceso a sudo sin que se le solicite una contraseña.

> **Note**
>
> `username` hace referencia al usuario de SSH que utiliza IDT para obtener acceso al dispositivo bajo prueba.

Para añadir el usuario al grupo sudo

1. En el dispositivo bajo prueba, ejecute `sudo usermod -aG sudo` *`<username>`*.
2. Cierre la sesión y, a continuación, vuelva a iniciar sesión para que los cambios surtan efecto.
3. Para comprobar que su nombre de usuario se haya añadido correctamente, ejecute sudo echo test. Si no se le solicita una contraseña, el usuario se ha configurado correctamente.
4. Añada el archivo `/etc/sudoers` y agregue la siguiente línea al final del archivo:

   ```
   <ssh-username> ALL=(ALL) NOPASSWD: ALL
   ```

Configure una función de intercambio de fichas personalizada

Puede optar por utilizar una función de IAM personalizada como función de intercambio de fichas que el dispositivo objeto de prueba asume para interactuar con AWS los recursos. Para obtener información sobre la creación de un rol de IAM, consulte Creación de roles de IAM en la Guía del usuario de IAM.

Debe cumplir los siguientes requisitos para permitir que IDT utilice su función de IAM personalizada. Le recomendamos encarecidamente que añada solo las acciones políticas mínimas requeridas a este rol.

- El archivo de configuración userdata.json debe actualizarse para establecer el `GreengrassV2TokenExchangeRole` parámetro en. `true`
- La función de IAM personalizada debe configurarse con la siguiente política de confianza mínima:

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

- El rol de IAM personalizado debe configurarse con la siguiente política de permisos mínimos:

```
{
   "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":"*"
      }
   ]
}
```

- El nombre del rol de IAM personalizado debe coincidir con el recurso del rol de IAM que especifique en los permisos de IAM para el usuario de prueba. De forma predeterminada, la política de usuarios de prueba permite el acceso a los roles de IAM que tienen el `idt-` prefijo en sus nombres de rol. Si el nombre de su rol de IAM no usa este prefijo, añada el `arn:aws:iam::*:role/`*`custom-iam-role-name`* recurso a la `roleAliasResources` declaración y a la declaración a su política de usuarios de prueba, como se muestra en los siguientes ejemplos: `passRoleForResources`

  Example **`passRoleForResources`** instrucción

```
{
   "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`** instrucción

```
{
   "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"
   ]
}
```

Configurar el dispositivo para probar características opcionales

En esta sección se describen los requisitos del dispositivo para ejecutar pruebas de IDT para las funciones opcionales de Docker y aprendizaje automático (ML). Las funciones de aprendizaje automático solo son compatibles con la versión 4.9.3 de IDT. Debe asegurarse de que su dispositivo cumpla estos requisitos solo si desea probar estas funciones. De lo contrario, siga en the section called “Configuración de los ajustes de IDT”.

Temas

Requisitos de cualificación de Docker

IDT for AWS IoT Greengrass V2 ofrece pruebas de calificación de Docker para validar que sus dispositivos pueden usar el componente de administrador de aplicaciones AWS de Docker suministrado para descargar imágenes de contenedores de Docker que usted puede ejecutar con componentes de contenedores de Docker personalizados. Para obtener información sobre la creación de componentes Docker personalizados, consulte. Ejecute un contenedor Docker

Para ejecutar las pruebas de calificación de Docker, los dispositivos que se estén probando deben cumplir los siguientes requisitos para implementar el componente de administrador de aplicaciones de Docker.

- Docker Engine 1.9.1 o posterior instalado en el dispositivo principal de Greengrass. Se ha comprobado que la versión 20.10 es la última versión que funciona con el software Core. AWS IoT Greengrass Debe instalar Docker directamente en el dispositivo principal antes de implementar los componentes que ejecutan contenedores de Docker.
- El daemon de Docker se inició y se ejecutó en el dispositivo principal antes de implementar este componente.
- El usuario del sistema que ejecute un componente contenedor de Docker debe tener permisos de raíz o administrador, o bien debe configurar Docker para que se ejecute como un usuario no de raíz o no administrador.
  - En los dispositivos Linux, puede añadir un usuario al `docker` grupo para `docker` ejecutar comandos sin él. `sudo`
  - En los dispositivos Windows, puede añadir un usuario al `docker-users` grupo para que invoque `docker` comandos sin privilegios de administrador.

  Linux or Unix

  Para añadir `ggc_user` al `docker` grupo el usuario no root que utiliza para ejecutar los componentes del contenedor de Docker, ejecute el siguiente comando.

  ```
  sudo usermod -aG docker ggc_user
  ```

  Para obtener más información, consulta Administrar Docker como usuario no root.

  Windows Command Prompt (CMD)

  Para añadir `ggc_user` al `docker-users` grupo o el usuario que utiliza para ejecutar los componentes del contenedor de Docker, ejecute el siguiente comando como administrador.

```
net localgroup docker-users ggc_user /add
```

Windows PowerShell

Para añadir `ggc_user` al `docker-users` grupo o al usuario que utiliza para ejecutar los componentes del contenedor de Docker, ejecute el siguiente comando como administrador.

```
Add-LocalGroupMember -Group docker-users -Member ggc_user
```

Requisitos de calificación de ML

> **Note**
>
> La función de aprendizaje automático solo es compatible con la versión 4.9.3 de IDT.

IDT for AWS IoT Greengrass V2 ofrece pruebas de calificación de aprendizaje automático para validar que sus dispositivos puedan usar los componentes de aprendizaje automático AWS proporcionados para realizar inferencias de aprendizaje automático de forma local mediante los marcos Deep Learning Runtime o Lite ML. TensorFlow Para obtener más información sobre la ejecución de inferencias de aprendizaje automático en dispositivos Greengrass, consulte. Cómo realizar la inferencia de machine learning

Para ejecutar las pruebas de calificación de aprendizaje automático, los dispositivos que se estén probando deben cumplir los siguientes requisitos para implementar los componentes de aprendizaje automático.

- En los dispositivos principales de Greengrass que ejecutan Amazon Linux 2 o Ubuntu 18.04, se instala en el dispositivo la versión 2.27 o posterior de la Biblioteca C GNU (glibc).
- En los dispositivos ARMv7L, como Raspberry Pi, las dependencias para OpenCV-Python están instaladas en el dispositivo. Ejecute el siguiente comando para instalar las dependencias.

  ```
  sudo apt-get install libopenjp2-7 libilmbase23 libopenexr-dev libavcodec-dev
   libavformat-dev libswscale-dev libv4l-dev libgtk-3-0 libwebp-dev
  ```
- Los dispositivos Raspberry Pi que ejecutan el sistema operativo Bullseye de Raspberry Pi deben cumplir los siguientes requisitos:

- NumPy 1.22.4 o una versión posterior instalada en el dispositivo. Raspberry Pi OS Bullseye incluye una versión anterior de NumPy, por lo que puede ejecutar el siguiente comando para actualizar NumPy el dispositivo.

  ```
  pip3 install --upgrade numpy
  ```

- La pila de cámaras antigua habilitada en el dispositivo. El Raspberry Pi OS Bullseye incluye una nueva pila de cámaras que está habilitada de forma predeterminada y no es compatible, por lo que debes activar la pila de cámaras antigua.

  Para activar la pila de cámaras antigua

  1. Ejecute el siguiente comando para abrir la herramienta de configuración de Raspberry Pi.

     ```
     sudo raspi-config
     ```

  2. Seleccione Opciones de interfaz.
  3. Seleccione Cámara antigua para activar la pila de cámaras antiguas.
  4. Reinicie el Raspberry Pi.

Requisitos de calificación de HSM

AWS IoT Greengrass proporciona el componente PKCS #11 para integrarlo con el módulo de seguridad de hardware (HSM) del PKCS del dispositivo. La configuración del HSM depende del dispositivo y del módulo HSM que haya elegido. Siempre que se proporcione la configuración de HSM esperada, tal como se documenta en los ajustes de configuración de IDT, IDT dispondrá de la información necesaria para realizar esta prueba de calificación de funciones opcional.

Configure los ajustes de IDT para ejecutar el conjunto de AWS IoT Greengrass cualificación

Antes de ejecutar las pruebas, debe configurar los ajustes de las AWS credenciales y los dispositivos del ordenador host.

Configure AWS las credenciales en config.json

Debe configurar sus credenciales de usuario de IAM en el archivo *<device_tester_extract_location>*`/configs/config.json`. Utilice las credenciales

del usuario de IDT para AWS IoT Greengrass V2 creado enthe section called “Cree y configure un Cuenta de AWS”. Puede especificar sus credenciales de una de las dos formas siguientes:

- En un archivo de credenciales
- Como variables de entorno.

Configure AWS las credenciales con un archivo de credenciales

IDT utiliza el mismo archivo de credenciales que la AWS CLI. Para obtener más información, consulte Archivos de configuración y credenciales.

La ubicación del archivo de credenciales varía en función del sistema operativo que utilice:

- macOS, Linux: `~/.aws/credentials`
- Windows: `C:\Users\UserName\.aws\credentials`

Añada sus AWS credenciales al `credentials` archivo en el siguiente formato:

```
[default]
aws_access_key_id = <your_access_key_id>
aws_secret_access_key = <your_secret_access_key>
```

Para configurar IDT para AWS IoT Greengrass V2 de modo que utilice AWS las credenciales del `credentials` archivo, edítelo `config.json` de la siguiente manera:

```
{
  "awsRegion": "region",
  "auth": {
    "method": "file",
    "credentials": {
      "profile": "default"
    }
  }
}
```

> **Note**
>
> Si no usa el `default` AWS perfil, asegúrese de cambiar el nombre del perfil en el `config.json` archivo. Para obtener más información, consulte Perfiles con nombre.

Configure AWS las credenciales con variables de entorno

Las variables de entorno son las variables que mantiene el sistema operativo y utilizan los comandos del sistema. No se guardan si cierra la sesión de SSH. IDT para AWS IoT Greengrass V2 puede usar las variables de `AWS_SECRET_ACCESS_KEY` entorno `AWS_ACCESS_KEY_ID` y para almacenar sus AWS credenciales.

Para establecer estas variables en Linux, MacOS, o Unix, utilice export:

```
export AWS_ACCESS_KEY_ID=<your_access_key_id>
export AWS_SECRET_ACCESS_KEY=<your_secret_access_key>
```

Para establecer estas variables en Windows, utilice set:

```
set AWS_ACCESS_KEY_ID=<your_access_key_id>
set AWS_SECRET_ACCESS_KEY=<your_secret_access_key>
```

Para configurar IDT para utilizar las variables de entorno, edite la sección `auth` de su archivo `config.json`. A continuación se muestra un ejemplo:

```
{
  "awsRegion": "region",
  "auth": {
    "method": "environment"
  }
}
```

Configurar device.json

> **Note**
>
> La versión 4.9.3 de IDT permite probar las funciones `ml` y`docker`. `streamManagement` La versión 4.9.4 de IDT y las versiones posteriores admiten las pruebas. `docker` Si no quiere probar estas funciones, defina el valor correspondiente en. no

Además de AWS las credenciales, IDT para AWS IoT Greengrass V2 necesita información sobre los dispositivos en los que se ejecutan las pruebas. Algunos ejemplos de información serían la dirección IP, la información de inicio de sesión, el sistema operativo y la arquitectura de la CPU.

Debe proporcionar esta información utilizando la plantilla `device.json` ubicada en *`<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**
>
> Especifique `privKeyPath` solo si `method` está establecido en `pki`
> Especifique `password` solo si `method` está establecido en `password`

Todas las propiedades que contienen valores son obligatorias, tal y como se describe a continuación:

`id`

Un ID alfanumérico definido por el usuario que identifica de forma única una colección de dispositivos que se conoce como grupo de dispositivos. Los dispositivos que pertenecen a un grupo deben tener idéntico hardware. Al ejecutar un conjunto de pruebas, los dispositivos del grupo se utilizan para paralelizar la carga de trabajo. Se utilizan varios dispositivos para ejecutar diferentes pruebas.

`sku`

Un valor alfanumérico que identifica de forma única el dispositivo a prueba. El SKU se utiliza para realizar un seguimiento de placas cualificadas.

> **Note**
>
> Si quieres incluir tu dispositivo en el catálogo de AWS Partner dispositivos, el SKU que especifiques aquí debe coincidir con el SKU que utilices en el proceso de publicación.

`features`

Una matriz que contenga las características compatibles del dispositivo. Todas las características son obligatorias.

`arch`

Las arquitecturas de sistemas operativos compatibles que valida la prueba. Los valores válidos son:

- `x86_64`
- `armv6l`
- `armv7l`
- `aarch64`

`ml`

Valida que el dispositivo cumpla con todas las dependencias técnicas necesarias para utilizar los componentes de aprendizaje automático AWS(ML) proporcionados.

Al habilitar esta función, también se valida que el dispositivo pueda realizar inferencias de aprendizaje automático mediante los marcos Deep Learning Runtime y TensorFlow Lite ML.

Los valores válidos son cualquier combinación de `dlr` `ytensorflowlite`, o. no

`docker`

Valida que el dispositivo cumpla con todas las dependencias técnicas necesarias para utilizar el componente Docker application manager () AWS proporcionado. `aws.greengrass.DockerApplicationManager`

Al habilitar esta función, también se valida que el dispositivo pueda descargar una imagen de contenedor de Docker desde Amazon ECR.

Los valores válidos son cualquier combinación de o. `yes` `no`

`streamManagement`

Valida que el dispositivo pueda descargar, instalar y ejecutar el administrador de AWS IoT Greengrass transmisiones.

Los valores válidos son cualquier combinación de `yes` ono.

`hsi`

Valida que el dispositivo pueda autenticar las conexiones a los AWS IoT AWS IoT Greengrass servicios mediante una clave privada y un certificado almacenados en un módulo de seguridad de hardware (HSM). Esta prueba también verifica que el componente del proveedor PKCS #11 AWS proporcionado pueda interactuar con el HSM mediante una biblioteca PKCS #11 proporcionada por el proveedor. Para obtener más información, consulte Integración de la seguridad de hardware.

Los valores válidos son `hsm` o no.

> **Note**
>
> La prueba solo está disponible con la versión 4.9.3 de `hsi` IDT y versiones posteriores.

`devices.id`

Un identificador único y definido por el usuario para el dispositivo que se está probando.

`devices.operatingSystem`

El sistema operativo del dispositivo. Los valores admitidos son `Linux` y `Windows`.

`connectivity.protocol`

El protocolo de comunicación que se usará para la comunicación con este dispositivo. Actualmente, el único valor admitido es `ssh` para dispositivos físicos.

`connectivity.ip`

La dirección IP del dispositivo que se está probando.

Esta propiedad solo se aplica si `connectivity.protocol` está establecido en `ssh`.

`connectivity.port`

Opcional. El número de puerto que se va a utilizar para las conexiones SSH.

El valor predeterminado es 22.

Esta propiedad solo se aplica si `connectivity.protocol` está establecido en `ssh`.

`connectivity.publicKeyPath`

Opcional. La ruta completa a la clave pública utilizada para autenticar las conexiones al dispositivo que se está probando.

Al especificar la `publicKeyPath`, IDT valida la clave pública del dispositivo cuando establece una conexión SSH con el dispositivo que se está probando. Si no se especifica este valor, IDT crea una conexión SSH, pero no valida la clave pública del dispositivo.

Le recomendamos encarecidamente que especifique la ruta a la clave pública y que utilice un método seguro para obtenerla. En el caso de los clientes SSH estándar basados en la línea de comandos, la clave pública se proporciona en el archivo `known_hosts`. Si especifica un archivo de clave pública independiente, este archivo debe usar el mismo formato que el archivo `known_hosts`, es decir, *`ip-address key-type public-key`*. Si hay varias entradas con la misma dirección IP, la entrada del tipo de clave utilizada por IDT debe estar antes que las demás entradas del archivo.

`connectivity.auth`

Información de autenticación para la conexión.

Esta propiedad solo se aplica si `connectivity.protocol` está establecido en `ssh`.

`connectivity.auth.method`

El método de autenticación que se utiliza para acceder a un dispositivo a través de un determinado protocolo de conectividad.

Los valores admitidos son:

- `pki`
- `password`

`connectivity.auth.credentials`

Las credenciales que se utilizan para la autenticación.

`connectivity.auth.credentials.password`

La contraseña que se utiliza para iniciar sesión en el dispositivo que se va a probar.

Este valor solo se aplica si `connectivity.auth.method` está establecido en `password`.

`connectivity.auth.credentials.privKeyPath`

La ruta completa a la clave privada que se utiliza para iniciar sesión en el dispositivo que se está probando.

Este valor solo se aplica si `connectivity.auth.method` está establecido en `pki`.

`connectivity.auth.credentials.user`

El nombre de usuario para iniciar sesión en el dispositivo que se está probando.

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**
>
> Especifique `privKeyPath` solo si `method` está establecido en `pki`
> Especifique `password` solo si `method` está establecido en `password`

Todas las propiedades que contienen valores son obligatorias, tal y como se describe a continuación:

`id`

Un ID alfanumérico definido por el usuario que identifica de forma única una colección de dispositivos que se conoce como grupo de dispositivos. Los dispositivos que pertenecen a un grupo deben tener idéntico hardware. Al ejecutar un conjunto de pruebas, los dispositivos del grupo se utilizan para paralelizar la carga de trabajo. Se utilizan varios dispositivos para ejecutar diferentes pruebas.

`sku`

Un valor alfanumérico que identifica de forma única el dispositivo a prueba. El SKU se utiliza para realizar un seguimiento de placas cualificadas.

> **Note**
>
> Si quieres incluir tu dispositivo en el catálogo de AWS Partner dispositivos, el SKU que especifiques aquí debe coincidir con el SKU que utilices en el proceso de publicación.

`features`

Una matriz que contenga las características compatibles del dispositivo. Todas las características son obligatorias.

`arch`

Las arquitecturas de sistemas operativos compatibles que valida la prueba. Los valores válidos son:

- `x86_64`
- `armv6l`
- `armv7l`
- `aarch64`

`docker`

Valida que el dispositivo cumpla con todas las dependencias técnicas necesarias para utilizar el componente Docker application AWS manager () suministrado. `aws.greengrass.DockerApplicationManager`

Al habilitar esta función, también se valida que el dispositivo pueda descargar una imagen de contenedor de Docker desde Amazon ECR.

Los valores válidos son cualquier combinación de o. `yes` `no`

`hsi`

Valida que el dispositivo pueda autenticar las conexiones a los AWS IoT AWS IoT Greengrass servicios mediante una clave privada y un certificado almacenados en un módulo de seguridad de hardware (HSM). Esta prueba también verifica que el componente del proveedor PKCS #11 AWS proporcionado pueda interactuar con el HSM mediante una biblioteca PKCS #11 proporcionada por el proveedor. Para obtener más información, consulte Integración de la seguridad de hardware.

Los valores válidos son `hsm` o `no`.

> **Note**
>
> La prueba solo está disponible con la versión 4.9.3 de `hsi` IDT y versiones posteriores.

`devices.id`

Un identificador único y definido por el usuario para el dispositivo que se está probando.

`devices.operatingSystem`

El sistema operativo del dispositivo. Los valores admitidos son `Linux` y `Windows`.

`connectivity.protocol`

El protocolo de comunicación que se usará para la comunicación con este dispositivo. Actualmente, el único valor admitido es `ssh` para dispositivos físicos.

`connectivity.ip`

La dirección IP del dispositivo que se está probando.

Esta propiedad solo se aplica si `connectivity.protocol` está establecido en `ssh`.

`connectivity.port`

Opcional. El número de puerto que se va a utilizar para las conexiones SSH.

El valor predeterminado es 22.

Esta propiedad solo se aplica si `connectivity.protocol` está establecido en `ssh`.

`connectivity.publicKeyPath`

Opcional. La ruta completa a la clave pública utilizada para autenticar las conexiones al dispositivo que se está probando.

Al especificar la `publicKeyPath`, IDT valida la clave pública del dispositivo cuando establece una conexión SSH con el dispositivo que se está probando. Si no se especifica este valor, IDT crea una conexión SSH, pero no valida la clave pública del dispositivo.

Le recomendamos encarecidamente que especifique la ruta a la clave pública y que utilice un método seguro para obtenerla. En el caso de los clientes SSH estándar basados en la línea de comandos, la clave pública se proporciona en el archivo `known_hosts`. Si especifica un archivo de clave pública independiente, este archivo debe usar el mismo formato que el

archivo known_hosts, es decir, *ip-address key-type public-key*. Si hay varias entradas con la misma dirección IP, la entrada del tipo de clave utilizada por IDT debe estar antes que las demás entradas del archivo.

`connectivity.auth`

Información de autenticación para la conexión.

Esta propiedad solo se aplica si `connectivity.protocol` está establecido en `ssh`.

`connectivity.auth.method`

El método de autenticación que se utiliza para acceder a un dispositivo a través de un determinado protocolo de conectividad.

Los valores admitidos son:

- `pki`
- `password`

`connectivity.auth.credentials`

Las credenciales que se utilizan para la autenticación.

`connectivity.auth.credentials.password`

La contraseña que se utiliza para iniciar sesión en el dispositivo que se va a probar.

Este valor solo se aplica si `connectivity.auth.method` está establecido en `password`.

`connectivity.auth.credentials.privKeyPath`

La ruta completa a la clave privada que se utiliza para iniciar sesión en el dispositivo que se está probando.

Este valor solo se aplica si `connectivity.auth.method` está establecido en `pki`.

`connectivity.auth.credentials.user`

El nombre de usuario para iniciar sesión en el dispositivo que se está probando.

Configure userdata.json

IDT para AWS IoT Greengrass V2 también necesita información adicional sobre la ubicación de los artefactos y el software de prueba. AWS IoT Greengrass

Debe proporcionar esta información utilizando la plantilla `userdata.json` ubicada en *<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"
    }
}
```

Todas las propiedades que contienen valores son obligatorias, tal y como se describe aquí:

`TempResourcesDirOnDevice`

La ruta completa a una carpeta temporal del dispositivo que se está probando en la que se almacenan los artefactos de prueba. Asegúrese de que no se requieren permisos de sudo para escribir en este directorio.

> **Note**
>
> IDT borra el contenido de esta carpeta cuando termina de ejecutar una prueba.

`InstallationDirRootOnDevice`

La ruta completa a la carpeta del dispositivo en la que se va a instalar. AWS IoT Greengrass En el PreInstalled caso de Greengrass, esta es la ruta al directorio de instalación de Greengrass.

Debe establecer los permisos de archivo necesarios para esta carpeta. Ejecute el siguiente comando para cada carpeta de la ruta de instalación.

```
sudo chmod 755 folder-name
```

`GreengrassNucleusZip`

La ruta completa al archivo ZIP (`greengrass-nucleus-latest.zip`) del núcleo de Greengrass en el equipo host. Este campo no es obligatorio para realizar pruebas con PreInstalled Greengrass.

> **Note**
>
> Para obtener información sobre las versiones compatibles del núcleo de Greengrass para IDT, consulte. AWS IoT GreengrassÚltima versión de IDT para V2 AWS IoT Greengrass Para descargar la versión más reciente del software de Greengrass, consulte Descargar el AWS IoT Greengrass software.

`PreInstalled`

Esta función solo está disponible para IDT v4.5.8 y versiones posteriores en dispositivos Linux.

(Opcional) Si el valor es *sí*, IDT asumirá que la ruta mencionada es el directorio en `InstallationDirRootOnDevice` el que está instalado Greengrass.

Para obtener más información sobre cómo instalar Greengrass en su dispositivo, consulte. Instale el software AWS IoT Greengrass principal con aprovisionamiento automático de recursos Si la instalación se realiza con aprovisionamiento manual, incluya el paso « AWS IoT Añadir el elemento a un grupo de objetos nuevo o existente» al crear un elemento manualmente AWS IoT . IDT supone que la cosa y el grupo de cosas se crean durante la configuración de la instalación. Asegúrese de que estos valores se reflejen en el `effectiveConfig.yaml` archivo. IDT busca el archivo que aparece `effectiveConfig.yaml` debajo`<InstallationDirRootOnDevice>/config/effectiveConfig.yaml`.

Para ejecutar pruebas con HSM, asegúrese de que el `aws.greengrass.crypto.Pkcs11Provider` campo esté actualizado en. `effectiveConfig.yaml`

`GreengrassV2TokenExchangeRole`

(Opcional) La función de IAM personalizada que desea utilizar como función de intercambio de fichas que el dispositivo objeto de prueba asume para interactuar con AWS los recursos.

> **Note**
>
> IDT utiliza esta función de IAM personalizada en lugar de crear la función de intercambio de fichas predeterminada durante la ejecución de la prueba. Si usa un rol personalizado, puede actualizar los permisos de IAM del usuario de prueba para excluir la `iamResourcesUpdate` declaración que permite al usuario crear y eliminar roles y políticas de IAM.

Para obtener más información sobre cómo crear una función de IAM personalizada como función de intercambio de fichas, consulte. Configure una función de intercambio de fichas personalizada

`hsm`

Esta función está disponible para la versión 4.5.1 de IDT y versiones posteriores.

(Opcional) La información de configuración para realizar pruebas con un módulo de seguridad de AWS IoT Greengrass hardware (HSM). De lo contrario, la propiedad `hsm` debe omitirse. Para obtener más información, consulte Integración de la seguridad de hardware.

Esta propiedad solo se aplica si `connectivity.protocol` está establecido en `ssh`.

> **Warning**
>
> La configuración del HSM puede considerarse información confidencial si IDT y otro sistema comparten el módulo de seguridad de hardware. En esta situación, puede evitar proteger estos valores de configuración en texto plano almacenándolos en un AWS parámetro del almacén SecureString de parámetros y configurando IDT para que los recupere durante la ejecución de la prueba. Para obtener más información, consulte ???.

`hsm.greengrassPkcsPluginJar`

La ruta completa al componente del proveedor PKCS #11 que se descarga en la máquina host de IDT. AWS IoT Greengrass proporciona este componente como un archivo JAR que puede descargar para especificarlo como complemento de aprovisionamiento durante la instalación. Puede descargar la última versión del archivo JAR del componente en la siguiente URL: https://d2s8p88vqu9w66.cloudfront.net/releases/Pkcs11Provider/aws.greengrass.crypto.Pkcs11Provider-latest.jar.

`hsm.pkcs11ProviderLibrary`

La ruta completa a la biblioteca PKCS #11 que proporciona el proveedor del módulo de seguridad de hardware (HSM) para interactuar con el HSM.

`hsm.slotId`

El identificador de ranura que se utiliza para identificar la ranura del HSM en la que se cargan la clave y el certificado.

`hsm.slotLabel`

La etiqueta de ranura que se utiliza para identificar la ranura HSM en la que se cargan la clave y el certificado.

`hsm.slotUserPin`

El PIN de usuario que IDT utiliza para autenticar el software AWS IoT Greengrass principal en el HSM.

> **Note**
>
> Como práctica recomendada de seguridad, no utilices el mismo PIN de usuario en los dispositivos de producción.

`hsm.keyLabel`

Es la etiqueta que se utiliza para identificar la clave en el módulo de hardware. Tanto la clave como el certificado deben usar la misma etiqueta de clave.

`hsm.preloadedCertificateArn`

El nombre de recurso de Amazon (ARN) del certificado del dispositivo cargado en la AWS IoT nube.

Debe haber generado previamente este certificado con la clave del HSM, haberlo importado a su HSM y haberlo subido a la nube. AWS IoT Para obtener información sobre cómo generar e importar el certificado, consulte la documentación de su HSM.

Debe cargar el certificado en la misma cuenta y región que proporcionó en config.json. . Para obtener más información sobre cómo cargar el certificado a AWS IoT, consulte Registrar un certificado de cliente manualmente en la Guía para AWS IoT desarrolladores.

`hsm.rootCAPath`

(Opcional) La ruta completa de la máquina host de IDT hasta la entidad emisora de certificados (CA) raíz que firmó el certificado. Esto es obligatorio si el certificado creado en el HSM no está firmado por la CA raíz de Amazon.

Obtenga la configuración del almacén de parámetros AWS

AWS IoT El Device Tester (IDT) incluye una función opcional para obtener los valores de configuración del almacén de parámetros de AWS Systems Manager. AWS El almacén de parámetros permite almacenar las configuraciones de forma segura y cifrada. Cuando está configurado, IDT puede recuperar los parámetros del almacén de AWS parámetros en lugar de almacenarlos en texto plano dentro del archivo. `userdata.json` Esto resulta útil para cualquier dato confidencial que deba almacenarse cifrado, como contraseñas, números PIN y otros datos secretos.

1. Para utilizar esta función, debe actualizar los permisos utilizados al crear su usuario de IDT para permitir la GetParameter acción en los parámetros para los que IDT está configurado. A continuación se muestra un ejemplo de una declaración de permiso que se puede añadir al usuario de IDT. Para obtener más información, consulte la guía del AWS Systems Manager usuario.

   ```
   {
          "Sid":"parameterStoreResources",
          "Effect": "Allow",
           "Action": [
               "ssm:GetParameter"
           ],
           "Resource": "arn:aws:ssm:*:*:parameter/IDT*"
   }
   ```

   El permiso anterior está configurado para permitir la obtención de todos los parámetros cuyo nombre comience por`IDT`, mediante el uso del carácter comodín. * Debe personalizarlo según sus necesidades para que IDT tenga acceso a cualquier parámetro configurado en función del nombre de los parámetros que esté utilizando.

2. Debe almacenar los valores de configuración en AWS Paramater Store. Esto se puede hacer desde la AWS consola o desde la AWS CLI. AWS El almacén de parámetros le permite elegir un almacenamiento cifrado o no cifrado. Para almacenar valores confidenciales, como secretos,

contraseñas y números PIN, debe utilizar la opción cifrada, que es un tipo de parámetro de SecureString. Para cargar un parámetro mediante la AWS CLI, puede utilizar el siguiente comando:

```
aws ssm put-parameter --name IDT-example-name --value IDT-example-value --type
 SecureString
```

Puede comprobar que un parámetro está almacenado mediante el siguiente comando. (Opcional) Utilice la `--with-decryption` marca para obtener un parámetro descifrado. SecureString

```
aws ssm get-parameter --name IDT-example-name
```

El uso de la AWS CLI cargará el parámetro en la AWS región del usuario de CLI actual e IDT obtendrá los parámetros de la región configurada. `config.json` Para comprobar su región desde la AWS CLI, utilice lo siguiente:

```
aws configure get region
```

3. Una vez que tenga un valor de configuración en la AWS nube, puede actualizar cualquier valor de la configuración de IDT para obtenerlo de la AWS nube. Para ello, utilice un marcador de posición en la configuración de IDT del formulario `{{AWS.Parameter.parameter_name}}` para buscar el parámetro con ese nombre en el almacén de parámetros. AWS

   Por ejemplo, supongamos que quiere usar el `IDT-example-name` parámetro del paso 2 como HSM KeyLabel en su configuración de HSM. Para ello, puede actualizar el suyo `userdata.json` de la siguiente manera:

   ```
   "hsm": {
           "keyLabel": "{{AWS.Parameter.IDT-example-name}}",
           [...]
       }
   ```

   IDT obtendrá el valor de este parámetro en tiempo de ejecución que se configuró `IDT-example-value` en el paso 2. Esta configuración es similar a la configuración, `"keyLabel": "IDT-example-value"` pero, en cambio, ese valor se almacena cifrado en la AWS nube.

Ejecute el conjunto de cualificación de AWS IoT Greengrass

Después de configurar los ajustes necesarios, puede iniciar las pruebas. El tiempo de ejecución del conjunto completo de pruebas depende de su hardware. Como referencia, se tarda aproximadamente 30 minutos en completar el conjunto de pruebas completo en una unidad Raspberry Pi 3B.

Usa el siguiente `run-suite` comando para ejecutar un conjunto de pruebas.

```
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
```

Todas las opciones son opcionales. Por ejemplo, puede omitir `pool-id` si solo tiene un grupo de dispositivos, que es un conjunto de dispositivos idénticos, definido en el `device.json` archivo. O bien, puede omitir `suite-id` si desea ejecutar la última versión del conjunto de pruebas en la carpeta `tests`.

> **Note**
>
> IDT le pregunta si está disponible en línea una versión más reciente del conjunto de pruebas. Para obtener más información, consulte the section called “Versiones del conjunto de pruebas”.

Ejemplos de comandos para ejecutar el conjunto de calificaciones

Los siguientes ejemplos de línea de comandos muestran cómo ejecutar las pruebas de calificación de un grupo de dispositivos. Para obtener más información acerca de `run-suite` y otros comandos IDT, consulte the section called “Comandos de IDT”.

Use el siguiente comando para ejecutar todos los grupos de pruebas de un conjunto de pruebas específico. El `list-suites` comando muestra los conjuntos de pruebas que se encuentran en la `tests` carpeta.

```
devicetester_[linux | mac | win]_x86-64 run-suite \
```

```
    --suite-id GGV2Q_1.0.0 \
    --pool-id <pool-id> \
    --userdata userdata.json
```

Use el siguiente comando para ejecutar un grupo de pruebas específico en un conjunto de pruebas. El `list-groups` comando muestra los grupos de pruebas de un conjunto de pruebas.

```
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
```

Use el siguiente comando para ejecutar un caso de prueba específico en un grupo de pruebas.

```
devicetester_[linux | mac | win]_x86-64 run-suite \
    --group-id <group-id> \
    --test-id <test-id> \
    --userdata userdata.json
```

Use el siguiente comando para ejecutar varios casos de prueba en un grupo de pruebas.

```
devicetester_[linux | mac | win]_x86-64 run-suite \
    --group-id <group-id> \
    --test-id <test-id1>,<test-id2>
    --userdata userdata.json
```

Use el siguiente comando para enumerar todos los casos de prueba de un grupo de pruebas.

```
devicetester_[linux | mac | win]_x86-64 list-test-cases --group-id <group-id>
```

Le recomendamos que ejecute el conjunto completo de pruebas de calificación, que ejecuta las dependencias de los grupos de pruebas en el orden correcto. Si decide ejecutar grupos de prueba específicos, le recomendamos que ejecute primero el grupo de prueba del comprobador de dependencias para asegurarse de que todas las dependencias de Greengrass estén instaladas antes de ejecutar los grupos de prueba relacionados. Por ejemplo:

- Ejecute `coredependencies` antes de ejecutar los grupos de prueba de cualificación del núcleo.

Comandos IDT para V2 AWS IoT Greengrass

Los comandos IDT se encuentran en el directorio *<device-tester-extract-location>*/bin. Para ejecutar un conjunto de pruebas, debe proporcionar el comando en el siguiente formato:

`help`

Enumera información acerca del comando especificado.

`list-groups`

Muestra los grupos de un conjunto de prueba determinado.

`list-suites`

Muestra los conjuntos de prueba disponibles.

`list-supported-products`

Enumera los productos compatibles, en este caso las versiones de AWS IoT Greengrass y las versiones del conjunto de pruebas para la versión actual de IDT.

`list-test-cases`

Enumera los casos de prueba en un grupo de prueba determinado. Se admite la siguiente opción:

- `group-id`. El grupo de pruebas que se va a buscar. Esta opción es necesaria y debe especificar un solo grupo.

`run-suite`

Ejecuta un conjunto de pruebas en un grupo de dispositivos. Las siguientes son algunas de las opciones admitidas:

- `suite-id`. La versión del conjunto de pruebas que se va a ejecutar. Si no se especifica, IDT utiliza la versión más reciente de la carpeta `tests`.
- `group-id`. Los grupos de pruebas que se van a ejecutar, como una lista separada por comas. Si no se especifica, IDT ejecuta todos los grupos de pruebas adecuados del conjunto de pruebas en función de los ajustes configurados en`device.json`. IDT no ejecuta ningún grupo de pruebas que el dispositivo no admita en función de los ajustes configurados, incluso si esos grupos de prueba están especificados en la `group-id` lista.
- `test-id`. Los casos de prueba que se van a ejecutar, como una lista separada por comas. Cuando se especifique, `group-id` debe especificar un solo grupo.
- `pool-id`. El grupo de dispositivos que se va a probar. Debe especificar un grupo si tiene varios grupos de dispositivos definidos en el archivo `device.json`.

- `stop-on-first-failure`. Configura IDT para que deje de ejecutarse en el primer fallo. Utilice esta opción `group-id` cuando desee depurar los grupos de prueba especificados. No utilice esta opción cuando ejecute un conjunto de pruebas completo para generar un informe de cualificación.
- `update-idt`. Establece la respuesta a la solicitud de actualización de IDT. La `Y` respuesta detiene la ejecución de la prueba si IDT detecta que hay una versión más reciente. La `N` respuesta continúa con la ejecución de la prueba.
- `userdata`. La ruta completa al `userdata.json` archivo que contiene información sobre las rutas de los artefactos de prueba. Esta opción es necesaria para el `run-suite` comando. *`El userdata.json archivo debe estar ubicado en el directorio devicetester_extract_location /devicetester_ggv2_ [win|mac|linux] / configs/.`*

Para obtener más información acerca de `run-suite` las opciones, utilice la opción `help`:

```
devicetester_[linux | mac | win]_x86-64 run-suite -h
```

Descripción de los resultados y de los registros

En esta sección se describe cómo ver e interpretar registros e informes de resultados de IDT.

Para solucionar errores, consulteSolución de problemas de IDT paraAWS IoT GreengrassV2.

Ver los resultados

Mientras ejecuta, IDT escribe errores en la consola, en archivos de registro y en informes de prueba. Una vez que IDT completa el conjunto de pruebas de cualificación, genera dos informes de prueba. Estos informes se encuentran en*`<device-tester-extract-location>`*`/results/`*`<execution-id>`*`/`. Ambos informes capturan los resultados de la ejecución del conjunto de pruebas de calificación.

`awsiotdevicetester_report.xml`Es el informe de prueba de aptitud al que se enviaAWS para incluir su dispositivo en el catálogo deAWS Partner dispositivos. El informe contiene los componentes siguientes:

- La versión de IDT.
- La versión AWS IoT Greengrass que se ha probado.

- El SKU y el grupo de dispositivos especificado en el archivo `device.json`.
- Las características del grupo de dispositivos especificado en el archivo `device.json`.
- El resumen de agregación de los resultados de las pruebas.
- Un desglose de los resultados de las pruebas por bibliotecas que se probaron en función de las funciones del dispositivo, como el acceso a los recursos locales, la sombra y MQTT.

El informe `GGV2Q_Result.xml` está en formato XML JUnit. Puede integrarlo en plataformas de integración/implementación continua como Jenkins, Bamboo, etc. El informe contiene los componentes siguientes:

- Resumen de agregación de los resultados de pruebas.
- Desglose de resultados de pruebas por funcionalidad de AWS IoT Greengrass probada.

Interpretación deAWS IoT Device Tester resultados

La sección de informe en `awsiotdevicetester_report.xml` o `awsiotdevicetester_report.xml` enumera las pruebas que se ejecutaron y los resultados.

La primera etiqueta XML`<testsuites>` contiene el resumen de la ejecución de la prueba. Por ejemplo:

```
<testsuites name="GGQ results" time="2299" tests="28" failures="0" errors="0"
 disabled="0">
```

Atributos que se utilizan en la etiqueta **`<testsuites>`**

`name`

El nombre del grupo de prueba.

`time`

El tiempo, en segundos, que se ejecutación el conjunto de calificaciones.

`tests`

El número de pruebas que se realizaron.

`failures`

El número de pruebas que se ejecutaron, pero que no se superaron.

`errors`

La cantidad de pruebas que IDT no pudo ejecutar.

`disabled`

Ignore este atributo. No se utiliza.

El archivo `awsiotdevicetester_report.xml` contiene una etiqueta `<awsproduct>` que tiene información sobre el producto que se está probando y las características del producto que se han validado después de ejecutar un conjunto de pruebas.

Atributos que se utilizan en la etiqueta **`<awsproduct>`**

`name`

El nombre del producto que se está probando.

`version`

La versión del producto que se está probando.

`features`

Las características validadas. Las características marcadas como `required` son necesarias para solicitar la cualificación de la placa. En el siguiente fragmento se muestra cómo aparece esta información en el archivo `awsiotdevicetester_report.xml`:

```
<name="aws-iot-greengrass-v2-core" value="supported" type="required"></feature>
```

Si no hay errores de pruebas para las características requeridas, el dispositivo cumple los requisitos técnicos para ejecutar AWS IoT Greengrass y puede interoperar con servicios de AWS IoT. Si quieres incluir tu dispositivo en el catálogo deAWS Partner dispositivos, puedes utilizar este informe como prueba de aptitud.

Si se producen errores en pruebas, puede identificar la prueba fallido revisando las etiquetas XML `<testsuites>`. Las etiquetas XML `<testsuite>` dentro de la etiqueta `<testsuites>` muestran el resumen del resultado de la prueba de un grupo de prueba. Por ejemplo:

```
<testsuite name="combination" package="" tests="1" failures="0" time="161" disabled="0"
 errors="0" skipped="0">
```

El formato es similar a la etiqueta `<testsuites>`, pero con un atributo `skipped` que no se utiliza y que se puede pasar por alto. Dentro de cada etiqueta`<testsuite>` XML, hay`<testcase>` etiquetas para cada prueba que se ejecutó para un grupo de prueba. Por ejemplo:

```
<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>>
```

Atributos que se utilizan en la etiqueta **`<testcase>`**

`name`

El nombre de la prueba.

`attempts`

El número de veces que IDT ejecutó el caso de prueba.

Cuando una prueba genera un error o si se produce un error, las etiquetas `<failure>` o `<error>` se añaden a la etiqueta `<testcase>` con información para la resolución de problemas. Por ejemplo:

```
<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>
```

Visualización de registros de

IDT genera registros a partir de las ejecuciones de pruebas en*`<devicetester-extract-location>`*`/results/`*`<execution-id>`*`/logs`. Se generan dos conjuntos de registros:

`test_manager.log`

Registros generados a partir del componente Test Manager deAWS IoT Device Tester (por ejemplo, registros relacionados con la configuración, la secuenciación de pruebas y la generación de informes).

`<test-case-id>``.log (for example, lambdaDeploymentTest.log)`

Registros del caso de prueba dentro del grupo de prueba, incluidos los registros del dispositivo que se está probando. A partir de IDT v4.2.0, IDT agrupa los registros de pruebas de cada caso

de prueba en una carpeta *<test-case-id >* independiente dentro del*<devicetester-extract-location>*/results/*<execution-id>*/logs/*<test-group-id>/* directorio.

Uso de IDT para desarrollar y ejecutar sus propios conjuntos de pruebas

A partir de la versión 4.0.1 de IDT, IDT para AWS IoT Greengrass V2 combina una configuración y un formato de resultados estandarizados con un entorno de conjuntos de pruebas que le permite desarrollar conjuntos de pruebas personalizados para sus dispositivos y su software. Puede añadir pruebas personalizadas para su propia validación interna o proporcionárselas a sus clientes para que verifiquen los dispositivos.

Utilice IDT para desarrollar y ejecutar conjuntos de pruebas personalizados, de la siguiente manera:

Para desarrollar conjuntos de pruebas personalizados

- Cree conjuntos de pruebas con lógica de prueba personalizada para el dispositivo Greengrass que desee probar.
- Proporcione a IDT sus conjuntos de pruebas personalizados para los corredores de pruebas. Incluya información sobre configuraciones de ajustes específicas de los conjuntos de pruebas.

Para ejecutar conjuntos de pruebas personalizados

- Configure el dispositivo que desea probar.
- Implemente las configuraciones de los ajustes que requieran los conjuntos de pruebas que desee usar.
- Utilice IDT para ejecutar sus conjuntos de pruebas personalizados.
- Vea los resultados de las pruebas y los registros de ejecución de las pruebas realizadas por IDT.

Descargue la última versión de for AWS IoT Device TesterAWS IoT Greengrass

Descargue la última versión de IDT y extraiga el software en una ubicación (*< device-tester-extract-location >*) del sistema de archivos en la que tenga permisos de lectura/escritura.

> **Note**
>
> IDT no admite la ejecución por parte de varios usuarios desde una ubicación compartida, como un directorio NFS o una carpeta compartida de red de Windows. Le recomendamos que extraiga el paquete IDT en una unidad local y ejecute el binario IDT en su estación de trabajo local.
> Windows tiene una limitación de longitud de ruta de 260 caracteres. Si utiliza Windows, extraiga IDT en un directorio raíz como `C:\` o `D:\` para mantener las rutas por debajo del límite de 260 caracteres.

Flujo de trabajo de creación de un conjunto de pruebas

Los conjuntos de pruebas se componen de tres tipos de archivos:

- Archivos de configuración que proporcionan a IDT información sobre cómo ejecutar el conjunto de pruebas.
- Archivos ejecutables de prueba que IDT utiliza para ejecutar casos de prueba.
- Archivos adicionales necesarios para ejecutar las pruebas.

Complete los siguientes pasos básicos para crear pruebas de IDT personalizadas:

1. Cree archivos de configuración para su conjunto de pruebas.
2. Cree ejecutables de casos de prueba que contengan la lógica de prueba de su conjunto de pruebas.
3. Verifique y documente la información de configuración necesaria para que los ejecutores las pruebas ejecuten el conjunto de pruebas.
4. Compruebe que IDT pueda ejecutar su conjunto de pruebas y generar los resultados de las pruebas según lo esperado.

Para crear rápidamente un conjunto personalizado de muestra y ejecutarlo, siga las instrucciones que se indican en Tutorial: crear y ejecutar el ejemplo del conjunto de pruebas de IDT.

Para empezar a crear un conjunto de pruebas personalizado en Python, consulte Tutorial: Desarrollar un conjunto de pruebas de IDT sencillo.

Tutorial: crear y ejecutar el ejemplo del conjunto de pruebas de IDT

La descarga de AWS IoT Device Tester incluye el código fuente de un conjunto de pruebas de muestra. Puedes completar este tutorial para crear y ejecutar el conjunto de pruebas de ejemplo para entender cómo puedes usar IDT AWS IoT Greengrass para ejecutar conjuntos de pruebas personalizados.

En este tutorial, debe completar los siguientes pasos:

1. Creación del conjunto de pruebas de muestra
2. Uso de IDT para ejecutar el conjunto de pruebas de muestra

Requisitos previos

Necesitará lo siguiente para completar este tutorial:

- Requisitos del equipo host
  - Última versión de AWS IoT Device Tester
  - Python 3.7 o posterior

    Para comprobar la versión de Python instalada en el equipo, ejecute el siguiente comando:

    ```
    python3 --version
    ```

    En Windows, si el uso de este comando devuelve un error, utilice `python --version` en su lugar. Si el número de versión devuelto es 3.7 o superior, ejecute el siguiente comando en un terminal de Powershell para establecer `python3` como alias para el comando `python`.

    ```
    Set-Alias -Name "python3" -Value "python"
    ```

    Si no se devuelve información sobre la versión o si la versión es inferior a 3.7, siga las instrucciones que se indican en Descarga de Python) para instalar Python 3.7 o superior. Para obtener más información, consulte la documentación de Python.
  - urllib3

    Para comprobar que `urllib3` se ha instalado correctamente, ejecute el siguiente comando:

```
python3 -c 'import urllib3'
```

Si `urllib3` no está instalado, ejecute el siguiente comando para instalarlo:

```
python3 -m pip install urllib3
```

- Requisitos de los dispositivos
  - Un dispositivo con un sistema operativo Linux y una conexión de red a la misma red que el equipo host.

    Le recomendamos que utilice un Raspberry Pi con el sistema operativo Raspberry Pi. Asegúrese de que tiene configurado SSH en el Raspberry Pi para conectarse de forma remota.

Configuración de la información del dispositivo para IDT

Configure la información del dispositivo para que IDT ejecute la prueba. Debe actualizar la plantilla `device.json` ubicada en la carpeta *<device-tester-extract-location>*`/configs` con la siguiente información.

```
[
  {
    "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>"
            }
          }
        }
```

```
      }
    ]
  }
]
```

En el objeto `devices`, indique la siguiente información:

`id`

Un identificador único definido por el usuario para el dispositivo.

`connectivity.ip`

La dirección IP del dispositivo.

`connectivity.port`

Opcional. El número de puerto que se va a utilizar para las conexiones SSH al dispositivo.

`connectivity.auth`

Información de autenticación para la conexión.

Esta propiedad solo se aplica si `connectivity.protocol` está establecido en `ssh`.

`connectivity.auth.method`

El método de autenticación que se utiliza para acceder a un dispositivo a través de un determinado protocolo de conectividad.

Los valores admitidos son:

- `pki`
- `password`

`connectivity.auth.credentials`

Las credenciales que se utilizan para la autenticación.

`connectivity.auth.credentials.user`

El nombre de usuario utilizado para iniciar sesión en el dispositivo.

`connectivity.auth.credentials.privKeyPath`

La ruta completa a la clave privada que se utiliza para iniciar sesión en el dispositivo.

Este valor solo se aplica si `connectivity.auth.method` está establecido en `pki`.

`devices.connectivity.auth.credentials.password`

La contraseña que se utiliza para iniciar sesión en el dispositivo.

Este valor solo se aplica si `connectivity.auth.method` está establecido en `password`.

> **Note**
>
> Especifique `privKeyPath` solo si `method` está establecido en `pki`
> Especifique `password` solo si `method` está establecido en `password`

Creación del conjunto de pruebas de muestra

La carpeta *<device-tester-extract-location>*`/samples/python` contiene ejemplos de archivos de configuración, código fuente y el SDK de cliente de IDT que puede combinar en un conjunto de pruebas mediante los scripts de creación proporcionados. El siguiente árbol de directorios muestra la ubicación de estos archivos de muestra:

```
<device-tester-extract-location>
### ...
### tests
### samples
#   ### ...
#   ### python
#       ### configuration
#       ### src
#       ### build-scripts
#           ### build.sh
#           ### build.ps1
### sdks
    ### ...
    ### python
        ### idt_client
```

Para crear el conjunto de pruebas, ejecute los siguientes comandos en el equipo host:

Windows

```
cd <device-tester-extract-location>/samples/python/build-scripts
```

```
./build.ps1
```

Linux, macOS, or UNIX

```
cd <device-tester-extract-location>/samples/python/build-scripts
./build.sh
```

Esto crea el conjunto de pruebas de muestra en la carpeta `IDTSampleSuitePython_1.0.0`, dentro de la carpeta *<device-tester-extract-location>*`/tests`. Revise los archivos de la `IDTSampleSuitePython_1.0.0` carpeta para comprender cómo está estructurado el conjunto de pruebas de muestra y para ver varios ejemplos de ejecutables de casos de prueba y archivos JSON de configuración de pruebas.

> **Note**
>
> El conjunto de pruebas de muestra incluye el código fuente de Python. No incluya información confidencial en el código del conjunto de pruebas.

Siguiente paso: Uso de IDT para ejecutar el conjunto de pruebas de muestra que creó.

Uso de IDT para ejecutar el conjunto de pruebas de muestra

Para ejecutar el conjunto de pruebas de muestra, ejecute los siguientes comandos en el equipo host:

```
cd <device-tester-extract-location>/bin
./devicetester_[linux | mac | win_x86-64] run-suite --suite-id IDTSampleSuitePython
```

IDT ejecuta el conjunto de pruebas de muestra y transmite los resultados a la consola. Cuando la prueba termine de ejecutarse, verá la siguiente información:

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

Solución de problemas

Utilice la siguiente información para resolver cualquier problema al completar el tutorial.

El caso de prueba no se ejecuta correctamente

Si la prueba no se ejecuta correctamente, IDT transmite los registros de errores a la consola para ayudarle a solucionar los problemas con la ejecución de la prueba. Asegúrese de que cumple con todos los requisitos previos de este tutorial.

No se puede conectar al dispositivo que se está probando

Compruebe lo siguiente:

- El archivo `device.json` contiene la dirección IP, el puerto y la información de autenticación correctos.
- Puede conectarse a su dispositivo a través de SSH desde su equipo host.

Tutorial: Desarrollar un conjunto de pruebas de IDT sencillo

Un conjunto de pruebas combina lo siguiente:

- Ejecutables de prueba que contienen la lógica de la prueba
- Archivos de configuración que describen el conjunto de pruebas

Este tutorial le muestra cómo usar IDT para AWS IoT Greengrass para desarrollar un conjunto de pruebas de Python que contenga un único caso de prueba. En este tutorial, completará los siguientes pasos:

1. Creación de un directorio del conjunto de pruebas
2. Creación de archivos de configuración

3. Creación del ejecutable del caso de prueba
4. Ejecución del conjunto de pruebas

Requisitos previos

Necesitará lo siguiente para completar este tutorial:

- Requisitos del equipo host
  - Última versión de AWS IoT Device Tester
  - Python 3.7 o posterior

    Para comprobar la versión de Python instalada en el equipo, ejecute el siguiente comando:

    ```
    python3 --version
    ```

    En Windows, si el uso de este comando devuelve un error, utilice `python --version` en su lugar. Si el número de versión devuelto es 3.7 o superior, ejecute el siguiente comando en un terminal de Powershell para establecer `python3` como alias para el comando `python`.

    ```
    Set-Alias -Name "python3" -Value "python"
    ```

    Si no se devuelve información sobre la versión o si la versión es inferior a 3.7, siga las instrucciones que se indican en Descarga de Python) para instalar Python 3.7 o superior. Para obtener más información, consulte la documentación de Python.
  - urllib3

    Para comprobar que `urllib3` se ha instalado correctamente, ejecute el siguiente comando:

    ```
    python3 -c 'import urllib3'
    ```

    Si `urllib3` no está instalado, ejecute el siguiente comando para instalarlo:

    ```
    python3 -m pip install urllib3
    ```
- Requisitos de los dispositivos
  - Un dispositivo con un sistema operativo Linux y una conexión de red a la misma red que el equipo host.

Le recomendamos que utilice un Raspberry Pi con el sistema operativo Raspberry Pi. Asegúrese de que tiene configurado SSH en el Raspberry Pi para conectarse de forma remota.

Creación de un directorio del conjunto de pruebas

IDT separa de forma lógica los casos de prueba en grupos de pruebas dentro de cada conjunto de pruebas. Cada caso de prueba debe estar dentro de un grupo de pruebas. Para este tutorial, cree una carpeta llamada `MyTestSuite_1.0.0` y cree el siguiente árbol de directorios dentro de esta carpeta:

```
MyTestSuite_1.0.0
### suite
    ### myTestGroup
        ### myTestCase
```

Creación de archivos de configuración

El conjunto de pruebas debe contener los siguientes archivos de configuración necesarios:

Archivos de configuración necesarios

`suite.json`

Contiene información sobre el conjunto de pruebas. Consulte Configuración de suite.json.

`group.json`

Contiene información sobre un grupo de pruebas. Debe crear un archivo `group.json` para cada grupo de pruebas del conjunto de pruebas. Consulte Configuración de group.json.

`test.json`

Contiene información sobre un caso de prueba. Debe crear un archivo `test.json` para cada caso de prueba de su conjunto de pruebas. Consulte Configuración de test.json.

1. En la carpeta `MyTestSuite_1.0.0/suite`, cree un archivo `suite.json` con la estructura siguiente:

   ```
   {
       "id": "MyTestSuite_1.0.0",
   ```

```
    "title": "My Test Suite",
    "details": "This is my test suite.",
    "userDataRequired": false
}
```

2. En la carpeta `MyTestSuite_1.0.0/myTestGroup`, cree un archivo `group.json` con la estructura siguiente:

```
{
    "id": "MyTestGroup",
    "title": "My Test Group",
    "details": "This is my test group.",
    "optional": false
}
```

3. En la carpeta `MyTestSuite_1.0.0/myTestGroup/myTestCase`, cree un archivo `test.json` con la estructura siguiente:

```
{
    "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"
            ]
        }
    }
```

```
}
```

El árbol de directorios de la carpeta `MyTestSuite_1.0.0` debe ser similar al siguiente:

```
MyTestSuite_1.0.0
### suite
    ### suite.json
    ### myTestGroup
        ### group.json
        ### myTestCase
            ### test.json
```

Obtención del SDK de cliente de IDT

Utilice el SDK de cliente de IDT para permitir que IDT interactúe con el dispositivo que se está probando e informe de los resultados de las pruebas. Para este tutorial, utilizará la versión de Python del SDK.

Desde la carpeta *<device-tester-extract-location>*`/sdks/python/`, copie la carpeta `idt_client` a su carpeta `MyTestSuite_1.0.0/suite/myTestGroup/myTestCase`.

Para comprobar que el SDK se ha copiado correctamente, ejecute el siguiente comando.

```
cd MyTestSuite_1.0.0/suite/myTestGroup/myTestCase
python3 -c 'import idt_client'
```

Creación del ejecutable del caso de prueba

Los ejecutables del caso de prueba contienen la lógica de prueba que desea ejecutar. Un conjunto de pruebas puede contener varios ejecutables de casos de prueba. Para este tutorial, creará solo un ejecutable de caso de prueba.

1. Cree el archivo del conjunto de pruebas.

   En la carpeta `MyTestSuite_1.0.0/suite/myTestGroup/myTestCase`, cree un archivo `myTestCase.py` con el siguiente contenido:

   ```
   from idt_client import *

   def main():
   ```

```
    # Use the client SDK to communicate with IDT
    client = Client()

if __name__ == "__main__":
    main()
```

2. Utilice las funciones del SDK del cliente para añadir la siguiente lógica de prueba al archivo `myTestCase.py`:

   a. Ejecute un comando SSH en el dispositivo que se está probando.

```
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. Envíe el resultado de la prueba a IDT.

```
from idt_client import *

def main():
    # Use the client SDK to communicate with IDT
    client = Client()

    # Create an execute on device request
    exec_req = ExecuteOnDeviceRequest(ExecuteOnDeviceCommand("echo 'hello
 world'"))

    # Run the command
```

```
    exec_resp = client.execute_on_device(exec_req)

    # Print the standard output
    print(exec_resp.stdout)

    # Create a send result request
    sr_req = SendResultRequest(TestResult(passed=True))

    # Send the result
    client.send_result(sr_req)

if __name__ == "__main__":
    main()
```

Configuración de la información del dispositivo para IDT

Configure la información del dispositivo para que IDT ejecute la prueba. Debe actualizar la plantilla `device.json` ubicada en la carpeta *<device-tester-extract-location>*`/configs` con la siguiente información.

```
[
  {
    "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>"
            }
          }
        }
      }
    ]
```

```
  }
]
```

En el objeto `devices`, indique la siguiente información:

`id`

Un identificador único definido por el usuario para el dispositivo.

`connectivity.ip`

La dirección IP del dispositivo.

`connectivity.port`

Opcional. El número de puerto que se va a utilizar para las conexiones SSH al dispositivo.

`connectivity.auth`

Información de autenticación para la conexión.

Esta propiedad solo se aplica si `connectivity.protocol` está establecido en `ssh`.

`connectivity.auth.method`

El método de autenticación que se utiliza para acceder a un dispositivo a través de un determinado protocolo de conectividad.

Los valores admitidos son:

- `pki`
- `password`

`connectivity.auth.credentials`

Las credenciales que se utilizan para la autenticación.

`connectivity.auth.credentials.user`

El nombre de usuario utilizado para iniciar sesión en el dispositivo.

`connectivity.auth.credentials.privKeyPath`

La ruta completa a la clave privada que se utiliza para iniciar sesión en el dispositivo.

Este valor solo se aplica si `connectivity.auth.method` está establecido en `pki`.

`devices.connectivity.auth.credentials.password`

La contraseña que se utiliza para iniciar sesión en el dispositivo.

Este valor solo se aplica si `connectivity.auth.method` está establecido en `password`.

> **Note**
>
> Especifique `privKeyPath` solo si `method` está establecido en `pki`
> Especifique `password` solo si `method` está establecido en `password`

Ejecución del conjunto de pruebas

Después de crear el conjunto de pruebas, querrá asegurarse de que funciona como se espera. Siga los pasos que se indican a continuación para ejecutar el conjunto de pruebas con su grupo de dispositivos existente.

1. Copie su carpeta `MyTestSuite_1.0.0` en *<device-tester-extract-location>*`/tests`.
2. Ejecute los comandos siguientes:

```
cd <device-tester-extract-location>/bin
./devicetester_[linux | mac | win_x86-64] run-suite --suite-id MyTestSuite
```

IDT ejecuta el conjunto de pruebas y transmite los resultados a la consola. Cuando la prueba termine de ejecutarse, verá la siguiente información:

```
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
```

Solución de problemas

Utilice la siguiente información para resolver cualquier problema al completar el tutorial.

El caso de prueba no se ejecuta correctamente

Si la prueba no se ejecuta correctamente, IDT transmite los registros de errores a la consola para ayudarle a solucionar los problemas con la ejecución de la prueba. Antes de consultar los registros de errores, compruebe lo siguiente:

- El SDK del cliente IDT se encuentra en la carpeta correcta, tal y como se describe en este paso.
- Cumple todos los requisitos previos de este tutorial.

No se puede conectar al dispositivo que se está probando

Compruebe lo siguiente:

- El archivo `device.json` contiene la dirección IP, el puerto y la información de autenticación correctos.
- Puede conectarse a su dispositivo a través de SSH desde su equipo host.

Creación de archivos de configuración para el conjunto de pruebas de IDT

En esta sección se describen los formatos en los que se crean los archivos de configuración que se incluyen al escribir un conjunto de pruebas personalizado.

Archivos de configuración necesarios

`suite.json`

Contiene información sobre el conjunto de pruebas. Consulte Configuración de suite.json.

`group.json`

Contiene información sobre un grupo de pruebas. Debe crear un archivo `group.json` para cada grupo de pruebas del conjunto de pruebas. Consulte Configuración de group.json.

`test.json`

Contiene información sobre un caso de prueba. Debe crear un archivo `test.json` para cada caso de prueba de su conjunto de pruebas. Consulte Configuración de test.json.

Archivos de configuración opcionales

`test_orchestrator.yaml` o `state_machine.json`

Define cómo se ejecutan las pruebas cuando IDT ejecuta el conjunto de pruebas. Consulte Configuración de test_orchestrator.yaml.

> **Note**
>
> A partir de IDT v4.5.1, se utiliza el `test_orchestrator.yaml` archivo para definir el flujo de trabajo de la prueba. En las versiones anteriores de IDT, se utiliza el archivo `state_machine.json`. Para obtener más información sobre la máquina de estados, consulte Configurar el equipo de estado IDT.

`userdata_schema.json`

Define el esquema del archivo `userdata.json` que los ejecutores de pruebas pueden incluir en su configuración de ajustes. El archivo `userdata.json` se utiliza para cualquier información de configuración adicional necesaria para ejecutar la prueba, pero que no está presente en el archivo `device.json`. Consulte Configuración de userdata_schema.json.

Los archivos de configuración se colocan en su *<custom-test-suite-folder>*, tal y como se muestra aquí.

```
<custom-test-suite-folder>
### suite
    ### suite.json
    ### test_orchestrator.yaml
    ### userdata_schema.json
    ### <test-group-folder>
        ### group.json
        ### <test-case-folder>
            ### test.json
```

Configuración de suite.json

El archivo `suite.json` establece las variables de entorno y determina si los datos del usuario son necesarios para ejecutar el conjunto de pruebas. Utilice la siguiente plantilla para configurar el archivo *<custom-test-suite-folder>*`/suite/suite.json`:

```
{
    "id": "<suite-name>_<suite-version>",
    "title": "<suite-title>",
    "details": "<suite-details>",
    "userDataRequired": true | false,
    "environmentVariables": [
        {
            "key": "<name>",
            "value": "<value>",
        },
        ...
        {
            "key": "<name>",
            "value": "<value>",
        }
    ]
}
```

Todos los campos que contienen valores son obligatorios tal y como se describe aquí:

`id`

Un ID único definido por el usuario para el conjunto de pruebas. El valor de `id` debe coincidir con el nombre de la carpeta del conjunto de pruebas en la que se encuentra el archivo `suite.json`. El nombre y la versión del conjunto también deben cumplir los siguientes requisitos:

- *<suite-name>* no puede contener guiones bajos.
- *<suite-version>* se indica como *x.x.x*, donde x es un número.

El ID se muestra en los informes de prueba generados por IDT.

`title`

Un nombre definido por el usuario para el producto o la característica que se está probando en este conjunto de pruebas. El nombre se muestra en la CLI de IDT para los ejecutores de las pruebas.

`details`

Una descripción corta de la finalidad del conjunto de pruebas.

`userDataRequired`

Define si los ejecutores de las pruebas deben incluir información personalizada en un archivo `userdata.json`. Si establece este valor en `true`, también debe incluir el archivo `userdata_schema.json` en la carpeta del conjunto de pruebas.

`environmentVariables`

Opcional. Una matriz de variables de entorno que se va a establecer para este conjunto de pruebas.

`environmentVariables.key`

El nombre de la variable de entorno.

`environmentVariables.value`

El valor de la variable de entorno.

Configuración de group.json

El archivo `group.json` define si un grupo de pruebas es obligatorio u opcional. Utilice la siguiente plantilla para configurar el archivo *<custom-test-suite-folder>*`/suite/`*<test-group>*`/group.json`:

```
{
    "id": "<group-id>",
```

```
    "title": "<group-title>",
    "details": "<group-details>",
    "optional": true | false,
}
```

Todos los campos que contienen valores son obligatorios tal y como se describe aquí:

`id`

Un ID único definido por el usuario para el conjunto de pruebas. El valor de `id` debe coincidir con el nombre de la carpeta del grupo de pruebas en la que se encuentra el `group.json` archivo y no puede contener guiones bajos (). _ El ID se utiliza en los informes de prueba generados por IDT.

`title`

Un nombre descriptivo para el grupo de prueba. El nombre se muestra en la CLI de IDT para los ejecutores de las pruebas.

`details`

Una descripción corta de la finalidad del grupo de pruebas.

`optional`

Opcional. Establézcalo en `true` para mostrar este grupo de pruebas como un grupo opcional una vez que IDT termine de ejecutar las pruebas requeridas. El valor predeterminado es `false`.

Configuración de test.json

El `test.json` archivo determina los ejecutables del caso de prueba y las variables de entorno que utiliza un caso de prueba. Para obtener más información sobre cómo crear ejecutables de casos de prueba, consulte Cree ejecutables de casos de prueba de IDT.

Utilice la siguiente plantilla para configurar el archivo *<custom-test-suite-folder>*/suite/*<test-group>*/*<test-case>*/test.json:

```
{
    "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>",
        }
    ]
}
```

Todos los campos que contienen valores son obligatorios tal y como se describe aquí:

`id`

Un ID único definido por el usuario para el caso de prueba. El valor de `id` debe coincidir con el nombre de la carpeta del caso de prueba en la que se encuentra el `test.json` archivo y no puede contener guiones bajos (_). El ID se utiliza en los informes de pruebas generados por IDT.

`title`

Un nombre descriptivo para el caso de prueba. El nombre se muestra en la CLI de IDT para los ejecutores de las pruebas.

`details`

Una breve descripción de la finalidad del caso de prueba.

`requireDUT`

Opcional. Establézcalo en `true` si se requiere un dispositivo para ejecutar esta prueba; de lo contrario, establézcalo en `false`. El valor predeterminado es `true`. Los ejecutores de las pruebas configurarán los dispositivos que utilizarán para ejecutar la prueba en su archivo `device.json`.

`requiredResources`

Opcional. Una matriz que proporciona información sobre los dispositivos de recursos necesarios para ejecutar esta prueba.

`requiredResources.name`

El nombre exclusivo que se asignará al dispositivo de recursos cuando se ejecute esta prueba.

`requiredResources.features`

Una matriz de características del dispositivo de recursos definidas por el usuario.

`requiredResources.features.name`

El nombre de la característica. La característica del dispositivo para la que desea utilizar este dispositivo. Este nombre se coteja con el nombre de la característica que proporciona el ejecutor de las pruebas en el archivo `resource.json`.

`requiredResources.features.version`

Opcional. La versión de la característica. Este valor se coteja con la versión de la característica proporcionada por el ejecutor de las pruebas en el archivo `resource.json`.

Si no se proporciona una versión, la característica no se comprueba. Si no se necesita un número de versión para la característica, deje este campo en blanco.

`requiredResources.features.jobSlots`

Opcional. El número de pruebas simultáneas que puede admitir esta característica. El valor predeterminado es 1. Si desea que IDT utilice distintos dispositivos para características individuales, le recomendamos que establezca este valor en 1.

`execution.timeout`

La cantidad de tiempo (en milisegundos) que IDT espera a que la prueba termine de ejecutarse. Para obtener más información sobre cómo establecer este valor, consulte Cree ejecutables de casos de prueba de IDT.

`execution.os`

Los ejecutables del caso de prueba que se ejecutarán en función del sistema operativo del equipo host que ejecuta IDT. Los valores admitidos son `linux`, `mac` y `win`.

`execution.os.cmd`

La ruta al ejecutable del caso de prueba que desea ejecutar para el sistema operativo especificado. Esta ubicación debe estar en la ruta del sistema.

`execution.os.args`

Opcional. Los argumentos que se deben proporcionar para ejecutar el ejecutable del caso de prueba.

`environmentVariables`

Opcional. Una matriz de variables de entorno definidas para este caso de prueba.

`environmentVariables.key`

El nombre de la variable de entorno.

`environmentVariables.value`

El valor de la variable de entorno.

> **Note**
>
> Si especifica la misma variable de entorno en el archivo `test.json` y en el archivo `suite.json`, el valor del archivo `test.json` tiene prioridad.

Configuración de test_orchestrator.yaml

Un orquestador de pruebas es un constructo que controla el flujo de ejecución del conjunto de pruebas. Determina el estado inicial de un conjunto de pruebas, gestiona las transiciones de estado en función de las reglas definidas por el usuario y continúa pasando por esos estados hasta alcanzar el estado final.

Si su conjunto de pruebas no incluye un orquestador de pruebas definido por el usuario, IDT generará un orquestador de pruebas para usted.

El orquestador de pruebas predeterminado realiza las siguientes funciones:

- Ofrece a los ejecutores de pruebas la posibilidad de seleccionar y ejecutar grupos de pruebas específicos, en lugar de todo el conjunto de pruebas.
- Si no se seleccionan grupos de pruebas específicos, ejecuta todos los grupos de pruebas del conjunto de pruebas en orden aleatorio.
- Genera informes e imprime un resumen de la consola que muestra los resultados de las pruebas de cada grupo y caso de prueba.

Para obtener más información sobre cómo funciona el orquestador de pruebas, consulte Configurar el orquestrador de pruebas IDT.

Configuración de userdata_schema.json

El archivo `userdata_schema.json` determina el esquema en el que los ejecutores de las pruebas proporcionan los datos de usuario. Los datos de usuario son necesarios si su conjunto de pruebas requiere información que no está presente en el archivo `device.json`. Por ejemplo, es posible que las pruebas necesiten credenciales de red Wi-Fi, puertos abiertos específicos o certificados que deba proporcionar un usuario. Esta información se puede proporcionar a IDT como un parámetro de entrada denominado`userdata`, cuyo valor es un archivo `userdata.json`, que los usuarios crean en su carpeta *`<device-tester-extract-location>`*`/config`. El formato del archivo `userdata.json` se basa en el archivo `userdata_schema.json` que se incluye en el conjunto de pruebas.

Para indicar que los ejecutores de pruebas deben proporcionar un archivo `userdata.json`:

1. En el archivo `suite.json`, establezca `userDataRequired` en `true`.
2. En su *`<custom-test-suite-folder>`*, cree un archivo `userdata_schema.json`.

3. Edite el archivo `userdata_schema.json` para crear un borrador del esquema JSON v4 de IETF válido.

Cuando IDT ejecuta su conjunto de pruebas, lee automáticamente el esquema y lo usa para validar el archivo `userdata.json` proporcionado por el ejecutor de la prueba. Si es válido, el contenido del archivo `userdata.json` está disponible tanto en el contexto de IDT como en el contexto del orquestador de pruebas.

Configurar el orquestrador de pruebas IDT

A partir de IDT v4.5.1, IDT incluye un nuevoorquestación de pruebascomponente. El orquestrador de pruebas es un componente IDT que controla el flujo de ejecución del conjunto de pruebas y genera el informe de pruebas después de que IDT finalice la ejecución de todas las pruebas. El orquestrador de pruebas determina la selección de pruebas y el orden en que se ejecutan las pruebas según las reglas definidas por el usuario.

Si el conjunto de pruebas no incluye un orquestador de pruebas definido por el usuario, IDT generará un orquestador de pruebas para usted.

El orquestación de pruebas predeterminado realiza las siguientes funciones:

- Ofrece a los ejecutores de pruebas la capacidad de seleccionar y ejecutar grupos de pruebas específicos, en lugar de todo el conjunto de pruebas.
- Si no se seleccionan grupos de pruebas específicos, ejecuta todos los grupos de pruebas del conjunto de pruebas en orden aleatorio.
- Genera informes e imprime un resumen de la consola que muestra los resultados de las pruebas de cada grupo de prueba y caso de prueba.

El orquestador de pruebas reemplaza al orquestador de pruebas IDT. Le recomendamos encarecidamente que utilice el orquestrador de pruebas para desarrollar sus conjuntos de pruebas en lugar del orquestador de pruebas IDT. El orquestrador de pruebas proporciona las siguientes funciones mejoradas:

- Utiliza un formato declarativo en comparación con el formato imperativo que utiliza la máquina de estado IDT. Esto le permiteespecificarqué pruebas quieres ejecutar ycuandoquieres dirigirlos.

- Gestiona el manejo de grupos específicos, la generación de informes, el tratamiento de errores y el seguimiento de resultadospara que no se te requierapara administrar manualmente estas acciones.
- Utiliza el formato YAML, que admite comentarios de forma predeterminada.
- requiere80 por cientomenos espacio en disco que el orquestador de pruebas para definir el mismo flujo de trabajo.
- Añade validación previa a la prueba para verificar que la definición de flujo de trabajo no contiene identificadores de prueba incorrectos ni dependencias circulares.

Formato orquestador de pruebas

Puede utilizar la siguiente plantilla para configurar la suya propia*<custom-test-suite-folder>*/`suite/test_orchestrator.yaml`file:

```
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
```

Todos los campos que contienen valores son obligatorios tal y como se describe aquí:

`Aliases`

Opcional. Cadenas definidas por el usuario que se asignan a expresiones de contexto. Los alias permiten generar nombres descriptivos paraidentificar expresiones de contexto en la configuración del orquestrador de pruebas. Esto es especialmente útil si está creando expresiones de contexto complejas o expresiones que utiliza en varios lugares.

Puede utilizar expresiones de contexto para almacenar consultas de contexto que le permiten acceder a datos de otras configuraciones de IDT. Para obtener más información, consulte Acceda a los datos en el contexto.

Example Ejemplo

```
Aliases:
    FizzChosen: "'{{$pool.features[?(@.name == 'Fizz')].value[0]}}' == 'yes'"
    BuzzChosen: "'{{$pool.features[?(@.name == 'Buzz')].value[0]}}' == 'yes'"
    FizzBuzzChosen: "'{{$aliases.FizzChosen}}' && '{{$aliases.BuzzChosen}}'"
```

`ConditionalTests`

Opcional. Una lista de condiciones y los casos de prueba correspondientes que se ejecutan cuando se cumple cada condición. Cada afección puede tener varios casos de prueba; sin embargo, puede asignar un caso de prueba determinado a una sola condición.

De forma predeterminada, IDT ejecuta cualquier caso de prueba que no esté asignado a una condición de esta lista. Si no especifica esta sección, IDT ejecuta todos los grupos de prueba del conjunto de pruebas.

Cada artículo del`ConditionalTests`list incluye los siguientes parámetros:

`Condition`

Una expresión de contexto que tiene el valor debooleanoValor . Si el valor evaluado es verdadero, IDT ejecuta los casos de prueba especificados en el`Tests`parámetro.

`Tests`

La lista de descriptores de prueba.

Cada descriptor de prueba utiliza el ID del grupo de pruebas y uno o más identificadores de casos de prueba para identificar las pruebas individuales que se van a ejecutar de un grupo de pruebas específico. El descriptor de prueba utiliza el siguiente formato:

```
GroupId: group-id
CaseIds: [test-id, test-id] # optional
```

Example Ejemplo

En el siguiente ejemplo se utilizan expresiones de contexto genéricas que se pueden definir como`Aliases`.

```
ConditionalTests:
    - Condition: "{{$aliases.Condition1}}"
      Tests:
          - GroupId: A
          - GroupId: B
    - Condition: "{{$aliases.Condition2}}"
      Tests:
          - GroupId: D
    - Condition: "{{$aliases.Condition1}} || {{$aliases.Condition2}}"
      Tests:
          - GroupId: C
```

En función de las condiciones definidas, IDT selecciona los grupos de prueba de la siguiente manera:

- Si`Condition1`es cierto, IDT realiza las pruebas en los grupos de prueba A, B y C.
- Si`Condition2`es cierto, IDT ejecuta las pruebas en los grupos de prueba C y D.

`Order`

Opcional. El orden en que se ejecutan las pruebas. El orden de prueba se especifica en el nivel del grupo de pruebas. Si no especifica esta sección, IDT ejecuta todos los grupos de pruebas aplicables en orden aleatorio. El valor de`Order`es una lista de listas de descriptores de grupo. Cualquier grupo de prueba en el que no incluyas la lista`Order`, se puede ejecutar en parallel con cualquier otro grupo de prueba de la lista.

Cada lista de descriptores de grupo contiene uno de los más descriptores de grupo e identifica el orden en que se ejecutan los grupos especificados en cada descriptor. Puede utilizar los siguientes formatos para definir descriptores de grupo individuales:

- *`group-id`*: el ID de grupo de un grupo de prueba existente.
- [*`group-id`*, *`group-id`*]: lista de grupos de pruebas que se pueden ejecutar en cualquier orden en relación entre sí.

- `"*"`—comodín. Esto equivale a la lista de todos los grupos de prueba que aún no se han especificado en la lista de descriptores de grupos actual.

El valor de`Order`debe cumplir también los siguientes requisitos:

- Los ID de grupo de pruebas especificados en un descriptor de grupo deben existir en el conjunto de pruebas.
- Cada lista de descriptores de grupo debe incluir al menos un grupo de prueba.
- Cada lista de descriptores de grupo debe contener identificadores de grupo únicos. No se puede repetir un ID de grupo de pruebas dentro de los descriptores de grupo individuales.
- Una lista de descriptores de grupo puede tener al menos un descriptor de grupo comodín. El descriptor de grupo comodín debe ser el primer o el último elemento de la lista.

Example Ejemplos

Para un conjunto de pruebas que contiene grupos de pruebas A, B, C, D y E, en la siguiente lista de ejemplos se muestran diferentes formas de especificar que IDT debe ejecutar primero el grupo de pruebas A, luego ejecutar el grupo de pruebas B y, a continuación, ejecutar los grupos de prueba C, D y E en cualquier orden.

- ```
  Order:
      - - A
        - B
        - [C, D, E]
  ```
- ```
  Order:
      - - A
        - B
        - "*"
  ```
- ```
  Order:
      - - A
        - B

      - - B
        - C

      - - B
        - D

      - - B
  ```
  ```

```
      - E
```

`Features`

Opcional. La lista de características del producto que desea que IDT agregue a la`awsiotdevicetester_report.xml`file. Si no especificas esta sección, IDT no añadirá ninguna característica del producto al informe.

Una característica de producto es información definida por el usuario sobre criterios específicos que un dispositivo podría cumplir. Por ejemplo, la función del producto MQTT puede indicar que el dispositivo publica correctamente los mensajes MQTT. En`awsiotdevicetester_report.xml`, las características del producto se establecen como`supported`,`not-supported`, o un valor personalizado definido por el usuario, en función de si se han superado las pruebas especificadas.

Cada artículo del`Features`list consta de los siguientes parámetros:

`Name`

El nombre de la característica.

`Value`

Opcional. El valor personalizado que desea utilizar en el informe en lugar de`supported`. Si no se especifica este valor, IDT basado establece el valor de la entidad en`supported`o`not-supported`basado en los resultados de pruebas. Si prueba la misma función con condiciones diferentes, puede utilizar un valor personalizado para cada instancia de esa entidad en el`Features`e IDT concatena los valores de entidad para las condiciones admitidas. Para obtener más información, consulte

`Condition`

Una expresión de contexto que tiene el valor de`booleanoValor` . Si el valor evaluado es verdadero, IDT añade la función al informe de prueba una vez finalizado la ejecución del grupo de pruebas. Si el valor evaluado es falso, la prueba no se incluye en el informe.

`Tests`

Opcional. La lista de descriptores de prueba. Todas las pruebas especificadas en esta lista deben pasar para que se admita la función.

Cada descriptor de prueba de esta lista utiliza el ID del grupo de pruebas y uno o más identificadores de casos de prueba para identificar las pruebas individuales que se van

a ejecutar de un grupo de pruebas específico. El descriptor de prueba utiliza el siguiente formato:

```
GroupId: group-id
CaseIds: [test-id, test-id] # optional
```

Debe especificarTestsoOneOfTestspara cada función delFeatureslista.

`OneOfTests`

Opcional. La lista de descriptores de prueba. Debe pasar al menos una de las pruebas especificadas en esta lista para que se admita la función.

Cada descriptor de prueba de esta lista utiliza el ID del grupo de pruebas y uno o más identificadores de casos de prueba para identificar las pruebas individuales que se van a ejecutar de un grupo de pruebas específico. El descriptor de prueba utiliza el siguiente formato:

```
GroupId: group-id
CaseIds: [test-id, test-id] # optional
```

Debe especificarTestsoOneOfTestspara cada función delFeatureslista.

`IsRequired`

El valor booleano que define si la función es necesaria en el informe de prueba. El valor predeterminado es `false`.

Example

Contexto de orquestación de prueba

El contexto del orquestrador de pruebas es un documento JSON de solo lectura que contiene datos disponibles para el orquestador de pruebas durante la ejecución. El contexto del orquestrador de pruebas solo es accesible desde el orquestador de pruebas y contiene información que determina el flujo de pruebas. Por ejemplo, puede utilizar la información configurada por los ejecutores de pruebas en el`userdata.json`para determinar si se requiere una prueba específica para ejecutarse.

El contexto del orquestación de pruebas utiliza el siguiente formato:

```
{
```

```
    "pool": {
        <device-json-pool-element>
    },
    "userData": {
        <userdata-json-content>
    },
    "config": {
        <config-json-content>
    }
}
```

`pool`

Información sobre el grupo de dispositivos seleccionado para la ejecución de prueba. Para un grupo de dispositivos seleccionado, esta información se recupera del elemento de matriz de grupos de dispositivos de nivel superior correspondiente definido en el`device.json`file.

`userData`

Información de en la`userdata.json`file.

`config`

Información de en la`config.json`file.

Puede consultar el contexto mediante la notación JSONPath. La sintaxis de las consultas JSONPath en las definiciones de estado es`{{query}}`. Cuando acceda a los datos desde el contexto del orquestrador de pruebas, asegúrese de que cada valor se evalúe en una cadena, un número o unbooleano.

Para obtener más información sobre el uso de la notación JSONPath para acceder a los datos del contexto, consulteUtilizar el contexto IDT.

Configurar el equipo de estado IDT

> **Important**
>
> A partir de IDT v4.5.1, esta máquina de estado está obsoleta. Le recomendamos que utilice el nuevo orquestador de pruebas. Para obtener más información, consulte Configurar el orquestrador de pruebas IDT.

Un equipo de estado es un componente fijo que controla el flujo de ejecución del conjunto de pruebas. Determina el estado inicial de un conjunto de pruebas, administra las transiciones de estado en función de reglas definidas por el usuario y continúa realizando la transición a través de esos estados hasta que alcanza el estado final.

Si el conjunto de pruebas no incluye un equipo de estado definido por el usuario, IDT generará un equipo de estado para usted. El equipo de estado predeterminado realiza las siguientes funciones:

- Proporciona a los ejecutores de pruebas la capacidad de seleccionar y ejecutar grupos de pruebas específicos, en lugar de todo el conjunto de pruebas.
- Si no se seleccionan grupos de pruebas específicos, ejecuta todos los grupos de pruebas del conjunto de pruebas en orden aleatorio.
- Genera informes e imprime un resumen de la consola que muestra los resultados de las pruebas de cada grupo de prueba y caso de prueba.

La máquina de estado de un grupo de pruebas de IDT debe cumplir los siguientes criterios:

- Cada estado corresponde a una acción que IDT debe llevar a cabo, como ejecutar un grupo de prueba o un producto un archivo de informe.
- La transición a un estado ejecuta la acción asociada al estado.
- Cada estado define la regla de transición para el siguiente estado.
- El estado final debe ser o`Succeed`o`Fail`.

Formato de la máquina de estado

Puede utilizar la siguiente plantilla para configurar la suya propia*<custom-test-suite-folder>*/`suite/state_machine.json`file:

```
{
  "Comment": "<description>",
  "StartAt": "<state-name>",
  "States": {
    "<state-name>": {
      "Type": "<state-type>",
      // Additional state configuration
    }

    // Required states
```

```
    "Succeed": {
      "Type": "Succeed"
    },
    "Fail": {
      "Type": "Fail"
    }
  }
}
```

Todos los campos que contienen valores son obligatorios tal y como se describe aquí:

`Comment`

Una descripción de la máquina de estado.

`StartAt`

Nombre del estado en que IDT comienza a ejecutar el grupo de pruebas. El valor de`StartAt`debe establecerse en uno de los estados que figuran en la`States`objeto.

`States`

Objeto que asigna nombres de estados definidos por el usuario a estados IDT válidos. Cada uno de los Estados.*`state-name`*contiene la definición de un estado válido asignado al*`state-name`*.

La`States`objeto debe incluir el`Succeed`y`Fail`estados. Para obtener información sobre los estados válidos, consulteEstados válidos y definiciones de estado.

Estados válidos y definiciones de estado

En esta sección se describen las definiciones de estado de todos los estados válidos que se pueden utilizar en el equipo de estado IDT. Algunos de los siguientes estados admiten configuraciones a nivel de caso de prueba. Sin embargo, le recomendamos que configure las reglas de transición de estado a nivel de grupo de pruebas en lugar del caso de prueba, a menos que sea absolutamente necesario.

Definiciones de estado

- RunTask
- Opción
- Parallel
- Añadir características del producto

- Informar
- Mensaje de registro
- Seleccionar grupo
- Fail
- Succeed

RunTask

LaRunTaskstate ejecuta casos de prueba de un grupo de prueba definido en el conjunto de pruebas.

```
{
    "Type": "RunTask",
    "Next": "<state-name>",
    "TestGroup": "<group-id>",
    "TestCases": [
        "<test-id>"
    ],
    "ResultVar": "<result-name>"
}
```

Todos los campos que contienen valores son obligatorios tal y como se describe aquí:

`Next`

Nombre del estado que se va a adoptar después de ejecutar las acciones en el estado actual.

`TestGroup`

Opcional. Identificador del grupo de prueba que se va a ejecutar. Si no se especifica este valor, IDT ejecuta el grupo de prueba que selecciona el corredor de prueba.

`TestCases`

Opcional. Matriz de identificadores de casos de prueba del grupo especificado en`TestGroup`. Sobre la base de los valores de`TestGroup`y`TestCases`, IDT determina el comportamiento de ejecución de la prueba de la siguiente manera:

- Cuando ambos`TestGroup`y`TestCases`se especifican, IDT ejecuta los casos de prueba especificados del grupo de prueba.
- Cuando`TestCases`se especifican pero`TestGroup`no se especifica, IDT ejecuta los casos de prueba especificados.

- CuandoTestGroupse especifica, peroTestCasesno se especifica, IDT ejecuta todos los casos de prueba del grupo de prueba especificado.
- Cuando ningunoTestGroupoTestCases, IDT ejecuta todos los casos de prueba del grupo de prueba que el ejecutor de prueba selecciona de la CLI de IDT. Para habilitar la selección de grupos para los ejecutores de pruebas, debe incluir ambosRunTaskyChoiceestados en tustate_machine.jsonfile. Para ver un ejemplo de cómo funciona, consulteMáquina de estado de ejemplo: Ejecutar grupos de prueba seleccionados por el usuario.

  Para obtener más información acerca de la habilitación de los comandos de la CLI de IDT para ejecutores de prueba, consultethe section called “Habilitar comandos de la CLI”.

`ResultVar`

El nombre de la variable de contexto que se va a establecer con los resultados de la prueba. No especifique este valor si no ha especificado un valor para`TestGroup`. IDT establece el valor de la variable que define en`ResultVar`a`true`o`false`basado en lo siguiente:

- Si el nombre de la variable pertenece al formulario*`text`*`_`*`text`*`_passed`, el valor se establece en si todas las pruebas del primer grupo de pruebas han pasado o se han omitido.
- En todos los demás casos, el valor se establece en si se han omitido o se han omitido todas las pruebas de todos los grupos de pruebas.

Normalmente, utilizará`RunTask`estado para especificar un ID de grupo de pruebas sin especificar identificadores de casos de prueba individuales, de modo que IDT ejecute todos los casos de prueba del grupo de prueba especificado. Todos los casos de prueba ejecutados por este estado se ejecutan en parallel, en orden aleatorio. Sin embargo, si todos los casos de prueba requieren que se ejecute un dispositivo y solo hay un dispositivo disponible, los casos de prueba se ejecutarán secuencialmente.

Error handling (Control de errores)

Si alguno de los grupos de prueba o ID de casos de prueba especificados no es válido, este estado emite la`RunTaskError`error de ejecución de. Si el estado encuentra un error de ejecución, también establece el`hasExecutionError`variable en el contexto de la máquina de estado para`true`.

Opción

La`Choice`state le permite establecer dinámicamente el siguiente estado al que se va a realizar la transición en función de las condiciones definidas por el usuario.

```
{
    "Type": "Choice",
    "Default": "<state-name>",
    "FallthroughOnError": true | false,
    "Choices": [
        {
            "Expression": "<expression>",
            "Next": "<state-name>"
        }
    ]
}
```

Todos los campos que contienen valores son obligatorios tal y como se describe aquí:

`Default`

El estado predeterminado que se va a adoptar si no tiene lugar ninguna de las expresiones definidas en`Choices`se puede evaluar para`true`.

`FallthroughOnError`

Opcional. Especifica el comportamiento cuando el estado encuentra un error en la evaluación de expresiones. Establezca en`true`si desea omitir una expresión si la evaluación produce un error. Si no hay expresiones que coincidan, el equipo de estados hace la transición a`Default`estado. Si el archivo de`FallthroughOnError`no se especifica el valor, el valor predeterminado es`false`.

`Choices`

Conjunto de expresiones y estados para determinar a qué estado realizar la transición después de ejecutar las acciones en el estado actual.

`Choices.Expression`

Cadena de expresión que se evalúa en un valor booleano. Si la expresión se evalúa como`true`, a continuación, la máquina de estados cambia al estado definido en`Choices.Next`. Las cadenas de expresión recuperan valores del contexto de la máquina de estado y, a continuación, realizan operaciones en ellas para obtener un valor booleano. Para obtener información sobre cómo acceder al contexto de la máquina de estado, consulteContexto de máquina de estado.

`Choices.Next`

Nombre del estado que se va a adoptar si la expresión definida en`Choices.Expression`evalúa para`true`.

Error handling (Control de errores)

La`Choice`state puede requerir la gestión de errores en los siguientes casos:

- Algunas variables de las expresiones de elección no existen en el contexto de la máquina de estados.
- El resultado de una expresión no es un valor booleano.
- El resultado de una búsqueda JSON no es una cadena, un número ni un booleano.

No puede utilizar un`Catch`bloque para tratar los errores de este estado. Si desea dejar de ejecutar la máquina de estado cuando se produce un error, debe configurar`FallthroughOnError`a`false`. Sin embargo, le recomendamos que establezca`FallthroughOnError`a`true`y, en función de su caso de uso, lleve a cabo uno de los procedimientos siguientes:

- Si se espera que no exista una variable a la que está accediendo en algunos casos, utilice el valor de`Default`y adicionales`Choices`bloques para especificar el siguiente estado.
- Si siempre debe existir una variable a la que está accediendo, establezca la`Default`estado a`Fail`.

Parallel

La`Parallel`state le permite definir y ejecutar nuevos equipos de estado en parallel entre sí.

```
{
    "Type": "Parallel",
    "Next": "<state-name>",
    "Branches": [
        <state-machine-definition>
    ]
}
```

Todos los campos que contienen valores son obligatorios tal y como se describe aquí:

`Next`

Nombre del estado que se va a adoptar después de ejecutar las acciones en el estado actual.

`Branches`

Matriz de definiciones de máquinas de estado que se van a ejecutar. Cada definición de máquina de estado debe contener su propia`StartAt`,`Succeed`, y`Fail`estados. Las definiciones de máquina de estado de esta matriz no pueden hacer referencia a estados fuera de su propia definición.

> **Note**
>
> Dado que cada máquina de estado de rama comparte el mismo contexto de máquina de estado, establecer variables en una rama y luego leer esas variables de otra rama podría dar lugar a un comportamiento inesperado.

La`Parallel`state se mueve al siguiente estado solo después de ejecutar todas las máquinas de estado de sucursal. Cada estado que requiera un dispositivo esperará a ejecutarse hasta que el dispositivo esté disponible. Si hay varios dispositivos disponibles, este estado ejecuta casos de prueba de varios grupos en parallel. Si no hay suficientes dispositivos disponibles, los casos de prueba se ejecutarán secuencialmente. Dado que los casos de prueba se ejecutan en orden aleatorio cuando se ejecutan en parallel, se pueden utilizar distintos dispositivos para ejecutar pruebas del mismo grupo de pruebas.

Error handling (Control de errores)

Asegúrese de que tanto la máquina de estado de sucursal como la máquina de estado padre pasen al`Fail`estado para gestionar los errores de ejecución.

Dado que los equipos de estado de sucursal no transmiten errores de ejecución al equipo de estado principal, no se puede utilizar un`Catch`bloque para gestionar errores de ejecución en máquinas de estado de sucursal. En su lugar, utilice la`hasExecutionErrors`valor en el contexto de máquina de estado compartido. Para ver un ejemplo de cómo funciona, consulteMáquina de estado de ejemplo: Ejecutar dos grupos de prueba en parallel.

Añadir características del producto

La`AddProductFeatures`state le permite añadir características del producto a la`awsiotdevicetester_report.xml`archivo generado por IDT.

Una característica de producto es información definida por el usuario sobre criterios específicos que un dispositivo podría cumplir. Por ejemplo, la`MQTT`la función del producto puede indicar que el dispositivo publica correctamente los mensajes MQTT. En el informe, las características del producto se establecen como`supported`,`not-supported`, o un valor personalizado, según si se han superado las pruebas especificadas.

> **Note**
>
> La`AddProductFeatures`state no genera informes por sí mismo. Este estado debe pasar al`Report`statepara generar informes.

```
{
    "Type": "Parallel",
    "Next": "<state-name>",
    "Features": [
        {
            "Feature": "<feature-name>",
            "Groups": [
                "<group-id>"
            ],
            "OneOfGroups": [
                "<group-id>"
            ],
            "TestCases": [
                "<test-id>"
            ],
            "IsRequired": true | false,
            "ExecutionMethods": [
                "<execution-method>"
            ]
        }
    ]
}
```

Todos los campos que contienen valores son obligatorios tal y como se describe aquí:

`Next`

Nombre del estado que se va a adoptar después de ejecutar las acciones en el estado actual.

`Features`

Una serie de características del producto que se pueden mostrar en el`awsiotdevicetester_report.xml`file.

`Feature`

El nombre de la característica

`FeatureValue`

Opcional. El valor personalizado que se utilizará en el informe en lugar de`supported`. Si no se especifica este valor, en función de los resultados de la prueba, el valor de la entidad se establece en`supported`o`not-supported`.

Si utiliza un valor personalizado para`FeatureValue`, puede probar la misma entidad con condiciones diferentes e IDT concatena los valores de entidad para las condiciones admitidas. Por ejemplo, en el siguiente fragmento se muestra la`MyFeature`entidad con dos valores de elemento independientes:

```
...
{
    "Feature": "MyFeature",
    "FeatureValue": "first-feature-supported",
    "Groups": ["first-feature-group"]
},
{
    "Feature": "MyFeature",
    "FeatureValue": "second-feature-supported",
    "Groups": ["second-feature-group"]
},
...
```

Si ambos grupos de prueba pasan, el valor de la entidad se establece en`first-feature-supported, second-feature-supported`.

`Groups`

Opcional. Una matriz de los ID de grupos de pruebas. Todas las pruebas de cada grupo de pruebas especificado deben pasar para que se admita la función.

`OneOfGroups`

Opcional. Una matriz de los ID de grupos de pruebas. Todas las pruebas de al menos uno de los grupos de pruebas especificados deben pasar para que se admita la función.

`TestCases`

Opcional. Una matriz de los ID de casos de prueba de prueba. Si especifica este valor, se aplica lo siguiente:

- Se deben aprobar todos los casos de prueba especificados para que se admita la función.
- `Groups`Debe contener un único ID de grupo de prueba.
- `OneOfGroups`no debe especificarse.

`IsRequired`

Opcional. Establezca en`false`para marcar esta función como elemento opcional en el informe. El valor predeterminado es `true`.

`ExecutionMethods`

Opcional. Matriz de métodos de ejecución que coinciden con el`protocol`valor especificado en la`device.json`file. Si se especifica este valor, los ejecutores de prueba deben especificar un`protocol`valor que coincide con uno de los valores de esta matriz para incluir la entidad en el informe. Si no se especifica este valor, la función siempre se incluirá en el informe.

Para utilizar el`AddProductFeatures`state, debes establecer el valor de`ResultVar`en la`RunTask`estado de uno de los siguientes valores:

- Si ha especificado identificadores de casos de prueba individuales, establezca`ResultVar`a*`group-id_test-id`*`_passed`.
- Si no especificó ID de caso de prueba individuales, establezca`ResultVar`a*`group-id`*`_passed`.

La`AddProductFeatures`Comprobaciones de estado de los resultados de las pruebas de la siguiente manera:

- Si no especificó ningún identificador de caso de prueba, el resultado de cada grupo de prueba se determina a partir del valor del*`group-id`*`_passed`variable en el contexto de la máquina de estado.
- Si especificó ID de caso de prueba, el resultado de cada una de las pruebas se determina a partir del valor de la*`group-id_test-id`*`_passed`variable en el contexto de la máquina de estado.

Error handling (Control de errores)

Si un ID de grupo proporcionado en este estado no es un ID de grupo válido, este estado da como resultado el`AddProductFeaturesError`error de ejecución de. Si el estado encuentra un error de ejecución, también establece el`hasExecutionErrors`variable en el contexto de la máquina de estado para`true`.

Informar

La`Report`state genera el*`suite-name`*`_Report.xml`y`awsiotdevicetester_report.xml`archivos. Este estado también transmite el informe a la consola.

```
{
    "Type": "Report",
    "Next": "<state-name>"
}
```

Todos los campos que contienen valores son obligatorios tal y como se describe aquí:

`Next`

Nombre del estado que se va a adoptar después de ejecutar las acciones en el estado actual.

Siempre debes hacer la transición a la`Report`estado hacia el final del flujo de ejecución de la prueba para que los ejecutores de pruebas puedan ver los resultados de las pruebas. Normalmente, el siguiente estado después de este estado es`Succeed`.

Error handling (Control de errores)

Si este estado encuentra problemas al generar los informes, emite el`ReportError`error de ejecución de.

Mensaje de registro

La`LogMessage`state genera el`test_manager.log`y transmite el mensaje de registro a la consola.

```
{
    "Type": "LogMessage",
    "Next": "<state-name>"
    "Level": "info | warn | error"
    "Message": "<message>"
```

```
}
```

Todos los campos que contienen valores son obligatorios tal y como se describe aquí:

`Next`

Nombre del estado que se va a adoptar después de ejecutar las acciones en el estado actual.

`Level`

Nivel de error en el que se va a crear el mensaje de registro. Si especifica un nivel que no es válido, este estado genera un mensaje de error y lo descarta.

`Message`

El mensaje que se va a registrar.

Seleccionar grupo

La`SelectGroup`state actualiza el contexto de la máquina de estado para indicar qué grupos están seleccionados. Los valores establecidos por este estado los utiliza cualquier subsiguiente`Choice`estados.

```
{
    "Type": "SelectGroup",
    "Next": "<state-name>"
    "TestGroups": [
        <group-id>"
    ]
}
```

Todos los campos que contienen valores son obligatorios tal y como se describe aquí:

`Next`

Nombre del estado que se va a adoptar después de ejecutar las acciones en el estado actual.

`TestGroups`

Matriz de grupos de prueba que se marcarán como seleccionados. Para cada ID de grupo de pruebas de esta matriz, el*group-id*`_selected`variable se establece en`true`en el contexto. Asegúrese de proporcionar ID de grupo de pruebas válidos porque IDT no valida si existen los grupos especificados.

Fail

La`Fail`state indica que la máquina de estado no se ha ejecutado correctamente. Este es un estado final de la máquina de estado y cada definición de máquina de estado debe incluir este estado.

```
{
    "Type": "Fail"
}
```

Succeed

La`Succeed`state indica que la máquina de estado se ha ejecutado correctamente. Este es un estado final de la máquina de estado y cada definición de máquina de estado debe incluir este estado.

```
{
    "Type": "Succeed"
}
```

Contexto de máquina de estado

El contexto de máquina de estado es un documento JSON de solo lectura que contiene datos disponibles para el equipo de estado durante la ejecución. Solo se puede acceder al contexto de la máquina de estado desde el equipo de estado y contiene información que determina el flujo de prueba. Por ejemplo, puede utilizar la información configurada por los ejecutores de pruebas en el`userdata.json`para determinar si se requiere una prueba específica para ejecutarse.

El contexto de la máquina de estados utiliza el siguiente formato:

```
{
    "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`

Información sobre el grupo de dispositivos seleccionado para la ejecución de prueba. Para un grupo de dispositivos seleccionado, esta información se recupera del elemento de matriz de grupos de dispositivos de nivel superior correspondiente definido en el`device.json`file.

`userData`

Información de la sección de`userdata.json`file.

`config`

Información de anclar el`config.json`file.

`suiteFailed`

El valor se establece en`false`cuando se inicia la máquina de estado. Si un grupo de prueba falla en un`RunTask`state, a continuación, este valor se establece en`true`durante el resto de la ejecución de la máquina de estado.

`specificTestGroups`

Si el ejecutor de pruebas selecciona grupos de pruebas específicos para ejecutarlos en lugar de todo el conjunto de pruebas, esta clave se crea y contiene la lista de identificadores de grupos de pruebas específicos.

`specificTestCases`

Si el ejecutor de pruebas selecciona casos de prueba específicos para ejecutarse en lugar de todo el conjunto de pruebas, esta clave se crea y contiene la lista de identificadores de casos de prueba específicos.

`hasExecutionErrors`

No sale cuando se inicia el equipo de estado. Si algún estado encuentra errores de ejecución, esta variable se crea y se establece en`true`durante el resto de la ejecución de la máquina de estado.

Puede consultar el contexto mediante la notación JSONPath. La sintaxis de las consultas JSONPath en las definiciones de estado es`{{$.query}}`. Puede utilizar las consultas JSONPath como cadenas de marcador de posición dentro de algunos estados. IDT reemplaza las cadenas de marcador de posición por el valor de la consulta JSONPath evaluada del contexto. Puede utilizar marcadores de posición para los siguientes valores:

- La`TestCases`valor de en`RunTask`estados.
- La`Expression`value`Choice`estado.

Al acceder a los datos desde el contexto de la máquina de estado, asegúrese de que se cumplan las siguientes condiciones:

- Las rutas de JSON deben comenzar por`$.`
- Cada valor debe evaluarse en una cadena, un número o un booleano.

Para obtener más información acerca de la utilización de la notación JSONPath para acceder a los datos del contexto, consulteUtilizar el contexto IDT.

Errores de ejecución

Los errores de ejecución son errores en la definición de máquina de estado que encuentra el equipo de estado al ejecutar un estado. IDT registra información sobre cada error en el`test_manager.log`y transmite el mensaje de registro a la consola.

Puede utilizar los siguientes métodos para tratar los errores de ejecución:

- Adición de un`Catch`bloqueen la definición de estado.
- Compruebe el valor de la`hasExecutionErrors`valueen el contexto de la máquina de estados.

Coger

Para utilizar`Catch`, añada lo siguiente a la definición de estado:

```
"Catch": [
    {
        "ErrorEquals": [
            "<error-type>"
        ]
```

```
        "Next": "<state-name>"
    }
]
```

Todos los campos que contienen valores son obligatorios tal y como se describe aquí:

`Catch.ErrorEquals`

Matriz de los tipos de errores que se catch. Si un error de ejecución coincide con uno de los valores especificados, la máquina de estado adopta el estado especificado en`Catch.Next`. Consulte cada definición de estado para obtener información sobre el tipo de error que produce.

`Catch.Next`

El siguiente estado al que se va a realizar la transición si el estado actual encuentra un error de ejecución que coincida con uno de los valores especificados en`Catch.ErrorEquals`.

Los bloques de captura se manejan secuencialmente hasta que uno coincida. Si los errores no coinciden con los que aparecen en los bloques Catch, las máquinas de estado siguen ejecutándose. Dado que los errores de ejecución son el resultado de definiciones de estado incorrectas, le recomendamos que haga la transición al estado Fallo cuando un estado detecte un error de ejecución.

Tiene error de ejecución

Cuando algunos estados encuentran errores de ejecución, además de emitir el error, también establecen el`hasExecutionError`valor de `atrue`en el contexto de la máquina de estados. Puede utilizar este valor para detectar cuándo se produce un error y, a continuación, utilizar un`Choice`estate para pasar la máquina de estado a la`Fail`estado.

Este método tiene las siguientes características:

- El equipo de estado no se inicia con ningún valor asignado a`hasExecutionError`, y este valor no está disponible hasta que un estado determinado lo establezca. Esto significa que debe establecer explícitamente la`FallthroughOnError`a`false`para la`Choice`indica que acceden a este valor para evitar que la máquina de estado se detenga si no se producen errores de ejecución.
- Una vez que esté configurado en`true`,`hasExecutionError`nunca se establece en false ni se elimina del contexto. Esto significa que este valor solo es útil la primera vez que se establece en`true`, y para todos los estados posteriores, no proporciona un valor significativo.

- La`hasExecutionError`valor se comparte con todos los equipos de estado de rama del`Parallel`state, que puede dar lugar a resultados inesperados según el orden en que se acceda a él.

Debido a estas características, no recomendamos que utilice este método si puede utilizar un bloque Catch en su lugar.

Máquinas de estado de ejemplo

En esta sección se proporcionan ejemplos de configuraciones de máquina de estado.

Ejemplos

- Máquina de estado de ejemplo: Ejecutar un único grupo de pruebas
- Máquina de estado de ejemplo: Ejecutar grupos de prueba seleccionados por el usuario
- Máquina de estado de ejemplo: Ejecutar un único grupo de pruebas con características de producto
- Máquina de estado de ejemplo: Ejecutar dos grupos de prueba en parallel

Máquina de estado de ejemplo: Ejecutar un único grupo de pruebas

Esta máquina de estado de:

- Ejecuta el grupo de prueba con id`GroupA`, que debe estar presente en la suite de un`group.json`file.
- Comprueba si hay errores de ejecución y transiciones a`Fail`si se encuentra alguno.
- Genera un informe y hace transiciones a`Succeed`si no hay errores, y`Fail`De lo contrario, .

```
{
    "Comment": "Runs a single group and then generates a report.",
    "StartAt": "RunGroupA",
    "States": {
        "RunGroupA": {
            "Type": "RunTask",
            "Next": "Report",
            "TestGroup": "GroupA",
            "Catch": [
                {
```

```
                    "ErrorEquals": [
                        "RunTaskError"
                    ],
                    "Next": "Fail"
                }
            ]
        },
        "Report": {
            "Type": "Report",
            "Next": "Succeed",
            "Catch": [
                {
                    "ErrorEquals": [
                        "ReportError"
                    ],
                    "Next": "Fail"
                }
            ]
        },
        "Succeed": {
            "Type": "Succeed"
        },
        "Fail": {
            "Type": "Fail"
        }
    }
}
```

Máquina de estado de ejemplo: Ejecutar grupos de prueba seleccionados por el usuario

Esta máquina de estado de:

- Comprueba si el corredor de prueba ha seleccionado grupos de pruebas específicos. El equipo de estado no comprueba casos de prueba específicos porque los ejecutores de pruebas no pueden seleccionar casos de prueba sin seleccionar también un grupo de pruebas.
- Si se seleccionan grupos de prueba:
  - Ejecuta los casos de prueba dentro de los grupos de prueba seleccionados. Para ello, el equipo de estado no especifica explícitamente ningún grupo de prueba ni casos de prueba en el`RunTask`estado.
  - Genera un informe después de ejecutar todas las pruebas y salidas.
- Si no se seleccionan grupos de prueba:

- Realiza pruebas en el grupo de pruebas`GroupA`.
- Genera informes y sale.

```
{
    "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"
        }
    }
}
```

Máquina de estado de ejemplo: Ejecutar un único grupo de pruebas con características de producto

Esta máquina de estado de:

- Ejecute el grupo de pruebas`GroupA`.
- Comprueba si hay errores de ejecución y transiciones a`Fail`si se encuentra alguno.
- Adición de la`FeatureThatDependsOnGroupA`característica de la`awsiotdevicetester_report.xml`file:
  - Si`GroupA`pases, la característica está establecida en`supported`.
  - La función no está marcada como opcional en el informe.
- Genera un informe y hace transiciones a`Succeed`si no hay errores, y`Fail`de lo contrario

```
{
    "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"
        }
```

```
    }
}
```

Máquina de estado de ejemplo: Ejecutar dos grupos de prueba en parallel

Esta máquina de estado de:

- Ejecute la`GroupA`y`GroupB`grupos de prueba en parallel. La`ResultVar`variables almacenadas en el contexto por el`RunTask`estados de las máquinas de estado de sucursal por están disponibles para el`AddProductFeatures`estado.
- Comprueba si hay errores de ejecución y transiciones a`Fail`si se encuentra alguno. Este equipo de estado no utiliza un`Catch`bloque porque ese método no detecta errores de ejecución en máquinas de estado de sucursal.
- Añade funciones a la`awsiotdevicetester_report.xml`archivo basado en los grupos que pasan
  - Si`GroupA`pases, la característica está establecida en`supported`.
  - La función no está marcada como opcional en el informe.
- Genera un informe y hace transiciones a`Succeed`si no hay errores, y`Fail`de lo contrario

Si hay dos dispositivos configurados en el grupo de dispositivos, ambos`GroupA`y`GroupB`puede ejecutarse al mismo tiempo. Sin embargo, si alguno de ellos`GroupA`o`GroupB`tiene varias pruebas y, a continuación, ambos dispositivos pueden asignarse a esas pruebas. Si solo se configura un dispositivo, los grupos de prueba se ejecutarán secuencialmente.

```
{
    "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"
        }
    }
}
```

Cree ejecutables de casos de prueba de IDT

Puede crear y colocar los ejecutables de casos de prueba en una carpeta del conjunto de pruebas de las siguientes maneras:

- Para los conjuntos de pruebas que utilizan argumentos o variables de entorno de los`test.json` archivos para determinar qué pruebas ejecutar, puede crear un único ejecutable de caso de prueba para todo el conjunto de pruebas o un ejecutable de prueba para cada grupo de pruebas del conjunto de pruebas.
- Para un conjunto de pruebas en el que desee ejecutar pruebas específicas en función de comandos especificados, cree un ejecutable de caso de prueba para cada caso de prueba del conjunto de pruebas.

Como redactor de pruebas, puede determinar qué enfoque es el adecuado para su caso de uso y estructurar el ejecutable del caso de prueba en consecuencia. Asegúrese de proporcionar la ruta del ejecutable de mayúsculas y minúsculas de prueba correcta en cada`test.json` archivo y de que el ejecutable especificado se ejecute correctamente.

Cuando todos los dispositivos estén listos para ejecutar un caso de prueba, IDT lee los siguientes archivos:

- El`test.json` para el caso de prueba seleccionado determina los procesos que se van a iniciar y las variables de entorno que se van a configurar.
- El conjunto`suite.json` de pruebas determina las variables de entorno que se van a configurar.

IDT inicia el proceso ejecutable de prueba requerido en función de los comandos y argumentos especificados en el`test.json` archivo y pasa las variables de entorno necesarias al proceso.

Utilice el SDK del cliente IDT

Los SDK de IDT Client te permiten simplificar la forma en que escribes la lógica de prueba en tu ejecutable de prueba con comandos de API que puedes usar para interactuar con IDT y los dispositivos que se están probando. IDT ofrece actualmente los siguientes SDK:

- SDK para Python
- SDK for Go
- SDK for Java

Estos SDK se encuentran en la`<device-tester-extract-location>`/`sdks` carpeta. Al crear un nuevo ejecutable de caso de prueba, debes copiar el SDK que deseas usar en la carpeta que contiene el ejecutable de caso de prueba y hacer referencia al SDK en tu código. En esta sección se proporciona una breve descripción de los comandos de API disponibles que puedes usar en los ejecutables de tu caso de prueba.

En esta sección

Interacción entre dispositivos

Los siguientes comandos le permiten comunicarse con el dispositivo que se está probando sin tener que implementar ninguna función adicional de administración de conectividad e interacción del dispositivo.

`ExecuteOnDevice`

Permite que los conjuntos de pruebas ejecuten comandos de shell en un dispositivo que admita conexiones de shell SSH o Docker.

`CopyToDevice`

Permite que los conjuntos de pruebas copien un archivo local de la máquina host que ejecuta IDT a una ubicación específica en un dispositivo que admita conexiones SSH o Docker shell.

`ReadFromDevice`

Permite que los conjuntos de pruebas lean desde el puerto serie de los dispositivos que admiten conexiones UART.

> **Note**
>
> Como IDT no gestiona las conexiones directas a los dispositivos que se realizan mediante la información de acceso a los dispositivos del contexto, recomendamos utilizar estos comandos de la API de interacción de dispositivos en los ejecutables de los casos de prueba. Sin embargo, si estos comandos no cumplen con los requisitos del caso de prueba, puede recuperar la información de acceso al dispositivo del contexto de IDT y utilizarla para establecer una conexión directa al dispositivo desde el conjunto de pruebas.
> Para establecer una conexión directa, recupere la información de los`device.connectivity`resource.devices.connectivity campos del dispositivo que se está probando y de los dispositivos de recursos, respectivamente. Para obtener más información acerca del uso del contexto de IDT, consulteUtilizar el contexto IDT.

Interacción IDT

Los siguientes comandos permiten que sus conjuntos de pruebas se comuniquen con IDT.

`PollForNotifications`

Permite que los conjuntos de pruebas comprueben las notificaciones de IDT.

`GetContextValue` y `GetContextString`

Permite que los conjuntos de pruebas recuperen valores del contexto IDT. Para obtener más información, consulte Utilizar el contexto IDT.

`SendResult`

Permite que los conjuntos de pruebas informen de los resultados de los casos de prueba a IDT. Este comando debe invocarse al final de cada caso de prueba en un conjunto de pruebas.

Interacción con el anfitrión

El siguiente comando permite que los conjuntos de pruebas se comuniquen con la máquina host.

`PollForNotifications`

Permite que los conjuntos de pruebas comprueben las notificaciones de IDT.

`GetContextValue` y `GetContextString`

Permite que los conjuntos de pruebas recuperen valores del contexto IDT. Para obtener más información, consulte Utilizar el contexto IDT.

`ExecuteOnHost`

Permite que los conjuntos de pruebas ejecuten comandos en la máquina local y permite a IDT gestionar el ciclo de vida de los ejecutables de los casos de prueba.

Habilitar comandos de la CLI

El `run-suite` comando IDT CLI proporciona varias opciones que permiten al ejecutor de pruebas personalizar la ejecución de la prueba. Para permitir que los ejecutores de pruebas utilicen estas opciones para ejecutar su conjunto de pruebas personalizado, implemente la compatibilidad con la CLI de IDT. Si no implementa el soporte, los ejecutores de pruebas podrán seguir ejecutándolas, pero algunas opciones de la CLI no funcionarán correctamente. Para ofrecer una experiencia de cliente ideal, le recomendamos que implemente la compatibilidad con los siguientes argumentos para el `run-suite` comando en la CLI de IDT:

`timeout-multiplier`

Especifica un valor superior a 1.0 que se aplicará a todos los tiempos de espera durante la ejecución de las pruebas.

Los ejecutores de pruebas pueden usar este argumento para aumentar el tiempo de espera de los casos de prueba que desean ejecutar. Cuando un ejecutor de pruebas especifica este argumento en su `run-suite` comando, IDT lo usa para calcular el valor de la variable de entorno IDT_TEST_TIMEOUT y establece el `config.timeoutMultiplier` campo en el contexto de IDT. Para respaldar este argumento, debe hacer lo siguiente:

- En lugar de utilizar directamente el valor de tiempo de espera del `test.json` archivo, lea la variable de entorno IDT_TEST_TIMEOUT para obtener el valor de tiempo de espera calculado correctamente.
- Recupera el `config.timeoutMultiplier` valor del contexto de IDT y aplícalo a tiempos de espera prolongados.

Para obtener más información sobre cómo salir anticipadamente debido a eventos de tiempo de espera, consulteEspecificar el comportamiento de salida.

`stop-on-first-failure`

Especifica que IDT debe dejar de ejecutar todas las pruebas si se produce un error.

Cuando un ejecutor de pruebas especifica este argumento en `surun-suite` comando, IDT dejará de ejecutar las pruebas tan pronto como detecte un error. Sin embargo, si los casos de prueba se ejecutan en parallel, esto puede generar resultados inesperados. Para implementar el soporte, asegúrese de que, si IDT detecta este evento, su lógica de prueba indique a todos los casos de prueba en ejecución que se detengan, limpien los recursos temporales e informen del resultado de la prueba a IDT. Para obtener más información acerca de cómo salir anticipadamente en caso de errores, consulteEspecificar el comportamiento de salida.

`group-id` y `test-id`

Especifica que IDT debe ejecutar solo los grupos de prueba o casos de prueba seleccionados.

Los ejecutores de pruebas pueden usar estos argumentos con `surun-suite` comando para especificar el siguiente comportamiento de ejecución de la prueba:

- Ejecute todas las pruebas dentro de los grupos de pruebas especificados.
- Ejecute una selección de pruebas dentro de un grupo de pruebas especificado.

Para respaldar estos argumentos, el orquestador de pruebas de su conjunto de pruebas debe incluir un conjuntoRunTask `yChoice` estados específicos en su orquestador de pruebas. Si no utiliza una máquina de estados personalizada, el orquestador de pruebas IDT predeterminado incluye los estados necesarios para usted y no necesita realizar ninguna acción adicional. Sin embargo, si utiliza un orquestador de pruebas personalizado, utilíceloMáquina de estado de ejemplo: Ejecutar grupos de prueba seleccionados por el usuario como ejemplo para añadir los estados requeridos en su orquestador de pruebas.

Para obtener más información acerca de los comandos de la CLI de IDT, consulteDepurar y ejecutar conjuntos de pruebas personalizadas.

Escribir registros de eventos

Mientras se ejecuta la prueba, se envían datos a la consola`stdout` `ystderr` se escriben registros de eventos y mensajes de error en ella. Para obtener información acerca del formato de los mensajes de la consola, consulteFormato de mensajes de consola.

Cuando el IDT termine de ejecutar el conjunto de pruebas, esta información también estará disponible en el`test_manager.log` archivo ubicado en la*<devicetester-extract-location>*`/results/`*<execution-id>*`/logs` carpeta.

Puede configurar cada caso de prueba para que escriba los registros de su ejecución de prueba, incluidos los registros del dispositivo que se está probando, en el*<group-id>*`_`*<test-id>* archivo que se encuentra en la*<device-tester-extract-location>*`/results/`*execution-id*`/logs` carpeta. Para ello, recupere la ruta al archivo de registro del contexto IDT con la`testData.logFilePath` consulta, cree un archivo en esa ruta y escriba el contenido que desee en él. IDT actualiza automáticamente la ruta en función del caso de prueba que se esté ejecutando. Si decide no crear el archivo de registro para un caso de prueba, no se generará ningún archivo para ese caso de prueba.

También puede configurar el ejecutable de texto para crear archivos de registro adicionales según sea necesario en la*<device-tester-extract-location>*`/logs` carpeta. Le recomendamos que especifique prefijos únicos para los nombres de los archivos de registro para que no se sobrescriban.

Informe los resultados a IDT

IDT escribe los resultados de las pruebas en`awsiotdevicetester_report.xml` y en los*suite-name*`_report.xml` archivos. Estos archivos de informes están ubicados en*<device-tester-extract-location>*`/results/`*<execution-id>*`/`. Ambos informes capturan los resultados de la ejecución del conjunto de pruebas. Para obtener más información sobre los esquemas que IDT utiliza para estos informes, consulteRevisar los resultados y registros de las pruebas IDT

Para rellenar el contenido del*suite-name*`_report.xml` archivo, debe utilizar el`SendResult` comando para informar de los resultados de las pruebas a IDT antes de que finalice la ejecución de la prueba. Si el IDT no puede encontrar los resultados de una prueba, emite un error para el caso de prueba. El siguiente extracto de Python muestra los comandos para enviar el resultado de una prueba a IDT:

```
request-variable = SendResultRequest(TestResult(result))
client.send_result(request-variable)
```

Si no notifica los resultados a través de la API, IDT busca los resultados de las pruebas en la carpeta de artefactos de prueba. La ruta a esta carpeta se almacena en el`testData.testArtifactsPath` archivo en el contexto IDT. En esta carpeta, IDT utiliza el primer archivo XML ordenado alfabéticamente que encuentra como resultado de la prueba.

Si su lógica de prueba produce resultados XML de JUnit, puede escribir los resultados de la prueba en un archivo XML de la carpeta artefactos para proporcionar los resultados directamente a IDT en lugar de analizar los resultados y, a continuación, utilizar la API para enviarlos a IDT.

Si utiliza este método, asegúrese de que la lógica de prueba resuma con precisión los resultados de la prueba y dé formato al archivo de resultados en el mismo formato que el*suite-name*_report.xml archivo. IDT no realiza ninguna validación de los datos que usted proporciona, con las siguientes excepciones:

- IDT ignora todas las propiedades de latestsuites etiqueta. En su lugar, calcula las propiedades de las etiquetas a partir de los resultados de otros grupos de prueba informados.
- Debe existir al menos unatestsuite etiqueta en su interiortestsuites.

Dado que IDT utiliza la misma carpeta de artefactos para todos los casos de prueba y no elimina los archivos de resultados entre las ejecuciones de las pruebas, este método también puede generar informes erróneos si IDT lee el archivo incorrecto. Le recomendamos que utilice el mismo nombre para el archivo de resultados XML generado en todos los casos de prueba para sobrescribir los resultados de cada caso de prueba y asegurarse de que los resultados correctos estén disponibles para que IDT los utilice. Si bien puede utilizar un enfoque mixto para generar informes en su conjunto de pruebas, es decir, utilizar un archivo de resultados XML para algunos casos de prueba y enviar los resultados a través de la API para otros, no recomendamos este enfoque.

Especificar el comportamiento de salida

Configure su ejecutable de texto para que salga siempre con un código de salida igual a 0, incluso si un caso de prueba indica un error o el resultado de un error. Utilice códigos de salida distintos de cero únicamente para indicar que un caso de prueba no se ejecutó o si el ejecutable del caso de prueba no pudo comunicar ningún resultado a IDT. Cuando IDT recibe un código de salida distinto de cero, indica que el caso de prueba ha encontrado un error que ha impedido su ejecución.

IDT podría solicitar o esperar que un caso de prueba deje de ejecutarse antes de que finalice en los siguientes eventos. Utilice esta información para configurar el ejecutable del caso de prueba para detectar cada uno de estos eventos del caso de prueba:

Timeout (Tiempo de espera)

Se produce cuando un caso de prueba se ejecuta durante más tiempo que el valor de tiempo de espera especificado en eltest.json archivo. Si el ejecutor de la prueba utilizó eltimeout-

`multiplier` argumento para especificar un multiplicador de tiempo de espera, IDT calcula el valor del tiempo de espera con el multiplicador.

Para detectar este evento, utilice la variable de entorno IDT_TEST_TIMEOUT. Cuando un ejecutor de pruebas lanza una prueba, IDT establece el valor de la variable de entorno IDT_TEST_TIMEOUT en el valor del tiempo de espera calculado (en segundos) y pasa la variable al ejecutable del caso de prueba. Puede leer el valor de la variable para configurar un temporizador adecuado.

Interrumpir

Se produce cuando el ejecutor de la prueba interrumpe el IDT. Por ejemplo, pulsandoCtrl+C.

Como los terminales propagan las señales a todos los procesos secundarios, puede configurar simplemente un controlador de señales en sus casos de prueba para detectar las señales de interrupción.

Como alternativa, puedes sondear periódicamente la API para comprobar el valor del`CancellationRequested` booleano en la respuesta de la`PollForNotifications` API. Cuando IDT recibe una señal de interrupción, establece el valor del`CancellationRequested` booleano en`true`.

Se detiene en el primer error

Se produce cuando un caso de prueba que se ejecuta en parallel con el caso de prueba actual falla y el ejecutor de la prueba utiliza el`stop-on-first-failure` argumento para especificar que IDT debe detenerse cuando se produce algún error.

Para detectar este evento, puedes sondear periódicamente la API para comprobar el valor del`CancellationRequested` booleano en la respuesta de la`PollForNotifications` API. Cuando IDT detecta un error y se configura para detenerse en el primer error, establece el valor del`CancellationRequested` booleano en`true`.

Cuando se produce alguno de estos eventos, IDT espera 5 minutos hasta que los casos de prueba que se estén ejecutando actualmente terminen de ejecutarse. Si todos los casos de prueba en ejecución no salen en 5 minutos, el IDT obliga a detener cada uno de sus procesos. Si IDT no ha recibido los resultados de las pruebas antes de que finalicen los procesos, marcará los casos de prueba como agotados. Como práctica recomendada, debe asegurarse de que los casos de prueba realicen las siguientes acciones cuando se encuentren con uno de los eventos:

1. Deje de ejecutar la lógica de prueba normal.

2. Limpia todos los recursos temporales, como los artefactos de prueba del dispositivo que se está probando.
3. Informe el resultado de una prueba a IDT, como una falla o un error en la prueba.
4. Salida.

Utilizar el contexto IDT

Cuando IDT ejecuta un conjunto de pruebas, el conjunto de pruebas puede acceder a un conjunto de datos que se pueden utilizar para determinar cómo se ejecuta cada prueba. Estos datos se denominan contexto IDT. Por ejemplo, la configuración de datos de usuario proporcionada por los ejecutores de pruebas en un`userdata.json`se pone a disposición de los grupos de pruebas en el contexto IDT.

El contexto IDT se puede considerar un documento JSON de solo lectura. Los conjuntos de pruebas pueden recuperar datos y escribirlos en el contexto utilizando tipos de datos JSON estándar, como objetos, matrices, números, etc.

Esquema contextual

El contexto IDT utiliza el siguiente formato:

```
{
    "config": {
        <config-json-content>
        "timeoutMultiplier": timeout-multiplier
    },
    "device": {
        <device-json-device-element>
    },
    "devicePool": {
        <device-json-pool-element>
    },
    "resource": {
        "devices": [
            {
                <resource-json-device-element>
                "name": "<resource-name>"
            }
        ]
    },
```

```
    "testData": {
        "awsCredentials": {
            "awsAccessKeyId": "<access-key-id>",
            "awsSecretAccessKey": "<secret-access-key>",
            "awsSessionToken": "<session-token>"
        },
        "logFilePath": "/path/to/log/file"
    },
    "userData": {
        <userdata-json-content>
    }
}
```

`config`

Información del`config.json`archivo. La`config`también contiene el siguiente campo adicional:

`config.timeoutMultiplier`

El multiplicador del valor de tiempo de espera utilizado por el conjunto de pruebas. Este valor lo especifica el ejecutor de prueba de la CLI de IDT. El valor predeterminado es 1.

`device`

Información sobre el dispositivo seleccionado para la prueba. Esta información equivale a la`devices`elemento array en el`device.json`archivopara el dispositivo seleccionado.

`devicePool`

Información sobre el grupo de dispositivos seleccionado para la ejecución de prueba. Esta información es equivalente al elemento de matriz de grupos de dispositivos de nivel superior definido en el`device.json`para el grupo de dispositivos seleccionado.

`resource`

Información sobre los dispositivos de recursos de la`resource.json`file.

`resource.devices`

Esta información equivale a la`devices`matriz definida en el`resource.json`file. Cada`devices`incluye el siguiente campo adicional:

`resource.device.name`

El nombre del dispositivo de recursos. Este valor se establece en el`requiredResource.name`valor en el`test.json`file.

`testData.awsCredentials`

LaAWScredenciales utilizadas por la prueba para conectarse alAWSnube. Esta información se obtiene del`config.json`file.

`testData.logFilePath`

Ruta del archivo de registro en el que el caso de prueba escribe mensajes de registro. El conjunto de pruebas crea este archivo si no existe.

`userData`

Información proporcionada por el corredor de pruebas en el`userdata.json`archivo.

Acceda a los datos en el contexto

Puede consultar el contexto mediante la notación JSONPath desde los archivos JSON y desde el ejecutable de texto con el`GetContextValue`y`GetContextString`APIs. La sintaxis de las cadenas JSONPath para acceder al contexto IDT varía de la siguiente manera:

- En`suite.json`y`test.json`, utiliza`{{`*`query`*`}}`. Es decir, no uses el elemento raíz`$.`para empezar tu expresión.
- En`test_orchestrator.yaml`, utiliza`{{`*`query`*`}}`.

  Si utiliza la máquina de estado obsoleta, entonces en`state_machine.json`, utiliza`{{$.`*`query`*`}}`.
- En los comandos de la API, utilizas*`query`*o`{{$.`*`query`*`}}`, según el comando. Para obtener más información, consulte la documentación en línea de los SDK.

En la siguiente tabla se describen los operadores de una expresión JSONPath típica:

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

Operator	Description
	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 `Región $.config.awsRegion` .
`[start:end]`	Filters elements from an array, retrieving items beginning from the `iniciar` index and going up to the `fin` index, both inclusive.
`[índice 1, índice 2,..., 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.

Para crear expresiones de filtro, utilice la siguiente sintaxis:

```
<jsonpath> | <value> operator <jsonpath> | <value>
```

En esta sintaxis:

- `jsonpath`es un JSONPath que utiliza sintaxis JSON estándar.
- `value`es cualquier valor personalizado que utilice la sintaxis JSON estándar.
- `operator`es uno de los siguientes operadores:
  - <(Menor que)
  - <=(Menor o igual que)
  - ==(Igual que)

    Si el valor o JSONPath de su expresión es un valor de matriz, booleano o objeto, este es el único operador binario compatible que puede utilizar.
  - >=(Mayor o igual que)
  - >(Mayor que)

- =~(coincidencia de expresiones regulares). Para utilizar este operador en una expresión de filtro, el JSONPath o el valor de la parte izquierda de la expresión deben evaluarse en una cadena y el lado derecho debe ser un valor de patrón que siga elSintaxis RE2.

Puede utilizar consultas JSONPath en el formulario {{*query*}} como cadenas de marcador de posición dentro de`args`y`environmentVariables`campos en`test.json`archivos y dentro de`environmentVariables`campos en`suite.json`archivos. IDT realiza una búsqueda de contexto y rellena los campos con el valor evaluado de la consulta. Por ejemplo, en el`suite.json`, puede utilizar cadenas de marcador de posición para especificar valores de variables de entorno que cambian con cada caso de prueba e IDT rellenará las variables de entorno con el valor correcto para cada caso de prueba. Sin embargo, cuando utiliza cadenas de marcador de posición en`test.json`y`suite.json`, se aplican las siguientes consideraciones para sus consultas:

- Debe tener cada vez que se produzca el`devicePool`clave en la consulta en minúsculas. Es decir, use`devicepool`en su lugar.
- Para matrices, solo puede utilizar matrices de cadenas. Además, los arreglos utilizan un elemento no estándar`item1, item2,...,itemN`formato. Si la matriz contiene solo un elemento, se serializa como`item`, lo que lo hace indistinguible de un campo de cadena.
- No se pueden utilizar marcadores de posición para recuperar objetos del contexto.

Debido a estas consideraciones, recomendamos que, siempre que sea posible, utilice la API para acceder al contexto de su lógica de prueba en lugar de cadenas de marcador de posición en`test.json`y`suite.json`archivos. Sin embargo, en algunos casos podría resultar más conveniente utilizar marcadores de posición JSONPath para recuperar cadenas individuales para definirlas como variables de entorno.

Configure los ajustes para los ejecutores de pruebas

Para ejecutar conjuntos de pruebas personalizados, los ejecutores de pruebas deben configurar sus ajustes en función del conjunto de pruebas que desean ejecutar. Los ajustes se especifican en función de las plantillas del archivo de configuración que se encuentran en la carpeta *<device-tester-extract-location>*`/configs/`. Si es necesario, los ejecutores de las pruebas también deben configurar las credenciales de AWS que IDT utilizará para conectarse a la nube de AWS.

Como redactor de pruebas, necesitará configurar estos archivos para depurar su conjunto de pruebas. Debe proporcionar instrucciones a los ejecutores de pruebas para que puedan configurar los siguientes ajustes según sea necesario para ejecutar sus conjuntos de pruebas.

Configurar device.json

El archivo `device.json` contiene información sobre los dispositivos en los que se ejecutan las pruebas (por ejemplo, dirección IP, información de inicio de sesión, sistema operativo y arquitectura de la CPU).

Los ejecutores de pruebas pueden proporcionar esta información mediante el siguiente archivo `device.json` de plantilla que se encuentra en la carpeta *<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>",
            }
        }
    ]
}
]
```

Todos los campos que contienen valores son obligatorios tal y como se describe aquí:

`id`

Un ID alfanumérico definido por el usuario que identifica de forma única una colección de dispositivos que se conoce como grupo de dispositivos. Los dispositivos que pertenecen a un grupo deben tener idéntico hardware. Al ejecutar un conjunto de pruebas, los dispositivos del grupo se utilizan para paralelizar la carga de trabajo. Se utilizan varios dispositivos para ejecutar diferentes pruebas.

`sku`

Un valor alfanumérico que identifica de forma única el dispositivo a prueba. El SKU se utiliza para realizar un seguimiento de los dispositivos cualificados.

> **Note**
>
> Si desea enumerar la placa en el catálogo de dispositivos de AWS Partner, el SKU que especifique aquí debe coincidir con el SKU que utilice en el proceso de publicación.

`features`

Opcional. Una matriz que contenga las características compatibles del dispositivo. Las características del dispositivo son valores definidos por el usuario que se configuran en el conjunto de pruebas. Debe proporcionar a los ejecutores de las pruebas información sobre los nombres y valores de las características que desee incluir en el archivo `device.json`. Por ejemplo, si quiere probar un dispositivo que funciona como servidor MQTT para otros dispositivos, puede configurar la lógica de prueba para validar los niveles admitidos específicos para una característica denominada `MQTT_QOS`. Los ejecutores de las pruebas proporcionan el nombre de esta característica y establecen su valor en los niveles de QOS compatibles con su dispositivo. Puede recuperar la información proporcionada del contexto de IDT con la `devicePool.features` consulta o del contexto del orquestador de pruebas con la `pool.features` consulta.

`features.name`

El nombre de la característica.

`features.value`

Los valores de la característica admitidos.

`features.configs`

Los ajustes de configuración de la característica, si son necesarios.

`features.config.name`

El nombre del ajuste de configuración.

`features.config.value`

Los valores de configuración admitidos.

`devices`

Una matriz de dispositivos en el grupo que se va a probar. Se requiere al menos un dispositivo.

`devices.id`

Un identificador único y definido por el usuario para el dispositivo que se está probando.

`connectivity.protocol`

El protocolo de comunicación que se usará para la comunicación con este dispositivo. Cada dispositivo de un grupo debe usar el mismo protocolo.

Actualmente, los únicos valores que se admiten son `ssh` y `uart` para dispositivos físicos, y `docker` para contenedores de Docker.

`connectivity.ip`

La dirección IP del dispositivo que se está probando.

Esta propiedad solo se aplica si `connectivity.protocol` está establecido en `ssh`.

`connectivity.port`

Opcional. El número de puerto que se va a utilizar para las conexiones SSH.

El valor predeterminado es 22.

Esta propiedad solo se aplica si `connectivity.protocol` está establecido en `ssh`.

`connectivity.auth`

Información de autenticación para la conexión.

Esta propiedad solo se aplica si `connectivity.protocol` está establecido en `ssh`.

`connectivity.auth.method`

El método de autenticación que se utiliza para acceder a un dispositivo a través de un determinado protocolo de conectividad.

Los valores admitidos son:

- `pki`
- `password`

`connectivity.auth.credentials`

Las credenciales que se utilizan para la autenticación.

`connectivity.auth.credentials.password`

La contraseña que se utiliza para iniciar sesión en el dispositivo que se va a probar.

Este valor solo se aplica si `connectivity.auth.method` está establecido en `password`.

`connectivity.auth.credentials.privKeyPath`

La ruta completa a la clave privada que se utiliza para iniciar sesión en el dispositivo que se está probando.

Este valor solo se aplica si `connectivity.auth.method` está establecido en `pki`.

`connectivity.auth.credentials.user`

El nombre de usuario para iniciar sesión en el dispositivo que se está probando.

`connectivity.serialPort`

Opcional. El puerto serie al que está conectado el dispositivo.

Esta propiedad solo se aplica si `connectivity.protocol` está establecido en `uart`.

`connectivity.containerId`

El ID de contenedor o el nombre del contenedor de Docker que se está probando.

Esta propiedad solo se aplica si `connectivity.protocol` está establecido en `ssh`.

`connectivity.containerUser`

Opcional. El nombre del usuario que se va a utilizar dentro del contenedor. El valor predeterminado es el usuario proporcionado en el Dockerfile.

El valor predeterminado es 22.

Esta propiedad solo se aplica si `connectivity.protocol` está establecido en `ssh`.

> **Note**
>
> Para comprobar si los ejecutores de la prueba configuran la conexión de dispositivo incorrecta para una prueba, puedes consultarla `pool.Devices[0].Connectivity.Protocol` desde el contexto del orquestador de pruebas y compararla con el valor esperado en un estado. `Choice` Si se utiliza un protocolo incorrecto, imprima un mensaje con el estado `LogMessage` y haga la transición al estado `Fail`.
> Como alternativa, puede utilizar un código de gestión de errores para informar de un fallo en la prueba para los tipos de dispositivos incorrectos.

(Opcional) Configuración de userdata.json

El archivo `userdata.json` contiene cualquier información adicional que requiera un conjunto de pruebas, pero que no esté especificada en el archivo `device.json`. El formato de este archivo depende del archivo `userdata_scheme.json` definido en el conjunto de pruebas. Si es un redactor

de pruebas, asegúrese de proporcionar esta información a los usuarios que van a ejecutar los conjuntos de pruebas que escriba.

(Opcional) Configuración de resource.json

El archivo `resource.json` contiene información sobre los dispositivos que se van a utilizar como dispositivos de recursos. Los dispositivos de recursos son dispositivos que se requieren para probar ciertas capacidades de un dispositivo que se está probando. Por ejemplo, para probar la capacidad Bluetooth de un dispositivo, puede usar un dispositivo de recursos para comprobar si el dispositivo se puede conectar correctamente a él. Los dispositivos de recursos son opcionales y puede requerir tantos dispositivos de recursos como necesite. Como redactor de pruebas, utilice el archivo test.json para definir las características del dispositivo de recursos que se requieren para una prueba. A continuación, los ejecutores de pruebas utilizan el archivo `resource.json` para proporcionar un grupo de dispositivos de recursos que tengan las funciones necesarias. Asegúrese de proporcionar esta información a los usuarios que vayan a ejecutar los conjuntos de pruebas que escriba.

Los ejecutores de pruebas pueden proporcionar esta información mediante el siguiente archivo `resource.json` de plantilla que se encuentra en la carpeta *<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>",
                }
            }
        ]
    }
]
```

Todos los campos que contienen valores son obligatorios tal y como se describe aquí:

`id`

Un ID alfanumérico definido por el usuario que identifica de forma única una colección de dispositivos que se conoce como grupo de dispositivos. Los dispositivos que pertenecen a un grupo deben tener idéntico hardware. Al ejecutar un conjunto de pruebas, los dispositivos del grupo se utilizan para paralelizar la carga de trabajo. Se utilizan varios dispositivos para ejecutar diferentes pruebas.

`features`

Opcional. Una matriz que contenga las características compatibles del dispositivo. La información requerida en este campo se define en los archivos test.json del conjunto de pruebas y determina qué pruebas se van a ejecutar y cómo se van a ejecutar. Si el conjunto de pruebas no requiere ninguna característica, este campo no es obligatorio.

`features.name`

El nombre de la característica.

`features.version`

La versión de la característica.

`features.jobSlots`

Configuración para indicar cuántas pruebas pueden utilizar el dispositivo simultáneamente. El valor predeterminado es 1.

`devices`

Una matriz de dispositivos en el grupo que se va a probar. Se requiere al menos un dispositivo.

`devices.id`

Un identificador único y definido por el usuario para el dispositivo que se está probando.

`connectivity.protocol`

El protocolo de comunicación que se usará para la comunicación con este dispositivo. Cada dispositivo de un grupo debe usar el mismo protocolo.

Actualmente, los únicos valores que se admiten son `ssh` y `uart` para dispositivos físicos, y `docker` para contenedores de Docker.

`connectivity.ip`

La dirección IP del dispositivo que se está probando.

Esta propiedad solo se aplica si `connectivity.protocol` está establecido en `ssh`.

`connectivity.port`

Opcional. El número de puerto que se va a utilizar para las conexiones SSH.

El valor predeterminado es 22.

Esta propiedad solo se aplica si `connectivity.protocol` está establecido en `ssh`.

`connectivity.auth`

Información de autenticación para la conexión.

Esta propiedad solo se aplica si `connectivity.protocol` está establecido en `ssh`.

`connectivity.auth.method`

El método de autenticación que se utiliza para acceder a un dispositivo a través de un determinado protocolo de conectividad.

Los valores admitidos son:

- `pki`
- `password`

`connectivity.auth.credentials`

Las credenciales que se utilizan para la autenticación.

`connectivity.auth.credentials.password`

La contraseña que se utiliza para iniciar sesión en el dispositivo que se va a probar.

Este valor solo se aplica si `connectivity.auth.method` está establecido en `password`.

`connectivity.auth.credentials.privKeyPath`

La ruta completa a la clave privada que se utiliza para iniciar sesión en el dispositivo que se está probando.

Este valor solo se aplica si `connectivity.auth.method` está establecido en `pki`.

`connectivity.auth.credentials.user`

El nombre de usuario para iniciar sesión en el dispositivo que se está probando.

`connectivity.serialPort`

Opcional. El puerto serie al que está conectado el dispositivo.

Esta propiedad solo se aplica si `connectivity.protocol` está establecido en `uart`.

`connectivity.containerId`

El ID de contenedor o el nombre del contenedor de Docker que se está probando.

Esta propiedad solo se aplica si `connectivity.protocol` está establecido en `ssh`.

`connectivity.containerUser`

Opcional. El nombre del usuario que se va a utilizar dentro del contenedor. El valor predeterminado es el usuario proporcionado en el Dockerfile.

El valor predeterminado es 22.

Esta propiedad solo se aplica si `connectivity.protocol` está establecido en `ssh`.

(Opcional) Configuración de config.json

El archivo `config.json` contiene la información de configuración para IDT. Por lo general, los ejecutores de pruebas no necesitarán modificar este archivo excepto para proporcionar sus credenciales de usuario de AWS para IDT y, opcionalmente, una región de AWS. Si se proporcionan las credenciales de AWS con los permisos necesarios, AWS IoT Device Tester recopila y envía las métricas de uso a AWS. Se trata de una característica opcional que se utiliza para mejorar la funcionalidad de IDT. Para obtener más información, consulte Métricas de uso de IDT.

Los ejecutores de pruebas pueden configurar sus credenciales de AWS de una de las siguientes maneras:

- Archivo de credenciales

  IDT utiliza el mismo archivo de credenciales que la AWS CLI. Para obtener más información, consulte Archivos de configuración y credenciales.

  La ubicación del archivo de credenciales varía en función del sistema operativo que utilice:

  - macOS, Linux: `~/.aws/credentials`
  - Windows: `C:\Users\UserName\.aws\credentials`
- Variables de entorno

  Las variables de entorno son las variables que mantiene el sistema operativo y utilizan los comandos del sistema. Las variables definidas durante una sesión de SSH no están disponibles una vez cerrada la sesión. IDT puede usar las variables de entorno `AWS_ACCESS_KEY_ID` y `AWS_SECRET_ACCESS_KEY` para almacenar sus credenciales de AWS.

  Para establecer estas variables en Linux, MacOS, o Unix, utilice export:

  ```
  export AWS_ACCESS_KEY_ID=<your_access_key_id>
  export AWS_SECRET_ACCESS_KEY=<your_secret_access_key>
  ```

  Para establecer estas variables en Windows, utilice set:

  ```
  set AWS_ACCESS_KEY_ID=<your_access_key_id>
  set AWS_SECRET_ACCESS_KEY=<your_secret_access_key>
  ```

Para configurar las credenciales de AWS para IDT, los ejecutores de pruebas editan la sección `auth` del archivo `config.json` ubicado en la carpeta *<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>"
        }
    }
}
]
```

Todos los campos que contienen valores son obligatorios tal y como se describe aquí:

> **Note**
>
> Todas las rutas de este archivo se definen en relación con *< device-tester-extract-location >*.

`log.location`

La ruta a la carpeta de registros de la carpeta *< device-tester-extract-location >*.

`configFiles.root`

La ruta a la carpeta que contiene los archivos de configuración.

`configFiles.device`

La ruta al archivo `device.json`.

`testPath`

La ruta a la carpeta que contiene los conjuntos de pruebas.

`reportPath`

La ruta a la carpeta que contendrá los resultados de las pruebas después de que IDT ejecute un conjunto de pruebas.

`awsRegion`

Opcional. La región de AWS que utilizarán los conjuntos de pruebas. Si no se establece, los conjuntos de pruebas utilizarán la región predeterminada especificada en cada conjunto de pruebas.

`auth.method`

El método que IDT utiliza para recuperar las credenciales de AWS. Los valores admitidos son `file` para recuperar las credenciales de un archivo de credenciales y `environment` para recuperar las credenciales mediante variables de entorno.

`auth.credentials.profile`

El perfil de credenciales que se va a utilizar del archivo de credenciales. Esta propiedad solo se aplica si `auth.method` está establecido en `file`.

Depurar y ejecutar conjuntos de pruebas personalizadas

Una vez establecida la configuración requerida, IDT puede ejecutar su conjunto de pruebas. El tiempo de ejecución del conjunto de pruebas completa depende del hardware y de la composición del conjunto de pruebas. Como referencia, se tarda aproximadamente 30 minutos en completar el conjunto AWS IoT Greengrass de pruebas completo en una unidad Raspberry Pi 3B.

Mientras escribe su conjunto de pruebas, puede usar IDT para ejecutarla en modo de depuración, comprobar el código antes de ejecutarla o proporcionárselo a los ejecutores de pruebas.

Ejecución de IDT en modo de depuración

Como los conjuntos de pruebas dependen de IDT para interactuar con los dispositivos, proporcionar el contexto y recibir los resultados, no puede simplemente depurar sus conjuntos de pruebas en un IDE sin ninguna interacción con IDT. Para ello, la CLI de IDT proporciona el comando `debug-test-suite`, que permite ejecutar IDT en modo de depuración. Ejecute el siguiente comando para ver las opciones disponibles para `debug-test-suite`:

```
devicetester_[linux | mac | win_x86-64] debug-test-suite -h
```

Cuando se ejecuta IDT en modo de depuración, IDT no inicia realmente el conjunto de pruebas ni ejecuta orquestador de pruebas, sino que interactúa con el IDE para responder a las solicitudes realizadas desde el conjunto de pruebas que se ejecuta en el IDE e imprime los registros en la consola. IDT no agota el tiempo de espera y espera a salir hasta que se interrumpa manualmente. En el modo de depuración, IDT tampoco ejecuta el orquestador de pruebas y no generará ningún archivo de informe. Para depurar su conjunto de pruebas, debe usar su IDE para proporcionar cierta información que IDT suele obtener de los archivos JSON de configuración. Asegúrese de que dispone de la siguiente información:

- Variables de entorno y argumentos para cada prueba. IDT no leerá esta información de `test.json` ni `suite.json`.
- Argumentos para seleccionar los dispositivos de recursos. IDT no leerá esta información de `test.json`.

Para depurar los conjuntos de pruebas, complete los pasos siguientes:

1. Cree los archivos de configuración de ajustes necesarios para ejecutar el conjunto de pruebas. Por ejemplo, si su conjunto de pruebas requiere `device.json`, `resource.json` y `user data.json`, asegúrese de configurarlos todos según sea necesario.
2. Ejecute el siguiente comando para establecer IDT en modo de depuración y seleccione los dispositivos necesarios para ejecutar la prueba.

   ```
   devicetester_[linux | mac | win_x86-64] debug-test-suite [options]
   ```

   Tras ejecutar este comando, IDT espera las solicitudes del conjunto de pruebas y, a continuación, responde a ellas. IDT también genera las variables de entorno que se requieran para el proceso de casos para el SDK de cliente de IDT.
3. En su IDE, utilice la configuración `run` o `debug` para hacer lo siguiente:

   a. Establecer los valores de las variables de entorno generadas por IDT.

   b. Establecer el valor de cualquier variable de entorno o argumento que haya especificado en el archivo `test.json` y `suite.json`.

   c. Establecer los puntos de interrupción según sea necesario.
4. Ejecute el conjunto de pruebas en su IDE.

Puede depurar y volver a ejecutar el conjunto de pruebas tantas veces como sea necesario. En el modo de depuración, IDT no agota el tiempo de espera.

5. Una vez completada la depuración, interrumpa IDT para salir del modo de depuración.

Comandos de la CLI de IDT para ejecutar pruebas

En la sección siguiente se describen los comandos de la CLI de IDT.

IDT v4.0.0

`help`

Enumera información acerca del comando especificado.

`list-groups`

Muestra los grupos de un conjunto de prueba determinado.

`list-suites`

Muestra los conjuntos de prueba disponibles.

`list-supported-products`

Enumera los productos compatibles con su versión de IDT, en este caso las versionesAWS IoT Greengrass de cualificaciones AWS IoT Greengrass y las versiones del conjunto de pruebas para la versión actual de IDT.

`list-test-cases`

Enumera los casos de prueba en un grupo de prueba determinado. Se admite la siguiente opción:

- `group-id`. El grupo de pruebas que se va a buscar. Esta opción es necesaria y debe especificar un solo grupo.

`run-suite`

Ejecuta un conjunto de pruebas en un grupo de dispositivos. Estas son algunas opciones que suelen utilizarse:

- `suite-id`. La versión del conjunto de pruebas que se va a ejecutar. Si no se especifica, IDT utiliza la versión más reciente de la carpeta `tests`.

- `group-id`. Los grupos de pruebas que se van a ejecutar, como una lista separada por comas. Si no se especifica, IDT ejecuta todos los grupos de prueba del conjunto de pruebas.
- `test-id`. Los casos de prueba que se van a ejecutar, como una lista separada por comas. Cuando se especifique, `group-id` debe especificar un solo grupo.
- `pool-id`. El grupo de dispositivos que se va a probar. Los ejecutores de las pruebas deben especificar un grupo si tienen varios grupos de dispositivos definidos en el archivo `device.json`.
- `timeout-multiplier`. Configura IDT para modificar el tiempo de espera de ejecución de la prueba especificado en el archivo `test.json` para una prueba con un multiplicador definido por el usuario.
- `stop-on-first-failure`. Configura IDT para detener la ejecución en el primer error. Esta opción debe utilizarse con `group-id` para depurar los grupos de prueba especificados.
- `userdata`. Establece el archivo que contiene la información sobre los datos del usuario necesarios para ejecutar el conjunto de pruebas. Esto solo es necesario si `userdataRequired` está establecido en verdadero en el archivo `suite.json` del conjunto de pruebas.

Para obtener más información acerca de `run-suite` las opciones, utilice la opción `help`:

```
devicetester_[linux | mac | win_x86-64] run-suite -h
```

`debug-test-suite`

Ejecute el conjunto de pruebas en modo de depuración. Para obtener más información, consulte Ejecución de IDT en modo de depuración.

Revisar los resultados y registros de las pruebas IDT

En esta sección se describe el formato en que IDT genera registros de consola e informes de prueba.

Formato de mensajes de consola

AWS IoT Device Testerutiliza un formato estándar para imprimir mensajes en la consola cuando inicia un conjunto de pruebas. En el fragmento siguiente, se muestra un ejemplo de un mensaje de consola generado por IDT.

```
time="2000-01-02T03:04:05-07:00" level=info msg=Using suite: MyTestSuite_1.0.0
executionId=9a52f362-1227-11eb-86c9-8c8590419f30
```

La mayoría de los mensajes de consola constan de los campos siguientes:

`time`

Marca de hora ISO 8601 completa para el evento registrado.

`level`

Nivel de mensaje del evento registrado. Normalmente, el nivel de mensajes registrados es uno de los`info`,`warn`, o bien`error`. El IDT emite un`fatal`o`panic`mensaje si encuentra un evento esperado que hace que salga antes de tiempo.

`msg`

El mensaje registrado.

`executionId`

Cadena de ID exclusiva para el proceso IDT actual. Este ID se utiliza para diferenciar entre ejecuciones IDT individuales.

Los mensajes de consola generados a partir de un conjunto de pruebas proporcionan información adicional sobre el dispositivo en prueba y el conjunto de pruebas, el grupo de pruebas y los casos de prueba que IDT ejecuta. En el siguiente extracto se muestra un ejemplo de un mensaje de consola generado a partir de un conjunto de pruebas.

```
time="2000-01-02T03:04:05-07:00" level=info msg=Hello world! suiteId=MyTestSuite
groupId=myTestGroup testCaseId=myTestCase deviceId=my-device
executionId=9a52f362-1227-11eb-86c9-8c8590419f30
```

La parte específica del conjunto de pruebas del mensaje de la consola contiene los siguientes campos:

`suiteId`

El nombre del conjunto de pruebas que se está ejecutando actualmente.

`groupId`

ID del grupo de prueba que se está ejecutando actualmente.

`testCaseId`

El ID del caso de prueba que se está ejecutando.

`deviceId`

Identificador del dispositivo que se está probando que está utilizando el caso de prueba actual.

Para imprimir un resumen de prueba en la consola cuando un IDT termina de ejecutar una prueba, debe incluir un`Reportstate`en tu orquestador de pruebas. El resumen de la prueba contiene información sobre el conjunto de pruebas, los resultados de las pruebas de cada grupo que se ha ejecutado y las ubicaciones de los registros generados y los archivos de informe. En el siguiente ejemplo se muestra un mensaje de resumen de prueba.

```
========== Test Summary ==========
Execution Time:     5m00s
Tests Completed:    4
Tests Passed:       3
Tests Failed:       1
Tests Skipped:      0
----------------------------------
Test Groups:
    GroupA:         PASSED
    GroupB:         FAILED
----------------------------------
Failed Tests:
    Group Name: GroupB
        Test Name: TestB1
            Reason: Something bad happened
----------------------------------
Path to IoT Device Tester Report: /path/to/awsiotdevicetester_report.xml
Path to Test Execution Logs: /path/to/logs
Path to Aggregated JUnit Report: /path/to/MyTestSuite_Report.xml
```

AWS IoT Device TesterEsquema de informe

`awsiotdevicetester_report.xml`es un informe firmado que contiene la siguiente información:

- La versión de IDT.
- La versión del conjunto de pruebas.
- Firma del informe y clave utilizadas para firmar el informe.

- El SKU del dispositivo y el grupo de dispositivos especificado en`device.json`file.
- La versión del producto y las características del dispositivo probadas.
- El resumen de agregación de los resultados de las pruebas. Esta información es la misma que la contenida en el*suite-name*`_report.xml`file.

```
<apnreport>
    <awsiotdevicetesterversion>idt-version</awsiotdevicetesterversion>
    <testsuiteversion>test-suite-version</testsuiteversion>
    <signature>signature</signature>
    <keyname>keyname</keyname>
    <session>
        <testsession>execution-id</testsession>
        <starttime>start-time</starttime>
        <endtime>end-time</endtime>
    </session>
    <awsproduct>
        <name>product-name</name>
        <version>product-version</version>
        <features>
            <feature name="<feature-name>" value="supported | not-supported | <feature-
value>" type="optional | required"/>
        </features>
    </awsproduct>
    <device>
        <sku>device-sku</sku>
        <name>device-name</name>
        <features>
            <feature name="<feature-name>" value="<feature-value>"/>
        </features>
        <executionMethod>ssh | uart | docker</executionMethod>
    </device>
    <devenvironment>
        <os name="<os-name>"/>
    </devenvironment>
    <report>
        <suite-name-report-contents>
    </report>
</apnreport>
```

El archivo `awsiotdevicetester_report.xml` contiene una etiqueta `<awsproduct>` que tiene información sobre el producto que se está probando y las características del producto que se han validado después de ejecutar un conjunto de pruebas.

Atributos que se utilizan en la etiqueta **`<awsproduct>`**

`name`

El nombre del producto que se está probando.

`version`

La versión del producto que se está probando.

`features`

Las características validadas. Funciones marcadas como`required`son necesarios para que el conjunto de pruebas valide el dispositivo. En el siguiente fragmento se muestra cómo aparece esta información en el archivo `awsiotdevicetester_report.xml`:

```
<feature name="ssh" value="supported" type="required"></feature>
```

Funciones marcadas como`optional`no son necesarios para la validación. Los siguientes fragmentos muestran características opcionales:

```
<feature name="hsi" value="supported" type="optional"></feature>

<feature name="mqtt" value="not-supported" type="optional"></feature>
```

Esquema de informe del grupo de pruebas

El informe *`suite-name`*`_Result.xml` está en formato XML JUnit. Puede integrarlo en plataformas de integración/implementación continua como Jenkins, Bamboo, etc. El informe contiene un resumen de agregación de los resultados de pruebas.

```
<testsuites name="<suite-name> results" time="<run-duration>" tests="<number-of-test>"
 failures="<number-of-tests>" skipped="<number-of-tests>" errors="<number-of-tests>"
 disabled="0">
    <testsuite name="<test-group-id>" package="" tests="<number-of-tests>"
 failures="<number-of-tests>" skipped="<number-of-tests>" errors="<number-of-tests>"
 disabled="0">
        <!--success-->
```

```
        <testcase classname="<classname>" name="<name>" time="<run-duration>"/>
        <!--failure-->
        <testcase classname="<classname>" name="<name>" time="<run-duration>">
            <failure type="<failure-type>">
                reason
            </failure>
        </testcase>
        <!--skipped-->
        <testcase classname="<classname>" name="<name>" time="<run-duration>">
            <skipped>
                reason
            </skipped>
        </testcase>
        <!--error-->
        <testcase classname="<classname>" name="<name>" time="<run-duration>">
            <error>
                reason
            </error>
        </testcase>
    </testsuite>
</testsuites>
```

La sección de informes de ambos`awsiotdevicetester_report.xml`o*`suite-name`*`_report.xml`Enumera las pruebas que se ejecutaron y los resultados.

La primera etiqueta XML `<testsuites>` contiene el resumen de la ejecución de las pruebas. Por ejemplo:

```
<testsuites name="MyTestSuite results" time="2299" tests="28" failures="0" errors="0"
 disabled="0">
```

Atributos que se utilizan en la etiqueta **`<testsuites>`**

`name`

El nombre del grupo de prueba.

`time`

El tiempo, en segundos, que se ha tardado en ejecutar el conjunto de pruebas.

`tests`

El número de pruebas ejecutadas.

`failures`

El número de pruebas que se ejecutaron, pero que no se superaron.

`errors`

El número de pruebas que IDT no ha podido ejecutar.

`disabled`

Este atributo no se utiliza y se puede omitir.

Si se producen errores en pruebas, puede identificar la prueba fallido revisando las etiquetas XML `<testsuites>`. Las etiquetas XML `<testsuite>` dentro de la etiqueta `<testsuites>` muestran el resumen del resultado de la prueba de un grupo de prueba. Por ejemplo:

```
<testsuite name="combination" package="" tests="1" failures="0" time="161" disabled="0"
 errors="0" skipped="0">
```

El formato es similar a la etiqueta `<testsuites>`, pero con un atributo `skipped` que no se utiliza y que se puede pasar por alto. Dentro de cada etiqueta XML `<testsuite>`, hay etiquetas `<testcase>` para cada prueba ejecutada para un grupo de prueba. Por ejemplo:

```
<testcase classname="Security Test" name="IP Change Tests" attempts="1"></testcase>>
```

Atributos que se utilizan en la etiqueta **`<testcase>`**

`name`

El nombre de la prueba.

`attempts`

El número de veces que IDT ha ejecutado el caso de prueba.

Cuando una prueba genera un error o si se produce un error, las etiquetas `<failure>` o `<error>` se añaden a la etiqueta `<testcase>` con información para la resolución de problemas. Por ejemplo:

```
<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>
```

Métricas de uso de IDT

Si proporciona AWS credenciales con los permisos necesarios, AWS IoT Device Tester recopila y envía las métricas de uso a AWS. Se trata de una característica opcional que se utiliza para mejorar la funcionalidad de IDT. IDT recopila información como la siguiente:

- El Cuenta de AWS ID utilizado para ejecutar IDT
- Los AWS CLI comandos de IDT utilizados para ejecutar las pruebas
- Los conjuntos de pruebas que se ejecutan
- Los conjuntos de pruebas de la carpeta *< device-tester-extract-location >*
- La cantidad de dispositivos configurados en el grupo de dispositivos
- Los nombres de casos de prueba y los tiempos de ejecución
- La información sobre los resultados de las pruebas, por ejemplo, si se han superado, si han fallado, si se han encontrado errores o si se han omitido
- Las características del producto probadas
- El comportamiento de salida de IDT, como salidas inesperadas o anticipadas

Toda la información que IDT envía también se registra en un archivo `metrics.log` de la carpeta *<device-tester-extract-location>*`/results/`*<execution-id>*`/`. Puede consultar el archivo de registro para ver la información recopilada durante la ejecución de una prueba. Este archivo se genera solo si elige recopilar métricas de uso.

Para deshabilitar la recopilación de métricas, no es necesario que realice ninguna acción adicional. Simplemente no almacene sus AWS credenciales y, si AWS las tiene almacenadas, no configure el `config.json` archivo para acceder a ellas.

Configure sus AWS credenciales

Si aún no tiene una Cuenta de AWS, debe crear una. Si ya tiene una Cuenta de AWS, solo tiene que configurar los permisos necesarios para su cuenta para que IDT pueda enviarle las métricas de uso AWS en su nombre.

Paso 1: crear una Cuenta de AWS

En este paso, cree y configure una Cuenta de AWS. Si ya tiene una Cuenta de AWS, vaya directamente a the section called “Paso 2: Configurar los permisos de IDT”.

Si no tiene una Cuenta de AWS, complete los siguientes pasos para crearla.

Para suscribirse a una Cuenta de AWS

1. Abra https://portal.aws.amazon.com/billing/signup.
2. Siga las instrucciones que se le indiquen.

   Parte del procedimiento de registro consiste en recibir una llamada telefónica e indicar un código de verificación en el teclado del teléfono.

   Cuando te registras en una Cuenta de AWS, Usuario raíz de la cuenta de AWSse crea una. El usuario raíz tendrá acceso a todos los Servicios de AWS y recursos de esa cuenta. Como práctica recomendada de seguridad, asigne acceso administrativo a un usuario y utilice únicamente el usuario raíz para realizar tareas que requieren acceso de usuario raíz.

Para crear un usuario administrador, elija una de las siguientes opciones.

Elegir una forma de administrar el administrador	Para	Haga esto	También puede
En IAM Identity Center (recomendado)	Usar credenciales a corto plazo para acceder a AWS. Esto se ajusta a las prácticas recomendadas de seguridad. Para obtener información sobre las prácticas recomendadas, consulte Prácticas	Siga las instrucciones en Introducción en la Guía del usuario de AWS IAM Identity Center .	Configure el acceso programático configurando el AWS CLI que se utilizará AWS IAM Identity Center en la Guía del AWS Command Line Interface usuario.

Elegir una forma de administrar el administrador	Para	Haga esto	También puede
	recomendadas de seguridad en IAM en la Guía del usuario de IAM.		
En IAM (no recomendado)	Usar credenciales a largo plazo para acceder a AWS.	Siga las instrucciones en Creación del primer grupo de usuarios y usuario de administrador de IAM en la Guía del usuario de IAM.	Configurar el acceso programático mediante Administración de las claves de acceso de los usuarios de IAM en la Guía del usuario de IAM.

Paso 2: Configurar los permisos de IDT

En este paso, configure los permisos que IDT utiliza para ejecutar las pruebas y recopilar datos de uso de IDT. Puede usar AWS Management Console o AWS Command Line Interface (AWS CLI) para crear una política de IAM y un usuario para IDT y, a continuación, adjuntar políticas al usuario.

- Configuración de permisos para IDT (consola)
- Configuración de permisos para IDT (AWS CLI)

Configuración de permisos de IDT (consola)

Siga estos pasos para usar la consola para configurar permisos para IDT para AWS IoT Greengrass.

1. Inicie sesión en la consola de IAM.
2. Crear una política administrada que conceda permisos para crear roles con permisos específicos.

a. En el panel de navegación, seleccione Políticas y, a continuación, Crear política.

b. En la pestaña JSON, reemplace el contenido del marcador de posición por la política siguiente.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot-device-tester:SendMetrics"
            ],
            "Resource": "*"
        }
    ]
}
```

c. Elija Revisar política.

d. En Nombre, ingrese **`IDTUsageMetricsIAMPermissions`**. En Summary (Resumen), revise los permisos concedidos por la política.

e. Elija Crear política.

3. Cree un usuario de IAM y adjunte los permisos al usuario.

a. Cree un usuario de IAM. Siga los pasos del 1 al 5 en Creación de usuarios de IAM (consola) en la Guía del usuario de IAM. Si ya ha creado un usuario de IAM, pase directamente al siguiente paso.

b. Adjunte los permisos a su usuario de IAM:

i. En la página Set permissions (Establecer permisos), elija Attach existing policies to user directly (Adjuntar políticas existentes al usuario directamente).

ii. Busque la política UsageMetricsIAMPermissions de IDT que creó en el paso anterior. Seleccione la casilla de verificación.

c. Elija Siguiente: etiquetas.

d. Elija Next: Review (Siguiente: revisar) para ver un resumen de sus opciones.

e. Seleccione la opción Crear un usuario.

f. Para ver las claves de acceso del usuario (ID de clave de acceso y claves de acceso secretas), elija Show (Mostrar) junto a la contraseña y la clave de acceso. Para guardar las

claves de acceso, elija Download.csv (Descargar archivo .csv) y, a continuación, guarde el archivo en un lugar seguro. Utilizará esta información más adelante para configurar el archivo de credenciales. AWS

Configuración de permisos de IDT (AWS CLI)

Siga estos pasos AWS CLI para configurar los permisos de IDT para AWS IoT Greengrass.

1. En su ordenador, instale y configure el AWS CLI si aún no está instalado. Siga los pasos que se indican en Instalación de la AWS CLI en la Guía del usuario de la AWS Command Line Interface .

   > **Note**
   >
   > AWS CLI Se trata de una herramienta de código abierto que puede utilizar para interactuar con los AWS servicios desde el shell de la línea de comandos.

2. Cree la siguiente política administrada por el cliente que otorgue permisos para administrar el IDT y las funciones. 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**
>
> Este paso incluye un ejemplo de símbolo del sistema de Windows porque utiliza una sintaxis JSON diferente a la de los comandos de terminal Linux, macOS o Unix.

PowerShell

```
aws iam create-policy --policy-name IDTUsageMetricsIAMPermissions --policy-
document '{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot-device-tester:SendMetrics"
            ],
            "Resource": "*"
        }
    ]
}'
```

3. Cree un usuario de IAM y adjunte los permisos requeridos por IDT para AWS IoT Greengrass.

   a. Cree un usuario de IAM.

   ```
   aws iam create-user --user-name user-name
   ```

   b. Adjunte la política `IDTUsageMetricsIAMPermissions` que ha creado a su nuevo usuario de IAM. Sustituya *el nombre de usuario* por su nombre de usuario de IAM y *<account-id>* en el comando con la identificación de su Cuenta de AWS.

```
aws iam attach-user-policy --user-name user-name --policy-arn
 arn:aws:iam::<account-id>:policy/IDTGreengrassIAMPermissions
```

4. Cree una clave de acceso secreta para el usuario.

   ```
   aws iam create-access-key --user-name user-name
   ```

   Almacene la salida en una ubicación segura. Utilizará esta información más adelante para configurar el archivo de AWS credenciales.

Proporcione AWS las credenciales a IDT

Para permitir que IDT acceda a sus AWS credenciales y envíe las métricas a ellas AWS, haga lo siguiente:

1. Guarde las AWS credenciales de su usuario de IAM como variables de entorno o en un archivo de credenciales:

   a. Para usar variables de entorno, ejecute los siguientes comandos.

      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. Para usar el archivo de credenciales, añada la siguiente información al `~/.aws/credentials` archivo.

      ```
      [profile-name]
      ```

```
aws_access_key_id=access-key
aws_secret_access_key=secret-access-key
```

2. Configure la sección `auth` del archivo `config.json`. Para obtener más información, consulte (Opcional) Configuración de config.json.

Solución de problemas de IDT paraAWS IoT GreengrassV2

IDT paraAWS IoT GreengrassV2 escribe los errores en varias ubicaciones según el tipo de error. IDT escribe los errores en la consola, los archivos de registro y los informes de las pruebas.

¿Dónde buscar errores

Los errores de alto nivel se muestran en la consola mientras se ejecuta la prueba y, una vez finalizadas todas las pruebas, aparece un resumen de las pruebas fallidas.`awsiotdevicetester_report.xml`contiene un resumen de todos los errores que provocaron el error de una prueba. IDT almacena los archivos de registro de cada ejecución de prueba en un directorio con un UUID para la ejecución de la prueba, que se muestra en la consola durante la ejecución de la prueba.

El directorio de registros de pruebas de IDT es*<device-tester-extract-location>*/`results/`*<execution-id>*`/logs/`. Este directorio contiene los siguientes archivos que se muestran en la tabla. Esto es útil a efectos de depuración.

Archivos	Descripción
`test_manager.log`	Los registros escritos en la consola mientras se estaba ejecutando la prueba. El resumen de los resultados al final de este archivo incluye una lista de las pruebas que no superaron. La advertencia y los registros de errores en este archivo pueden proporcionarle información acerca de los errores que se producen.
test-group-id /*test-case-id* /*test-name* `.log`	Registros detallados de la prueba específica en un grupo de prueba. En el caso de las pruebas que utilizan componentes de Greengrass, el

Archivos	Descripción
	archivo de registro de casos de prueba se denominagreengrass-test-run.log .
test-group-id /*test-case-id* / greengrass.log	Registros detallados deAWS IoT Greengras sSoftware básico. IDT copia este archivo del dispositivo que se está probando cuando ejecuta las pruebas de instalaciónAWS IoT GreengrassSoftware básico del dispositi vo. Para obtener más información sobre los mensajes de este archivo de registro, consulteSolución de problemas AWS IoT Greengrass V2.
test-group-id /*test-case-id*/*component-name* .log	Registros detallados de los componentes de Greengrass que se implementan durante las pruebas. IDT copia los archivos de registro de los componentes del dispositivo que se está probando cuando ejecuta pruebas en las que se despliegan componentes específic os. El nombre de cada archivo de registro de componentes corresponde al nombre del componente desplegado. Para obtener más información sobre los mensajes de este archivo de registro, consulteSolución de problemas AWS IoT Greengrass V2.

Resolver IDT paraAWS IoT GreengrassErrores V2

Antes de ejecutar IDT paraAWS IoT Greengrass, coloque los archivos de configuración correctos. Si recibe errores de análisis y configuración, lo primero que debe hacer es buscar y utilizar una plantilla de configuración adecuada para su entorno.

Si continúa teniendo problemas, consulte el siguiente proceso de depuración.

Temas

Errores de resolución de alias

Al ejecutar conjuntos de pruebas personalizados, es posible que aparezca el siguiente error en la consola y en el`test_manager.log`.

```
Couldn't resolve placeholders: couldn't do a json lookup: index out of range
```

Este error puede producirse cuando los alias configurados en el orquestador de pruebas de IDT no se resuelven correctamente o si los valores resueltos no están presentes en los archivos de configuración. Para resolver este error, asegúrese de que`device.json`y`userdata.json` contienen la información correcta requerida para su conjunto de pruebas. Para obtener información sobre la configuración requerida paraAWS IoT Greengrasscalificación, consulteConfigure los ajustes de IDT para ejecutar el conjunto de AWS IoT Greengrass cualificación.

Errores de conflicto

Es posible que aparezca el siguiente error al ejecutar elAWS IoT Greengrassconjunto de cualificaciones simultáneamente en más de un dispositivo.

```
ConflictException: Component [com.example.IDTHelloWorld : 1.0.0] for account [account-id] already exists with state: [DEPLOYABLE] { RespMetadata: { StatusCode: 409,
 RequestID: “id” }, Message_: “Component [com.example.IDTHelloWorld : 1.0.0] for
 account [account-id] already exists with state: [DEPLOYABLE]” }
```

La ejecución simultánea de pruebas aún no es compatible con elAWS IoT Greengrasssuite de calificación. Ejecute el conjunto de calificaciones de forma secuencial para cada dispositivo.

Error por la imposibilidad de iniciar una prueba

Es posible que encuentres errores que apunten a fallos que se produjeron cuando la prueba intentaba comenzar. Existen varias causas posibles, por lo que debe hacer lo siguiente:

- Asegúrese de que el nombre del grupo que aparece en el comando de ejecución realmente existe. IDT hace referencia al nombre del grupo directamente desde su`device.json`archivo.
- Asegúrese de que el dispositivo o dispositivos del grupo tienen parámetros de configuración correctos.

La imagen de calificación de Docker contiene errores

Las pruebas de calificación del administrador de aplicaciones de Docker utilizan el`amazon/amazon-ec2-metadata-mock`imagen del contenedor en Amazon ECR para calificar el dispositivo sometido a prueba.

Es posible que reciba el siguiente error si la imagen ya está presente en un contenedor Docker del dispositivo que se está probando.

```
The Docker image amazon/amazon-ec2-metadata-mock:version already exists on the device.
```

Si anteriormente descargó esta imagen y ejecutó el`amazon/amazon-ec2-metadata-mock`contenedor de su dispositivo, asegúrese de eliminar esta imagen del dispositivo que se está probando antes de realizar las pruebas de calificación.

No se pudo leer la credencial

Al probar dispositivos Windows, es posible que encuentre el`Failed to read credential`error en el`greengrass.log`archivo si el usuario que utiliza para conectarse al dispositivo que se está probando no está configurado en el administrador de credenciales de ese dispositivo.

Para resolver este error, configure el usuario y la contraseña del usuario IDT en el administrador de credenciales del dispositivo que se está probando.

Para obtener más información, consulte Configura las credenciales de usuario para los dispositivos Windows.

Controle los errores con PreInstalled Greengrass

Mientras ejecutas IDT con PreInstalled Greengrass, si encuentra un error de`GuiceoErrorInCustomProvider`, compruebe si el archivo`userdata.json`tiene el`InstalledDirRootOnDevice`establecido en la carpeta de instalación de Greengrass. IDT busca el archivo`effectiveConfig.yaml`debajo de`<InstallationDirRootOnDevice>/config/effectiveConfig.yaml`.

Para obtener más información, consulte Configura las credenciales de usuario para los dispositivos Windows.

Excepción de firma no válida

Al ejecutar las pruebas de calificación de Lambda, es posible que encuentre el`invalidsignatureexception`error si su máquina host IDT tiene problemas de acceso a la red. Reinicie el router y vuelva a ejecutar las pruebas.

Errores de calificación del aprendizaje automático

Al ejecutar pruebas de calificación de aprendizaje automático (ML), es posible que se produzcan errores de calificación si el dispositivo no cumple con los requisitosrequerimientospara desplegar elAWS-componentes de aprendizaje automático proporcionados. Para solucionar los errores de calificación del aprendizaje automático, haga lo siguiente:

- Busque los detalles de los errores en los registros de los componentes que se implementaron durante la ejecución de la prueba. Los registros de los componentes se encuentran en*<device-tester-extract-location>*/results/*<execution-id>*/logs/*<test-group-id>*directorio.

- Añada el`-Dgg.persist=installed.software`argumento al`test.json`archivo para el caso de prueba fallido. El`test.json`el archivo se encuentra en*<device-tester-extract-location>*`/tests/GGV2Q_`*`version`*` directory.`

Implementaciones fallidas de Open Test Framework (OTF)

Si las pruebas de OTF no completan la implementación, una causa probable podrían ser los permisos establecidos para la carpeta principal de`TempResourcesDirOnDevice`o`InstallationDirRootOnDevice`. Para configurar los permisos de esta carpeta correctamente, ejecute el siguiente comando. Sustituya*`folder-name`*por el nombre de la carpeta principal.

```
sudo chmod 755 folder-name
```

Errores de procesamiento

Los errores tipográficos en una configuración de JSON pueden provocar errores de análisis. En la mayoría de los casos, el problema es resultado de omitir un paréntesis, una coma o unas comillas en el archivo JSON. IDT realiza la validación JSON e imprime información de depuración. Imprime la línea en la que se produjo el error, el número de línea y el número de la columna del error de sintaxis. Esta información debería ser suficiente para ayudarte a corregir el error, pero si sigues sin poder localizarlo, puedes realizar la validación manualmente en tu IDE, en un editor de texto como Atom o Sublime, o a través de una herramienta online como JSONlint.

Errores de permiso denegado

IDT realiza operaciones en diversos directorios y archivos en un dispositivo que se está probando. Algunas de estas operaciones requieren acceso raíz. Para automatizar estas operaciones, IDT debe ser capaz de ejecutar comandos con sudo sin escribir una contraseña.

Siga estos pasos para permitir acceso sudo sin escribir una contraseña.

> **Note**
>
> `user`y`username`hacen referencia al usuario SSH que utiliza IDT para acceder al dispositivo a prueba.

1. Use sudo usermod -aG sudo ***<ssh-username>*** para añadir el usuario SSH al grupo sudo.

2. Cierre la sesión y, a continuación, vuelva a iniciar sesión para que los cambios surtan efecto.
3. Añada el archivo `/etc/sudoers` y, a continuación, agregue la siguiente línea al final del archivo: `<ssh-username> ALL=(ALL) NOPASSWD: ALL`

> **Note**
>
> Le recomendamos que utilice sudo visudo al editar `/etc/sudoers`.

Error al generar el informe de calificación

IDT admite las cuatro últimas*major*.*minor*versiones delAWS IoT Greengrasspaquete de cualificación V2 (GGV2Q) para generar informes de cualificación que puede enviar aAWS Partner Networkpara incluir sus dispositivos en elAWS PartnerCatálogo de dispositivos. Las versiones anteriores del paquete de cualificaciones no generan informes de cualificación.

Si tiene preguntas sobre la política de soporte, póngase en contacto conAWS Support.

Error por ausencia de un parámetro obligatorio

Cuando IDT agrega nuevas funciones, puede introducir cambios en los archivos de configuración. Utilizar un archivo de configuración antiguo podría romper la configuración. Si esto ocurre, el archivo *<test_case_id>*`.log` en `/results/`*<execution-id>*`/logs` enumera explícitamente todos los parámetros que faltan. IDT también valida los esquemas de los archivos de configuración JSON para comprobar que está utilizando la última versión compatible.

Excepción de seguridad en macOS

Cuando ejecutas IDT en un ordenador host macOS, se bloquea la ejecución de IDT. Para ejecutar IDT, conceda una excepción de seguridad a los ejecutables que forman parte de la funcionalidad de ejecución de IDT. Cuando aparezca el mensaje de advertencia en el equipo host, haga lo siguiente para cada uno de los ejecutables aplicables:

Para conceder una excepción de seguridad a los ejecutables de IDT

1. En el ordenador macOS, en el menú Apple, abrePreferencias del sistema.
2. EligeSeguridad y privacidad, luego en elGeneralpestaña, selecciona el icono de candado para realizar cambios en la configuración de seguridad.

3. En caso de bloqueo`devicetester_mac_x86-64`, busca el mensaje`"devicetester_mac_x86-64" was blocked from use because it is not from an identified developer.`y eligePermitir de todos modos.
4. Reanude las pruebas de IDT hasta que haya revisado todos los ejecutables involucrados.

Errores de conexión SSH

Cuando IDT no puede conectarse a un dispositivo que se está probando, registra los fallos de conexión`/results/<execution-id>/logs/<test-case-id>.log`. Los mensajes SSH aparecen en la parte superior de este archivo de registro porque la conexión a un dispositivo que se está probando es una de las primeras operaciones que realiza IDT.

La mayoría de las configuraciones de Windows utilizan la aplicación de terminal PuTTY para conectarse a los hosts Linux. Esta aplicación requiere que convierta los archivos de clave privada PEM estándar a un formato propietario de Windows denominado PPK. Si configura SSH en su`device.json`archivo, utilice archivos PEM. Si usa un archivo PPK, IDT no puede crear una conexión SSH con elAWS IoT Greengrassdispositivo y no puede ejecutar pruebas.

A partir de la versión 4.4.0 de IDT, si no has activado el SFTP en el dispositivo que estás probando, es posible que veas el siguiente error en el archivo de registro.

```
SSH connection failed with EOF
```

Para resolver este error, habilita SFTP en tu dispositivo.

Errores de calificación del administrador de transmisiones

Al ejecutar las pruebas de calificación de Stream Manager, es posible que veas el siguiente error en el`com.aws.StreamManagerExport.log`archivo.

```
Failed to upload data to S3
```

Este error puede producirse cuando el administrador de transmisiones usa elAWScredenciales en el`~/root/.aws/credentials`archiva en tu dispositivo en lugar de utilizar las credenciales de entorno que IDT exporta al dispositivo que se está probando. Para evitar este problema, elimine el`credentials`archiva en tu dispositivo y vuelve a ejecutar la prueba de calificación.

Errores de tiempo de espera

Puede aumentar el tiempo de espera de cada prueba especificando un multiplicador de tiempo de espera aplicado al valor predeterminado del tiempo de espera de cada prueba. Cualquier valor configurado para esta marca debe ser superior o igual a 1,0.

Para utilizar el multiplicador de tiempo de espera, utilice la marca `--timeout-multiplier` al ejecutar las pruebas. Por ejemplo:

```
./devicetester_linux run-suite --suite-id GGV2Q_1.0.0 --pool-id DevicePool1 --timeout-multiplier 2.5
```

Para obtener más información, ejecute `run-suite --help`.

Algunos errores de tiempo de espera se producen cuando los casos de prueba de IDT no se pueden completar debido a problemas de configuración. No puede resolver estos errores aumentando el multiplicador de tiempo de espera. Usa los registros de la ejecución de la prueba para solucionar los problemas de configuración subyacentes.

- Si los registros de los componentes MQTT o Lambda contienen`Access denied`si se producen errores, es posible que la carpeta de instalación de Greengrass no tenga los permisos de archivo correctos. Ejecute el siguiente comando para cada carpeta de la ruta de instalación que haya definido en su`userdata.json`archivo.

  ```
  sudo chmod 755 folder-name
  ```

- Si los registros de Greengrass indican que la implementación de la CLI de Greengrass no está completa, haga lo siguiente:
  - Compruebe que`bash`está instalado en el dispositivo que se está probando.
  - Si tu`userdata.json`el archivo incluye el`GreengrassCliVersion`parámetro de configuración, elimínelo. Este parámetro está obsoleto en IDT v4.1.0 y versiones posteriores. Para obtener más información, consulte Configure userdata.json.
- Si la prueba de despliegue de Lambda no ha tenido éxito y aparece el mensaje de error «Validar la publicación de Lambda: se ha agotado el tiempo de espera» y aparece un error en el archivo de registro de la prueba (`idt-gg2-lambda-function-idt-<resource-id>.log`) que dice`Error: Could not find or load main class com.amazonaws.greengrass.runtime.LambdaRuntime.`, haga lo siguiente:

- Compruebe para qué carpeta se utilizó`InstallationDirRootOnDevice`en el`userdata.json`archivo.
- Asegúrese de que los permisos de usuario correctos estén configurados en su dispositivo. Para obtener más información, consulta[Configura los permisos de usuario en tu dispositivo](#).

Errores de comprobación de versiones

IDT emite el siguiente error cuando[AWS](#)las credenciales de usuario del usuario de IDT no tienen los permisos de IAM necesarios.

```
Failed to check version compatibility
```

El[AWS](#)el usuario que no tiene los permisos de IAM necesarios.

Política de soporte AWS IoT Device Tester para AWS IoT Greengrass

AWS IoT Device Tester for AWS IoT Greengrass es una herramienta de automatización de pruebas que se utiliza para validar y [calificar](#) sus AWS IoT Greengrass dispositivos para su inclusión en el [catálogo de AWS Partner dispositivos](#). Le recomendamos que utilice la versión más reciente de AWS IoT Greengrass y AWS IoT Device Tester para probar o calificar sus dispositivos.

Hay al menos una versión de AWS IoT Device Tester disponible para cada versión compatible de AWS IoT Greengrass. Para ver las versiones compatibles de AWS IoT Greengrass, consulte las versiones del [núcleo de Greengrass](#). Para ver las versiones compatibles de AWS IoT Device Tester, consulte[Versiones compatibles de AWS IoT Device Tester para AWS IoT Greengrass V2](#).

También puede usar cualquiera de las versiones compatibles de AWS IoT Greengrass y AWS IoT Device Tester para probar o calificar sus dispositivos. Si bien puedes seguir usando versiones no compatibles de AWS IoT Device Tester, esas versiones no reciben correcciones de errores ni actualizaciones. Si tiene preguntas sobre la política de soporte, póngase en contacto con [AWS Support](#).

Soluciones de IoT basadas en Greengrass

El Everyware de Eurotech GreenEdge está en versión preliminar AWS IoT Greengrass y está sujeto a cambios. AWS no es compatible con esta solución. Si tiene algún problema con este dispositivo, debe ponerse en contacto con Eurotech.

AWS IoT Greengrassofrece soluciones de socios para optimizar su experiencia de instalación de Greengrass. La siguiente es una solución que se AWS ha asociado con Eurotech para ofrecer. Esta solución viene con el tiempo de ejecución AWS IoT Greengrass Core Edge y capacidades adicionales preinstaladas.

Eurotech

AWSse ha asociado con Eurotech para ofrecer una solución de IoT a los clientes que buscan un dispositivo que venga con el software AWS IoT Greengrass Core preinstalado. Everyware de Eurotech GreenEdge es un software avanzado de IoT preconfigurado y precalificado por. AWS Esta solución combina las capacidades de Greengrass y el Eurotech Everyware Software Framework (ESF) para ofrecer a los clientes una amplia conectividad hacia el sur a través de adaptadores de protocolo como: Modbus, OPC-UA Client/Server, S7, TwinCat, J1939, DNP3 Master/Outstation y más. Con esta solución, también puede enviar datos a todos los AWS servicios en dirección norte (como AWS IoT Core Amazon S3 Nube de AWS y Amazon Kinesis Video Streams) y conectarse a ellos. AWS IoT SiteWise AWS IoT Analytics En combinación con Everyware Cloud, la solución de administración de dispositivos de Eurotech, esta solución presenta un novedoso servicio de aprovisionamiento sin intervención previa, que simplifica la incorporación y el despliegue masivo de los dispositivos.

Para obtener más información sobre Eurotech, consulte Eurotech.

Solución de problemas AWS IoT Greengrass V2

Utilice la información y las soluciones de solución de problemas de esta sección para ayudar a resolver los problemas con. AWS IoT Greengrass Version 2

Temas

Consulte los registros AWS IoT Greengrass principales de software y componentes

El software AWS IoT Greengrass Core escribe registros en el sistema de archivos local que puede usar para ver información en tiempo real sobre el dispositivo principal. También puede configurar los dispositivos principales para que escriban registros en los CloudWatch registros, de modo que pueda solucionar los problemas de los dispositivos principales de forma remota. Estos registros pueden ayudarle a identificar problemas con los componentes, las implementaciones y los dispositivos principales. Para obtener más información, consulte Supervisar AWS IoT Greengrass registros.

AWS IoT Greengrass Problemas principales de software

Solucionar problemas AWS IoT Greengrass de software principal.

Temas

- No se pudo configurar el dispositivo principal
- No se puede iniciar el software AWS IoT Greengrass Core como un servicio del sistema
- No se puede configurar Nucleus como un servicio del sistema
- No se puede conectar a AWS IoT Core
- Error de memoria insuficiente
- No se puede instalar la CLI de Greengrass
- 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
- No se pudo configurar el servicio de 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

No se pudo configurar el dispositivo principal

Si el instalador del software AWS IoT Greengrass principal falla y no puedes configurar un dispositivo principal, es posible que tengas que desinstalar el software e intentarlo de nuevo. Para obtener más información, consulte Desinstale el software AWS IoT Greengrass principal.

No se puede iniciar el software AWS IoT Greengrass Core como un servicio del sistema

Si el software AWS IoT Greengrass principal no se inicia, compruebe los registros de servicio del sistema para identificar el problema. Un problema habitual es que Java no esté disponible en la variable de entorno PATH (Linux) o en la variable de sistema PATH (Windows).

No se puede configurar Nucleus como un servicio del sistema

Es posible que aparezca este error si el instalador del software AWS IoT Greengrass Core no se configura AWS IoT Greengrass como un servicio del sistema. En los dispositivos Linux, este error suele producirse si el dispositivo principal no tiene el sistema de inicio systemd. El instalador puede configurar correctamente el software AWS IoT Greengrass principal aunque no pueda configurar el servicio del sistema.

Realice una de las acciones siguientes:

- Configure y ejecute el software AWS IoT Greengrass principal como un servicio del sistema. Debe configurar el software como un servicio del sistema para poder utilizar todas las funciones de AWS IoT Greengrass. Puede instalar systemd o utilizar un sistema de inicio diferente. Para obtener más información, consulte Configurar el núcleo de Greengrass como un servicio del sistema.
- Ejecute el software AWS IoT Greengrass principal sin un servicio de sistema. Puede ejecutar el software mediante un script de carga que el instalador configura en la carpeta raíz de Greengrass. Para obtener más información, consulte Ejecute el software AWS IoT Greengrass principal sin un servicio de sistema.

No se puede conectar a AWS IoT Core

Es posible que aparezca este error cuando el software AWS IoT Greengrass principal no se pueda conectar AWS IoT Core para recuperar los trabajos de implementación, por ejemplo. Haga lo siguiente:

- Comprueba que tu dispositivo principal se pueda conectar a Internet y AWS IoT Core. Para obtener más información sobre el AWS IoT Core punto final al que se conecta el dispositivo, consulteConfigurar el software AWS IoT Greengrass principal.
- Comprueba que el dispositivo AWS IoT principal utilice un certificado que permita los `iot:Subscribe` permisos `iot:Connect` `iot:Publish``iot:Receive`,, y.
- Si tu dispositivo principal usa un proxy de red, comprueba que el dispositivo principal tenga una función de dispositivo y que esta función permita los `iot:Subscribe` permisos `iot:Connect` `iot:Publish``iot:Receive`,, y.

Error de memoria insuficiente

Este error suele producirse si el dispositivo no tiene memoria suficiente para asignar un objeto en el montón de Java. En los dispositivos con memoria limitada, es posible que tengas que especificar un tamaño máximo de pila para controlar la asignación de memoria. Para obtener más información, consulte Controle la asignación de memoria con las opciones de JVM.

No se puede instalar la CLI de Greengrass

Es posible que vea el siguiente mensaje de consola cuando utilice el `--deploy-dev-tools` argumento en el comando de instalación de 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
```

Esto ocurre cuando el componente CLI de Greengrass no está instalado porque el dispositivo principal es miembro de un grupo de cosas que tiene una implementación existente. Si ve este mensaje, puede implementar manualmente el componente CLI de Greengrass (`aws.greengrass.Cli`) en el dispositivo para instalar la CLI de Greengrass. Para obtener más información, consulte Instalación de la CLI de Greengrass.

User root is not allowed to execute

Es posible que aparezca este error cuando el usuario que ejecuta el software AWS IoT Greengrass principal (normalmente`root`) no tiene permiso para ejecutar `sudo` con ningún usuario o grupo. Para el usuario predeterminado `ggc_user` del sistema, este error tiene el siguiente aspecto:

```
Sorry, user root is not allowed to execute <command> as ggc_user:ggc_group.
```

Compruebe que el `/etc/sudoers` archivo da permiso al usuario para ejecutarse `sudo` como otros grupos. El permiso de entrada del usuario `/etc/sudoers` debería tener el aspecto que se muestra en el siguiente ejemplo.

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

com.aws.greengrass.lifecyclemanager.GenericExternalService: Could not determine user/group to run with

Es posible que aparezca este error cuando el dispositivo principal intente ejecutar un componente y el núcleo de Greengrass no especifique un usuario de sistema predeterminado para ejecutar los componentes.

Para solucionar este problema, configure el núcleo de Greengrass para especificar el usuario del sistema predeterminado que ejecuta los componentes. Para obtener más información, consulte Configure el usuario que ejecuta los componentes y Configure el usuario del componente predeterminado.

Failed to map segment from shared object: operation not permitted

Es posible que aparezca este error cuando el software AWS IoT Greengrass principal no se inicie porque la `/tmp` carpeta está montada con `noexec` permisos. La biblioteca AWS Common Runtime (CRT) usa la `/tmp` carpeta de forma predeterminada.

Realice una de las acciones siguientes:

- Ejecute el siguiente comando para volver a montar la `/tmp` carpeta con `exec` permisos e inténtelo de nuevo.

  ```
  sudo mount -o remount,exec /tmp
  ```

- Si ejecuta Greengrass nucleus v2.5.0 o posterior, puede configurar una opción de JVM para cambiar la carpeta que utiliza la biblioteca CRT. AWS Puede especificar el `jvmOptions` parámetro en la configuración del componente núcleo de Greengrass en una implementación o al instalar el software AWS IoT Greengrass Core. Sustituya */path/to/use* por la ruta a una carpeta que pueda usar la AWS biblioteca CRT.

```
{
  "jvmOptions": "-Daws.crt.lib.dir=\"/path/to/use\""
}
```

No se pudo configurar el servicio de Windows

Es posible que veas este error si instalas el software AWS IoT Greengrass Core en un dispositivo Microsoft Windows 2016. El software AWS IoT Greengrass principal no es compatible con Windows 2016; para obtener una lista de los sistemas operativos compatibles, consultePlataformas admitidas.

Si debe usar Windows 2016, puede hacer lo siguiente:

1. Descomprima el archivo de instalación AWS IoT Greengrass de Core descargado
2. Abre el `bin/greengrass.xml.template` archivo en el `Greengrass` directorio.
3. Añada la `<autoRefresh>` etiqueta al final del archivo justo antes de la `</service>` etiqueta.

```
  </log>
  <autoRefresh>false</autoRefresh>
</service>
```

com.aws.greengrass.util.exceptions.TLSAuthException: Failed to get trust manager

Es posible que aparezca este error al instalar el software AWS IoT Greengrass principal sin un archivo raíz de una entidad emisora de certificados (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
```

Compruebe que ha especificado un archivo de CA raíz válido con el `rootCaPath` parámetro en el archivo de configuración que ha proporcionado al instalador. Para obtener más información, consulte Instalación del software AWS IoT Greengrass Core.

com.aws.greengrass.deployment.IotJobsHelper: No connection available during subscribing to Iot Jobs descriptions topic. Will retry in sometime

Es posible que aparezca este mensaje de advertencia cuando el dispositivo principal no pueda conectarse para suscribirse AWS IoT Core a las notificaciones de los trabajos de despliegue. Haga lo siguiente:

- Compruebe que el dispositivo principal esté conectado a Internet y pueda acceder al punto final de AWS IoT datos que configuró. Para obtener más información sobre los terminales que utilizan los dispositivos principales, consultePermitir el tráfico del dispositivo a través de un proxy o firewall.
- Compruebe los registros de Greengrass para ver si hay otros errores que revelen otras causas principales.

software.amazon.awssdk.services.iam.model.IamException: The security token included in the request is invalid

Es posible que aparezca este error cuando instale el software AWS IoT Greengrass principal con aprovisionamiento automático y el instalador utilice un token de AWS sesión que no es válido. Haga lo siguiente:

- Si utilizas credenciales de seguridad temporales, comprueba que el token de sesión es correcto y que estás copiando y pegando el token de sesión completo.
- Si utilizas credenciales de seguridad de larga duración, comprueba que el dispositivo no tenga un token de sesión de una época en la que utilizaste anteriormente credenciales temporales. Haga lo siguiente:

  1. Ejecute el siguiente comando para desconfigurar la variable de entorno del token de sesión.

     Linux or Unix

     ```
     unset AWS_SESSION_TOKEN
     ```

Windows Command Prompt (CMD)

```
set AWS_SESSION_TOKEN=
```

PowerShell

```
Remove-Item Env:\AWS_SESSION_TOKEN
```

2. Compruebe si el archivo de AWS credenciales,`~/.aws/credentials`, contiene un token de sesión,`aws_session_token`. Si es así, elimine esa línea del archivo.

```
aws_session_token = AQoEXAMPLEH4aoAH0gNCAPyJxz4BlCFFxWNE1OPTgk5TthT
+FvwqnKwRc0IfrRh3c/LTo6UDdyJwOOvEVPvLXCrrrUtdnniCEXAMPLE/
IvU1dYUg2RVAJBanLiHb4IgRmpRV3zrkuWJOgQs8IZZaIv2BXIa2R40lgk
```

También puede instalar el software AWS IoT Greengrass Core sin proporcionar AWS credenciales. Para obtener más información, consulte Instale el software AWS IoT Greengrass principal con aprovisionamiento manual de recursos o Instale el software AWS IoT Greengrass principal con aprovisionamiento AWS IoT de flota.

software.amazon.awssdk.services.iot.model.IotException: User: <user> is not authorized to perform: iot:GetPolicy

Es posible que aparezca este error al instalar el software AWS IoT Greengrass principal con el aprovisionamiento automático y el instalador utilice AWS credenciales que no tienen los permisos necesarios. Para obtener más información sobre los permisos necesarios, consultePolítica de IAM mínima para que el instalador aprovisione recursos.

Compruebe los permisos de la identidad de IAM de las credenciales y otorgue a la identidad de IAM los permisos necesarios que falten.

Error: com.aws.greengrass.shadowmanager.sync.model.FullShadowSyncRequest: Could not execute cloud shadow get request

Es posible que aparezca este error cuando utilice el componente administrador de sombras para sincronizar las sombras de los dispositivos con ellas. AWS IoT Core El código de estado HTTP 403

indica que este error se produjo porque la AWS IoT política del dispositivo principal no concede permiso para realizar llamadas`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)
```

Para sincronizar con las sombras locales AWS IoT Core, la AWS IoT política del dispositivo principal debe conceder los siguientes permisos:

- `iot:GetThingShadow`
- `iot:UpdateThingShadow`
- `iot:DeleteThingShadow`

Comprueba la AWS IoT política del dispositivo principal y añade los permisos necesarios que falten. Para más información, consulte los siguientes temas:

- AWS IoT Core acciones políticas en la Guía para AWS IoT desarrolladores
- Actualice la política de un dispositivo principal AWS IoT

Operation aws.greengrass#<operation> is not supported by Greengrass

Es posible que aparezca este error cuando utilice una operación de comunicación entre procesos (IPC) en un componente personalizado de Greengrass y el componente requerido AWS proporcionado no esté instalado en el dispositivo principal.

Para solucionar este problema, añada el componente necesario como una dependencia en la receta de componentes, de modo que el software AWS IoT Greengrass principal instale el componente necesario al implementar el componente.

- Recupera valores secretos — `aws.greengrass.SecretManager`
- Interactúa con las sombras locales — `aws.greengrass.ShadowManager`

- Gestione las implementaciones y los componentes locales (versión `aws.greengrass.Cli` 2.6.0 o posterior)
- Autentique y autorice los dispositivos cliente: versión 2.2.0 o posterior `aws.greengrass.clientdevices.Auth`

java.io.FileNotFoundException: <stream-manager-store-root-dir>/stream_manager_metadata_store (Permission denied)

Es posible que veas este error en el archivo de registro del administrador de transmisiones (`aws.greengrass.StreamManager.log`) cuando configuras el administrador de transmisiones para que use una carpeta raíz que no existe o que no tiene los permisos correctos. Para obtener más información sobre cómo configurar esta carpeta, consulta la sección Configuración del administrador de flujos.

com.aws.greengrass.security.provider.pkcs11.PKCS11CryptoKeyService: Private key or certificate with label <label> does not exist

Este error se produce cuando el componente proveedor PKCS #11 no puede encontrar ni cargar la clave privada o el certificado que especificó al configurar el software AWS IoT Greengrass principal para que utilice un módulo de seguridad de hardware (HSM). Haga lo siguiente:

- Compruebe que la clave privada y el certificado estén almacenados en el HSM mediante la ranura, el PIN de usuario y la etiqueta de objeto para los que ha configurado el software AWS IoT Greengrass Core.
- Compruebe que la clave privada y el certificado utilizan la misma etiqueta de objeto en el HSM.
- Si su HSM admite identificadores de objeto, compruebe que la clave privada y el certificado utilicen el mismo identificador de objeto en el HSM.

Consulte la documentación de su HSM para obtener información sobre cómo consultar los detalles sobre los tokens de seguridad del HSM. Si necesitas cambiar la ranura, la etiqueta del objeto o el ID del objeto por un token de seguridad, consulta la documentación de tu HSM para obtener información sobre cómo hacerlo.

software.amazon.awssdk.services.secretsmanager.model.SecretsManagerExcep User: <user> is not authorized to perform: secretsmanager:GetSecretValue on resource: <arn>

Este error puede producirse cuando utilizas el componente de administrador de secretos para implementar un AWS Secrets Manager secreto. Si la función de IAM de intercambio de fichas del dispositivo principal no concede permiso para obtener el secreto, se produce un error en la implementación y los registros de Greengrass incluyen este error.

Para autorizar a un dispositivo principal a descargar un secreto

1. Agrega el `secretsmanager:GetSecretValue` permiso a la función de intercambio de fichas del dispositivo principal. El siguiente ejemplo de declaración de política otorga permiso para obtener el valor de un secreto.

   ```
   {
       "Effect": "Allow",
       "Action": [
           "secretsmanager:GetSecretValue"
       ],
       "Resource": [
           "arn:aws:secretsmanager:us-west-2:123456789012:secret:MyGreengrassSecret-
   abcdef"
       ]
   }
   ```

   Para obtener más información, consulte Autorizar a los dispositivos principales a interactuar con AWS los servicios.

2. Vuelva a aplicar la implementación al dispositivo principal. Realice una de las acciones siguientes:

   - Revise la implementación sin cambios. El dispositivo principal intenta volver a descargar el secreto cuando recibe la implementación revisada. Para obtener más información, consulte Revisar las implementaciones.
   - Reinicie el software AWS IoT Greengrass principal para volver a intentar la implementación. Para más información, consulte Ejecute el software AWS IoT Greengrass principal

   La implementación se realiza correctamente si el administrador de secretos descarga el secreto correctamente.

software.amazon.awssdk.services.secretsmanager.model.SecretsManagerExcep Access to KMS is not allowed

Este error puede producirse cuando se utiliza el componente de administrador de secretos para implementar un AWS Secrets Manager secreto cifrado mediante una AWS Key Management Service clave. Si la función de IAM de intercambio de fichas del dispositivo principal no concede permiso para descifrar el secreto, se produce un error en el despliegue y los registros de Greengrass incluyen este error.

Para solucionar el problema, añada el `kms:Decrypt` permiso a la función de intercambio de fichas del dispositivo principal. Para más información, consulte los siguientes temas:

- Cifrado y descifrado secretos en la Guía del AWS Secrets Manager usuario
- Autorizar a los dispositivos principales a interactuar con AWS los servicios

java.lang.NoClassDefFoundError: com/aws/greengrass/security/CryptoKeySpi

Es posible que veas este error cuando intentes instalar el software AWS IoT Greengrass Core con seguridad de hardware y utilices una versión anterior del núcleo de Greengrass que no admite la integración de seguridad de hardware. Para usar la integración de seguridad de hardware, debe usar Greengrass nucleus v2.5.3 o posterior.

com.aws.greengrass.security.provider.pkcs11.PKCS11CryptoKeyService: CKR_OPERATION_NOT_INITIALIZED

Es posible que aparezca este error cuando utilice la biblioteca TPM2 al ejecutar AWS IoT Greengrass Core como un servicio del sistema.

Este error indica que debe agregar una variable de entorno que proporcione la ubicación del almacén PKCS #11 en el archivo de servicio AWS IoT Greengrass Core systemd.

Para obtener más información, consulte la sección de requisitos de la documentación del Proveedor PKCS #11 componente.

Greengrass core device stuck on nucleus v2.12.3

Si su dispositivo principal de Greengrass no revisa la implementación de la versión 2.12.3 de nucleus, es posible que tenga que descargar y reemplazar el archivo `Greengrass.jar` por la versión 2.12.2 de Greengrass nucleus. Haga lo siguiente:

1. En su dispositivo principal de Greengrass, ejecute el siguiente comando para detener el software Greengrass Core.

   Linux or Unix

   ```
   sudo systemctl stop greengrass
   ```

   Windows Command Prompt (CMD)

   ```
   sc stop "greengrass"
   ```

   PowerShell

   ```
   Stop-Service -Name "greengrass"
   ```

2. En su dispositivo principal, descargue el AWS IoT Greengrass software en un archivo denominado. `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. Descomprime el software AWS IoT Greengrass principal en una carpeta de tu dispositivo. *GreengrassInstaller*Sustitúyalo por la carpeta que desee usar.

   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. Ejecute el siguiente comando para anular el archivo JAR de Greengrass de la versión 2.12.3 del núcleo por el archivo JAR de Greengrass de la versión 2.12.2 del núcleo.

   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. Ejecute el siguiente comando para iniciar el software Greengrass Core.

Linux or Unix

```
sudo systemctl start greengrass
```

Windows Command Prompt (CMD)

```
sc start "greengrass"
```

PowerShell

```
Start-Service -Name "greengrass"
```

AWS IoT Greengrass problemas con la nube

Usa la siguiente información para solucionar problemas con la AWS IoT Greengrass consola y la API. Cada entrada corresponde a un mensaje de error que puede aparecer al realizar una acción.

An error occurred (AccessDeniedException) when calling the CreateComponentVersion operation: User: arn:aws:iam::123456789012:user/<username> is not authorized to perform: null

Es posible que aparezca este error al crear una versión de componente desde la AWS IoT Greengrass consola o durante la CreateComponentVersionoperación.

Este error indica que tu receta no es válida en formato JSON o YAML. Comprueba la sintaxis de la receta, corrige cualquier problema de sintaxis y vuelve a intentarlo. Puedes usar un corrector de sintaxis JSON o YAML en línea para identificar los problemas de sintaxis en tu receta.

Invalid Input: Encountered following errors in Artifacts: {<s3ArtifactUri> = Specified artifact resource cannot be accessed}

Es posible que veas este error al crear una versión de un componente desde la AWS IoT Greengrass consola o durante la CreateComponentVersionoperación. Este error indica que un artefacto de S3 en la receta del componente no es válido.

Haga lo siguiente:

- Compruebe que el depósito de S3 esté en el mismo Región de AWS lugar en el que creó el componente. AWS IoT Greengrass no admite solicitudes de artefactos de componentes entre regiones.
- Comprueba que el URI del artefacto es una URL de objeto S3 válida y comprueba que el artefacto existe en esa URL de objeto S3.
- Comprueba que tienes Cuenta de AWS permiso para acceder al artefacto en la URL del objeto de S3.

INACTIVE deployment status

Es posible que obtengas un estado de `INACTIVE` implementación cuando llames a la ListDeploymentsAPI sin las AWS IoT políticas dependientes requeridas. Debe tener los permisos necesarios para obtener un estado de implementación preciso. Puede encontrar las acciones dependientes consultando las acciones definidas por AWS IoT Greengrass V2 y siguiendo los permisos necesarios`ListDeployments`. Sin los AWS IoT permisos dependientes necesarios, seguirá viendo el estado de la implementación, pero es posible que vea un estado de implementación inexacto de`INACTIVE`.

Problemas principales de implementación de dispositivos

Solucione problemas de implementación en los dispositivos principales de Greengrass. Cada entrada corresponde a un mensaje de registro que puede que vea en su dispositivo principal.

Temas

- 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

Es posible que aparezca este error cuando el software AWS IoT Greengrass principal no puede descargar un artefacto componente cuando el dispositivo principal realiza una implementación. La implementación falla como resultado de este error.

Cuando recibe este error, el registro también incluye un rastreo de pila que puede usar para identificar el problema específico. Cada una de las siguientes entradas corresponde a un mensaje que puede que veas en el seguimiento de la pila del mensaje de `Failed to download artifact` error.

Temas

- 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, ...)

El PackageDownloadException error puede incluir este rastreo de pila en los siguientes casos:

- El artefacto del componente no está disponible en la URL del objeto S3 que especificó en la receta del componente. Comprueba que has subido el artefacto al depósito de S3 y que el URI del artefacto coincide con la URL del objeto de S3 del artefacto del depósito.
- La función de intercambio de fichas del dispositivo principal no permite que el software AWS IoT Greengrass principal descargue el artefacto componente desde la URL del objeto S3 que especifiques en la receta del componente. Comprueba que la función de intercambio de fichas permita `s3:GetObject` la URL del objeto S3 en la que está disponible el artefacto.

software.amazon.awssdk.services.s3.model.S3Exception: Access Denied (Service: S3, Status Code: 403, Request ID: <requestID>

El PackageDownloadException error puede incluir este rastreo de pila cuando el dispositivo principal no tiene permiso para llamars`3:GetBucketLocation`. El mensaje de error también incluye el siguiente mensaje.

```
reason: Failed to determine S3 bucket location
```

Comprueba que la función de intercambio de fichas del dispositivo principal permita `s3:GetBucketLocation` el depósito S3 en el que está disponible el artefacto.

Error: com.aws.greengrass.componentmanager.exceptions.ArtifactChecksumMismatchI Integrity check for downloaded artifact failed. Probably due to file corruption.

Es posible que aparezca este error cuando el software AWS IoT Greengrass principal no pueda descargar un artefacto componente cuando el dispositivo principal realice una implementación. La

implementación falla porque la suma de verificación del archivo de artefacto descargado no coincide con la suma de verificación que AWS IoT Greengrass se calculó al crear el componente.

Haga lo siguiente:

- Compruebe si el archivo de artefactos ha cambiado en el depósito de S3 en el que lo aloja. Si el archivo ha cambiado desde que creó el componente, restáurelo a la versión anterior que esperaba el dispositivo principal. Si no puede restaurar el archivo a su versión anterior o si desea utilizar la nueva versión del archivo, cree una nueva versión del componente con el archivo de artefactos.
- Comprueba la conexión a Internet del dispositivo principal. Este error puede producirse si el archivo del artefacto se daña mientras se descarga. Cree una nueva implementación e inténtelo de nuevo.

Error: com.aws.greengrass.componentmanager.exceptions.NoAvailableComponentVers Failed to negotiate component <name> version with cloud and no local applicable version satisfying requirement <requirements>

Es posible que aparezca este error cuando un dispositivo principal no encuentre una versión de componente que cumpla con los requisitos de las implementaciones de ese dispositivo principal. El dispositivo principal busca el componente en el AWS IoT Greengrass servicio y en el dispositivo local. El mensaje de error incluye el destino de cada implementación y los requisitos de versión de esa implementación para el componente. El objetivo de despliegue puede ser una cosa, un grupo de cosas o bien `LOCAL_DEPLOYMENT` representar la implementación local en el dispositivo principal.

Este problema puede producirse en los siguientes casos:

- El dispositivo principal es el objetivo de varias implementaciones que tienen requisitos de versión de componentes contradictorios. Por ejemplo, el dispositivo principal puede ser el objetivo de varias implementaciones que incluyen un `com.example.HelloWorld` componente, en las que una implementación requiere la versión 1.0.0 y la otra requiere la versión 1.0.1. Es imposible tener un componente que cumpla ambos requisitos, por lo que la implementación falla.
- La versión del componente no existe en el AWS IoT Greengrass servicio ni en el dispositivo local. Es posible que el componente se haya eliminado, por ejemplo.
- Existen versiones de componentes que cumplen con los requisitos de la versión, pero ninguna es compatible con la plataforma del dispositivo principal.

- La AWS IoT política del dispositivo principal no concede el `greengrass:ResolveComponentCandidates` permiso. Busca `Status Code: 403` en el registro de errores para identificar este problema. Para resolver este problema, añade el `greengrass:ResolveComponentCandidates` permiso a la AWS IoT política del dispositivo principal. Para obtener más información, consulte AWS IoTPolítica mínima para los dispositivos AWS IoT Greengrass V2 principales.

Para resolver este problema, revise las implementaciones para incluir versiones de componentes compatibles o eliminar las incompatibles. Para obtener más información sobre cómo revisar las implementaciones en la nube, consulte. Revisar las implementaciones Para obtener más información sobre cómo revisar las implementaciones locales, consulte el comando AWS IoT Greengrass CLI deployment create.

software.amazon.awssdk.services.greengrassv2data.model.ResourceNotFoundE The latest version of Component <componentName> doesn't claim platform <coreDevicePlatform> compatibility

Es posible que aparezca este error al implementar un componente en un dispositivo principal y el componente no muestre una plataforma que sea compatible con la plataforma del dispositivo principal. Realice una de las acciones siguientes:

- Si el componente es un componente personalizado de Greengrass, puede actualizarlo para que sea compatible con el dispositivo principal. Agregue un nuevo manifiesto que coincida con la plataforma del dispositivo principal o actualice un manifiesto existente para que coincida con la plataforma del dispositivo principal. Para obtener más información, consulte AWS IoT Greengrass referencia de recetas de componentes.
- Si el componente lo proporciona AWS, compruebe si otra versión del componente es compatible con el dispositivo principal. Si ninguna versión es compatible, póngase en contacto con nosotros AWS re:Postmediante la AWS IoT Greengrass etiqueta o póngase en contacto con nosotros 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

Es posible que aparezca este error al implementar un componente que depende del núcleo de Greengrass y el dispositivo principal ejecute una versión del núcleo de Greengrass anterior a la última versión secundaria disponible. Este error se produce porque el software AWS IoT Greengrass Core intenta actualizar automáticamente los componentes a la última versión compatible. Sin embargo, el software AWS IoT Greengrass Core impide que el núcleo de Greengrass se actualice a una nueva versión secundaria, ya que varios de los componentes AWS proporcionados dependen de versiones secundarias específicas del núcleo de Greengrass. Para obtener más información, consulte Comportamiento de actualización del núcleo de Greengrass.

Debe revisar la implementación para especificar la versión del núcleo de Greengrass que quiere usar. Realice una de las acciones siguientes:

- Revise la implementación para especificar la versión del núcleo de Greengrass que ejecuta actualmente el dispositivo principal.
- Revise el despliegue para especificar una versión secundaria posterior del núcleo de Greengrass. Si elige esta opción, también debe actualizar las versiones de todos los componentes AWS proporcionados que dependen de versiones secundarias específicas del núcleo de Greengrass. Para obtener más información, consulte AWS-componentes proporcionados.

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

Es posible que aparezca este error al mover un dispositivo Greengrass de un grupo de cosas a otro y, después, volver al grupo original con implementaciones que requieren el reinicio de Greengrass.

Para resolver este problema, vuelva a crear el directorio de inicio del dispositivo. También recomendamos encarecidamente actualizar el núcleo de Greengrass a la versión 2.9.6 o posterior.

El siguiente es un script de Linux para recrear el directorio de inicio. Guarde el script en un archivo llamado`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
```

Para ejecutar el script, ejecute el siguiente comando:

```
[root@ip-172-31-27-165 ~]# ./fix_directory.sh /greengrass/v2 2.9.5
Relinking /greengrass/v2/alts/directory_fix to /greengrass/v2/alts/current
Relinking Nucleus artifacts to /greengrass/v2/alts/directory_fix/distro
```

Info: com.aws.greengrass.deployment.exceptions.RetryableDeploymentDocumentDow Greengrass Cloud Service returned an error when getting full deployment configuration

Es posible que aparezca este error cuando el dispositivo principal reciba un documento de despliegue de gran tamaño, es decir, un documento de despliegue de más de 7 KB (en el caso de las implementaciones orientadas a objetos) o 31 KB (en el caso de las implantaciones dirigidas a grupos de objetos). Para recuperar un documento de despliegue de gran tamaño, la AWS IoT política del dispositivo principal debe permitir el `greengrass:GetDeploymentConfiguration` permiso. Este error puede producirse cuando el dispositivo principal no tiene este permiso. Cuando se produce este error, la implementación se reintenta indefinidamente y su estado es En curso ()`IN_PROGRESS`.

Para resolver este problema, añade el `greengrass:GetDeploymentConfiguration` permiso a la política del AWS IoT dispositivo principal. Para obtener más información, consulte Actualice la política de un dispositivo principal AWS IoT.

Warn: com.aws.greengrass.deployment.DeploymentService: Failed to get thing group hierarchy

Es posible que veas esta advertencia cuando el dispositivo principal reciba una implementación y la AWS IoT política del dispositivo principal no permita el `greengrass:ListThingGroupsForCoreDevice` permiso. Al crear una implementación, el dispositivo principal usa este permiso para identificar sus grupos de cosas y eliminar los componentes de cualquier grupo de cosas de los que haya eliminado el dispositivo principal. Si el dispositivo principal ejecuta Greengrass nucleus v2.5.0, se produce un error en la implementación. Si el dispositivo principal ejecuta Greengrass nucleus v2.5.1 o posterior, la implementación continúa pero no elimina los componentes. Para obtener más información sobre el comportamiento de eliminación de grupos de cosas, consulte. Implemente AWS IoT Greengrass componentes en los dispositivos

Para actualizar el comportamiento del dispositivo principal a fin de eliminar componentes de los grupos de cosas de los que se elimina el dispositivo principal, añada el `greengrass:ListThingGroupsForCoreDevice` permiso a la AWS IoT política del dispositivo principal. Para obtener más información, consulte Actualice la política de un dispositivo principal AWS IoT.

Info: com.aws.greengrass.deployment.DeploymentDocumentDownloader: Calling Greengrass cloud to get full deployment configuration

Es posible que veas este mensaje de información impreso varias veces sin que se produzca ningún error, ya que el dispositivo principal registra el error a nivel de `DEBUG` registro. Este problema puede producirse cuando el dispositivo principal recibe un documento de implementación de gran tamaño. Cuando se produce este problema, la implementación se reintenta indefinidamente y su estado es En curso ()`IN_PROGRESS`. Para obtener más información sobre cómo resolver este problema, consulte esta entrada de solución de problemas.

Caused by: software.amazon.awssdk.services.greengrassv2data.model.GreengrassV2DataE: null (Service: GreengrassV2Data, Status Code: 403, Request ID: <some_request_id>, Extended Request ID: null)

Es posible que veas este error cuando una API de plano de datos no tiene `iot:Connect` permiso. Si no tienes la política correcta, recibirás un`GreengrassV2DataException: 403`. Para crear una política de permisos, sigue estas instrucciones:Creación de una política de AWS IoT.

Problemas con los componentes principales del dispositivo

Solucione los problemas de los componentes de Greengrass en los dispositivos principales.

Temas

- 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

Es posible que veas este error en los registros de un componente de Greengrass cuando el software AWS IoT Greengrass principal no ejecute un comando en el script de ciclo de vida del componente. El estado del componente pasa a ser el `BROKEN` resultado de este error. Este error puede producirse si el usuario del sistema que ejecuta el componente, por ejemplo`gc_user`, no encuentra el ejecutable del comando en las carpetas de la PATH.

En los dispositivos Windows, compruebe que la carpeta que contiene el ejecutable pertenece al `PATH` usuario del sistema que ejecuta el componente. Si falta en el`PATH`, realice una de las siguientes acciones:

- Añada la carpeta del ejecutable a la variable de `PATH` sistema, que está disponible para todos los usuarios. A continuación, reinicie el componente.

  Si ejecuta Greengrass nucleus 2.5.0, después de actualizar la variable de `PATH` sistema, debe reiniciar el software AWS IoT Greengrass Core para ejecutar los componentes con la actualización. `PATH` Si el software AWS IoT Greengrass principal no utiliza la actualización `PATH` después de reiniciar el software, reinicie el dispositivo e inténtelo de nuevo. Para obtener más información, consulte Ejecute el software AWS IoT Greengrass principal.
- Añada la carpeta del ejecutable a la `PATH` variable de usuario del sistema que ejecuta el componente.

El script de Python no registra los mensajes

Los dispositivos principales de Greengrass recopilan registros que puede utilizar para identificar problemas con los componentes. Si el script `stdout` y `stderr` los mensajes de Python no

aparecen en los registros de sus componentes, es posible que deba vaciar el búfer o deshabilitar el almacenamiento en búfer para estos flujos de salida estándar en Python. Realice uno de los siguientes procedimientos:

- Ejecute Python con el argumento -u para deshabilitar el almacenamiento en búfer en y. `stdout` `stderr`

  Linux or Unix

  ```
  python3 -u hello_world.py
  ```

  Windows

  ```
  py -3 -u hello_world.py
  ```

- Utilice Setenv en la receta de su componente para establecer la variable de entorno PYTHONUNBUFFERED en una cadena que no esté vacía. Esta variable de entorno `stdout` `stderr` desactiva el almacenamiento en búfer en y.
- Vacíe el búfer de las transmisiones o. `stdout` `stderr` Realice una de las acciones siguientes:
  - Vacíe un mensaje al imprimirlo.

    ```
    import sys

    print('Hello, error!', file=sys.stderr, flush=True)
    ```

  - Vacíe un mensaje después de imprimirlo. Puedes enviar varios mensajes antes de vaciar la transmisión.

    ```
    import sys

    print('Hello, error!', file=sys.stderr)
    sys.stderr.flush()
    ```

Para obtener más información sobre cómo comprobar que el script de Python genera mensajes de registro, consulteSupervisar AWS IoT Greengrass registros.

La configuración de los componentes no se actualiza al cambiar la configuración predeterminada

Al cambiar la receta `DefaultConfiguration` de un componente, la nueva configuración predeterminada no sustituirá a la configuración existente del componente durante la implementación. Para aplicar la nueva configuración predeterminada, debe restablecer la configuración del componente a sus valores predeterminados. Al implementar el componente, especifique una sola cadena vacía como actualización de restablecimiento.

Console

Restablezca las rutas

```
[""]
```

AWS CLI

El siguiente comando crea una implementación en un dispositivo principal.

```
aws greengrassv2 create-deployment --cli-input-json file://reset-configuration-
deployment.json
```

El `reset-configuration-deployment.json` archivo contiene el siguiente documento JSON.

```
{
  "targetArn": "arn:aws:iot:us-west-2:123456789012:thing/MyGreengrassCore",
  "deploymentName": "Deployment for MyGreengrassCore",
  "components": {
    "com.example.HelloWorld": {
      "componentVersion": "1.0.0",
      "configurationUpdate": {,
        "reset": [""]
      }
    }
  }
}
```

Greengrass CLI

El siguiente comando CLI de Greengrass crea una implementación local en un dispositivo principal.

```
sudo greengrass-cli deployment create \
  --recipeDir recipes \
  --artifactDir artifacts \
  --merge "com.example.HelloWorld=1.0.0" \
  --update-config reset-configuration-deployment.json
```

El `reset-configuration-deployment.json` archivo contiene el siguiente documento JSON.

```
{
  "com.example.HelloWorld": {
    "RESET": [""]
  }
}
```

awsiot.greengrasscoreipc.model.UnauthorizedError

Es posible que veas este error en los registros de un componente de Greengrass cuando el componente no tiene permiso para realizar una operación de IPC en un recurso. Para conceder permiso a un componente para llamar a una operación de IPC, defina una política de autorización de IPC en la configuración del componente. Para obtener más información, consulte Autorice a los componentes a realizar operaciones de IPC.

> **Tip**
>
> Si cambia la receta `DefaultConfiguration` de un componente, debe restablecer la configuración del componente a su nueva configuración predeterminada. Al implementar el componente, especifique una sola cadena vacía como actualización de restablecimiento. Para obtener más información, consulte La configuración de los componentes no se actualiza al cambiar la configuración predeterminada.

com.aws.greengrass.authorization.exceptions.AuthorizationException: Duplicate policy ID "<id>" for principal "<componentList>"

Es posible que aparezca este error si varias políticas de autorización de IPC, incluidos todos los componentes del dispositivo principal, utilizan el mismo identificador de política.

Compruebe las políticas de autorización de IPC de sus componentes, corrija los duplicados e inténtelo de nuevo. Para crear identificadores de política únicos, le recomendamos que combine el nombre del componente, el nombre del servicio de IPC y un contador. Para obtener más información, consulte Autorice a los componentes a realizar operaciones de IPC.

> **Tip**
>
> Si cambia la receta `DefaultConfiguration` de un componente, debe restablecer la configuración del componente a su nueva configuración predeterminada. Al implementar el componente, especifique una sola cadena vacía como actualización de restablecimiento. Para obtener más información, consulte La configuración de los componentes no se actualiza al cambiar la configuración predeterminada.

com.aws.greengrass.tes.CredentialRequestHandler: Error in retrieving AwsCredentials from TES (HTTP 400)

Es posible que aparezca este error cuando un dispositivo principal no puede obtener AWS las credenciales del servicio de intercambio de fichas. El código de estado HTTP 400 indica que este error se produjo porque la función de IAM de intercambio de fichas del dispositivo principal no existe o no tiene una relación de confianza que permita al proveedor de AWS IoT credenciales asumirla.

Haga lo siguiente:

1. Identifique la función de intercambio de fichas que utiliza el dispositivo principal. El mensaje de error incluye el alias de la AWS IoT función del dispositivo principal, que apunta a la función de intercambio de fichas. Ejecuta el siguiente comando en tu ordenador de desarrollo y *`MyGreengrassCoreTokenExchangeRoleAlias`*sustitúyelo por el nombre del alias del AWS IoT rol que aparece en el mensaje de error.

   ```
   aws iot describe-role-alias --role-alias MyGreengrassCoreTokenExchangeRoleAlias
   ```

La respuesta incluye el nombre de recurso de Amazon (ARN) de la función de IAM de intercambio de tokens.

```
{
  "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. Compruebe que el rol existe. Ejecute el siguiente comando y sustituya *MyGreengrassV2TokenExchangeRole* por el nombre de la función de intercambio de fichas.

```
aws iam get-role --role-name MyGreengrassV2TokenExchangeRole
```

Si el comando devuelve un `NoSuchEntity` error, el rol no existe y debes crearlo. Para obtener más información sobre cómo crear y configurar este rol, consulteAutorizar a los dispositivos principales a interactuar con AWS los servicios.

3. Compruebe que el rol tenga una relación de confianza que permita al proveedor de AWS IoT credenciales asumirlo. La respuesta del paso anterior contiene una`AssumeRolePolicyDocument`, que define las relaciones de confianza del rol. El rol debe definir una relación de confianza que `credentials.iot.amazonaws.com` permita asumirla. Este documento debe tener un aspecto similar al siguiente ejemplo.

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

```
  ]
}
```

Si las relaciones de confianza del rol no `credentials.iot.amazonaws.com` permiten asumirlo, debe agregar esta relación de confianza al rol. Para obtener más información, consulte Modificación de un rol en la Guía del usuario de AWS Identity and Access Management .

com.aws.greengrass.tes.CredentialRequestHandler: Error in retrieving AwsCredentials from TES (HTTP 403)

Es posible que aparezca este error cuando un dispositivo principal no puede obtener AWS las credenciales del servicio de intercambio de fichas. El código de estado HTTP 403 indica que este error se produjo porque las AWS IoT políticas del dispositivo principal no conceden el `iot:AssumeRoleWithCertificate` permiso para el alias de AWS IoT rol del dispositivo principal.

Revisa las AWS IoT políticas del dispositivo principal y añade el `iot:AssumeRoleWithCertificate` permiso para el alias de AWS IoT rol del dispositivo principal. El mensaje de error incluye el alias de AWS IoT rol actual del dispositivo principal. Para obtener más información sobre este permiso y sobre cómo actualizar las AWS IoT políticas del dispositivo principal, consulte AWS IoTPolítica mínima para los dispositivos AWS IoT Greengrass V2 principales yActualice la política de un dispositivo principal AWS IoT.

com.aws.greengrass.tes.CredentialsProviderError: Could not load credentials from any providers

Es posible que aparezca este error cuando el componente intente solicitar AWS credenciales y no pueda conectarse al servicio de intercambio de fichas.

Haga lo siguiente:

- Compruebe que el componente declara una dependencia del componente del servicio de intercambio de fichas,`aws.greengrass.TokenExchangeService`. Si no es así, añada la dependencia y vuelva a implementar el componente.
- Si el componente se ejecuta en docker, asegúrese de aplicar la configuración de red y las variables de entorno correctas, según. Utilice AWS las credenciales en los componentes del contenedor de Docker (Linux)
- Si el componente está escrito en Nodejs, configure dns. setDefaultResultOrdene para. **`ipv4first`**

- `/etc/hosts`Compruebe si hay una entrada que comience por `::1` y contenga`localhost`. Elimine la entrada para ver si provocó que el componente se conectara al servicio de intercambio de fichas en una dirección incorrecta.

Received error when attempting to retrieve ECS metadata: Could not connect to the endpoint URL: "<tokenExchangeServiceEndpoint>"

Es posible que aparezca este error cuando el componente no ejecuta el servicio de intercambio de fichas y un componente intenta solicitar AWS credenciales.

Haga lo siguiente:

- Compruebe que el componente declara una dependencia del componente del servicio de intercambio de fichas,`aws.greengrass.TokenExchangeService`. Si no es así, añada la dependencia y vuelva a implementar el componente.
- Compruebe si el componente utiliza AWS credenciales durante su `install` ciclo de vida. AWS IoT Greengrass no garantiza la disponibilidad del servicio de intercambio de fichas durante el `install` ciclo de vida. Actualice el componente para trasladar el código que usa AWS las credenciales al `run` ciclo de vida `startup` o, a continuación, vuelva a implementar el componente.

copyFrom: <configurationPath> is already a container, not a leaf

Es posible que aparezca este error al cambiar un valor de configuración de un tipo de contenedor (una lista o un objeto) a un tipo que no es de contenedor (cadena, número o booleano). Haga lo siguiente:

1. Compruebe la receta del componente para ver si su configuración predeterminada establece ese valor de configuración en una lista o un objeto. Si es así, elimine o cambie ese valor de configuración.
2. Cree una implementación para restablecer ese valor de configuración a su valor predeterminado. Para obtener más información, consulte Crear implementaciones y Actualizar las configuraciones de los componentes.

A continuación, puede establecer ese valor de configuración en una cadena, un número o un booleano.

com.aws.greengrass.componentmanager.plugins.docker.exceptions.DockerLoginException: Error logging into the registry using credentials - 'The stub received bad data.'

Es posible que aparezca este error en los registros del núcleo de Greengrass cuando el componente gestor de aplicaciones de Docker intenta descargar una imagen de Docker de un repositorio privado de Amazon Elastic Container Registry (Amazon ECR). Este error se produce si utiliza el asistente de credenciales de `wincred` Docker (). `docker-credential-wincred` Como resultado, Amazon ECR no puede almacenar las credenciales de inicio de sesión.

Realice una de las siguientes acciones:

- Si no usa el asistente de credenciales de `wincred` Docker, elimine el `docker-credential-wincred` programa del dispositivo principal.
- Si utilizas el asistente de credenciales de `wincred` Docker, haz lo siguiente:
  1. Cambie el nombre del `docker-credential-wincred` programa en el dispositivo principal. `wincred`Sustitúyalo por un nombre nuevo para el asistente de credenciales de Docker de Windows. Por ejemplo, puede cambiarle el nombre a. `docker-credential-wincredreal`
  2. Actualice la `credsStore` opción en el archivo de configuración de Docker (`.docker/config.json`) para usar el nuevo nombre del asistente de credenciales de Docker de Windows. Por ejemplo, si ha cambiado el nombre del programa a`docker-credential-wincredreal`, actualice la opción a. `credsStore` `wincredreal`

     ```
     {
       "credsStore": "wincredreal"
     }
     ```

java.io.IOException: Cannot run program "cmd" ...: [LogonUser] The password for this account has expired.

Es posible que aparezca este error en un dispositivo básico de Windows cuando el usuario del sistema que ejecuta los procesos del componente, por ejemplo`gc_user`, tiene una contraseña caducada. Como resultado, el software AWS IoT Greengrass principal no puede ejecutar los procesos de los componentes como ese usuario del sistema.

Para actualizar la contraseña de un usuario del sistema Greengrass

1. Ejecute el siguiente comando como administrador para establecer la contraseña del usuario. Sustituya *ggc_user* por el usuario del sistema y sustituya la *contraseña* por la contraseña que desee configurar.

   ```
   net user ggc_user password
   ```

2. Utilice la PsExec utilidad para almacenar la nueva contraseña del usuario en la instancia de Credential Manager de la cuenta. LocalSystem Sustituya la *contraseña* por la contraseña del usuario que haya establecido.

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

Tip

Según la configuración de Windows, es posible que la contraseña del usuario caduque en una fecha futura. Para garantizar que sus aplicaciones de Greengrass sigan funcionando, controle cuándo caduque la contraseña y actualícela antes de que caduque. También puede configurar la contraseña del usuario para que nunca caduque.

- Para comprobar cuándo caducan un usuario y su contraseña, ejecuta el siguiente comando.

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

- Para configurar la contraseña de un usuario para que no caduque nunca, ejecute el siguiente comando.

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

- Si utilizas Windows 10 o una versión posterior, donde el `wmic` comando está obsoleto, ejecuta el siguiente PowerShell comando.

  ```
  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

Al actualizar Stream Manager v2.0.7 a una versión entre la v2.0.8 y la v2.0.11, es posible que aparezca el siguiente error en los registros del componente de Stream Manager si el componente no se inicia.

```
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}
```

Si has implementado la versión 2.0.7 de Stream Manager y quieres actualizarla a una versión posterior, debes actualizar directamente a la versión 2.0.12 de Stream Manager. Para obtener más información sobre el componente de administrador de transmisiones, consulte. Administrador de transmisiones

Problemas con los componentes de la función Lambda del dispositivo principal

Solucione problemas con los componentes de la función Lambda en los dispositivos principales.

Temas

- 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

Es posible que aparezca este error al ejecutar una función Lambda en un contenedor en los siguientes casos:

- El dispositivo principal no tiene cgroup v1 activado para la memoria o los cgroups del dispositivo.
- El dispositivo principal tiene cgroups v2 activado. Las funciones Lambda de Greengrass requieren cgroups v1, y los cgroups v1 y v2 se excluyen mutuamente.

Para habilitar cgroups v1, arranque el dispositivo con los siguientes parámetros del núcleo de Linux.

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

> **Tip**
>
> En una Raspberry Pi, edite el `/boot/cmdline.txt` archivo para configurar los parámetros del núcleo del dispositivo.

ipc_client.py:64,HTTP Error 400:Bad Request, b'No subscription exists for the source <label-or-lambda-arn> and subject <label-or-lambda-arn>

Es posible que aparezca este error al ejecutar una función Lambda de la versión 1, que utiliza el SDK AWS IoT Greengrass principal, en un dispositivo principal de la versión 2 sin especificar una suscripción en el componente del router de suscripciones anterior. Para solucionar este problema, implementa y configura el router de suscripciones antiguo para especificar las suscripciones necesarias. Para obtener más información, consulte Importación de funciones Lambda V1.

Versión del componente discontinuada

Es posible que veas una notificación en tu Personal Health Dashboard (PHD) cuando se deje de fabricar una versión de un componente de tu dispositivo principal. La versión componente envía esta notificación a su médico en un plazo de 60 minutos desde su descatalogación.

Para ver qué implementaciones necesita revisar, haga lo siguiente mediante: AWS Command Line Interface

1. Ejecute el siguiente comando para obtener una lista de sus dispositivos principales.

   ```
   aws greengrassv2 list-core-devices
   ```

2. Ejecute el siguiente comando para recuperar el estado de los componentes de cada dispositivo principal del paso 1. *coreDeviceName*Sustitúyalo por el nombre de cada dispositivo principal que desee realizar la consulta.

   ```
   aws greengrassv2 list-installed-components --core-device-thing-name coreDeviceName
   ```

3. Reúna los dispositivos principales con la versión de componentes discontinuada instalada en los pasos anteriores.
4. Ejecute el siguiente comando para recuperar el estado de todos los trabajos de implementación de cada dispositivo principal del paso 3. *coreDeviceName*Sustitúyalo por el nombre del dispositivo principal que se va a consultar.

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

   La respuesta contiene la lista de trabajos de implementación del dispositivo principal. Puede revisar la implementación para elegir otra versión del componente. Para obtener más información sobre cómo revisar un despliegue, consulte Revisar despliegues.

Problemas con la interfaz de línea de comandos de Greengrass

Solucione problemas con la CLI de Greengrass.

Temas

- java.lang.RuntimeException: Unable to create ipc client

java.lang.RuntimeException: Unable to create ipc client

Es posible que aparezca este error al ejecutar un comando de la CLI de Greengrass y especificar una carpeta raíz diferente a la que está instalado el software AWS IoT Greengrass principal.

Realice una de las siguientes acciones para establecer la ruta raíz y */greengrass/v2* sustitúyala por la ruta a la instalación del software AWS IoT Greengrass principal:

- Establezca la variable de entorno `GGC_ROOT_PATH` en */greengrass/v2*.
- Añada el `--ggcRootPath` */greengrass/v2* argumento al comando, tal y como se muestra en el siguiente ejemplo.

```
greengrass-cli --ggcRootPath /greengrass/v2 <command> <subcommand> [arguments]
```

AWS Command Line Interface problemas

Solucionar AWS CLI problemas para. AWS IoT Greengrass V2

Temas

Error: Invalid choice: 'greengrassv2'

Es posible que aparezca este error al ejecutar un AWS IoT Greengrass V2 comando con AWS CLI (por ejemplo,`aws greengrassv2 list-core-devices`).

Este error indica que tienes una versión de la AWS CLI que no es compatible AWS IoT Greengrass V2. Para AWS IoT Greengrass V2 utilizarla con AWS CLI, debe tener una de las siguientes versiones o una posterior:

- Versión AWS CLI V1 mínima: v1.18.197
- Versión AWS CLI V2 mínima: v2.1.11

> **Tip**
>
> Puede ejecutar el siguiente comando para comprobar la versión de la AWS CLI que dispone.
>
> ```
> aws --version
> ```

Para resolver este problema, actualícelo AWS CLI a una versión posterior que sea compatible AWS IoT Greengrass V2. Para obtener más información, consulte Instalar, actualizar y desinstalar la AWS CLI en la Guía del usuario de AWS Command Line Interface .

Códigos de error de implementación detallados

Utilice los códigos de error y las soluciones de estas secciones para ayudar a resolver los problemas relacionados con la implementación de los componentes al utilizar la versión 2.8.0 o posterior del núcleo de Greengrass.

El núcleo de Greengrass informa de los errores de implementación en forma jerárquica, desde el código menos específico hasta el más específico disponible. Puede usar esta jerarquía para ayudar a identificar el motivo de un error de implementación. Por ejemplo, la siguiente es una posible jerarquía de errores:

- FALLO_DE_IMPLEMENTACIÓN
  - ERROR DE DESCARGA DEL ARTEFACTO
    - IO_ERROR
      - ESPACIO DE DISCO CRÍTICO

Los códigos de error se organizan en tipos. Cada tipo representa una clase de errores que pueden producirse.AWS IoT Greengrassinforma de estos tipos de errores en la consola, la API yAWS CLI. Puede haber más de un tipo de error, según los errores notificados en la jerarquía de errores. Para el ejemplo anterior, el tipo de error devuelto es`DEVICE_ERROR`.

Los tipos son:

- ERROR_DE_PERMISO— Se denegó el acceso a una operación que requiere permiso.
- ERROR_SOLICITUD— Se ha producido un error debido a un problema en el documento de despliegue.
- ERROR DE RECETA DEL COMPONENTE— Se ha producido un error debido a un problema en la receta de un componente.
- AWS_COMPONENT_ERROR— Se ha producido un error al iniciar o eliminar unAWScomponente proporcionado.
- ERROR_COMPONENT_DE_USUARIO— Se ha producido un error al iniciar o eliminar un componente de usuario.
- ERROR_COMPONENTE— Se ha producido un error al iniciar o eliminar un componente, pero el núcleo de Greengrass no pudo determinar si el componente es unAWSun componente proporcionado o un componente de usuario.

- ERROR DE DISPOSITIVO— Se ha producido un error con la E/S local o se ha producido otro error en el dispositivo.
- ERROR_DE_DEPENDENCIA— No se pudo descargar un artefacto de Amazon S3 ni extraer una imagen de un registro de ECR.
- HTTP_ERROR— Se ha producido un error con una solicitud HTTP.
- ERROR_DE_RED— Se ha producido un error en la red del dispositivo.
- NUCLEUS_ERROR— El núcleo de Greengrass no pudo localizar un componente o no pudo encontrar la versión del núcleo activo.
- ERROR_DE_SERVIDOR— Un servidor devolvió un error 500 en respuesta a una solicitud.
- ERROR DE SERVICIO EN LA NUBE— Se ha producido un error con elAWS IoT Greengrassservicio en la nube.
- ERROR_DESCONOCIDO— El componente ha lanzado una excepción no marcada.

Muchos de los errores de esta sección contienen información adicional en laAWS IoT GreengrassRegistros básicos. Estos registros se almacenan en el sistema de archivos local del dispositivo principal. Hay registros paraAWS IoT GreengrassSoftware básico y para cada componente individual. Para obtener información sobre el acceso a los registros, consulteAcceda a los registros del sistema de archivos.

Error de permiso

ACCESO_DENEGADO

Es posible que aparezca este error cuandoAWSla operación de servicio devuelve un error 403 porque los permisos no están configurados correctamente. Consulte el código de error más específico para obtener más información.

GET_DEPLOYMENT_CONFIGURATION_ACCESS_DENIED

Es posible que aparezca este error cuandoAWS IoTla política no permite el permiso para llamar al`GetDeploymentConfiguration`operación. Añada el`greengrass::GetDeploymentConfiguration`permiso para aplicar la política del dispositivo principal.

GET_COMPONENT_VERSION_ARTIFACT_ACCESS_DENIED

Es posible que aparezca este error cuando el dispositivo principalAWS IoTla política no permite la`greengrass:GetComponentVersionArtifact`permiso. Añada el permiso a la política del dispositivo principal.

RESOLVE_COMPONENT_CANDIDATES_ACCESO_DENEGADO

Es posible que aparezca este error cuando el dispositivo principalAWS IoTla política no permite la`greengrass:ResolveComponentCandidates`permiso. Añada el permiso a la política del dispositivo principal.

GET_ECR_CREDENTIAL_ERROR

Es posible que aparezca este error cuando la implementación no pueda autenticarse con un registro privado en ECR. Compruebe si hay algún error específico en el registro y, a continuación, vuelva a intentar la implementación.

USER_NOT_AUTHORIZED_FOR_DOCKER

Es posible que aparezca este error cuando el usuario de Greengrass no esté autorizado a usar Docker. Asegúrese de que está ejecutando Greengrass como root o de que el usuario esté agregado a`docker`grupo. A continuación, vuelva a intentar la implementación.

S3_ACCESS_DENIED

Es posible que aparezca este error cuando una operación de Amazon S3 devuelva un error 403. Consulte cualquier código de error o registro adicional para obtener más información.

S3_HEAD_OBJECT_ACCESS_DENIED

Es posible que aparezca este error cuando la función de intercambio de tokens del dispositivo no permita laAWS IoT GreengrassSoftware principal para descargar el artefacto componente desde la URL del objeto S3 que especifique en la receta del componente o si el artefacto del componente no está disponible. Compruebe que la función de intercambio de tokens permita`s3:GetObject`para la URL del objeto S3 donde el artefacto está disponible y donde el artefacto está presente.

S3_GET_BUCKET_LOCATION_ACCESS_DENIED

Es posible que aparezca este error cuando la función de intercambio de tokens del dispositivo no permita las`s3:GetBucketLocation`permiso para el bucket de Amazon S3 donde está disponible el artefacto. Compruebe que el dispositivo permite el permiso y, a continuación, vuelva a intentar la implementación.

S3_GET_OBJECT_ACCESS_DENIED

Es posible que aparezca este error cuando la función de intercambio de tokens del dispositivo no permita laAWS IoT GreengrassSoftware principal para descargar el artefacto componente desde la URL del objeto S3 que especifique en la receta del componente o si el artefacto del componente no está disponible. Compruebe que la función de intercambio de tokens permitas3:GetObjectpara la URL del objeto S3 donde el artefacto está disponible y donde el artefacto está presente.

Error en la solicitud

NÚCLEUS_FALTAS_CAPACIDADES_REQUERIDAS

Es posible que aparezca este error cuando la versión de núcleo de la implementación no permita realizar una operación solicitada, como descargar una configuración grande o establecer límites de recursos de Linux. Vuelva a intentar la implementación con una versión de núcleo que admita la operación.

ERROR RESUELTO DE MÚLTIPLES NÚCLEOS

Es posible que aparezca este error cuando una implementación intente implementar varios componentes del núcleo. Consulte el registro para ver la causa del error y, a continuación, consulte la página de actualización del software del núcleo para ver si el problema se ha corregido en una versión posterior del núcleo o póngase en contacto conAWS Support.

ERROR DE DEPENDENCIA CIRCULAR DEL COMPONENTE

Es posible que aparezca este error cuando dos componentes de la implementación dependan el uno del otro. Revise la configuración de los componentes para que los componentes de la implementación no dependan unos de otros.

UNAUTHORIZED_NUCLEUS_MINOR_VERSION_UPDATE

Es posible que aparezca este error cuando un componente de su implementación requiera una actualización de la versión secundaria de Nucleus, pero esa versión no esté especificada en la implementación. Esto ayuda a reducir las actualizaciones accidentales de versiones menores de los componentes que dependen de una versión diferente. Incluya la nueva versión del núcleo secundario en la implementación.

FALTA EL ADMINISTRADOR DE APLICACIONES DE DOCKER

Es posible que aparezca este error al implementar un componente de Docker sin implementar el administrador de aplicaciones de Docker. Asegúrese de que su implementación incluya el administrador de aplicaciones Docker.

FALTA EL SERVICIO_DE_INTERCAMBIO DE TOKENS

Es posible que aparezca este error cuando la implementación quiera descargar un artefacto de imagen de Docker de un registro ECR privado sin implementar el servicio de intercambio de tokens. Asegúrese de que la implementación incluya el servicio de intercambio de tokens.

LOS REQUISITOS DE VERSIÓN DEL COMPONENTE NO SE CUMPLEN

Es posible que aparezca este error cuando haya un conflicto de restricciones de versión o cuando no exista una versión de componente. Para obtener más información, consulte Error: com.aws.greengrass.componentmanager.exceptions.NoAvailableComponentVersionException: Failed to negotiate component <name> version with cloud and no local applicable version satisfying requirement <requirements>.

ERROR_DE_ACELERACIÓN

Es posible que aparezca este error cuandoAWSla operación del servicio superó una cuota tarifaria. Reintente la implementación.

SOLICITUD_CONFLICTIVA

Es posible que aparezca este error cuandoAWSla operación de servicio devuelve un error 409 porque la implementación intenta realizar más de una operación a la vez. Reintente la implementación.

RECURSO_NO_ENCONTRADO

Es posible que aparezca este error cuandoAWSla operación de servicio devuelve un error 404 porque no se pudo encontrar un recurso. Compruebe el registro para ver si falta el recurso.

RUN_WITH_CONFIG_NOT_VALID

Es posible que aparezca este error cuando`posixUser`,`posixGroup`, o`windowsUser`la información especificada para ejecutar el componente no es válida. Compruebe que el usuario es válido y, a continuación, vuelva a intentar la implementación.

REGIÓN_NO COMPATIBLE

Es posible que aparezca este error cuando la región especificada para la implementación no sea compatible conAWS IoT Greengrass. Compruebe la región y vuelva a intentar la implementación.

IOT_CRED_ENDPOINT_NO_VALID

Es posible que aparezca este error cuandoAWS IoTel punto final de la credencial especificado en la configuración no es válido. Compruebe el punto de conexión e intente realizar la solicitud de nuevo.

IOT_DATA_ENDPOINT_NO_VALID

Es posible que aparezca este error cuandoAWS IoTel punto final de datos especificado en la configuración no es válido. Compruebe el punto de conexión e intente realizar la solicitud de nuevo.

S3_HEAD_OBJECT_RESOURCE_NOT_FOUND

Es posible que aparezca este error cuando el artefacto del componente no esté disponible en la URL del objeto S3 que especifiques en la receta del componente. Compruebe que ha subido el artefacto al bucket de S3 y que el URI del artefacto coincide con la URL del objeto S3 del artefacto del bucket.

S3_GET_BUCKET_LOCATION_RESOURCE_NOT_FOUND

Es posible que aparezca este error si no se encuentra el bucket de Amazon S3. Compruebe que el bucket existe y vuelva a intentar la implementación.

S3_GET_OBJECT_RESOURCE_NOT_FOUND

Es posible que aparezca este error cuando el artefacto del componente no esté disponible en la URL del objeto S3 que especifiques en la receta del componente. Compruebe que ha subido el artefacto al bucket de S3 y que el URI del artefacto coincide con la URL del objeto S3 del artefacto del bucket.

IO_MAPPING_ERROR

Es posible que aparezca este error cuando se produce un error de E/S al analizar el documento o la receta de implementación. Consulte cualquier código de error o registro adicional para obtener más información.

Error de receta de componentes

RECIPE_PARSE_ERROR

Es posible que aparezca este error cuando no se pueda analizar la receta de despliegue porque hay un error en la estructura de la receta. Compruebe que la receta tenga el formato correcto y vuelva a intentar la implementación.

RECIPE_METADATA_PARSE_ERROR

Es posible que aparezca este error cuando no se puedan analizar los metadatos de la receta de implementación descargados de la nube. Póngase en contacto con AWS Support.

ARTEFACTO_URI_NO_VÁLIDO

Es posible que aparezca este error cuando el URI de un artefacto de una receta no tenga el formato correcto. Compruebe el registro para ver si el URI no es válido, actualice el URI de la receta y, a continuación, vuelva a intentar la implementación.

S3_ARTIFACT_URI_NOT_VALID

Es posible que aparezca este error cuando la URI de Amazon S3 de un artefacto de una receta no sea válida. Compruebe el registro para ver si el URI no es válido, actualice el URI de la receta y, a continuación, vuelva a intentar la implementación.

DOCKER_ARTIFACT_URI_NOT_VALID

Es posible que aparezca este error cuando la URI de Docker de un artefacto de una receta no sea válida. Compruebe el registro para ver si el URI no es válido, actualice el URI de la receta y, a continuación, vuelva a intentar la implementación.

ARTEFACT_URI VACÍO

Es posible que aparezca este error cuando la URI de un artefacto no esté especificada en una receta. Compruebe el registro del artefacto al que le falta un URI, actualice el URI de la receta y, a continuación, vuelva a intentar el despliegue.

ESQUEMA_DE_ARTEFACTO_VACÍO

Es posible que aparezca este error cuando no se haya definido un esquema de URI para un artefacto. Compruebe el registro para ver si el URI no es válido, actualice el URI de la receta y, a continuación, vuelva a intentar la implementación.

ESQUEMA_ARTE_DE_ARTEFACTO NO COMPATIBLE

Es posible que aparezca este error si la versión de núcleo en ejecución no admite un esquema de URI. Un URI no es válido o necesita actualizar la versión del núcleo. Si el URI no es válido, compruebe en el registro el URI no válido, actualice el URI de la receta y, a continuación, vuelva a intentar la implementación.

MANIFIESTO FALTANTE DE RECETA

Es posible que aparezca este error si la sección del manifiesto no está incluida en la receta. Añada el manifiesto a la receta y vuelva a intentar la implementación.

RECIPE_MISSING_ARTIFACT_HASH_ALGORITHM

Es posible que aparezca este error cuando se especifique un artefacto que no es local dentro de una receta sin un algoritmo de hash. Añada el algoritmo al artefacto y, a continuación, vuelva a intentar la solicitud.

ARTIFACT_CHECKSUM_DISCORDANCIA

Es posible que aparezca este error cuando un artefacto descargado tenga un resumen diferente al especificado en la receta. Asegúrese de que la receta contenga el resumen correcto y, a continuación, vuelva a intentar la implementación. Para obtener más información, consulte Error: com.aws.greengrass.componentmanager.exceptions.ArtifactChecksumMismatchException: Integrity check for downloaded artifact failed. Probably due to file corruption..

LA DEPENDENCIA DEL COMPONENTE NO ES VÁLIDA

Es posible que aparezca este error cuando el tipo de dependencia especificado en una receta de despliegue no sea válido. Comprueba la receta y, a continuación, vuelve a intentar la solicitud.

CONFIG_INTERPOLATE_ERROR

Es posible que aparezca este error al interpolar una variable de receta. Consulte el registro para obtener más información.

IO_MAPPING_ERROR

Es posible que aparezca este error cuando se produce un error de E/S al analizar el documento o la receta de implementación. Consulte cualquier código de error o registro adicional para obtener más información.

AWSerror de componente, error de componente del usuario, error de componente

Los siguientes códigos de error se devuelven cuando hay un problema con un componente. El tipo de error real notificado depende del componente específico que provocó el error. Si el núcleo de Greengrass identifica el componente como uno proporcionado porAWS IoT Greengrass, regresa`AWS_COMPONENT_ERROR`. Si el componente se identifica como un componente de usuario, el núcleo de Greengrass devuelve`USER_COMPONENT_ERROR`. Si el núcleo de Greengrass no puede decirlo, regresa`COMPONENT_ERROR`.

ERROR DE ACTUALIZACIÓN DEL COMPONENTE

Es posible que aparezca este error cuando un componente no se actualiza durante una implementación. Compruebe cualquier código de error adicional o consulte el registro para ver la causa del error.

COMPONENT_ROTO

Es posible que aparezca este error cuando un componente se interrumpe durante una implementación. Consulte el registro de componentes para ver los detalles del error y, a continuación, vuelva a intentar la implementación.

ELIMINAR_COMPONENT_ERROR

Es posible que aparezca este error cuando el núcleo no pueda eliminar un componente durante una implementación. Compruebe el registro para ver los detalles del error y, a continuación, vuelva a intentar la implementación.

COMPONENT_BOOTSTRAP_TIMEOUT

Es posible que aparezca este error cuando la tarea de arranque de un componente haya tardado más que el tiempo de espera configurado. Aumente el tiempo de espera o reduzca el tiempo de ejecución de la tarea de arranque y, a continuación, vuelva a intentar la implementación.

COMPONENT_BOOTSTRAP_ERROR

Es posible que aparezca este error cuando la tarea de arranque de un componente contenga un error. Compruebe el registro para ver los detalles del error y, a continuación, vuelva a intentar la implementación.

LA CONFIGURACIÓN DEL COMPONENTE NO ES VÁLIDA

Es posible que aparezca este error cuando el núcleo no pueda validar la configuración implementada para el componente. Compruebe el registro para ver los detalles del error y, a continuación, vuelva a intentar la implementación.

Error de dispositivo

IO_WRITE_ERROR

Es posible que aparezca este error al escribir en un archivo. Consulte el registro para obtener más información.

IO_READ_ERROR

Es posible que aparezca este error al leer un archivo. Consulte el registro para obtener más información.

ESPACIO DE DISCO CRÍTICO

Es posible que aparezca este error cuando no haya suficiente espacio en disco para completar una solicitud de despliegue. Debes tener al menos 20 Mb de espacio disponible, o lo suficiente para contener un artefacto más grande. Libere espacio en disco y, a continuación, vuelva a intentar la implementación.

IO_FILE_ATTRIBUTE_ERROR

Es posible que aparezca este error cuando no se pueda recuperar el tamaño del archivo existente del sistema de archivos. Consulte el registro para obtener más información.

SET_PERMISSION_ERROR

Es posible que aparezca este error cuando no se puedan configurar los permisos en un artefacto o directorio de artefactos descargado. Consulte el registro para obtener más información.

IO_UNZIP_ERROR

Es posible que aparezca este error cuando no se pueda descomprimir un artefacto. Consulte el registro para obtener más información.

LOCAL_RECIPE_NO_ENCONTRADA

Es posible que aparezca este error si no se encuentra la copia local de un archivo de recetas. Vuelva a intentar la implementación.

LOCAL_RECIPE_CORRUPTO

Es posible que aparezca este error cuando la copia local de la receta haya cambiado desde que se descargó. Elimine la copia existente de la receta y vuelva a intentar la implementación.

LOCAL_RECIPE_METADATA_NO_ENCONTRADA

Es posible que aparezca este error si no se encuentra la copia local del archivo de metadatos de la receta. Vuelva a intentar la implementación.

LAUNCH_DIRECTORY_CORRUPTO

Es posible que aparezca este error cuando el directorio utilizado para lanzar el núcleo de Greengrass (`/greengrass/v2/alts/current`) se ha modificado desde la última vez que se puso en marcha el núcleo. Reinicie el núcleo y, a continuación, vuelva a intentar el despliegue.

ALGORITMO DE HASHING_NO DISPONIBLE

Es posible que aparezca este error cuando la distribución de Java del dispositivo no admite el algoritmo de hash requerido o cuando el algoritmo de hash especificado en la receta de un componente no es válido.

DEVICE_CONFIG_NOT_VALID_PAR_ARTEFACT_DOWNLOAD

Es posible que aparezca este error cuando haya un error en la configuración del dispositivo que impida que la implementación descargue el artefacto de Amazon S3 o de la nube de Greengrass. Compruebe el registro para ver si hay algún error de configuración específico y, a continuación, vuelva a intentar la implementación.

Error de dependencia

DOCKER_ERROR

Es posible que aparezca este error al extraer una imagen de Docker. Consulte cualquier código de error o registro adicional para obtener más información.

DOCKER_SERVICE_UNAVAILABLE

Es posible que aparezca este error cuando Greengrass no pueda iniciar sesión en el registro de Docker. Compruebe si hay algún error específico en el registro y, a continuación, vuelva a intentar la implementación.

DOCKER_LOGIN_ERROR

Es posible que aparezca este error cuando se produzca un error inesperado al iniciar sesión en Docker. Compruebe si hay algún error específico en el registro y, a continuación, vuelva a intentar la implementación.

DOCKER_PULL_ERROR

Es posible que aparezca este error cuando se produzca un error inesperado al extraer una imagen de Docker del registro. Compruebe si hay algún error específico en el registro y, a continuación, vuelva a intentar la implementación.

DOCKER_IMAGE_NOT_VALID

Es posible que aparezca este error cuando la imagen de Docker solicitada no exista. Compruebe si hay algún error específico en el registro y vuelva a intentar la implementación.

DOCKER_IMAGE_QUERY_ERROR

Es posible que aparezca este error cuando se produzca un error inesperado al consultar Docker las imágenes disponibles. Compruebe el registro para ver el error específico y vuelva a intentar la implementación.

S3_ERROR

Es posible que aparezca este error al descargar un artefacto de Amazon S3. Consulte cualquier código de error o registro adicional para obtener más información.

S3_RESOURCE_NOT_FOUND

Es posible que aparezca este error cuando una operación de Amazon S3 devuelva un error 404. Consulte cualquier código de error o registro adicional para obtener más información.

S3_BAD_REQUEST

Es posible que aparezca este error cuando una operación de Amazon S3 devuelva un error 400. Compruebe si hay algún error específico en el registro e intente realizar la solicitud de nuevo.

Error HTTP

HTTP_REQUEST_ERROR

Es posible que aparezca este error cuando se produce un error al realizar una solicitud HTTP. Compruebe el registro para ver el error específico.

ERROR DE DESCARGAR_DEPLOYMENT_DOCUMENT_ERROR

Es posible que aparezca este error si se produce un error HTTP al descargar el documento de despliegue. Compruebe el registro para ver el error HTTP específico.

GET_GREENGRASS_ARTIFACT_SIZE_ERROR

Es posible que aparezca este error cuando se produce un error HTTP al obtener el tamaño de un artefacto de componente público. Compruebe el registro para ver el error HTTP específico.

DESCARGAR_GREENGRASS_ARTIFACT_ERROR

Es posible que aparezca este error cuando se produce un error HTTP al descargar un artefacto de componente público. Compruebe el registro para ver el error HTTP específico.

Error de red

ERROR_DE_RED

Es posible que aparezca este error cuando haya un problema de conexión durante una implementación. Compruebe la conexión del dispositivo a Internet y vuelva a intentar la implementación.

Error de núcleo

SOLICITUD_INCORRECTA

Es posible que aparezca este error cuandoAWSla operación en la nube devuelve un error 400. Consulte el registro para ver qué API causó el error y, a continuación, consulte la página de actualización del software del núcleo para ver si el problema se ha corregido en una versión posterior del núcleo o póngase en contacto conAWS Support.

NO SE ENCONTRÓ LA VERSIÓN_DEL NÚCLEO

Es posible que aparezca este error cuando un dispositivo central no pueda encontrar la versión del núcleo activo. Consulte el registro para ver la causa del error y, a continuación, consulte la página de actualización del software del núcleo para ver si el problema se ha corregido en una versión posterior del núcleo o póngase en contacto conAWS Support.

FALLO DE REINICIO DEL NÚCLEO

Es posible que aparezca este error cuando el núcleo no se reinicie durante ninguna implementación que requiera un reinicio del núcleo. Consulte el registro del cargador para ver la causa del error y, a continuación, consulte la página de actualización del software del núcleo para ver si el problema se ha corregido en una versión posterior del núcleo o póngase en contacto conAWS Support.

EL COMPONENTE INSTALADO NO SE ENCONTRÓ

Es posible que aparezca este error cuando el núcleo no pueda localizar un componente instalado. Consulte el registro para ver la causa del error y, a continuación, consulte la página de actualización del software del núcleo para ver si el problema se ha corregido en una versión posterior del núcleo o póngase en contacto conAWS Support.

EL DOCUMENTO DE DESPLIEGUE NO ES VÁLIDO

Es posible que aparezca este error cuando el dispositivo reciba un documento de despliegue que no sea válido. Compruebe cualquier código de error adicional o consulte el registro para ver la causa del error.

SOLICITU_DE_DESPLIEGUE VACÍA

Es posible que aparezca este error cuando un dispositivo reciba una solicitud de despliegue vacía. Consulte el registro para ver la causa del error y, a continuación, consulte la página de actualización del software del núcleo para ver si el problema se ha corregido en una versión posterior del núcleo o póngase en contacto conAWS Support.

DEPLOYMENT_DOCUMENT_PARSE_ERROR

Es posible que aparezca este error cuando el formato de la solicitud de despliegue no coincida con el formato esperado. Consulte el registro para ver la causa del error y, a continuación, consulte la página de actualización del software del núcleo para ver si el problema se ha corregido en una versión posterior del núcleo o póngase en contacto conAWS Support.

COMPONENT_METADATA_NO_VALID_IN_DEPLOYMENT

Es posible que aparezca este error cuando la solicitud de despliegue contenga metadatos de componentes que no sean válidos. Consulte el registro para ver la causa del error y, a continuación, consulte la página de actualización del software del núcleo para ver si el problema se ha corregido en una versión posterior del núcleo o póngase en contacto conAWS Support.

LAUNCH_DIRECTORY_CORRUPTO

Es posible que aparezca este error al mover un dispositivo de Greengrass de un grupo de cosas a otro y, después, volver al grupo original con implementaciones que requieren que Greengrass se reinicie. Para resolver el error, vuelva a crear el directorio de inicio de Greengrass en el dispositivo.

Para obtener más información, consulte 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.

Error de servidor

ERROR_DE_SERVIDOR

Es posible que aparezca este error cuandoAWSla operación de servicio devuelve un error 500 porque el servicio no puede procesar la solicitud en este momento. Vuelva a intentar la implementación más adelante.

S3_SERVER_ERROR

Es posible que aparezca este error cuando una operación de Amazon S3 devuelva un error 500. Consulte cualquier código de error o registro adicional para obtener más información.

Error de servicio en la nube

RESOLVE_COMPONENT_CANDIDATES_BAD_RESPONSE

Es posible que aparezca este error cuando el servicio en la nube de Greengrass envíe una respuesta incompatible a`ResolveComponentCandidates`operación. Consulte el registro para ver la causa del error y, a continuación, consulte la página de actualización del software del núcleo para ver si el problema se ha corregido en una versión posterior del núcleo o póngase en contacto conAWS Support.

SE HA SUPERADO EL TAMAÑO DEL DOCUMENTO DE DESPLIEGUE

Es posible que aparezca este error cuando el documento de despliegue solicitado supere la cuota de tamaño máximo. Reduzca el tamaño del documento de despliegue e inténtelo de nuevo.

GREENGRASS_ARTIFACT_SIZE_NO_FOUND

Es posible que aparezca este error cuando Greengrass no pueda obtener el tamaño de un artefacto de componente público. Consulte el registro para ver la causa del error y, a continuación, consulte la página de actualización del software del núcleo para ver si el problema se ha corregido en una versión posterior del núcleo o póngase en contacto conAWS Support.

EL DOCUMENTO DE DESPLIEGUE NO ES VÁLIDO

Es posible que aparezca este error cuando el dispositivo reciba un documento de despliegue que no sea válido. Compruebe cualquier código de error adicional o consulte el registro para ver la causa del error.

SOLICITU_DE_DESPLIEGUE VACÍA

Es posible que aparezca este error cuando un dispositivo reciba una solicitud de despliegue vacía. Consulte el registro para ver la causa del error y, a continuación, consulte la página de actualización del software del núcleo para ver si el problema se ha corregido en una versión posterior del núcleo o póngase en contacto conAWS Support.

DEPLOYMENT_DOCUMENT_PARSE_ERROR

Es posible que aparezca este error cuando el formato de la solicitud de despliegue no coincida con el formato esperado. Consulte el registro para ver la causa del error y, a continuación, consulte la página de actualización del software del núcleo para ver si el problema se ha corregido en una versión posterior del núcleo o póngase en contacto conAWS Support.

COMPONENT_METADATA_NO_VALID_IN_DEPLOYMENT

Es posible que aparezca este error cuando la solicitud de despliegue contenga metadatos de componentes que no sean válidos. Consulte el registro para ver la causa del error y, a continuación, consulte la página de actualización del software del núcleo para ver si el problema se ha corregido en una versión posterior del núcleo o póngase en contacto conAWS Support.

Errores genéricos

Estos errores genéricos no tienen un tipo de error asociado.

IMPLEMENTACIÓN_INTERRUMPIDA

Es posible que aparezca este error cuando no se pueda completar una implementación debido a un cierre del núcleo u otro evento externo. Consulte cualquier código de error o registro adicional para obtener más información.

ERROR DE DESCARGA DEL ARTEFACTO

Es posible que aparezca este error cuando haya un problema al descargar un artefacto. Consulte cualquier código de error o registro adicional para obtener más información.

NINGUNA VERSIÓN_COMPONENT_DISPONIBLE

Es posible que aparezca este error cuando la versión de un componente no exista en la nube o de forma local, o si hay un conflicto de resolución de dependencias. Consulte cualquier código de error o registro adicional para obtener más información.

ERROR DE CARGA DEL PAQUETE DEL COMPONENTE

Es posible que aparezca este error cuando se procesen los artefactos descargados. Consulte cualquier código de error o registro adicional para obtener más información.

ERROR DE API EN LA NUBE

Es posible que aparezca este error cuando se produce un error al llamar a unAWSAPI de servicio. Consulte cualquier código de error o registro adicional para obtener más información.

IO_ERROR

Es posible que aparezca este error cuando se produzca un error de E/S durante una implementación. Consulte cualquier código de error o registro adicional para obtener más información.

ERROR DE ACTUALIZACIÓN DEL COMPONENTE

Es posible que aparezca este error cuando un componente no se actualiza durante una implementación. Compruebe cualquier código de error adicional o consulte el registro para ver la causa del error.

Error desconocido

FALLO_DE_IMPLEMENTACIÓN

Es posible que aparezca este error cuando se produce un error en una implementación porque se ha producido una excepción no marcada. Consulte el registro para ver la causa del error y, a continuación, consulte la página de actualización del software del núcleo para ver si el problema se ha corregido en una versión posterior del núcleo o póngase en contacto conAWS Support.

TIPO_DE_DE_IMPLEMENTACIÓN_NO_VÁLIDO

Es posible que aparezca este error cuando el tipo de despliegue no sea válido. Consulte el registro para ver la causa del error y, a continuación, consulte la página de actualización del software del núcleo para ver si el problema se ha corregido en una versión posterior del núcleo o póngase en contacto conAWS Support.

Códigos de estado detallados de los componentes

Utilice los códigos de estado y las soluciones de estas secciones para ayudar a resolver problemas con los componentes al utilizar la versión 2.8.0 o posterior del núcleo de Greengrass.

Muchos de los estados de este tema incluyen información adicional en los registros AWS IoT Greengrass principales. Estos registros se almacenan en el sistema de archivos local del dispositivo principal. Hay registros para cada componente individual. Para obtener información sobre el acceso a los registros, consulteAcceda a los registros del sistema de archivos.

ERROR DE INSTALACIÓN

Es posible que aparezca cuando se produce un error al ejecutar un script de instalación. El código de error se indica en el registro del componente. Compruebe si hay errores en el script de instalación y vuelva a implementar el componente.

INSTALL_CONFIG_NOT_VALID

Es posible que aparezca este error cuando no se pueda completar la instalación de un componente porque la `Install` sección de la receta no es válida. Consulte la sección de instalación de su receta para ver si hay errores e intente la implementación de nuevo.

INSTALL_IO_ERROR

Esto puede ocurrir cuando se produce un error de E/S durante la instalación de un componente. Consulte el registro de errores del componente para obtener detalles sobre el error.

INSTALL_MISSING_DEFAULT_RUN WITH

Es posible que aparezca este error cuando no AWS IoT Greengrass pueda determinar el usuario o el grupo que se va a utilizar al instalar un componente. Asegúrese de que la `runWith` sección de la receta de instalación incluye un usuario o grupo válido.

TIEMPO DE ESPERA DE INSTALACIÓN

Es posible que aparezca este error si el script de instalación no finaliza dentro del período de tiempo de espera configurado. Aumente el `Timeout` período especificado en la `Install` sección de la receta o modifique el script de instalación para que finalice dentro del tiempo de espera configurado.

ERROR_DE_INICIO

Es posible que aparezca cuando se produce un error al ejecutar un script de inicio. El código de error se indica en el registro del componente. Compruebe si hay errores en el script de instalación y vuelva a implementar el componente.

STARTUP_CONFIG_NOT_VALID

Es posible que aparezca este error cuando no se pueda completar la instalación de un componente porque la `Startup` sección de la receta no es válida. Consulte la sección de inicio de su receta para ver si hay errores e intente la implementación de nuevo.

STARTUP_IO_ERROR

Esto puede ocurrir cuando se produce un error de E/S durante el inicio de un componente. Consulte el registro de errores del componente para obtener detalles sobre el error.

STARTUP_MISSING_DEFAULT_RUN CON

Es posible que aparezca este error cuando no AWS IoT Greengrass pueda determinar el usuario o el grupo que se va a utilizar al ejecutar un componente. Asegúrese de que la `runWith` sección de su receta de inicio incluya un usuario o grupo válido.

TIEMPO DE ESPERA DE INICIO

Es posible que aparezca este error si el script de inicio no finaliza dentro del período de tiempo de espera configurado. Aumente el `Timeout` período especificado en la `Startup` sección de la receta o modifique el script de inicio para que finalice dentro del tiempo de espera configurado.

RUN_ERROR

Es posible que aparezca cuando se produce un error al ejecutar un script de componente. El código de error se indica en el registro del componente. Compruebe si hay errores en el script de ejecución y vuelva a implementar el componente.

RUN_MISSING_DEFAULT_RUN WITH

Es posible que aparezca este error cuando no AWS IoT Greengrass pueda determinar el usuario o el grupo que se va a utilizar al ejecutar un componente. Asegúrese de que la `runWith` sección de su receta de ejecución incluye un usuario o grupo válido.

RUN_CONFIG_NOT_VALID

Es posible que aparezca este error cuando no se pueda ejecutar un componente porque la `Run` sección de la receta no es válida. Consulte la sección de ejecución de su receta para ver si hay errores y vuelva a intentar la implementación.

RUN_IO_ERROR

Esto puede ocurrir cuando se produce un error de E/S mientras se ejecuta el componente. Consulte el registro de errores del componente para obtener detalles sobre el error.

TIEMPO DE ESPERA DE EJECUCIÓN

Es posible que aparezca este error cuando el script de ejecución no finalice dentro del período de tiempo de espera configurado. Aumente el `Timeout` período especificado en la `Run` sección de la receta o modifique el script de ejecución para que finalice dentro del tiempo de espera configurado.

ERROR_DE_APAGADO

Es posible que aparezca cuando se produce un error al cerrar un script de componente. El código de error se indica en el registro del componente. Compruebe si hay errores en el script de apagado y vuelva a implementar el componente.

TIMEOUT DE CIERRE

Es posible que aparezca este error si el script de apagado no finaliza dentro del período de tiempo de espera configurado. Aumente el `Timeout` período especificado en la `Shutdown` sección de la receta o modifique el script de ejecución para que finalice dentro del tiempo de espera configurado.

Etiquetar los recursos de AWS IoT Greengrass Version 2

Las etiquetas le permiten organizar y administrar sus recursos en AWS IoT Greengrass. Puede usar etiquetas para asignar metadatos a sus recursos y puede usar etiquetas en las políticas de IAM para definir el acceso condicional a sus recursos.

> **Note**
>
> En la actualidad, no se admiten etiquetas de recursos de Greengrass con los grupos de facturación o los informes de asignación de costos de AWS IoT.

Uso de etiquetas en AWS IoT Greengrass V2

Puede usar etiquetas para categorizar los recursos de AWS IoT Greengrass por finalidad, propietario, entorno o cualquier otra clasificación en función del caso de uso. Cuando tiene muchos recursos del mismo tipo, las etiquetas le ayuden a identificar un recurso específico con más facilidad.

Cada etiqueta está formada por una clave y un valor opcional, ambos definidos por el usuario. Por ejemplo, podría definir un conjunto de etiquetas para sus dispositivos principales que le permita realizar un seguimiento por parte de los clientes propietarios de los dispositivos. Le recomendamos que cree un conjunto de claves de etiqueta que cumpla sus necesidades para cada tipo de recurso. Si utiliza un conjunto coherente de claves de etiquetas, le será más fácil administrar los recursos.

Etiqueta con elAWS Management Console

El Tag Editor (Editor de etiquetas) de la AWS Management Console proporciona una forma central y unificada para que pueda crear y administrar las etiquetas para los recursos de todos los servicios de AWS. Para obtener más información, consulte Tag Editor en la Guía del usuario de AWS Resource Groups.

Etiqueta con laAWS IoT Greengrass V2 API

También puede utilizar laAWS IoT Greengrass V2 API para trabajar con etiquetas. Antes de crear etiquetas, tenga en cuenta las restricciones de etiquetado. Para obtener más información, consulte este artículo sobre las convenciones de nomenclatura y el uso de etiquetas en la Referencia general de AWS.

- Para agregar etiquetas al crear un recurso, deberá definirlas en la propiedad `tags` del recurso.
- Para añadir etiquetas a un recurso existente o actualizar los valores de las etiquetas, utilice la TagResourceoperación.
- Para eliminar etiquetas de un recurso de, utilice la UntagResourceoperación.
- Para recuperar las etiquetas asociadas a un recurso, utilice la ListTagsForResourceoperación o describa el recurso e inspeccione `sutags` propiedad.

La siguiente tabla muestra los recursos que puede etiquetar mediante laAWS IoT Greengrass V2 API y sus correspondientes`CreateDescribe` `uGet` operaciones.

Recursos etiquetables de AWS IoT Greengrass V2

Resource	Crear operación	Describa o opere
Dispositivo central	Ninguno. Ejecute el softwareA WS IoT Greengrass Core en un dispositivo para crear un dispositivo central.	GetCoreDevice
Componente	CreateComponentVersion	DescribeComponent, GetComponent
Implementación	CreateDeployment	GetDeployment

Utilice las siguientes operaciones para ver y administrar etiquetas para los recursos que admiten etiquetas:

- TagResource— Añade etiquetas a un recurso o actualiza el valor de una etiqueta existente.
- ListTagsForResource— Muestra las etiquetas de un recurso.
- UntagResource— Elimina etiquetas de un recurso.

Puede agregar o eliminar etiquetas de un recurso en cualquier momento. Para cambiar el valor de una clave de etiqueta, añada una etiqueta al recurso que defina la misma clave y el nuevo valor. El nuevo valor sustituye al valor anterior. Puede establecer un valor como una cadena vacía, pero no puede definir un valor como nulo.

Al eliminar un recurso, también se eliminarán las etiquetas que este tenga asociadas.

Uso de etiquetas con políticas de IAM

En sus políticas de IAM, puede utilizar etiquetas de recursos para controlar el acceso y los permisos de los usuarios. Por ejemplo, las políticas pueden permitir a los usuarios crear solo aquellos recursos que tienen una etiqueta específica. Las políticas también puede limitar la creación o modificación de recursos que tengan determinadas etiquetas por parte de los usuarios.

> **Note**
>
> Si utiliza etiquetas para permitir o denegar el acceso de los usuarios a los recursos, debe denegar a los usuarios la capacidad de agregar o eliminar esas etiquetas para los mismos recursos. De lo contrario, un usuario podría eludir sus restricciones y obtener acceso a un recurso modificando sus etiquetas.

Puede utilizar las siguientes claves y valores de contexto de condiciones en el`Condition` elemento, también denominado`Condition` bloque, de una declaración de política.

`greengrassv2:ResourceTag/`*`tag-key`*`: `*`tag-value`*

Permitir o denegar acciones en recursos con etiquetas específicas.

`aws:RequestTag/`*`tag-key`*`: `*`tag-value`*

Exija que se utilice, o no, una etiqueta específica al crear o modificar un recurso etiquetable.

`aws:TagKeys: [`*`tag-key`*`, ...]`

Exija que se utilice, o no, un conjunto específico de claves de etiqueta al crear o modificar un recurso etiquetable.

> **Note**
>
> Las claves y valores del contexto de condiciones de una política de IAM se aplican únicamente a las acciones que tienen un recurso etiquetable como parámetro obligatorio. Por ejemplo, puede configurar el acceso condicional basado en etiquetas para ListCoreDevices.

Para obtener más información, consulte Controlar el acceso aAWS los recursos mediante etiquetas de recursos y la referencia a la política JSON de IAM en la Guía del usuario de IAM.

Creación de AWS IoT Greengrass recursos con AWS CloudFormation

AWS IoT Greengrass está integrado con AWS CloudFormation, un servicio que le ayuda a modelar y configurar sus recursos de AWS para que pueda dedicar menos tiempo a crear y administrar sus recursos e infraestructura. Creas una plantilla que describa todos losAWSlos recursos que desee (como las versiones de los componentes y las implementaciones), yAWS CloudFormationaprovisiona y configura esos recursos para usted.

Cuando utiliza AWS CloudFormation, puede volver a utilizar la plantilla para configurar sus recursos de AWS IoT Greengrass de forma coherente y repetida. Solo tiene que describir los recursos una vez y luego aprovisionar los mismos recursos una y otra vez en varias Cuentas de AWS y regiones.

AWS IoT GreengrassPlantillas de AWS CloudFormation y

Para aprovisionar y configurar los recursos de AWS IoT Greengrass y sus servicios relacionados, debe entender las plantillas de AWS CloudFormation. Las plantillas son archivos de texto con formato de tipo JSON o YAML. Estas plantillas describen los recursos que desea aprovisionar en sus pilas de AWS CloudFormation. Si no está familiarizado con JSON o YAML, puede utilizar Designer de AWS CloudFormation para comenzar a utilizar las plantillas de AWS CloudFormation. Para obtener más información, consulte ¿Qué es Designer de AWS CloudFormation? en la Guía del usuario de AWS CloudFormation.

AWS IoT Greengrassadmite la creación de versiones e implementaciones de componentes enAWS CloudFormation. Para obtener más información, incluidos ejemplos de de de de plantillas JSON y YAML para versiones de componentes e implementaciones de componentes, consulte laAWS IoT GreengrassReferencia de tipo de recursoen elAWS CloudFormationGuía del usuario de.

ComponentVersion de de de de

La siguiente es la plantilla YAML para una versión de un componente simple. La receta JSON incluye saltos de línea por motivos de legibilidad.

```
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 }
```

Ejemplo de de de de

El siguiente es un archivo YAML que define una plantilla sencilla para una implementación.

```
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
```

Obtener más información sobre AWS CloudFormation

Para obtener más información acerca de AWS CloudFormation, consulte los siguientes recursos:

- AWS CloudFormation
- Guía del usuario de AWS CloudFormation
- Referencia de la API de AWS CloudFormation
- Guía del usuario de la interfaz de la línea de comandos de AWS CloudFormation

Software AWS IoT Greengrass Core de código abierto

El entorno de ejecución AWS IoT Greengrass Version 2 perimetral (núcleo) y otros componentes del software AWS IoT Greengrass Core son de código abierto. Esto significa que puede revisar el código para solucionar problemas de interacción con sus aplicaciones. También puede personalizar y ampliar el software AWS IoT Greengrass Core para que se adapte a sus necesidades específicas de software y hardware.

Para obtener información sobre los repositorios de código abierto del software AWS IoT Greengrass Core, consulte la organización aws-greengrass en. GitHub El uso del software de código abierto se rige por la licencia de código abierto del repositorio correspondiente. GitHub

El uso del software y los componentes AWS IoT Greengrass principales que no estén sujetos a una licencia de código abierto se rige por la licencia de software principal de AWS Greengrass.

Historial de documentos de la Guía para AWS IoT Greengrass V2 desarrolladores

En la siguiente tabla se describe la documentación de esta versión de AWS IoT Greengrass Version 2.

- Versión de API: 30 de noviembre de 2020

Cambio	Descripción	Fecha
Lanzamiento de la versión 2.3.8 de Shadow Manager	Ya está disponible la versión 2.3.8 de Shadow manager. Esta versión corrige un problema que provocaba que el administrador de sombras creara una situación de bloqueo durante la conexión del cliente MQTT.	5 de junio de 2024
Lanzamiento de la versión 2.12.6 de Greengrass CLI	El componente CLI de Greengrass v2.12.6 está disponible.	24 de mayo de 2024
AWS IoT Greengrass Actualización de software Core v2.12.6	Esta versión proporciona la versión 2.12.6 del component e núcleo de Greengrass y actualiza los componentes proporcionados. AWS	24 de mayo de 2024
AWS IoT Device Tester Publicada la versión 4.9.4 con la versión 2.5.4 de GGV2Q	Está disponible la versión 4.9.4 de IDT para V2. AWS IoT Greengrass Esta versión incluye la suite de calificac ión AWS IoT Greengrass V2 (GGV2Q) v2.5.4 y es compatible con las versiones	3 de mayo de 2024

	2.12.0, 2.11.0, 2.10.0 y 2.9.5 del núcleo de Greengrass.	
Publicada la versión 1.0.19 de Secure Tunneling	Se encuentra disponible la versión 1.0.19 de Secure Tunneling. Esta versión actualiza el AWS IoT Device Client subyacente invocado por el componente de la versión 1.8.0 a la versión 1.9.0. La versión 1.0.19 de Secure Tunneling aumenta el límite de túneles simultáneos a 20 túneles a nivel de componente. Esta nueva versión también aumenta el tiempo de espera de AWS IoT Greengrass Core IPC de 3 a 10 segundos.	1 de mayo de 2024
Lanzamiento del conector Edge para el componente Kinesis Video Streams v1.0.5	Está disponible la versión 1.0.5 del componente Edge Connector para Kinesis Video Streams. Esta versión incluye mejoras y correcciones de errores generales.	29 de abril de 2024
Lanzamiento de la versión 2.12.5 de Greengrass CLI	El componente CLI de Greengrass, versión 2.12.5, está disponible.	25 de abril de 2024

Lanzamiento del componente de autenticación de dispositivos cliente, versión 2.5.0	El componente de autenticación de dispositivos cliente, versión 2.5.0, está disponible. Esta versión añade compatibilidad con variables de política para los nombres de las cosas. Esta versión también permite utilizar caracteres comodín para los recursos de políticas.	25 de abril de 2024
AWS IoT Greengrass Actualización de software Core v2.12.5	Esta versión proporciona la versión 2.12.5 del componente núcleo de Greengrass y AWS actualiza los componentes proporcionados.	25 de abril de 2024
AWS IoT Device Tester Publicada la versión 4.9.3 con la versión 2.5.3 de GGV2Q	Ya está disponible la versión 4.9.3 de IDT para V2. AWS IoT Greengrass Esta versión incluye la suite de calificación AWS IoT Greengrass V2 (GGV2Q) v2.5.3 y es compatible con las versiones 2.12.0, 2.11.0, 2.10.0 y 2.9.5 del núcleo de Greengrass.	5 de abril de 2024
Lanzamiento de la versión 2.12.4 de Greengrass CLI	El componente CLI de Greengrass v2.12.4 está disponible.	2 de abril de 2024
AWS IoT Greengrass Actualización de software Core v2.12.4	Esta versión proporciona la versión 2.12.4 del componente núcleo de Greengrass y actualiza los componentes proporcionados. AWS	2 de abril de 2024

Lanzamiento de la versión 2.3.7 de Shadow Manager	Ya está disponible la versión 2.3.7 de Shadow manager. Esta versión corrige un problema por el que el administrador de sombras registra periódicamente un `NullPointerException` error durante una sincroniz ación con el administrador de sombras.	27 de marzo de 2024
Publicada la versión 2.3.6 del bróker MQTT 3.1.1 de Moquette	Ya está disponible la versión 2.3.6 del componente broker MQTT 3.1.1 de Moquette. Esta versión incluye mejoras y correcciones de errores generales.	27 de marzo de 2024
Lanzamiento de la consola de depuración local, versión 2.4.2	Está disponible el component e de la consola de depuració n local, versión 2.4.2. Esta versión incluye mejoras y correcciones de errores generales.	27 de marzo de 2024
Lanzada la versión 2.3.3 de Lambda Manager	Está disponible el component e Lambda manager v2.3.3. Esta versión incluye mejoras y correcciones de errores generales.	27 de marzo de 2024
Lanzamiento de la versión 2.1.9 del detector de IP	Está disponible el component e detector IP v2.1.9. Esta versión ajusta el paso de IP adquirida para enviar solo los registros a nivel del registro de depuración.	27 de marzo de 2024

AWS IoT Lanzamiento del complemento de aprovisionamiento de flotas, versión 1.2.1	AWS IoT Está disponible la versión 1.2.1 del complemento de aprovisionamiento de flotas. Esta versión corrige un problema por el que el complemento de aprovisionamiento de flotas estaba fuera de línea durante el inicio de Greengrass Nucleus. El complemento de aprovisionamiento de flotas ahora reintenta indefinidamente las llamadas a MQTT connect.	27 de marzo de 2024
AWS IoT Greengrass Actualización del software Core v2.12.3	Esta versión proporciona la versión 2.12.3 del componente núcleo de Greengrass y actualiza los componentes proporcionados. AWS	27 de marzo de 2024
Lanzamiento de la versión 2.12.3 de Greengrass CLI	El componente CLI de Greengrass v2.12.3 está disponible.	25 de marzo de 2024
AWS IoT Device Tester Publicada la versión 4.9.2 con la versión 2.5.2 de GGV2Q	Está disponible la versión 4.9.2 de IDT para V2. AWS IoT Greengrass Esta versión incluye la suite de calificación AWS IoT Greengrass V2 (GGV2Q) v2.5.2 y es compatible con las versiones 2.12.0, 2.11.0, 2.10.0 y 2.9.5 del núcleo de Greengrass.	18 de marzo de 2024
Lanzada la versión 1.2.0 del agente Lookout for Vision edge	Ya está disponible la versión 1.2.0 del agente Lookout for Vision edge.	11 de marzo de 2024

AWS IoT Greengrass Actualización de software Core v2.12.2	Esta versión proporciona la versión 2.12.2 del component e núcleo de Greengrass y actualiza los componentes proporcionados. AWS	15 de febrero de 2024
Lanzamiento de la versión 2.3.6 de Shadow Manager	Ya está disponible la versión 2.3.6 de Shadow manager. Esta versión corrige un problema por el que las propiedades ocultas que se eliminan mediante Nube de AWS las actualizaciones mientras el dispositivo está fuera de línea siguen existiend o en la sombra local tras recuperar la conectividad.	14 de febrero de 2024
Lanzada la versión 2.0.13 del lanzador Lambda	Está disponible la versión 2.0.13 del componente Lanzador Lambda. Esta versión incluye mejoras y correcciones de errores generales.	14 de febrero de 2024
Lanzamiento de Disk spooler v1.0.3	Está disponible el component e Disk Spooler v1.0.3. Esta versión mejora el rendimiento al reutilizar las conexiones de bases de datos.	14 de febrero de 2024
Lanzada la versión 1.1.9 del agente Lookout for Vision edge	Ya está disponible la versión 1.1.9 del agente Lookout for Vision edge.	17 de enero de 2024

Kit de desarrollo Greengrass CLI v1.6.2	Está disponible la versión 1.6.2 del kit de desarrollo CLI de Greengrass. Esta versión corrige un problema por el que el archivo gradlew.bat de Windows no funcionab a debido a la ruta relativa. Esta versión también contiene mejoras adicionales.	16 de enero de 2024
Nuevos eventos CloudTrail de datos	Ahora puede registrar eventos de AWS CloudTrail datos para obtener información sobre las operaciones de recursos, como la obtención de un componente o la configura ción de una implementación. Utilice estos eventos para obtener información sobre el funcionamiento de sus dispositivos Greengrass.	20 de diciembre de 2023
Lanzada la versión 1.1.8 del agente Lookout for Vision edge	Ya está disponible la versión 1.1.8 del agente Lookout for Vision edge.	12 de diciembre de 2023
Publicada la versión 2.1.12 de Stream Manager	Ya está disponible la versión 2.1.12 de Stream Manager. Esta versión cambia el orden que Greengrass utiliza para seleccionar un conjunto de credenciales para las llamadas de AWS servicio.	8 de diciembre de 2023

Publicada la versión 2.3.1 de MQTT bridge	Ya está disponible la versión 2.3.1 del puente MQTT. Esta versión corrige un problema poco frecuente que provocaba que el cliente MQTT local entrara en un bucle de desconexión.	8 de diciembre de 2023
Publicada la versión 1.0.2 de Disk spooler	Está disponible el componente Disk Spooler v1.0.2. Esta versión corrige un problema por el que el campo de formato de mensaje MQTT no se conserva en algunos casos.	8 de diciembre de 2023
Se ha publicado la versión 2.4.5 del componente de autenticación de dispositivos cliente	Está disponible el componente de autenticación de dispositivos cliente, versión 2.4.5. Esta versión añade la compatibilidad con los caracteres comodín al final de los nombres de las cosas en una regla de selección y corrige un problema por el que, en algunos casos, los certificados no se actualizaban con nueva información de conectividad.	8 de diciembre de 2023
AWS IoT Greengrass Actualización de software Core v2.12.1	Esta versión proporciona la versión 2.12.1 del componente núcleo de Greengrass y actualiza los componentes proporcionados. AWS	8 de diciembre de 2023

Kit de desarrollo Greengrass CLI v1.6.1	Está disponible la versión 1.6.1 del kit de desarrollo CLI de Greengrass. Esta versión contiene correcciones de errores y mejoras.	6 de diciembre de 2023
Validación de recetas	Se agregó una función de validación de recetas que validará la receta de un componente al crear una versión del componente.	16 de noviembre de 2023
Componentes compatibles con Publisher	AWS IoT Greengrass ahora ofrece componentes compatibles con Publisher. Estos componentes son desarrollados, ofrecidos y mantenidos por proveedores externos.	16 de noviembre de 2023
Lanzamiento de Greengrass Testing Framework v1.2.0	Ya está disponible la versión 1.2.0 de Greengrass Testing Framework.	15 de noviembre de 2023

Kit de desarrollo Greengrass CLI v1.6.0	Está disponible la versión 1.6.0 del kit de desarrollo CLI de Greengrass. Esta versión añade una comprobación de validación de recetas con respecto al esquema de recetas de Greengrass durante los comandos `component build` y`component publish`. Esta actualización ayuda a los desarrolladores a identificar problemas procesables en sus recetas de component es en una fase temprana del proceso de creación de componentes. Esta versión también añade un conjunto de pruebas de confianza a la plantilla que se puede desplegar con un `test-e2e init` comando. Este conjunto de pruebas de confianza incluye ocho pruebas genéricas que se pueden utilizar y ampliar para adaptarse a las necesidades básicas de las pruebas de componentes.	15 de noviembre de 2023
AWS IoT Device Tester La versión 4.9.1 es compatible con la versión 2.12.0 del núcleo de Greengrass	La versión 4.9.1 de IDT para AWS IoT Greengrass V2 ahora es compatible con la versión 2.12.0 del núcleo de Greengrass.	7 de noviembre de 2023

AWS IoT Greengrass Actualización de software Core v2.12.0	Esta versión proporciona la versión 2.12.0 del component e núcleo de Greengrass y actualiza los componentes proporcionados. AWS	7 de noviembre de 2023
Opere un dispositivo central de Greengrass en VPC	Está disponible el funcionam iento de un dispositivo central Greengrass en VPC. Esta función le permite realizar despliegues en una VPC sin acceso público a Internet.	3 de noviembre de 2023
Lanzamiento de la versión 2.12.0 de Greengrass CLI	El componente CLI de Greengrass v2.12.0 está disponible.	30 de octubre de 2023
Publicada la versión 2.1.10 de Stream Manager	Ya está disponible la versión 2.1.10 de Stream Manager. Esta versión corrige un problema por el que la configuración del proxy HTTPS no confiaba en la cadena de certificados CA de Greengrass.	26 de octubre de 2023
Lanzada la versión 2.0.12 del lanzador Lambda	Está disponible la versión 2.0.12 del componente Lanzador Lambda. Esta versión corrige un problema por el que el lanzador Lambda podía generar un error si el proceso anterior no se detenía correctamente.	26 de octubre de 2023

Kit de desarrollo Greengrass CLI v1.5.0	Está disponible la versión 1.5.0 del kit de desarrollo CLI de Greengrass. Esta versión actualiza los patrones reconocidos por la opción de `excludes` compilación cuando `build_system` está. `zip` Esta versión ahora reconocerá los patrones globales que coincidan con los nombres de las rutas en función de sus caractere s comodín. Esto permite especificar de forma personali zada los directorios de los que se debe excluir.	26 de octubre de 2023
Lanzada la versión 1.1.7 del agente Lookout for Vision edge	Ya está disponible la versión 1.1.7 del agente Lookout for Vision edge.	24 de octubre de 2023
Lanzada la versión 2.3.4 de Shadow Manager	Ya está disponible la versión 2.3.4 de Shadow Manager. Esta versión añade compatibi lidad con documentos de estado oculto nulos y vacíos.	18 de octubre de 2023
Publicada la versión 2.3.6 de Log Manager	Está disponible el componente Log Manager v2.3.6.	18 de octubre de 2023
Lanzamiento de la consola de depuración local, versión 2.4.0	Está disponible el component e de la consola de depuración local, versión 2.4.0.	18 de octubre de 2023
Lanzada la versión 2.3.1 de Lambda Manager	Está disponible el componente Lambda Manager v2.3.1.	18 de octubre de 2023

Lanzamiento de la versión 2.11.3 de Greengrass CLI	El componente CLI de Greengrass, versión 2.11.3, está disponible.	18 de octubre de 2023
AWS IoT Greengrass Actualización de software Core v2.11.3	Esta versión proporciona la versión 2.11.3 del component e núcleo de Greengrass y actualiza los componentes proporcionados. AWS	18 de octubre de 2023
Publicada la versión 1.0.17 de Secure Tunneling	Se encuentra disponible la versión 1.0.17 de Secure Tunneling.	4 de octubre de 2023
Kit de desarrollo Greengrass CLI v1.4.0	Está disponible la versión 1.4.0 del kit de desarrollo CLI de Greengrass. Esta versión agrega un nuevo `config` comando que inicia un mensaje interactivo para modificar los campos de un archivo de configuración del GDK existente. Esta versión también modifica los `gdk component publish` comandos `gdk component build` y para comprobar que el tamaño de la receta cumple con los requisitos de Greengrass (<=16000 bytes) antes de continuar.	2 de octubre de 2023
Publicada la versión 2.3.5 del bróker MQTT 3.1.1 de Moquette	Está disponible el component e broker Moquette MQTT 3.1.1, versión 2.3.5. Esta versión actualiza Moquette a la versión 0.17.	28 de septiembre de 2023

Publicada la versión 2.3.0 de MQTT bridge	Ya está disponible la versión 2.3.0 del puente MQTT. Esta versión añade compatibilidad con MQTT 5 para establecer conexiones entre AWS IoT Core fuentes MQTT locales.	28 de septiembre de 2023
Lanzada la versión 1.1.6 del agente Lookout for Vision edge	Ya está disponible la versión 1.1.6 del agente Lookout for Vision edge.	27 de septiembre de 2023
Lanzamiento de Lambda Manager v2.3.0	Está disponible el componente Lambda Manager v2.3.0.	15 de septiembre de 2023
Lanzada la versión 2.0.11 del lanzador Lambda	Está disponible la versión 2.0.11 del componente Lanzador Lambda. Esta versión es compatible con Lambda Manager 2.3.0.	15 de septiembre de 2023
Publicada la versión 2.3.4 del broker MQTT 3.1.1 de Moquette	Ya está disponible la versión 2.3.4 del componente broker MQTT 3.1.1 de Moquette.	1 de septiembre de 2023
Marco de pruebas de Greengrass	El GTF es un conjunto de componentes básicos para respaldar end-to-end la automatización. Permite a los clientes AWS IoT Greengrass Version 2 internos utilizar el mismo marco de pruebas que utiliza el equipo de servicio para calificar los cambios de software, aceptarlos automáticamente y garantizar la calidad.	11 de agosto de 2023

AWS IoT Greengrass Actualización de software Core v2.11.2	Esta versión proporciona la versión 2.11.2 del componente núcleo de Greengrass y actualiza los componentes proporcionados. AWS	9 de agosto de 2023
Kit de desarrollo Greengrass CLI v1.3.0	Está disponible la versión 1.3.0 del kit de desarrollo CLI de Greengrass. Esta versión añade un nuevo `test-e2e` comando para permitir las end-to-end pruebas de componentes mediante Open Test Framework.	21 de julio de 2023
AWS IoT Greengrass Actualización de software Core v2.11.1	Esta versión proporciona la versión 2.11.1 del componente núcleo de Greengrass y actualiza los componentes proporcionados. AWS	21 de julio de 2023
Publicada la versión 1.0.0 de Disk spooler	Está disponible el componente Disk Spooler v1.0.0.	28 de junio de 2023
AWS IoT Greengrass Actualización de software Core v2.11.0	Esta versión proporciona la versión 2.11.0 del componente núcleo de Greengrass y actualiza los componentes proporcionados. AWS	28 de junio de 2023
AWS IoT Greengrass Actualización del software Core v2.10.3	Esta versión proporciona la versión 2.10.3 del componente núcleo de Greengrass y actualiza los componentes proporcionados. AWS	21 de junio de 2023

AWS IoT Greengrass Actualización de software Core v2.10.2	Esta versión proporciona la versión 2.10.2 del componente núcleo de Greengrass y AWS actualiza los componentes proporcionados.	5 de junio de 2023
AWS IoT Greengrass Actualización de software Core v2.10.1	Esta versión proporciona la versión 2.10.1 del componente núcleo de Greengrass y AWS actualiza los componentes proporcionados.	11 de mayo de 2023
AWS IoT Greengrass Actualización de software Core v2.10.0	Esta versión proporciona la versión 2.10.0 del component e núcleo de Greengrass y actualiza los componentes proporcionados. AWS	9 de mayo de 2023
SageMaker Edge Manager ha dejado de fabricarse	El componente Amazon SageMaker Edge Manager dejará de funcionar el 26 de abril de 2024.	28 de abril de 2023
AWS IoT Greengrass Actualización de software Core v2.9.6	Esta versión proporciona la versión 2.9.6 del componente núcleo de Greengrass y AWS actualiza los componentes proporcionados.	20 de abril de 2023
Publicada la versión 2.3.2 de Log Manager	Está disponible el componente Log Manager v2.3.2.	19 de abril de 2023

Publicada la versión 2.1.4 de Stream Manager	Ya está disponible la versión 2.1.4 de Stream Manager. Esta versión corrige un problema por el que las entradas del mismo activo inmobiliario con la misma marca de tiempo en un mismo lote se devuelven `ConflictingOperationException` desde la SiteWise API, lo que provoca que Stream Manager vuelva a intentarlo continuamente. Esta versión también actualiza el tiempo de espera de conexión predeterminado de 3 segundos a 1 minuto.	13 de abril de 2023
Kit de desarrollo Greengrass CLI v1.2.3	Está disponible la versión 1.2.3 del kit de desarrollo CLI de Greengrass. Esta versión contiene correcciones de errores.	13 de abril de 2023
Lanzamiento del componente de autenticación de dispositivos cliente, versión 2.4.0	El componente de autenticación de dispositivos cliente, versión 2.4.0, está disponible. Esta versión añade compatibilidad con la autenticación de dispositivos cliente para emitir métricas operativas que se pueden mostrar en el panel de dispositivos cliente de Greengrass.	10 de abril de 2023

Kit de desarrollo Greengrass CLI v1.2.2	Está disponible la versión 1.2.2 del kit de desarrollo CLI de Greengrass. Esta versión contiene mejoras y correcciones de errores.	7 de abril de 2023
AWS IoT Greengrass Actualización de software Core v2.9.5	Esta versión proporciona la versión 2.9.5 del componente núcleo de Greengrass y AWS actualiza los componentes proporcionados.	30 de marzo de 2023
Publicada la versión 2.1.3 de Stream Manager	Ya está disponible la versión 2.1.3 de Stream Manager. Esta versión corrige un problema de inicio en el sistema operativo Windows cuando se ejecuta como usuario del SISTEMA.	7 de marzo de 2023
Lanzamiento del adaptador de protocolo Modbus-RTU v2.1.5	Está disponible el componente adaptador de protocolo Modbus-RTU, versión 2.1.5. Esta versión corrige un problema con la operación. `ReadDiscreteInput`	7 de marzo de 2023

Se ha publicado la versión 2.3.2 del componente de autenticación de dispositivos cliente	El componente de autenticación de dispositivos cliente, versión 2.3.2, está disponible. Esta versión añade compatibilidad con el almacenamiento en caché de la información del nombre de host, de modo que el componente genere correctamente los sujetos de los certificados cuando se reinicie sin conexión a Internet.	7 de marzo de 2023
AWS IoT Device Tester La versión 4.7.0 es compatible con la versión 2.9.4 del núcleo de Greengrass	La versión 4.7.0 de IDT para AWS IoT Greengrass V2 ahora es compatible con la versión 2.9.4 del núcleo de Greengrass.	2 de marzo de 2023
Lanzamiento de la interfaz de línea de comandos Greengrass v1.2.0	Está disponible la interfaz de línea de comandos Greengrass v1.2.0.	28 de febrero de 2023
AWS IoT Greengrass Actualización de software Core v2.9.4	Esta versión proporciona la versión 2.9.4 del componente núcleo de Greengrass y AWS actualiza los componentes proporcionados.	24 de febrero de 2023

Publicada la versión 2.3.1 de Shadow Manager	Ya está disponible la versión 2.3.1 de Shadow Manager. Esta versión corrige una condición que podía impedir la sincronización de las actualizaciones de Cloud Shadow. Esta versión también corrige un problema por el que los cambios en la configuración de sincronización de sombras con nombre se aplicaban solo a una sombra con nombre.	21 de febrero de 2023
AWS IoT Device Tester La versión 4.7.0 es compatible con la versión 2.9.3 del núcleo de Greengrass	La versión 4.7.0 de IDT para AWS IoT Greengrass V2 ahora es compatible con la versión 2.9.3 del núcleo de Greengrass.	9 de febrero de 2023
Se actualizaron las mejores prácticas de IAM	Se ha actualizado la guía para implementar las prácticas recomendadas de IAM. Para obtener más información, consulte prácticas recomendadas de seguridad en IAM.	3 de febrero de 2023
AWS IoT Greengrass Actualización del software Core v2.9.3	Esta versión incluye la versión 2.9.3 del componente núcleo de Greengrass y AWS actualiza los componentes proporcionados.	1 de febrero de 2023
Publicada la versión 2.3.1 de Log Manager	Está disponible la versión 2.3.1 del administrador de registros.	27 de enero de 2023

AWS IoT Device Tester La versión 4.7.0 es compatible con la versión 2.9.2 del núcleo de Greengrass	La versión 4.7.0 de IDT para AWS IoT Greengrass V2 ahora es compatible con la versión 2.9.2 del núcleo de Greengrass.	3 de enero de 2023
Lanzada la versión 2.3.0 de Shadow Manager	Ya está disponible la versión 2.3.0 de Shadow Manager. Esta versión corrige un problema que podía impedir que las sombras se sincroniz aran cuando un dispositivo guarda la clave privada del dispositivo Greengrass en un módulo de seguridad de hardware.	29 de diciembre de 2022
AWS IoT Lanzamiento del complemento de aprovisio namiento de flotas, versión 1.2.0	AWS IoT Está disponible el complemento de aprovisio namiento de flotas v1.2.0. Esta versión añade compatibilidad con el aprovisionamiento de dispositivos mediante una solicitud de firma de certificado con una ruta de clave privada configurable.	22 de diciembre de 2022
AWS IoT Greengrass Actualización de software Core v2.9.2	Esta versión proporciona la versión 2.9.2 del componente núcleo de Greengrass y AWS actualiza los componentes proporcionados.	22 de diciembre de 2022

AWS IoT Device Tester Publicada la versión 4.7.0 con la versión 2.5.0 de GGV2Q	Está disponible la versión 4.7.0 de IDT para V2. AWS IoT Greengrass Esta versión incluye la suite de calificac ión AWS IoT Greengrass V2 (GGV2Q) v2.5.0 y es compatible con las versiones 2.9.1, 2.9.0, 2.8.1, 2.8.0, 2.7.0 y 2.6.0 del núcleo de Greengrass.	13 de diciembre de 2022
Lanzamiento de la versión 2.2.4 de Shadow Manager	Soluciona un problema por el que la validación del tamaño de la sombra no era coherente con el de la nube al actualiza r el documento de sombra local. Esto también soluciona un problema por el que el administrador oculto deja de escuchar las actualiza ciones de la configuración si una implementación realiza una `RESET` en los nodos de configuración.	8 de diciembre de 2022
Lanzamiento de Lookout for Vision Edge Agent 1.1.1	Ya está disponible la versión 1.1.1 del componente Lookout for Vision Edge Agent.	5 de diciembre de 2022
Publicada la versión 2.3.0 de Log Manager	Está disponible el componente Log Manager v2.3.0.	18 de noviembre de 2022
AWS IoT Device Tester La versión 4.5.11 es compatible con la versión 2.9.1 del núcleo de Greengrass	La versión 4.5.11 de IDT para AWS IoT Greengrass V2 ahora es compatible con la versión 2.9.1 del núcleo de Greengrass.	18 de noviembre de 2022

AWS IoT Greengrass Actualización de software Core v2.9.1	Esta versión proporciona la versión 2.9.1 del componente núcleo de Greengrass y AWS actualiza los componentes proporcionados.	18 de noviembre de 2022
AWS IoT Device Tester La versión 4.5.11 es compatible con la versión 2.9.0 del núcleo de Greengrass	La versión 4.5.11 de IDT para AWS IoT Greengrass V2 ahora es compatible con la versión 2.9.0 del núcleo de Greengrass.	17 de noviembre de 2022
Publicada la versión 2.1.2 de Stream Manager	Ya está disponible la versión 2.1.2 de Stream Manager. Esta versión corrige un problema en los sistemas operativos Windows que utilizan un idioma distinto del inglés.	15 de noviembre de 2022
AWS IoT Greengrass Actualización de software Core v2.9.0	Esta versión proporciona la versión 2.9.0 del componente núcleo de Greengrass y AWS actualiza los componentes proporcionados.	15 de noviembre de 2022
AWS IoT Device Tester La versión 4.5.11 es compatible con la versión 2.8.1 del núcleo de Greengrass	La versión 4.5.11 de IDT para AWS IoT Greengrass V2 ahora es compatible con la versión 2.8.1 del núcleo de Greengrass.	19 de octubre de 2022

AWS IoT Device Tester Publicada la versión 4.5.11 con la versión 2.4.1 de GGV2Q	Ya está disponible la versión 4.5.11 de IDT para V2. AWS IoT Greengrass Esta versión incluye la suite de calificación AWS IoT Greengrass V2 (GGV2Q) v2.4.1 y es compatible con las versiones 2.8.0, 2.7.0 y 2.6.0 del núcleo de Greengrass.	13 de octubre de 2022
AWS IoT Greengrass Actualización de software Core v2.8.1	Esta versión proporciona la versión 2.8.1 del componente núcleo de Greengrass y AWS actualiza los componentes proporcionados.	13 de octubre de 2022
AWS IoT Greengrass Actualización de software Core v2.8.0	Esta versión proporciona la versión 2.8.0 del componente núcleo de Greengrass y AWS actualiza los componentes proporcionados.	7 de octubre de 2022
Se agregó soporte AWS CloudFormation para despliegues	AWS CloudFormation ahora admite AWS IoT Greengrass las implementaciones como recurso.	6 de octubre de 2022

SageMaker Lanzamiento de Edge Manager v1.3.0	Está disponible el componente Amazon SageMaker Edge Manager v1.3.0. Esta versión añade compatibilidad con este componente para establecer el tamaño del disco para la caché del modelo TensorRT y mejora la simultaneidad de predicciones para aprovechar mejor los motores aceleradores de dispositivos, como las GPU.	1 de septiembre de 2022
Usa el cliente de comunicación entre procesos (IPC) V2	Se agregó información sobre el cliente IPC V2, lo que reduce la cantidad de código que hay que escribir para utilizar las operaciones de IPC y ayuda a evitar los errores habituales que pueden producirse con el cliente IPC V1.	12 de agosto de 2022
AWS IoT Device Tester Publicada la versión 4.5.8 y la versión 2.4.0 de GGV2Q	Está disponible la versión 4.5.8 de IDT para V2. AWS IoT Greengrass Esta versión incluye la suite de calificación AWS IoT Greengrass V2 (GGV2Q) v2.4.0 y es compatible con las versiones 2.7.0, 2.6.0 y 2.5.6 del núcleo de Greengrass.	12 de agosto de 2022

SageMaker Lanzamiento de Edge Manager v1.2.0	Está disponible el componente Amazon SageMaker Edge Manager v1.2.0. Esta versión añade soporte para que este componente recupere automáticamente los modelos SageMaker compilados en NEO que cargue en Amazon S3, de forma que pueda implementar nuevos modelos sin necesidad de crear una AWS IoT Greengrass implementación.	3 de agosto de 2022
AWS IoT Device Tester La versión 4.5.3 es compatible con la versión 2.7.0 del núcleo de Greengrass	La versión 4.5.3 de IDT para AWS IoT Greengrass V2 ahora es compatible con la versión 2.7.0 del núcleo de Greengrass.	1 de agosto de 2022
Publicada la versión 2.1.0 de Stream Manager	Ya está disponible la versión 2.1.0 de Stream Manager. Esta versión incluye soporte para enviar métricas de telemetría a Amazon. EventBridge	28 de julio de 2022
AWS IoT Greengrass Actualización de software Core v2.7.0	Esta versión proporciona la versión 2.7.0 del componente núcleo de Greengrass y AWS actualiza los componentes proporcionados. Incluye soporte para enviar métricas de telemetría a Amazon. EventBridge	28 de julio de 2022

Publicada la versión 2.2.0 de IoT SiteWise Publisher	Está disponible el componente IoT SiteWise Publisher v2.2.0. Esta versión actualiza el componente para comprimir los datos antes de enviarlos al AWS IoT SiteWise servicio, lo que reduce el uso de ancho de banda hasta en un 75 por ciento.	19 de julio de 2022
Tutorial: Desarrolle un componente que interactúe con las sombras de los dispositivos cliente	Se agregó un nuevo módulo al Tutorial: Interactúe con dispositivos IoT locales a través de MQTT que puede seguir para aprender a desarrollar un componente que interactúe con las sombras de los dispositivos del cliente.	18 de julio de 2022
Elija un bróker MQTT local	Se ha añadido información sobre cómo elegir un intermediario MQTT local en el que los dispositivos cliente se conecten a un dispositivo principal.	18 de julio de 2022
AWS IoT Device Tester La versión 4.5.3 es compatible con la versión 2.6.0 del núcleo de Greengrass	La versión 4.5.3 de IDT para AWS IoT Greengrass V2 ahora es compatible con la versión 2.6.0 del núcleo de Greengrass.	29 de junio de 2022

AWS IoT Greengrass Actualización de software Core v2.6.0	Esta versión proporciona la versión 2.6.0 del component e núcleo de Greengrass y AWS actualiza los component es proporcionados. Incluye soporte para sombras de dispositivos cliente y un intermediario MQTT 5 local para dispositivos cliente. También admite caracteres comodín en los temas locales de publicación o suscripci ón, variables de receta en las configuraciones de los componentes y caractere s comodín en las políticas de autorización de IPC. Estas funciones le permiten desarrollar y configurar más fácilmente los component es que se implementan en las flotas de dispositivos principales. Esta versión también incluye soporte para que los component es utilicen operaciones de IPC que gestionan las implementaciones locales y los componentes de un dispositivo principal.	27 de junio de 2022

Actualizaciones de componentes de dispositivos cliente	Están disponibles la versión 2.1.0 de autenticación de dispositivos cliente, la versión 2.1.0 del agente MQTT (Moquette), la versión 2.1.1 del puente MQTT y la versión 2.1.2 del detector de IP. Esta versión mejora la rotación de certificados, mejora el rendimiento del broker MQTT y corrige los problemas relacionados con la forma en que estos componentes gestionan las actualizaciones de restablecimiento de la configuración.	14 de junio de 2022
AWS IoT Device Tester La versión 4.5.3 es compatible con la versión 2.5.6 del núcleo de Greengrass	La versión 4.5.3 de IDT para AWS IoT Greengrass V2 ahora es compatible con la versión 2.5.6 del núcleo de Greengrass.	1 de junio de 2022
AWS IoT Greengrass Actualización de software Core v2.5.6	Esta versión proporciona la versión 2.5.6 del componente núcleo de Greengrass y AWS actualiza los componentes proporcionados. Incluye soporte para módulos de seguridad de hardware con claves ECC. También incluye otras correcciones de errores y mejoras.	31 de mayo de 2022

AWS IoT Lanzamiento del complemento de aprovisionamiento de flotas v1.1.0	AWS IoT Está disponible la versión 1.1.0 del complemento de aprovisionamiento de flotas. Esta versión añade compatibilidad con otros formatos de rutas de archivos al configurar el complemento en dispositivos Windows.	12 de mayo de 2022
Lanzamiento de nuevos tiempos de ejecución de Lambda	Se agregó soporte para los nuevos tiempos de ejecución de Lambda: Python 3.9, Java 11 y NodejS 14.	10 de mayo de 2022
Desarrolle un componente de Greengrass que aplace las actualizaciones de los componentes	Se ha añadido un tutorial que puede seguir para aprender a desarrollar un componente de Greengrass que aplaza las actualizaciones de los componentes de las implementaciones. Es posible que desee retrasar una actualización cuando el nivel de batería de un dispositivo sea bajo o cuando ejecute un proceso que no pueda interrumpirse, por ejemplo.	4 de mayo de 2022

CloudWatch Publicadas las versiones 3.1.0 y AWS IoT Device Defender 3.1.0 de Metrics	CloudWatch están disponibl es el componente de métricas v3.1.0 y el componente v3.1.0. AWS IoT Device Defender Estas versiones añaden compatibilidad con las configuraciones de proxy de red HTTPS. Para obtener más información, consulte Conectarse en el puerto 443 o mediante un proxy de red y Habilitar el dispositivo principal para que confíe en un proxy HTTPS.	27 de abril de 2022
Migre desde AWS IoT Greengrass Version 1	Se agregó una guía que puede seguir para migrar de AWS IoT Greengrass V1 a AWS IoT Greengrass V2.	26 de abril de 2022

AWS IoT Device Tester v4.5.3 con GGV2Q v2.3.1 actualizada e IDT v4.5.1 con GGV2Q v2.3.0 agregadas a las versiones compatibles	La versión 4.5.3 de IDT para AWS IoT Greengrass V2 con paquete de calificación AWS IoT Greengrass V2 (GGV2Q) v2.3.1 se ha actualizado para incluir compatibilidad con las versiones 2.5.5, 2.5.4 y 2.5.3 del núcleo de Greengrass. Esta actualización también incluye IDT 4.5.1 con la versión 2.3.0 del paquete de calificación V2 (GGV2Q) como versión compatible. AWS IoT Greengrass La versión 2.3.0 de IDT 4.5.1 con paquete de calificación AWS IoT Greengrass V2 (GGV2Q) es compatible con Greengrass nucleus versión 2.5.3.	25 de abril de 2022
Lanzamiento del adaptador de protocolo Modbus-RTU v2.1.0	Está disponible el component e adaptador de protocolo Modbus-RTU, versión 2.1.0. Esta versión añade nuevos parámetros que puede especificar para configurar la comunicación en serie con los dispositivos Modbus RTU.	20 de abril de 2022

CloudWatch Lanzamiento de metrics v2.1.0, Firehose v2.1.0 y Amazon SNS v2.1.0	CloudWatch Están disponibles el componente de métricas v2.1.0, el componente Firehose v2.1.0 y el componente Amazon SNS v2.1.0. Estas versiones añaden compatibilidad con las configuraciones de proxy de red HTTPS. Para obtener más información, consulte Conectarse en el puerto 443 o mediante un proxy de red y Habilitar el dispositivo principal para que confíe en un proxy HTTPS.	19 de abril de 2022
AWS IoT Device Tester Publicada la versión 4.5.3 con la versión 2.3.1 de GGV2Q	Está disponible la versión 4.5.3 de IDT para V2. AWS IoT Greengrass Esta versión incluye la suite de calificación AWS IoT Greengrass V2 (GGV2Q) v2.3.1 y es compatible con la versión 2.5.5 del núcleo de Greengrass.	15 de abril de 2022

AWS IoT Greengrass Actualización de software Core v2.5.5	Esta versión proporciona la versión 2.5.5 del component e núcleo de Greengrass y AWS actualiza los component es proporcionados. Añade compatibilidad con dispositi vos Windows que utilizan un idioma de visualización distinto del inglés. También soluciona un problema por el que el dispositivo principal no informaba de su estado al servicio en la AWS IoT Greengrass nube después del aprovisionamiento en determinadas situaciones.	6 de abril de 2022
AWS IoT Greengrass Actualización del software Core v2.5.4	Esta versión proporciona la versión 2.5.4 del component e núcleo de Greengrass y AWS actualiza los component es proporcionados. Incluye correcciones de errores y mejoras.	23 de marzo de 2022
Descarga AWS IoT Device Tester mediante programación	Se agregó información sobre cómo descargar IDT mediante programación. AWS IoT Greengrass V2	15 de marzo de 2022

Kit de desarrollo Greengrass CLI v1.1.0	Está disponible la versión 1.1.0 del kit de desarrollo CLI de Greengrass. Esta versión agrega nuevos argumentos a los comandos `component init` y `component publish`. Esta versión también actualiza el `component publish` comando para compilar el componente si no está compilado.	24 de febrero de 2022
Lanzamiento de la versión 2.1.0 de Shadow Manager	Está disponible el componente Shadow Manager v2.1.0. En esta versión se añade la opción de configurar el intervalo con el que el componente sincroniza las sombras. AWS IoT Core Por ejemplo, puede especificar un intervalo más largo para reducir el uso y los cargos del ancho de banda.	3 de febrero de 2022
Dockerfile e imágenes de Docker para AWS IoT Greengrass la versión 2.5.3 del software Core	Ya están disponibles el Dockerfile y la imagen de Docker para el software Core v2.5.3. AWS IoT Greengrass	12 de enero de 2022

AWS IoT Device Tester Publicada la versión 4.5.1 con la versión 2.3.0 de GGV2Q	Está disponible la versión 4.5.1 de IDT para V2. AWS IoT Greengrass Esta versión incluye la versión 2.3.0 del paquete de calificación AWS IoT Greengrass V2 (GGV2Q) y permite validar y calificar los dispositivos basados en Linux que utilizan un módulo de seguridad de hardware (HSM) para almacenar la clave privada y el certificado utilizados por el software Core. AWS IoT Greengrass	11 de enero de 2022
AWS IoT Greengrass Actualización de software Core v2.5.3	Esta versión proporciona la versión 2.5.3 del component e núcleo de Greengrass y AWS actualiza los component es proporcionados. Incluye soporte para configurar el software AWS IoT Greengras s Core para que utilice una clave privada y un certifica do que se almacenarán de forma segura en un módulo de seguridad de hardware (HSM).	6 de enero de 2022
Imágenes de Dockerfile y Docker para el software Core v2.5.2 AWS IoT Greengrass	Ya están disponibles el Dockerfile y la imagen de Docker para el software Core v2.5.2. AWS IoT Greengrass	20 de diciembre de 2021

AWS IoT Device Tester Publicada la versión 4.4.1 con la versión 2.2.1 de GGV2Q	Está disponible la versión 4.4.1 de IDT para V2. AWS IoT Greengrass Esta versión incluye la suite de calificac ión AWS IoT Greengrass V2 (GGV2Q) v2.2.1 y es compatible con Greengrass nucleus versión 2.5.2 para la calificación de dispositivos.	12 de diciembre de 2021
Realice inferencias de aprendizaje automático con Amazon Lookout for Vision	Se agregó información sobre cómo realizar inferencias de aprendizaje automático con Lookout for Vision en los dispositivos principales de Greengrass. Lookout for Vision utiliza la visión artificia l para encontrar defectos visuales en productos industria les.	8 de diciembre de 2021
AWS IoT Device Tester Publicada la versión 4.4.1 con la versión 2.2.0 de GGV2Q	Está disponible la versión 4.4.1 de IDT para V2. AWS IoT Greengrass Esta versión incluye el paquete de calificac ión AWS IoT Greengrass V2 (GGV2Q) v2.2.0 y es compatible con Greengrass nucleus versión 2.5.2 para la calificación de dispositivos.	6 de diciembre de 2021

AWS IoT Greengrass Actualización de software Core v2.5.2	Esta versión proporciona la versión 2.5.2 del component e núcleo de Greengrass y AWS actualiza los component es proporcionados. Corrige un problema con el servicio de Windows que se produce después de que se actualice el núcleo de Greengrass. También incluye soporte para el AWS IoT Device Defender componente en dispositivos Windows.	3 de diciembre de 2021
Nuevo conector perimetral para el componente Kinesis Video Streams	Está disponible la versión 1.0.0 del componente Edge Connector para Kinesis Video Streams. Este proveedor lee AWS las transmisiones de vídeo de las cámaras locales y publica las transmisiones en Kinesis Video Streams. Este componente se integra con AWS IoT TwinMaker, lo que le permite ver y administrar las transmisiones de vídeo y otros datos en los paneles de Grafana.	30 de noviembre de 2021

Gestione los dispositivos principales de Greengrass con AWS Systems Manager	Se agregó información sobre cómo administrar los dispositivos principales de Greengrass con. AWS Systems Manager Systems Manager es un AWS servicio que le permite ver los datos operativos, automatizar las tareas operativas y mantener la seguridad y el cumplimiento.	29 de noviembre de 2021
Kit de desarrollo de Greengrass (CLI)	Se agregó información sobre la interfaz de línea de comandos del kit de AWS IoT Greengrass desarrollo (GDK CLI), que es una herramienta que puede descargar en su computadora de desarrollo local para ayudarlo a desarrollar componentes personalizados de Greengrass. Puede usar la CLI de GDK para crear, compilar y publicar componentes personalizados.	29 de noviembre de 2021
Componentes de Greengrass proporcionados por la comunidad	Se agregó información sobre el catálogo de software de Greengrass, que es un índice de los componentes de Greengrass desarrollados por la comunidad de Greengrass. Desde este catálogo, puede descargar, modificar e implementar componentes para crear sus aplicaciones de Greengrass.	29 de noviembre de 2021

AWS IoT Greengrass Actualización del software Core v2.5.1	Esta versión proporciona la versión 2.5.1 del component e núcleo de Greengrass y AWS actualiza los component es proporcionados. Incluye soporte para Java de 32 bits en dispositivos Windows. También corrige los problemas relacionados con el nuevo comportamiento de eliminaci ón de grupos de cosas y la carga de las variables de entorno del sistema en los dispositivos Windows.	23 de noviembre de 2021
AWS IoT Device Tester Publicada la versión 4.4.0 con la versión 2.1.0 de GGV2Q	Está disponible la versión 4.4.0 de IDT para V2. AWS IoT Greengrass Esta versión incluye la suite de calificac ión AWS IoT Greengrass V2 (GGV2Q) v2.1.0 y admite la calificación de dispositi vos Greengrass basados en Windows que ejecutan Greengrass nucleus versión 2.5.0.	19 de noviembre de 2021

AWS IoT Greengrass Actualización de software Core v2.5.0	Esta versión proporciona la versión 2.5.0 del component e núcleo de Greengrass y AWS actualiza los component es proporcionados. Incluye soporte para ejecutar el software AWS IoT Greengrass Core en dispositivos Windows. También cambia el comportam iento de eliminación de grupos de cosas y añade compatibi lidad con los proxies HTTPS.	12 de noviembre de 2021
SageMaker Lanzamiento de Edge Manager v1.1.0	Ya está disponible el componente Amazon SageMaker Edge Manager v1.1.0. Esta versión añade compatibilidad con los dispositi vos principales de Greengras s que ejecutan Amazon Linux 2 y añade un nuevo parámetro de configuración para especificar la ubicación de la carpeta de datos de captura en el dispositivo.	3 de noviembre de 2021
Actualización de la prevenció n del suplente confuso entre servicios	AWS IoT Greengrass V2 admite el uso de las claves de contexto `aws:SourceArn` y de condición `aws:Sourc eAccount` global en las políticas de recursos de IAM para evitar el confuso problema de los adjuntos.	1 de noviembre de 2021

Actualizaciones de componentes del dispositivo cliente	Están disponibles la autenticación de dispositivos cliente v2.0.3, el detector de IP v2.1.0, el puente MQTT v2.1.0 y el agente MQTT (Moquette) v2.0.2. Esta versión añade compatibilidad total con los puertos de broker MQTT no predeterminados e incluye otras correcciones de errores y mejoras.	28 de octubre de 2021
Lanzada la versión 2.0.4 de Shadow Manager	Está disponible el componente Shadow Manager v2.0.4. Esta versión corrige un problema que provocaba que el administrador de sombras eliminara las versiones recién creadas de cualquier sombra que se hubiera eliminado anteriormente. A partir de esta versión, la operación `DeleteThingShadow` IPC incrementa la versión oculta.	20 de octubre de 2021
Publicada la versión 2.2.0 de Log Manager	Está disponible el componente Log Manager v2.2.0. El administrador de registros ahora admite el uso de un mapa de configuración para proporcionar las configuraciones de registro de los componentes.	20 de octubre de 2021

Lanzada la versión 2.1.4 de Lambda Manager	Está disponible el componente Lambda manager v2.1.4. Esta versión corrige un problema que provocaba que las funciones de Lambda que utilizan tiempos de ejecución de Nodejs procesaran solo un mensaje.	20 de octubre de 2021
Utilice la comunicación entre procesos, AWS las credenciales y el administrador de flujos en los componentes del contenedor de Docker	Se agregó información sobre cómo usar la comunicación entre procesos (IPC), AWS las credenciales y el administrador de flujos en sus componentes de contenedor Docker personalizados.	19 de octubre de 2021
Nuevo componente emisor de telemetría de núcleo	Está disponible la versión 1.0.0 del componente emisor de telemetría del núcleo. Este componente AWS proporcionado recopila datos de telemetría de salud del sistema y los publica continuamente en un tema local y en un tema de MQTT. AWS IoT Core	30 de septiembre de 2021
Permita el tráfico del dispositivo a través de un proxy o firewall	Se agregó información sobre los puntos finales y los puertos que utilizan los dispositivos principales de Greengrass, para que pueda restringir el tráfico como medida de seguridad.	16 de septiembre de 2021

AWS IoT Device Tester Publicada la versión 4.2.0 con la versión 2.0.1 de GGV2Q	La versión 4.2.0 de IDT para la versión 2 se ha actualizado con la versión 2.0.1 de la suite de AWS IoT Greengrass calificación V2 (GGV2Q). AWS IoT Greengrass Esta versión es compatible con la versión 2.4.0 del núcleo de Greengrass para la calificación de los dispositivos.	31 de agosto de 2021
Se actualizaron los componentes del instalador de aprendizaje automático	Están disponibles el componente instalador DLR v1.6.5 y el componente instalador TensorFlow Lite v2.5.4. Estas versiones de componentes incluyen el nuevo parámetro de `UseInstaller` configuración que permite deshabilitar el script de instalación predeterminado.	30 de agosto de 2021
Soporte integrado de Linux para AWS IoT Greengrass	La BitBake receta AWS IoT Greengrass V2 está disponible en el `meta-aws` proyecto en GitHub. Puede utilizar esta receta para crear un sistema operativo personalizado basado en Linux mediante el proyecto Yocto.	20 de agosto de 2021

Integridad del código	Se agregó información sobre cómo se AWS IoT Greengras s V2 verifica la integridad del software que los dispositivos principales de Greengrass descargan desde. Nube de AWS	19 de agosto de 2021
Puntos de conexión de VPC (AWS PrivateLink)	AWS IoT Greengrass ahora admite puntos finales de VPC de interfaz (AWS PrivateLink) para el AWS IoT Greengras s plano de control. Puede establecer una conexión privada entre la VPC y el plano de AWS IoT Greengrass control.	16 de agosto de 2021
Publicada la versión 2.0.12 de Stream Manager	Ya está disponible la versión 2.0.12 de Stream Manager. Esta versión corrige un problema que impedía actualizar la versión 2.0.7 del componente Stream Manager a una versión entre la v2.0.8 y la v2.0.11.	10 de agosto de 2021
Dockerfile e imágenes de Docker para la versión 2.4.0 del software Core AWS IoT Greengrass	Ya están disponibles el Dockerfile y la imagen de Docker para el software Core v2.4.0. AWS IoT Greengrass	9 de agosto de 2021

AWS IoT Greengrass Actualización del software Core v2.4.0	Esta versión proporciona la versión 2.4.0 del component e núcleo de Greengrass y AWS actualiza los component es proporcionados. Incluye compatibilidad con los límites de recursos del sistema de componentes, las operaciones de IPC para pausar y reanudar los componentes y el aprovisio namiento de complementos.	3 de agosto de 2021
Componentes nuevos AWS IoT SiteWise	Se agregaron los siguiente s componentes AWS proporcionados para AWS IoT SiteWise: el recopilador SiteWise OPC-UA de IoT, el SiteWiseeditor de IoT y el procesador de IoT. SiteWise	29 de julio de 2021
AWS IoT Device Tester Lanzamiento de la versión 4.2.0 con la versión 2.0.0 de GGV2Q	Está disponible la versión 4.2.0 de IDT para V2. AWS IoT Greengrass Esta versión incluye el paquete de calificac ión AWS IoT Greengrass V2 (GGV2Q) v2.0.0 e incluye soporte para pruebas de calificación opcionales para los componentes de Docker, el aprendizaje automático y el administrador de transmisi ones.	14 de julio de 2021

AWS IoT Greengrass La biblioteca Core IPC está disponible en C++ v2 SDK para dispositivos con AWS IoT	La versión 1.13.0 de la versión SDK para dispositivos con AWS IoT para C++ es compatible con AWS IoT Greengrass Core IPC, por lo que puede desarrollar componentes en C++ que interactúen con el software Core. AWS IoT Greengrass	14 de julio de 2021
SageMaker Publicada la versión 1.0.2 del componente Edge Manager	Ya está disponible el componente Amazon SageMaker Edge Manager v1.0.2. Esta versión actualiza el script de instalación en el ciclo de vida del componente. Sus dispositivos principales ahora deben tener Python 3.6 o posterior, incluida `pip` su versión de Python, instalado en el dispositivo antes de implementar este componente.	12 de julio de 2021
Actualización de soporte AWS IoT Device Tester para AWS IoT Greengrass V2	La versión 4.1.0 de IDT para AWS IoT Greengrass V2 ahora admite el uso de la versión 2.3.0 del núcleo de Greengrass para la calificación de los dispositivos.	8 de julio de 2021
Dockerfile e imágenes de Docker para el software Core v2.3.0 AWS IoT Greengrass	Ya están disponibles el Dockerfile y la imagen de Docker para el software Core v2.3.0. AWS IoT Greengrass	7 de julio de 2021

AWS políticas gestionadas	Se agregó información sobre las políticas AWS administradas para AWS IoT Greengrass.	2 de julio de 2021
Nuevas opciones de JVM recomendadas	Se agregó información sobre las opciones de JVM recomendadas para controlar la asignación de memoria para el software AWS IoT Greengrass Core.	30 de junio de 2021
AWS IoT Greengrass Actualización del software Core v2.3.0	Esta versión proporciona la versión 2.3.0 del componente núcleo de Greengrass y AWS actualiza los componentes proporcionados. Incluye soporte para documentos de configuración de componentes de gran tamaño en las implementaciones.	29 de junio de 2021
Dockerfile e imágenes de Docker para el software Core v2.2.0 AWS IoT Greengrass	Ya están disponibles el Dockerfile y la imagen de Docker para el software Core v2.2.0. AWS IoT Greengrass	28 de junio de 2021
AWS IoT Device Tester Publicada la versión 4.1.0 con la versión 1.1.1 de GGV2Q	Ya está disponible la versión 4.1.0 de IDT para V2. AWS IoT Greengrass Esta versión incluye el paquete de calificación AWS IoT Greengrass V2 (GGV2Q) v1.1.1 y admite el uso de Greengrass nucleus v2.2.0, v2.1.0 y v2.0.5 para la calificación de dispositivos.	18 de junio de 2021

AWS IoT Greengrass Actualización de software Core v2.2.0	Esta versión proporciona la versión 2.2.0 del component e núcleo de Greengrass y AWS actualiza los component es proporcionados. Incluye componentes que puede implementar para añadir soporte a los dispositivos cliente y añadir el servicio paralelo local.	18 de junio de 2021
Lanzada la versión 2.0.6 del lanzador Lambda	Está disponible la versión 2.0.6 del componente Lanzador Lambda. Esta versión incluye mejoras de rendimiento y correcciones de errores.	13 de junio de 2021
Lanzamiento SageMaker del nuevo componente Edge Manager	La versión 1.0.0 del componente Amazon SageMaker Edge Manager está disponible para AWS IoT Greengrass. Este componente instala el binario del agente de SageMaker Edge Manager en los dispositivos principales de Greengrass.	10 de junio de 2021
Tipos de componentes	Se agregó información sobre los tipos de componentes en AWS IoT Greengrass. El tipo de componente especifica cómo el software AWS IoT Greengrass principal ejecuta un componente.	3 de junio de 2021

AWS IoT Device Tester Publicada la versión 4.0.2 y la versión 1.1.0 de GGV2Q	Ya está disponible la versión 4.0.2 de IDT para V2. AWS IoT Greengrass Esta versión incluye el paquete de calificación AWS IoT Greengrass V2 (GGV2Q) v1.1.0 y admite el uso de Greengrass nucleus v2.1.0 con Greengrass CLI v2.1.0 para la calificación de dispositivos. Esto también incluye los nuevos grupos de pruebas necesarios para MQTT y Lambda, así como otras mejoras y correcciones de errores menores.	5 de mayo de 2021
Imágenes de Dockerfile y Docker para la versión 2.1.0 del software Core AWS IoT Greengrass	Ya están disponibles el Dockerfile y la imagen de Docker para el software Core v2.1.0. AWS IoT Greengrass La imagen de Docker le permite ejecutar el software AWS IoT Greengrass principal en un contenedor de Docker que usa Amazon Linux 2 como sistema operativo base.	27 de abril de 2021

AWS IoT Greengrass Actualización del software Core v2.1.0	Esta versión proporciona la versión 2.1.0 del componente núcleo de Greengrass y AWS actualiza los componentes proporcionados. Incluye un nuevo componente que puede usar para descargar imágenes de Docker desde repositor ios privados de Amazon ECR y nuevos componentes de muestra para realizar inferenci as de aprendizaje automático con Lite. TensorFlow	26 de abril de 2021
Ejemplo de componente que usa Secrets Manager	Se agregó un componente de ejemplo que imprime el valor de un AWS Secrets Manager secreto que se implementa en un dispositivo principal.	8 de abril de 2021
AWS IoT Política mínima para los dispositivos principales de Greengrass	Se agregó información sobre el conjunto mínimo de permisos necesarios para admitir la funcionalidad básica de Greengrass en un dispositi vo principal.	2 de abril de 2021
Suscríbase a las transmisi ones de eventos del IPC	Se agregó información sobre cómo usar las operacion es de comunicación entre procesos (IPC) para suscribir se a transmisiones de eventos en un dispositivo central de Greengrass.	1 de abril de 2021

Actualización de soporte AWS IoT Device Tester para AWS IoT Greengrass	La versión 4.0.1 de IDT para AWS IoT Greengrass V2 ahora admite el uso de la versión 2.0.5 del núcleo de Greengrass con la versión 2.0.5 de la CLI de Greengras s para la calificación de los dispositivos.	17 de marzo de 2021
Cree componentes personali zados que usen el administr ador de flujos	Se agregó información sobre cómo configurar las recetas de componentes y los artefactos para desarrollar aplicaciones que administren los flujos de datos.	9 de marzo de 2021
AWS IoT Greengrass Actualización del software Core v2.0.5	Esta versión proporciona la versión 2.0.5 del componente núcleo de Greengrass y AWS actualiza los componentes proporcionados. Soluciona un problema con la compatibi lidad con el proxy de red y un problema con el punto final del plano de datos de Greengras s en las regiones de AWS China.	9 de marzo de 2021

Referencia de variables de entorno de componentes	Se agregó información sobre las variables de entorno que el software AWS IoT Greengrass principal establece para los componentes. Puede usar estas variables de entorno para obtener el nombre de la cosa y la Región de AWS versión del núcleo de Greengrass.	23 de febrero de 2021
Instalación manual	Se agregó información sobre cómo crear AWS los recursos necesarios manualmente o instalarlos detrás de un firewall o un proxy de red. Al realizar una instalación manual, no es necesario dar permiso al instalador para crear recursos en la suya Cuenta de AWS, ya que crea los recursos necesarios AWS IoT y los de IAM. También puedes configurar el dispositivo para que se conecte al puerto 443 o a través de un proxy de red.	17 de febrero de 2021
AWS IoT Greengrass Actualización de la biblioteca Core IPC SDK para dispositivos con AWS IoT para Python v2	La versión 1.5.4 de la versión SDK para dispositivos con AWS IoT para Python simplifica los pasos necesarios para conectarse al servicio AWS IoT Greengrass Core IPC.	11 de febrero de 2021

Actualización de soporte AWS IoT Device Tester para AWS IoT Greengrass	La versión 4.0.1 de IDT para AWS IoT Greengrass V2 ahora admite el uso de la versión 2.0.4 del núcleo de Greengrass con la versión 2.0.4 de la CLI de Greengras s para la calificación de los dispositivos.	5 de febrero de 2021
Nuevo tutorial para importar funciones Lambda	Se agregó un nuevo tutorial basado en consola para importar una función Lambda como un componente que se ejecuta en el dispositivo principal de Greengrass.	5 de febrero de 2021
AWS IoT Greengrass Actualización del software Core v2.0.4	Esta versión proporciona la versión 2.0.4 del component e núcleo de Greengrass. Incluye el nuevo `greengras sDataPlanePort` parámetro para configurar la comunicación HTTPS a través del puerto 443 y corrige errores. La política de IAM mínima ahora requiere que se ejecute el instalador del software AWS IoT Greengras s principal `iam:GetPolicy` y `sts:GetCallerIdent ity` cuando se ejecute con `--provision true` él.	4 de febrero de 2021

Lanzamiento de un nuevo componente de tunelización segura	La versión 1.0.0 del componente de tunelización segura está disponible para. AWS IoT Greengrass Este componente AWS proporcionado utiliza una tunelización AWS IoT segura para establecer una comunicación bidireccional segura con un dispositivo central de Greengrass que se encuentra detrás de firewalls restringidos.	21 de enero de 2021
AWS IoT Device Tester para la versión 4.0.1 publicada AWS IoT Greengrass	Está disponible la versión 4.0.1 de IDT para AWS IoT Greengrass V2. Esta versión le permite usar IDT para desarrollar y ejecutar sus conjuntos de pruebas personalizadas para la validación de dispositivos. Esto también incluye aplicaciones IDT firmadas con código para macOS y Windows.	22 de diciembre de 2020
Versión inicial de AWS IoT Greengrass Version 2	AWS IoT Greengrass V2 es una nueva versión principal de AWS IoT Greengrass. Esta versión añade varias funciones, como componentes de software modulares e implementaciones continuas. Estas funciones le facilitan el desarrollo y la administración de aplicaciones perimetrales.	15 de diciembre de 2020

AWS Glosario

Para obtener la AWS terminología más reciente, consulte el AWS glosario de la Glosario de AWS Referencia.

Las traducciones son generadas a través de traducción automática. En caso de conflicto entre la traducción y la version original de inglés, prevalecerá la version en inglés.