

---

# AWS IoT

## Guía para desarrolladores



## AWS IoT: Guía para desarrolladores

Copyright © 2017 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

## Table of Contents

¿Qué es AWS IoT .....	1
Componentes de AWS IoT .....	1
Primeros pasos con AWS IoT .....	2
Acceso a AWS IoT .....	2
Servicios relacionados .....	2
Cómo funciona AWS IoT .....	3
Introducción a AWS IoT .....	5
Inicio de sesión en la consola de AWS IoT .....	5
Registro de un dispositivo en el registro de objetos .....	6
Creación y activación de un certificado de dispositivo .....	8
Creación de una política de AWS IoT .....	10
Asociación de una política de AWS IoT a un certificado de dispositivo .....	13
Asociación de un certificado a un objeto .....	14
Configuración del dispositivo .....	17
Configuración de un botón AWS IoT .....	17
Configuración de otro dispositivo .....	18
Visualización de los mensajes MQTT de dispositivo con el cliente MQTT de AWS IoT .....	18
Configuración y comprobación de reglas .....	21
Creación de un tema de SNS .....	21
Suscripción a un tema de Amazon SNS .....	23
Creación de una regla .....	24
Comprobación de la regla de Amazon SNS .....	29
Pasos siguientes .....	30
Inicios rápidos del botón AWS IoT .....	31
Inicio rápido del asistente para el botón AWS IoT .....	32
Inicio rápido AWS CloudFormation del botón AWS IoT .....	40
Pasos siguientes .....	45
Tutoriales de reglas de AWS IoT .....	46
Creación de un regla de DynamoDB .....	47
Creación de un regla Lambda .....	55
Creación de la función Lambda .....	55
Comprobación de la función Lambda .....	63
Creación de un regla Lambda .....	65
Comprobación de la regla Lambda .....	68
Creación de un regla de Amazon SNS .....	70
Tutoriales de SDK de AWS IoT .....	78
Conexión de Raspberry Pi .....	78
Requisitos previos .....	78
Inicio de sesión en la consola de AWS IoT .....	79
Cree y asocie un objeto (dispositivo) .....	80
Uso de AWS IoT SDK para Embedded C .....	87
Configuración del entorno de tiempo de ejecución para AWS IoT SDK para Embedded C .....	87
Configuración de aplicación de muestra .....	87
Ejecución de las aplicaciones de muestra .....	89
Uso de AWS IoT Device SDK para JavaScript .....	90
Configuración del entorno de tiempo de ejecución para AWS IoT Device SDK para JavaScript .....	90
Instalación de AWS IoT Device SDK para JavaScript .....	92
Preparación antes de ejecutar las aplicaciones de muestra .....	92
Ejecución de las aplicaciones de muestra .....	92
Administración de objetos con AWS IoT .....	94
Administración de objetos con el registro de objetos .....	95
Creación de un objeto .....	95
Lista de objetos .....	95
Buscar objetos .....	95

Actualización de un objeto .....	97
Eliminación de un objeto .....	97
Asociar un principal a un objeto .....	97
Desvincular un principal de un objeto .....	97
Tipos de objeto .....	98
Creación de un tipo de objeto .....	98
Lista de los tipos de objeto .....	98
Descripción de un tipo de objeto .....	99
Asociación de un tipo de objeto a un objeto .....	99
Descartar un tipo de objeto .....	99
Eliminación de un tipo de objeto .....	100
Seguridad e identidad .....	101
Autenticación en AWS IoT .....	102
Certificados X.509 .....	102
Usuarios, grupos y roles de IAM .....	108
Identidades de Amazon Cognito .....	109
Autorización .....	109
Políticas de AWS IoT .....	111
Políticas de IoT de IAM .....	131
Acceso entre cuentas .....	134
Seguridad de transporte .....	135
Compatibilidad con el conjunto de cifrado de TLS .....	135
Agente de mensajes .....	136
Protocolos .....	136
Protocolo/mapeos de puerto .....	136
MQTT .....	137
HTTP .....	137
MQTT sobre el protocolo WebSocket .....	138
Temas .....	141
Temas reservados .....	142
Eventos del ciclo de vida .....	145
Política necesaria para recibir eventos de ciclo de vida .....	145
Eventos de conexión/desconexión .....	146
Eventos de suscripción/cancelación de suscripción .....	146
Reglas .....	148
Concesión a AWS IoT del acceso requerido .....	149
Transmisión de los permisos de rol .....	150
Creación de una regla de AWS IoT .....	151
Visualización de las reglas .....	154
Versiones de SQL .....	154
Novedades de la versión del motor de reglas SQL del 23/03/2016 .....	155
Solución de problemas de una regla .....	156
Eliminación de una regla .....	156
Acciones de las reglas de AWS IoT .....	156
Acción de alarma de CloudWatch .....	157
Acción de métrica de CloudWatch .....	158
Acción DynamoDB .....	159
Acción DynamoDBv2 .....	160
Acción Amazon ES .....	161
Acción Firehose .....	162
Acción Kinesis .....	162
Acción Lambda .....	163
Acción Republish .....	164
Acción S3 .....	165
Acción SNS .....	166
Acción SQS .....	166
Acción Salesforce .....	167

Referencias de SQL en AWS IoT .....	168
Tipos de datos .....	169
Operadores .....	172
Funciones .....	178
Cláusula SELECT .....	211
Cláusula FROM .....	212
Cláusula WHERE .....	213
Literales .....	214
Instrucciones case .....	214
Extensiones JSON .....	215
Plantillas de sustitución .....	216
Sombras de objeto .....	217
Flujo de datos de sombras de objeto .....	217
Detección de un objeto conectado .....	224
Documentos de sombras de objeto .....	225
Propiedades del documento .....	225
Control de versiones de una sombra de objeto .....	226
Token de cliente .....	226
Ejemplo de documento .....	226
Secciones vacías .....	227
Matrices .....	227
Uso de sombras de objeto .....	228
Compatibilidad del protocolo .....	228
Actualización de una sombra de objeto .....	228
Recuperación de un documento de sombra de objeto .....	229
Eliminación de datos .....	232
Eliminación de una sombra de objeto .....	233
Estado delta .....	233
Observación de los cambios de estado .....	235
Orden de los mensajes .....	235
Supresión de mensajes de sombras de objeto .....	236
API RESTful .....	237
GetThingShadow .....	237
UpdateThingShadow .....	238
DeleteThingShadow .....	239
Temas de publicación/suscripción MQTT .....	239
/update .....	240
/update/accepted .....	241
/update/documents .....	241
/update/rejected .....	242
/update/delta .....	242
/get .....	243
/get/accepted .....	243
/get/rejected .....	244
/delete .....	244
/delete/accepted .....	245
/delete/rejected .....	245
Sintaxis del documento .....	246
Documentos de estado de la solicitud .....	246
Documentos de estado de la respuesta .....	246
Documentos de respuesta de error .....	247
Mensajes de error .....	248
SDK de AWS IoT .....	249
SDK de AWS Mobile para Android .....	249
Arduino Yún SDK .....	249
AWS IoT Device SDK para Embedded C .....	250
AWS Mobile SDK para iOS .....	250

AWS IoT Device SDK para Java .....	250
AWS IoT Device SDK para JavaScript .....	250
AWS IoT Device SDK para Python .....	251
Monitorización .....	252
Herramientas de monitorización .....	253
Herramientas automatizadas .....	253
Herramientas manuales .....	253
Monitorización con Amazon CloudWatch .....	254
Dimensiones y métricas .....	254
Uso de las métricas de AWS IoT .....	258
Creación de alarmas de CloudWatch .....	259
Registro de llamadas a la API en AWS IoT con AWS CloudTrail .....	261
Información de AWS IoT en CloudTrail .....	261
Entradas de archivos de log de AWS IoT .....	262
Solución de problemas .....	264
Diagnóstico de problemas de conectividad .....	264
Autenticación .....	264
Autorización .....	264
Configuración de CloudWatch Logs .....	265
Configuración de un rol de IAM para registro .....	265
Formato de entrada de los logs de CloudWatch .....	266
Eventos de registro y códigos de error .....	267
Diagnóstico de problemas de las reglas .....	270
Diagnóstico de problemas en las sombras de objeto .....	270
Diagnosticar problemas con acciones de Salesforce .....	271
Registro de seguimiento de ejecución .....	271
Éxito y error de una acción .....	271

# ¿Qué es AWS IoT?

AWS IoT proporciona una comunicación bidireccional segura entre objetos conectados a Internet (dispositivos como sensores, actuadores, microcontroladores integrados o aparatos inteligentes) y la nube de AWS. Esto le permite recopilar datos de telemetría de varios objetos y almacenar y analizar los datos. También puede crear aplicaciones que permitan a sus usuarios controlar estos dispositivos desde sus teléfonos o tablets.

## Componentes de AWS IoT

AWS IoT está formado por los siguientes componentes:

Gateway para dispositivos

Permite a los dispositivos comunicarse de forma segura y eficaz con AWS IoT.

Agente de mensajes

Proporciona un mecanismo seguro para que los objetos y las aplicaciones de AWS IoT publiquen y reciban mensajes entre sí. Puede utilizar el protocolo MQTT directamente o MQTT sobre WebSocket para publicar y suscribirse. Puede utilizar la interfaz HTTP REST para publicar.

Motor de reglas

Proporciona funciones de procesamiento de mensajes y de integración con otros servicios de AWS. Puede utilizar un lenguaje basado en SQL para seleccionar datos de cargas de mensajes y procesar y enviar datos a otros servicios, como Amazon S3, Amazon DynamoDB y AWS Lambda. También puede utilizar el agente de mensajes para volver a publicar mensajes para otros suscriptores.

Servicio de seguridad e identidad

Comparte la responsabilidad de la seguridad en la nube de AWS. Sus objetos deben proteger las credenciales para enviar datos de forma segura al agente de mensajes. El agente de mensajes y el motor de reglas utilizan las características de seguridad de AWS para enviar datos de forma segura a dispositivos u otros servicios de AWS.

Registro de objetos

En ocasiones, se denomina registro de dispositivos. Organiza los recursos asociados a cada objeto. Usted registra los objetos y asocia hasta tres atributos a cada objeto. También puede asociar certificados e ID de cliente MQTT a cada objeto para mejorar su capacidad para administrar sus objetos y solucionar los problemas que estos presenten.

#### Sombra de objeto

En ocasiones, se denomina sombra de dispositivo. Documento JSON utilizado para almacenar y recuperar información del estado actual de un objeto (dispositivo, aplicación, etc.).

#### Servicio Thing Shadows

Proporciona una representación persistente de sus objetos en la nube de AWS. Puede publicar información de estado actualizada en una sombra de objeto y este puede sincronizar su estado cuando se conecte. Sus objetos también pueden publicar su estado actual en una sombra de objeto para que lo usen las aplicaciones o los dispositivos.

Para obtener información sobre los límites de AWS IoT, consulte [Límites de AWS IoT](#).

## Primeros pasos con AWS IoT

- Para obtener más información sobre AWS IoT, consulte [Cómo funciona AWS IoT \(p. 3\)](#).
- Para obtener información sobre cómo conectar un objeto a AWS IoT, consulte [Introducción a AWS IoT \(p. 5\)](#).

## Acceso a AWS IoT

AWS IoT ofrece las interfaces siguientes para crear e interactuar con sus objetos:

- AWS Command Line Interface (AWS CLI) - Ejecuta comandos para AWS IoT en Windows, OS X y Linux. Estos comandos le permiten crear y administrar objetos, certificados, reglas y políticas. Para comenzar, consulte la [AWS Command Line Interface Guía del usuario](#). Para obtener más información acerca de los comandos para AWS IoT, consulte `iot` en la AWS Command Line Interface Reference.
- AWS IoT API - Cree sus aplicaciones IoT mediante solicitudes HTTP o HTTPS. Estas API le permiten crear y administrar objetos, certificados, reglas y políticas mediante programación. Para obtener más información acerca de las acciones de la API para AWS IoT, consulte las [acciones](#) en las referencias de API de AWS IoT.
- AWS SDKs - Cree sus aplicaciones IoT mediante API específicas de un lenguaje. Estos SDK integran las API de HTTP/HTTPS y le permiten programar en cualquiera de los lenguajes admitidos. Para obtener más información, consulte [SDK y herramientas de AWS](#).
- AWS IoT Device SDKs - Cree aplicaciones que se ejecutan en sus dispositivos para enviar y recibir mensajes de AWS IoT. Para obtener más información, consulte los [SDK de AWS IoT](#).

## Servicios relacionados

AWS IoT se integra directamente con los siguientes servicios de AWS:

- Amazon Simple Storage Service - Proporciona almacenamiento escalable en la nube de AWS. Para obtener más información, consulte [Amazon S3](#).
- Amazon DynamoDB - Proporciona bases de datos NoSQL administradas. Para obtener más información, consulte [Amazon DynamoDB](#).
- Amazon Kinesis - Permite realizar el procesamiento de datos de streaming a gran escala. Para obtener más información, consulte [Amazon Kinesis](#).
- AWS Lambda - Ejecuta el código en servidores virtuales de Amazon EC2 en respuesta a eventos. Para obtener más información, consulte [AWS Lambda](#).

- Amazon Simple Notification Service - Envía o recibe notificaciones. Para obtener más información, consulte [Amazon SNS](#).
- Amazon Simple Queue Service - Almacena de datos en una cola para que los recuperen las aplicaciones. Para obtener más información, consulte [Amazon SQS](#).

## Cómo funciona AWS IoT

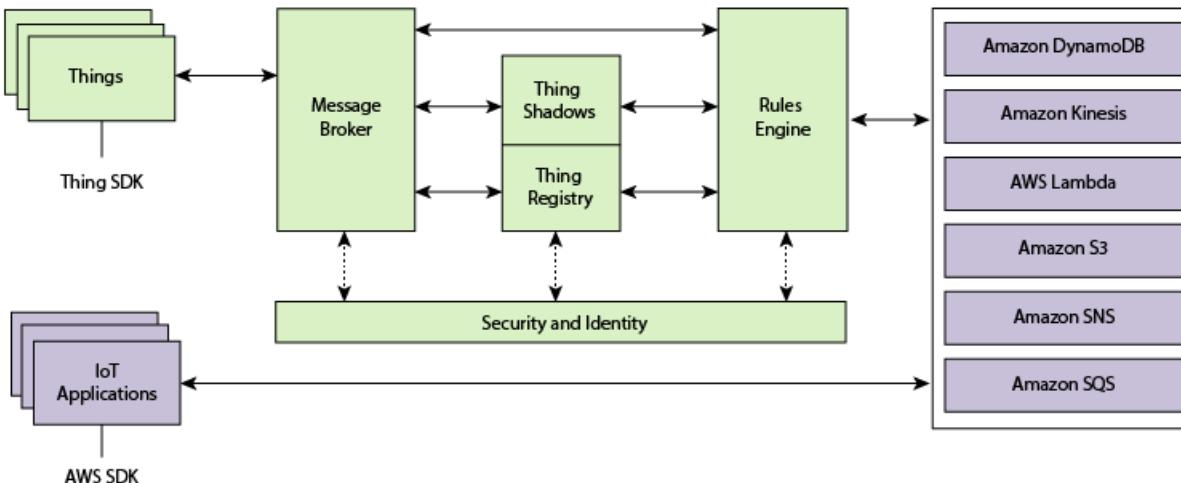
AWS IoT permite a objetos conectados a Internet conectarse a la nube de AWS. Asimismo, permite a las aplicaciones de la nube interactuar con objetos conectados a Internet. Las aplicaciones de IoT habituales recopilan y procesan telemetría de dispositivos, o bien permiten que los usuarios controlen un dispositivo de forma remota.

Los objetos indican su estado publicando mensajes en formato JSON, en temas MQTT. Cada tema MQTT tiene un nombre jerárquico que identifica el objeto cuyo estado se actualiza. Cuando un mensaje se publica en un tema MQTT, el mensaje se envía al agente de mensajes MQTT de AWS IoT, el cual se encarga de enviar todos los mensajes publicados en un tema MQTT a todos los clientes suscritos a dicho tema.

La comunicación entre un objeto y AWS IoT se protege mediante certificados X.509. AWS IoT puede generar un certificado, o bien usted puede utilizar el suyo propio. En ambos casos, el certificado debe registrarse y activarse en AWS IoT y, a continuación, copiarse en su objeto. Cuando su objeto se comunica con AWS IoT, presenta el certificado a AWS IoT como credencial.

Recomendamos que todos los objetos que se conecten con AWS IoT tengan una entrada en el registro de objetos. El registro de objetos almacena información sobre un objeto y los certificados que este utiliza para proteger la comunicación con AWS IoT.

Puede crear reglas que definan una o varias acciones que deben ejecutarse en función de los datos de un mensaje. Por ejemplo, puede insertar, actualizar o consultar una tabla de DynamoDB o invocar una función Lambda. Las reglas usan expresiones para filtrar los mensajes. Cuando una regla coincide con un mensaje, el motor de reglas invoca la acción mediante las propiedades seleccionadas. Las reglas también contienen un rol IAM que concede permiso a AWS IoT sobre los recursos de AWS utilizados para desempeñar la acción.



Cada objeto tiene una sombra de objeto que almacena y recupera información de estado. Cada elemento de la información de estado tiene dos entradas: el último estado notificado por el objeto y el estado deseado solicitado por una aplicación. Una aplicación puede solicitar la información de estado actual de un objeto. La sombra responde a la solicitud proporcionando un documento JSON con la información de estado (tanto notificado como deseado), los metadatos y un número de versión. Una aplicación puede controlar un objeto solicitando un cambio de estado. La sombra acepta la solicitud de cambio de estado, actualiza su información de estado y envía un mensaje para indicar que dicha información se ha actualizado. El objeto recibe el mensaje, cambia su estado y, a continuación, notifica su nuevo estado.

# Introducción a AWS IoT

En este tutorial se muestra cómo crear los recursos necesarios para enviar, recibir y procesar mensajes MQTT de dispositivos que utilizan AWS IoT.

Necesitará lo siguiente para completar este tutorial:

- Un equipo con acceso a wifi.
- Si tiene un botón AWS IoT (foto aquí), puede utilizarlo para completar este tutorial.
- Si no dispone de un botón, puede adquirir uno [aquí](#) o puede utilizar el cliente MQTT de la consola de AWS IoT para emular un dispositivo.



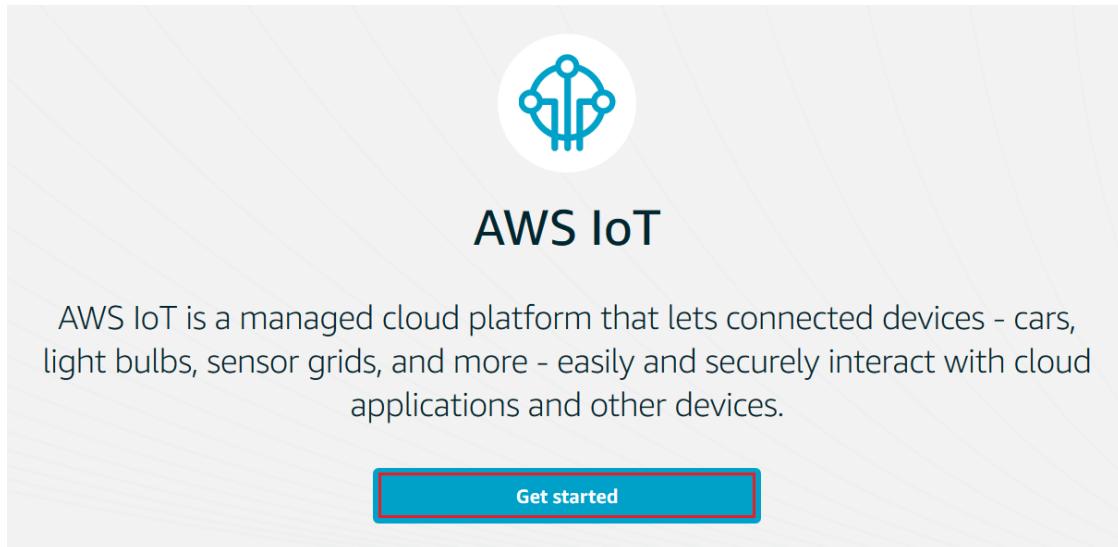
Para obtener más información acerca de AWS IoT, consulte [Qué es AWS IoT \(p. 1\)](#).

## Inicio de sesión en la consola de AWS IoT

Si no tiene una cuenta de AWS, deberá crear una.

Para crear una cuenta de AWS:

1. Abra la [página de inicio de AWS](#) y seleccione Abre una cuenta gratuita.
2. Siga las instrucciones en línea. Parte del procedimiento de inscripción consiste en recibir una llamada telefónica e introducir un número PIN con su teclado de teléfono.
3. Inicie sesión en la consola de administración de AWS y abra la [consola de AWS IoT](#).
4. En la página Welcome, elija Get started.



Si es la primera vez que utiliza la consola de AWS IoT, verá la página Welcome to the AWS IoT Console (tal y como se muestra en la siguiente imagen).

## Registro de un dispositivo en el registro de objetos

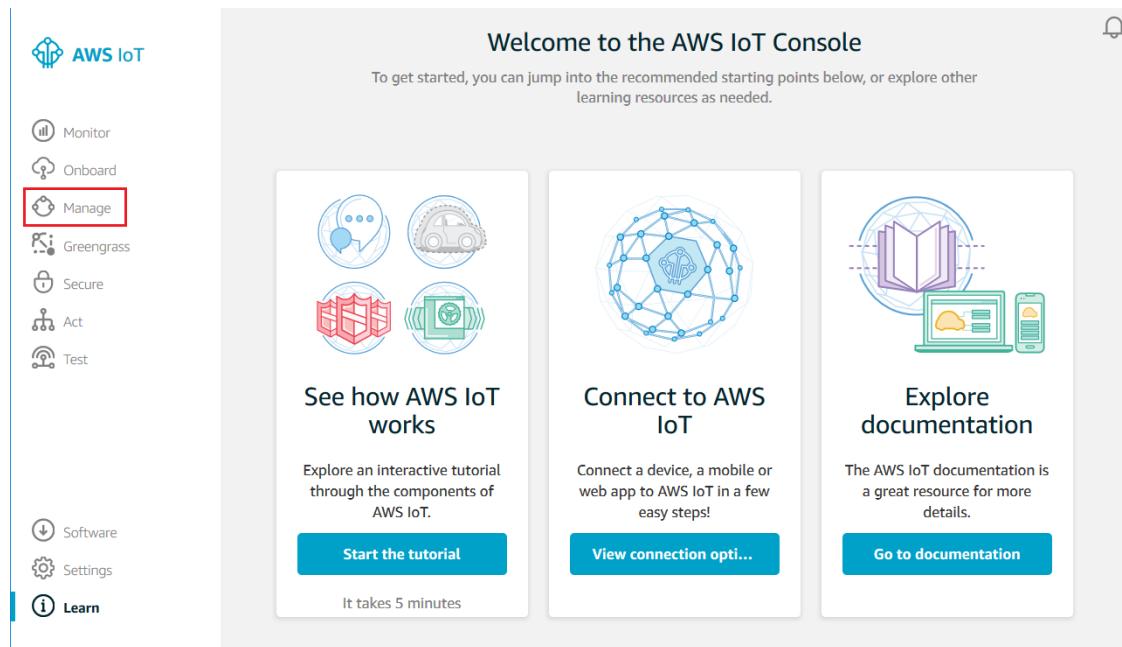
Los dispositivos conectados a AWS IoT están representados por objetos en el registro de objetos. El registro de objetos le permite mantener un registro de todos los dispositivos que están conectados a su cuenta de AWS IoT.

La manera más rápida de empezar a usar su botón AWS IoT es descargar la aplicación móvil para iOS o Android. La aplicación móvil creará automáticamente los recursos de AWS IoT necesarios y añadirá un origen del evento al botón, que invocará una nueva función AWS Lambda que usted elija mediante un proyecto Lambda. Los proyectos son funciones Lambda preconfiguradas que le permiten conectar rápidamente el clic de un botón a las funciones que prefiera, como enviar mensajes de correo o mensajes de texto automatizados o implementar otros servicios de AWS. Puede descargar las aplicaciones móviles desde: [Apple App Store](#) o [Google Play](#).

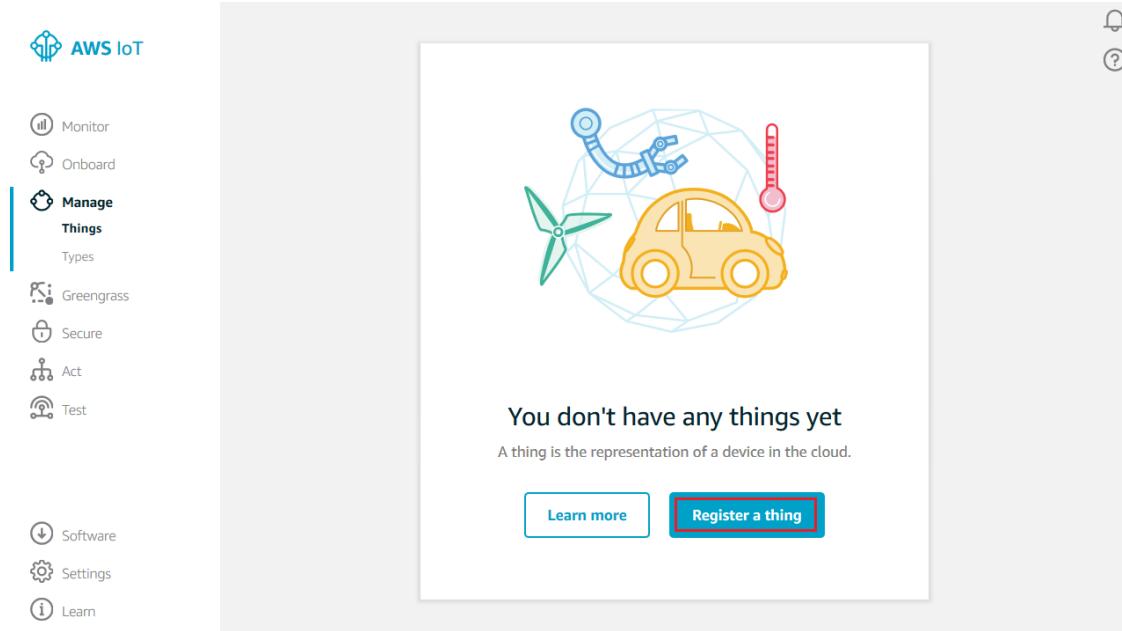
Si no puede utilizar las aplicaciones móviles, consulte las instrucciones que se indican a continuación.

Para registrar su dispositivo en el registro de objetos:

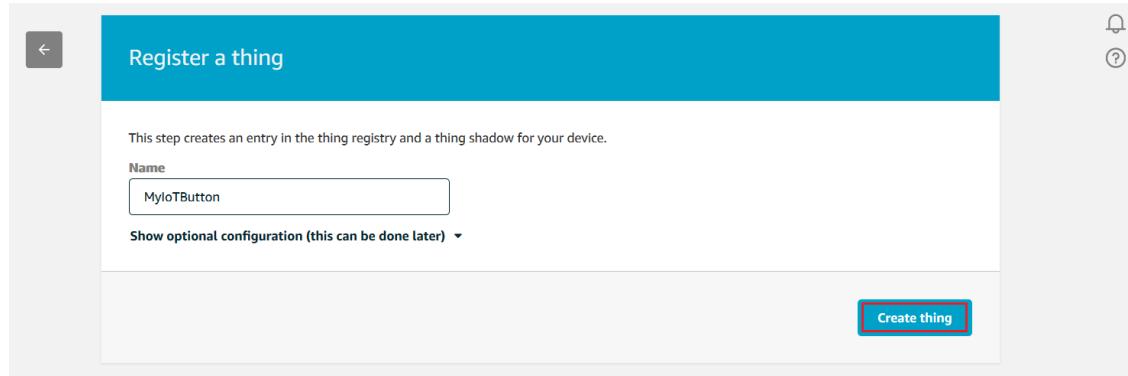
1. En el panel de navegación izquierdo de la página Welcome to the AWS IoT Console, seleccione Manage para ampliar las opciones y, a continuación, elija Things (según proceda).



2. En la página donde se indica You don't have any things yet, elija Register a thing.



3. En la página Register a thing, en el campo Name, escriba un nombre para el dispositivo, como **MyIoTButton**. Elija Create thing para agregar su dispositivo al registro de objetos.

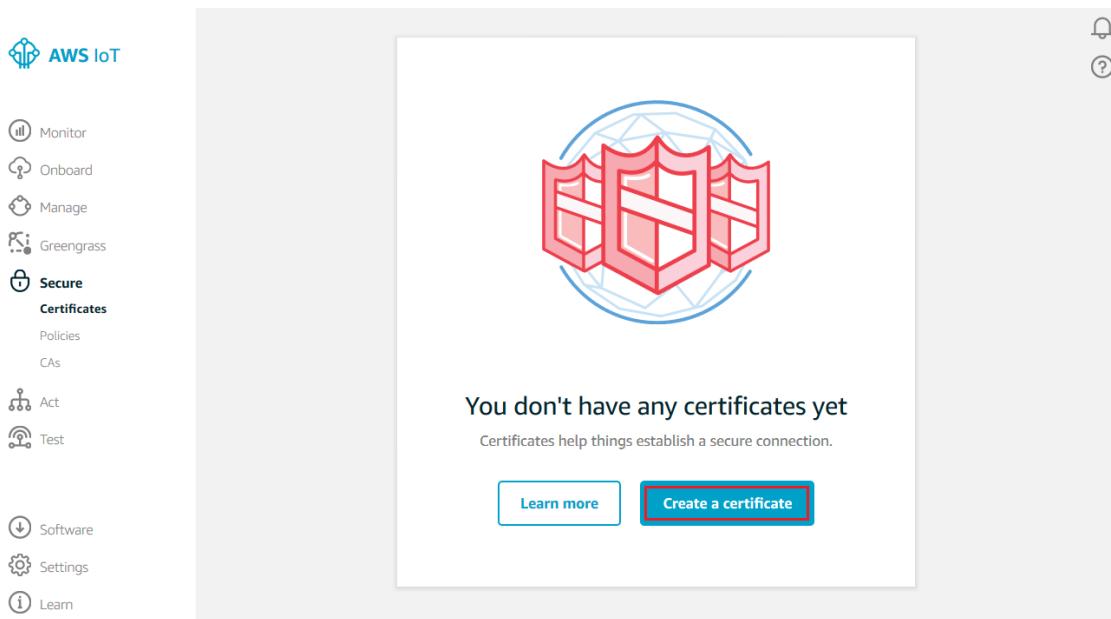


Se obtiene el siguiente resultado:

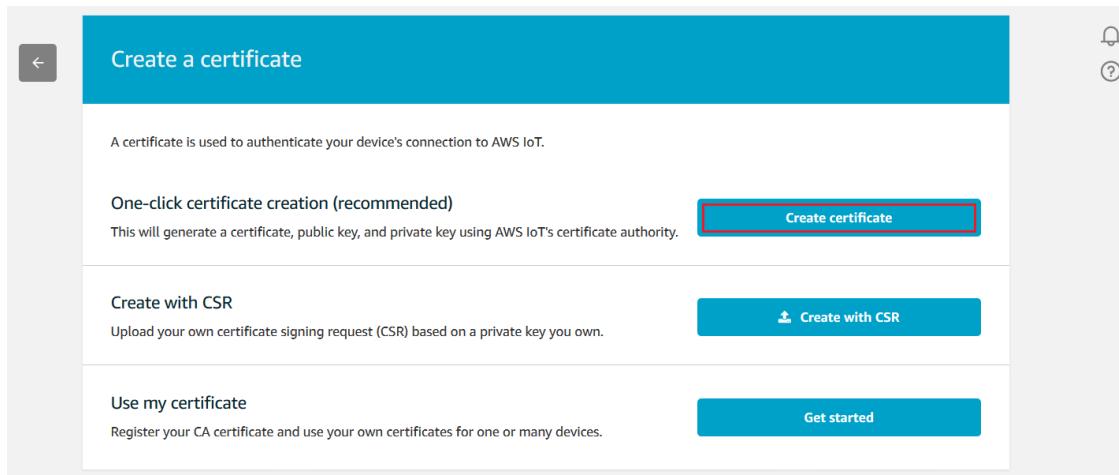
## Creación y activación de un certificado de dispositivo

La comunicación entre su dispositivo y AWS IoT se protege mediante certificados X.509. AWS IoT puede generar un certificado, o bien usted puede utilizar su propio certificado X.509. En este tutorial se presupone que AWS IoT generará el certificado X.509 para usted. Los certificados deben activarse para poder usarlos.

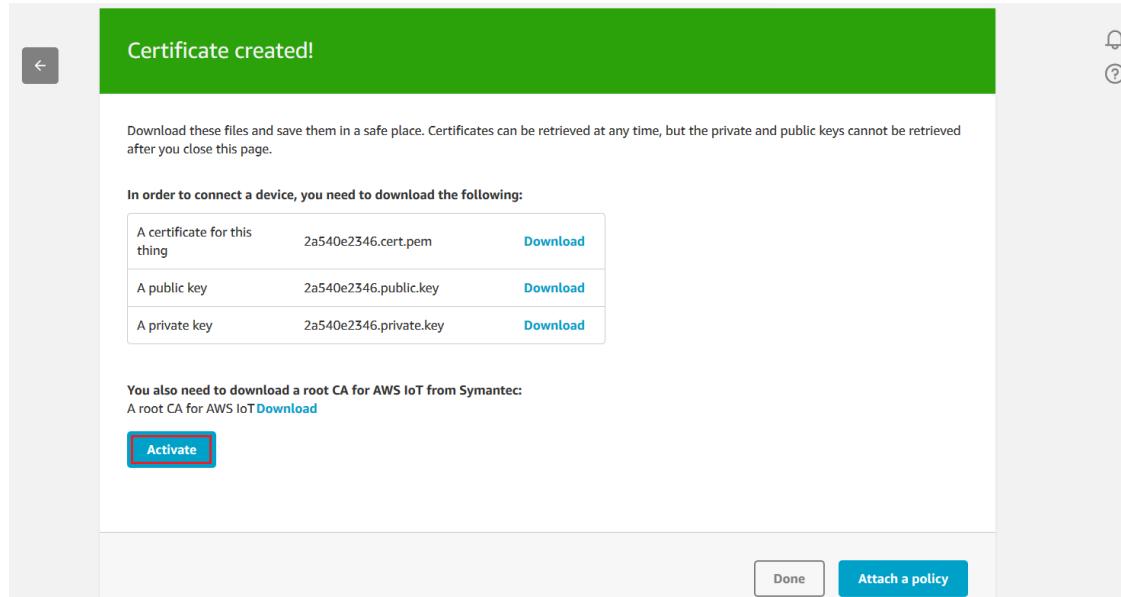
1. En el panel de navegación izquierdo, seleccione Secure, Certificates (según sea necesario) y, a continuación, Create a certificate.



2. En la página Create a certificate, seleccione Create certificate.



3. En la página Certificate created!, elija Download para el certificado, la clave privada y el certificado de CA raíz de AWS IoT (no es necesario descargar la clave privada). Guárdelos en su equipo y, a continuación, elija Activate para continuar.



Tenga en cuenta que los nombres de los archivos descargados pueden aparecer de forma distinta a los que figuran en la página Certificate created!. Por ejemplo:

- 2a540e2346-certificate.pem.crt.txt
- 2a540e2346-private.pem.key
- 2a540e2346-public.pem.key

#### Note

Aunque sea poco probable, los certificados de CA raíz están sujetos a vencimiento o revocación. Si esto ocurriese, deberá copiar un certificado de CA raíz nuevo en su dispositivo.

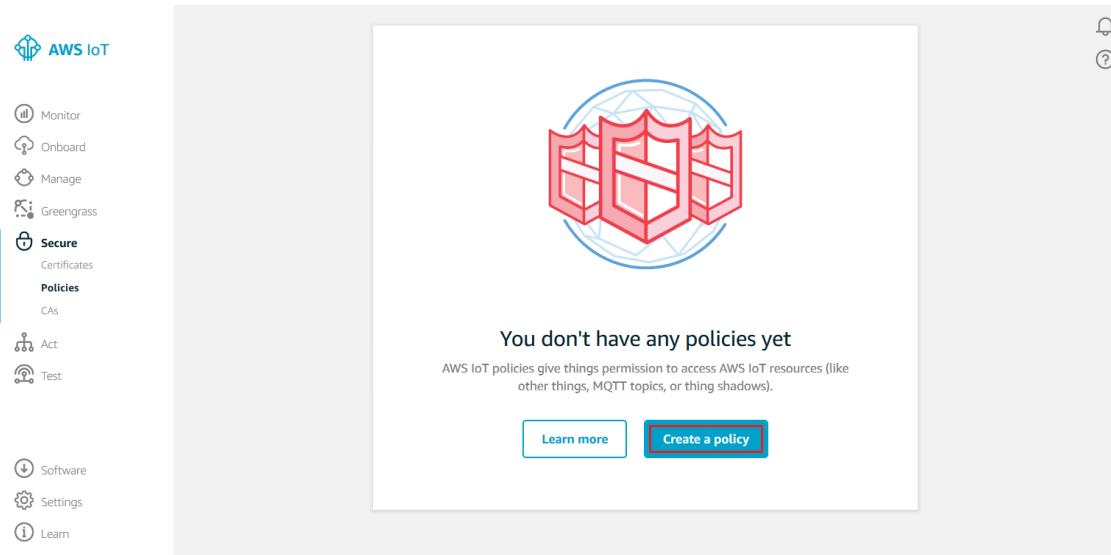
4. Seleccione Done.

## Creación de una política de AWS IoT

Los certificados X.509 se utilizan para autenticar su dispositivo en AWS IoT. Las políticas de AWS IoT se utilizan para autorizar a su dispositivo a ejecutar operaciones de AWS IoT, como suscribirse a temas MQTT o publicar en ellos. Su dispositivo presentará su certificado cuando envíe mensajes a AWS IoT. Para permitir a su dispositivo llevar a cabo operaciones de AWS IoT, debe crear una política de AWS IoT y asociarla al certificado de su dispositivo.

#### Para crear una política de AWS IoT

1. En el panel de navegación de la izquierda, seleccione Secure y, a continuación, elija Policies. En la página You don't have a policy yet, elija Create a policy.



2. En la página Create a policy, en el campo Name, escriba un nombre para la política (por ejemplo, **MyIoTButtonPolicy**). En el campo Action, escriba iot:Connect. En el campo Resource ARN, escriba \*. Seleccione la casilla Allow. Esto permite que todos los clientes se conecten a AWS IoT.

## Create a policy

Create a policy to define a set of authorized actions. You can authorize actions on one or more resources (things, topics, topic filters).

Name

MyIoTButtonPolicy

### Add statements

Policy statements define the types of actions that can be performed by a resource.

**Advanced mode**

Action

iot:Connect

Resource ARN

\*

Effect

Allow  Deny

Remove

Add statement

#### Note

Puede restringir qué clientes (dispositivos) pueden conectarse, especificando un ARN de cliente como recurso. Los ARN de cliente tienen este formato:

`arn:aws:iot:<your-region>:<your-aws-account>:client/<my-client-id>`

Seleccione el botón Add Statement para agregar otra instrucción de política. En el campo Action, escriba iot:Publish. En el campo Resource ARN, escriba el ARN del tema en el que su dispositivo va a publicar.

#### Note

El ARN de tema tiene este formato:

`arn:aws:iot:<your-region>:<your-aws-account>:topic/iotbutton/<your-button-serial-number>`

Por ejemplo:

`arn:aws:iot:us-east-1:123456789012:topic/iotbutton/G030JF055364XVRB`  
El número de serie se encuentra en la parte inferior del botón.

Si no utiliza un botón AWS IoT, coloque el tema en el que publica su dispositivo después de topic/ en el ARN. Por ejemplo:

`arn:aws:iot:us-east-1:123456789012:topic/my/topic/here`

Por último, marque la casilla Allow. Esto permite a su dispositivo publicar mensajes en el tema especificado.

- Una vez que haya especificado la información de su política, elija Create.

The screenshot shows the 'Create a policy' interface in the AWS IoT console. The policy name is 'MyIoTButtonPolicy'. A single statement is being defined with the following details:

- Action:** iot:Publish
- Resource ARN:** arn:aws:iot:us-east-1:...:topic/iotbutton/G030JF053216F1BS
- Effect:**  Allow  Deny

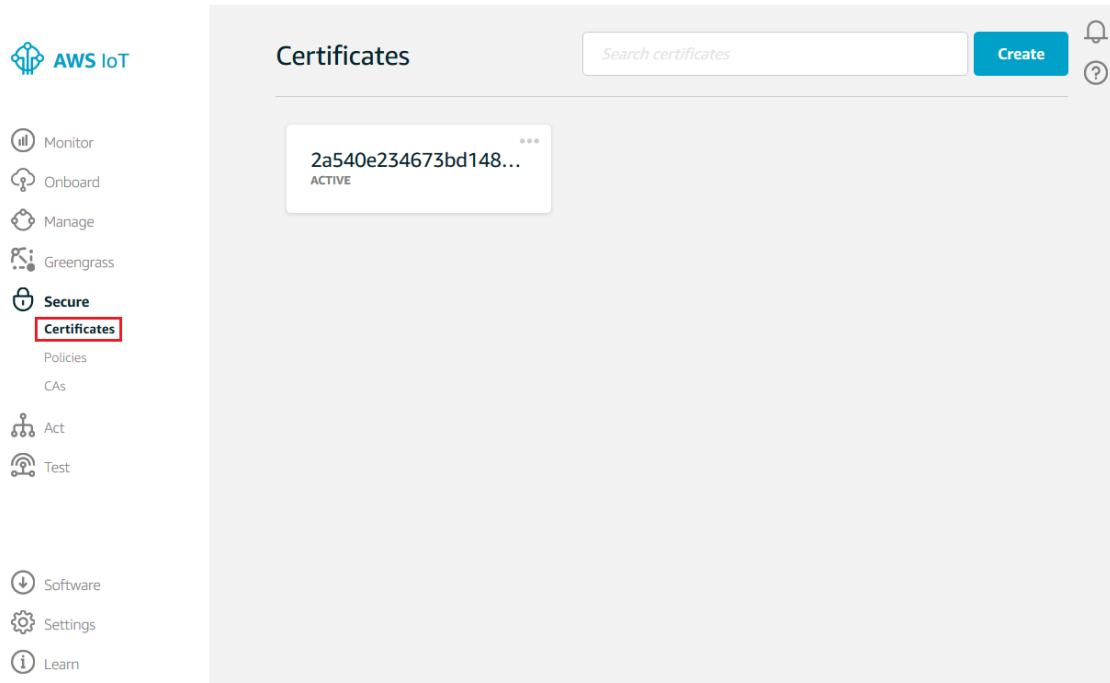
At the bottom right of the statement editor is a 'Remove' button. Below the statement editor is an 'Add statement' button. In the bottom right corner of the entire screen is a large blue 'Create' button.

Para obtener más información, consulte la sección acerca de [administración de políticas de AWS IoT](#).

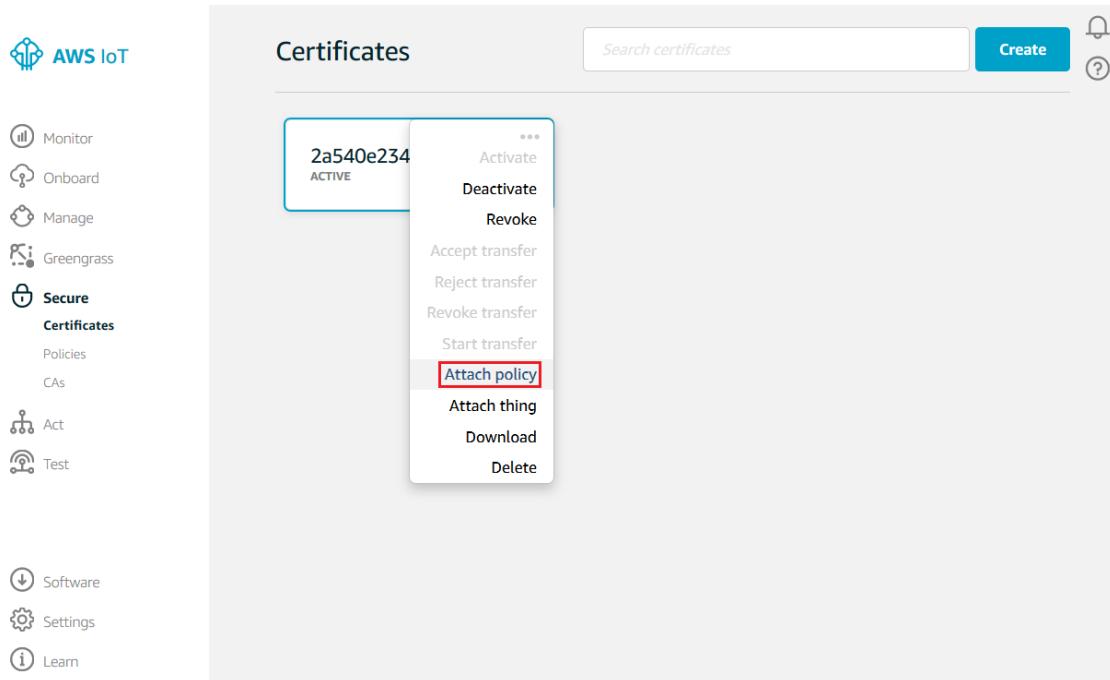
## Asociación de una política de AWS IoT a un certificado de dispositivo

Ahora que ha creado una política, debe asociarla a su certificado de dispositivo. Si asocia una política de AWS IoT a un certificado, dará al dispositivo los permisos especificados en la política.

1. En el panel de navegación de la izquierda, seleccione Secure y, a continuación, elija Certificates.



2. En la casilla del certificado que ha creado, elija ... para abrir un menú desplegable y, a continuación, elija Attach policy.



3. En el cuadro de diálogo **Attach policies to certificate(s)**, marque la casilla de verificación junto a la política que ha creado en el paso anterior y, a continuación, seleccione **Attach**.

The screenshot shows the 'Attach policies to certificate(s)' dialog box. It displays the message: 'Policies will be attached to the following certificate(s): 2a540e234673bd148f1e710ef6529602d66eaf638b3ee120a493082e4a5f11d7'. Below this, it says 'Choose one or more policies' and shows a search bar with 'Search policies' and a list item 'MyIoTButtonPolicy' with a checked checkbox. At the bottom right of the dialog are 'Cancel' and 'Attach' buttons, with 'Attach' being highlighted with a red box.

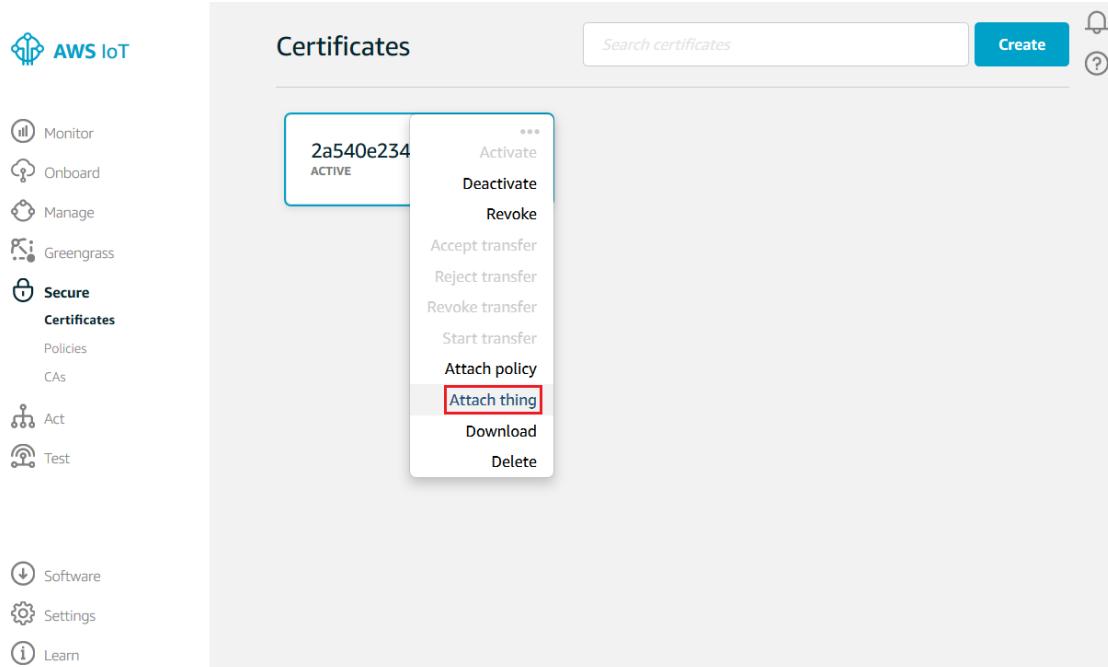
## Asociación de un certificado a un objeto

Un dispositivo debe tener un certificado, una clave privada y un certificado de CA raíz para autenticarse en AWS IoT. También le recomendamos asociar el certificado del dispositivo al objeto que representa a su dispositivo en AWS IoT. Esto le permite crear políticas de AWS IoT que concedan permisos según los

certificados asociados a sus objetos. Para obtener más información, consulte [Variables de la política de objeto \(p. 117\)](#)

Para asociar un certificado al objeto que representa su dispositivo en el registro de objetos:

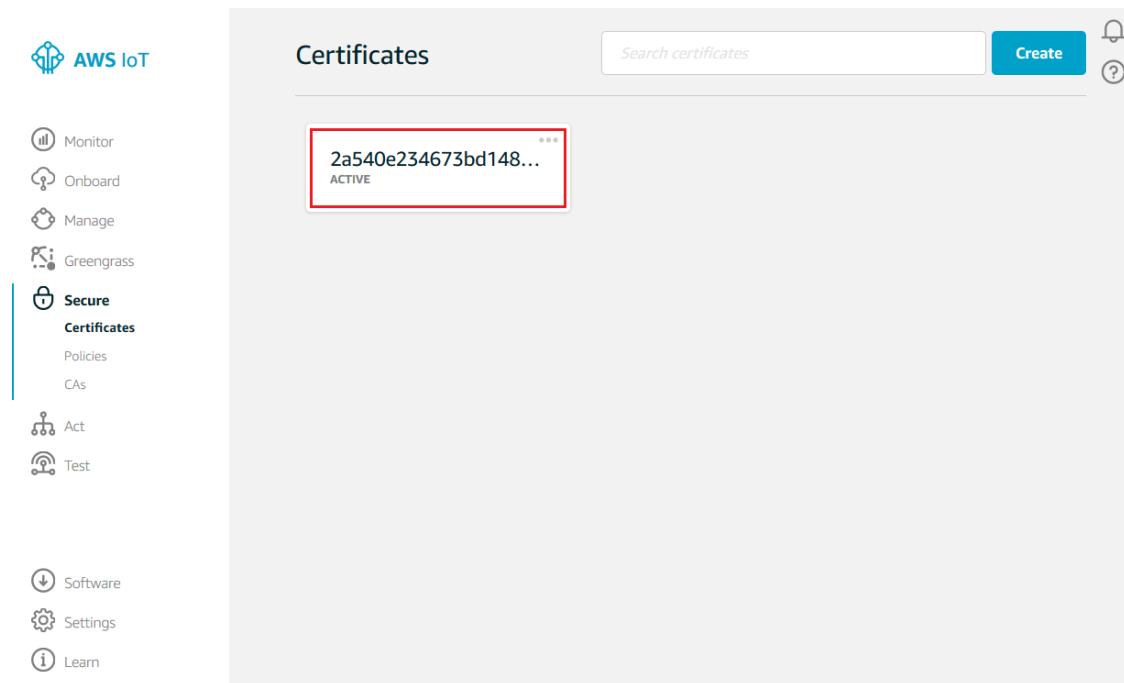
1. En la casilla del certificado que ha creado, elija ... para abrir un menú desplegable y, a continuación, elija Attach thing.



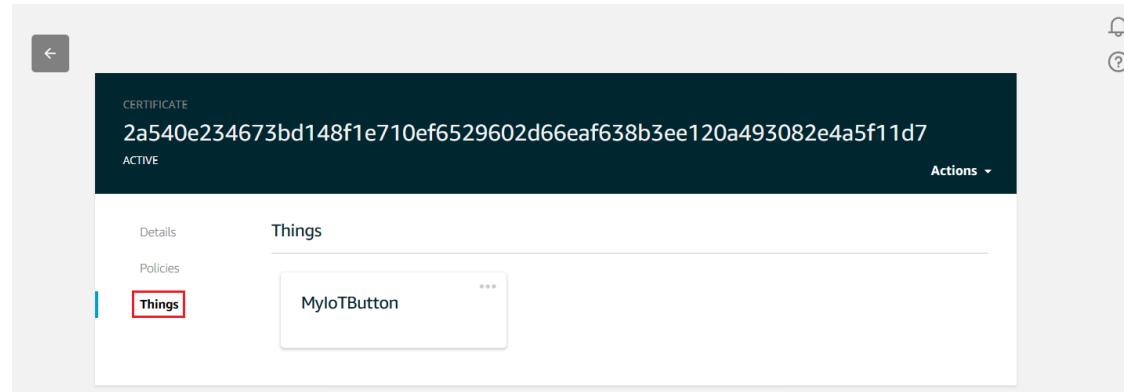
2. En el cuadro de diálogo Attach things to certificate(s) , marque la casilla situada junto al objeto que ha registrado y, a continuación, elija Attach.

The screenshot shows the 'Attach things to certificate(s)' dialog box. It contains the message: 'Things will be attached to the following certificate(s): 2a540e234673bd148f1e710ef6529602d66eaf638b3ee120a493082e4a5f11d7'. Below this is a section titled 'Choose one or more things' with a search bar labeled 'Search things' and a list containing 'MyloTButton' with a checked checkbox. At the bottom right of the dialog are buttons for 'Cancel' and 'Attach'.

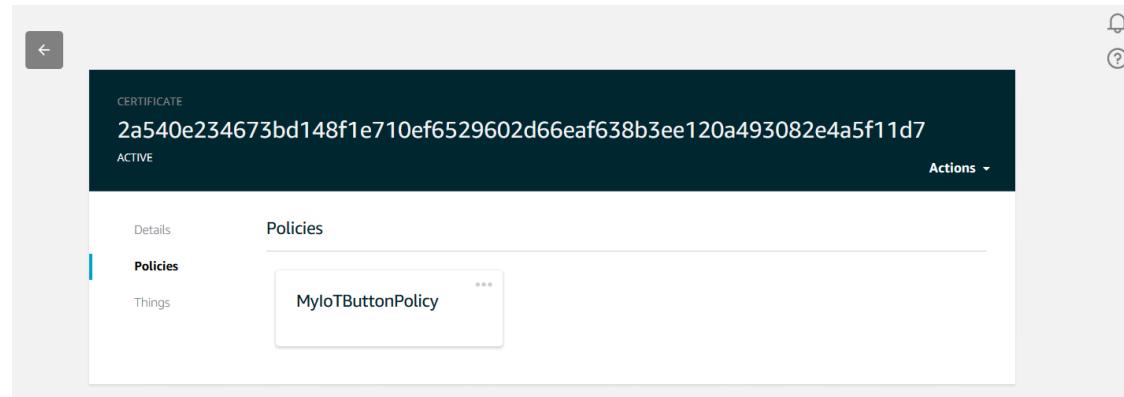
3. Para verificar que el objeto esté asociado, seleccione la casilla que representa el certificado.



4. En el panel de navegación izquierdo de la página Details del certificado, elija Things.



5. Para verificar que la política se haya asociado, en el panel de navegación izquierdo de la página Details del certificado, elija Policies.



# Configuración del dispositivo

La configuración del dispositivo permite conectarlo a su red wifi. Su dispositivo debe estar conectado a la red wifi para instalar los archivos requeridos y enviar mensajes a AWS IoT. Todos los dispositivos deben instalar un certificado de dispositivo, la clave privada y el certificado de CA raíz para comunicarse con AWS IoT.

## Configuración de un botón AWS IoT

La forma más sencilla de configurar el botón AWS IoT es utilizar la aplicación del botón AWS IoT para smartphones. Puede descargarla en [Apple App Store](#) o [Google Play Store](#). Si no puede utilizar la aplicación para smartphones, siga estas instrucciones para configurar el botón:

Para configurar el botón AWS IoT:

### Active su dispositivo

1. Extraiga el botón AWS IoT de su embalaje y, a continuación, manténgalo pulsado hasta que aparezca una luz azul parpadeante. (Esta operación no debe tardar más de 15 segundos).
2. El botón sirve de punto de acceso wifi, por lo que cuando su equipo busque redes wifi, encontrará una red denominada Button ConfigureMe - XXX, donde XXX es una cadena de tres caracteres generada por el botón. Utilice su equipo para conectarse con el punto de acceso wifi del botón.

#### Note

Cuando la luz azul deje de parpadear, el botón ya no aparece como un punto de acceso wifi. Por lo tanto, si no puede completar el siguiente procedimiento lo suficientemente rápido, es posible que deba invocar la luz azul que parpadea varias veces para hacerlo. Una vez configurado, no es necesario que el dispositivo se presente a sí mismo como un punto de acceso wifi, y se comunica con Internet como cualquier otro equipo a través de la red wifi local.

3. La primera vez que se conecte al punto de acceso wifi del botón, se le solicitará que indique la contraseña WPA2-PSK. Escriba los últimos 8 caracteres del número de serie del dispositivo (DSN). Encontrará el DSN en la parte posterior del dispositivo, tal y como se muestra aquí:



### Copie el certificado del dispositivo y la clave privada en su botón AWS IoT.

Para conectarse a AWS IoT, debe copiar la clave privada y el certificado del dispositivo en el botón AWS IoT.

1. En un navegador, vaya a <http://192.168.0.1/index.html>.
2. Rellene el formulario de configuración:
  - Escriba el SSID y la contraseña de la red wifi.
  - Busque y seleccione el certificado y la clave privada. Por ejemplo, 2a540e2346-certificate.pem.crt.txt y 2a540e2346-private.pem.key, respectivamente.

- Busque el punto de enlace personalizado en la [consola de AWS IoT](#). (En el panel de navegación de la izquierda, elija Manage y, a continuación, Things. Seleccione la casilla que representa su botón para mostrar su página de detalles. En el panel de navegación izquierdo de la página de detalles, seleccione Interact y busque la sección HTTPS, cerca de la parte superior). Su punto de enlace tendrá un aspecto similar al siguiente:

ABCDEF1234567.iot.us-east-2.amazonaws.com

donde ABCDEF1234567 es el subdominio y us-east-2 es la región.

- En la página Button ConfigureMe, escriba el subdominio y, a continuación, elija la región que coincide con la región de su punto de enlace de AWS IoT.
- Marque la casilla de verificación de Terms and Conditions. Ahora, su configuración debe tener el aspecto siguiente:

**Button ConfigureMe**

Enter the value for any field that you wish to change for device: G030JF055364XVRB

**Wi-Fi Configuration:**

SSID	Guest
Security	<input checked="" type="checkbox"/> Open Network(No Password)
Password	None (Unsecured)

**AWS IoT Configuration:**

Certificate	<a href="#">Choose File</a> MyIoTButtonCert.pem
Private Key	<a href="#">Choose File</a> MyIoTButtonKey.pem
Endpoint Subdomain	AMUN9F6MTZ77O
Endpoint Region	<a href="#">Choose Region</a>
Final Endpoint	AMUN9F6MTZ77O.iot.us-east-1.amazonaws.com

By clicking this box, you agree to the [AWS IoT Button Terms and Conditions](#).

[Configure](#)

- Elija Configure. Ahora, su botón debe poder conectarse a la red wifi.

## Configuración de otro dispositivo

Consulte la documentación de su dispositivo para saber cómo conectarse a él y copiar el certificado de dispositivo, la clave privada y el certificado de CA raíz.

## Visualización de los mensajes MQTT de dispositivo con el cliente MQTT de AWS IoT

Puede utilizar el cliente MQTT de AWS IoT para comprender mejor los mensajes MQTT enviados por un dispositivo.

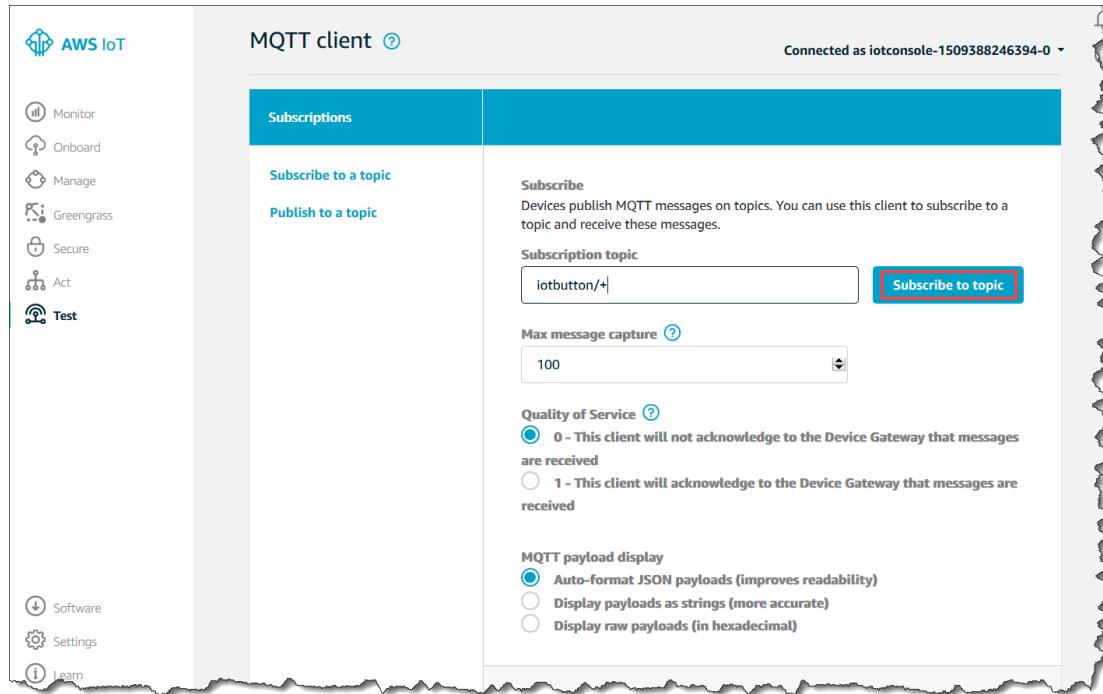
Los dispositivos publican los mensajes MQTT en temas. Puede utilizar el cliente MQTT de AWS IoT para suscribirse a estos temas para ver los mensajes.

Para ver los mensajes MQTT:

- En el panel de navegación izquierdo de la [consola de AWS IoT](#), elija Test.



2. Suscríbase al tema en el que publica su objeto. En el caso del botón AWS IoT, puede suscribirse a **iotbutton/+** (tenga en cuenta que **+** es el comodín). En **Subscribe to a topic**, en el campo **Subscription topic** escriba **iotbutton/+y**, a continuación, elija **Subscribe to topic**.



Si elige **Subscribe to topic** anteriormente, aparecerá el tema **iotbutton/+** en la columna **Subscriptions**.

AWS IoT Guía para desarrolladores  
Visualización de los mensajes MQTT de dispositivo con el cliente MQTT de AWS IoT

The screenshot shows the AWS IoT MQTT client interface. On the left, there's a sidebar with icons for Monitor, Onboard, Manage, Greengrass, Secure, Act, Test, Software, Settings, and Learn. The main area is titled "MQTT client" and shows a subscription to "iotbutton/+". In the "Publish" section, a message is being sent to the topic "iotbutton/+". The message content is displayed in a code editor-like box:

```
1 {
2   "message": "Hello from AWS IoT console"
3 }
```

3. Pulse el botón AWS IoT y, a continuación, consulte el mensaje que se obtiene en el cliente MQTT de AWS IoT. Si no dispone de botón, simulará que pulsa un botón en el paso siguiente.

The screenshot shows the AWS IoT MQTT client interface. On the left, there's a sidebar with icons for Monitor, Onboard, Manage, Greengrass, Secure, Act, Test, Software, Settings, and Learn. The main area is titled "MQTT client" and shows a subscription to "iotbutton/+". In the "Publish" section, a message has been sent to the topic "iotbutton/+". The message content is displayed in a code editor-like box:

```
1 {
2   "message": "Hello from AWS IoT console"
3 }
```

Below the publish section, there's a message received from the topic "iotbutton/G030JF055364XVRB" at Oct 31, 2017 3:24:41 PM -0700. The message content is:

```
{
  "serialNumber": "G030JF055364XVRB",
  "batteryVoltage": "1666mV",
  "clickType": "SINGLE"
}
```

#### Note

En la sección [AWS IoT Button FAQs](#) encontrará información útil sobre el patrón de colores del botón LED.

4. Para publicar un mensaje mediante la consola de AWS IoT:

En la página del cliente MQTT, en la sección Publish, campo Specifiy a topic and a message to publish..., escriba **iotbutton/ABCDEFG12345**. En la sección de carga del mensaje, escriba el siguiente JSON:

```
{
  "serialNumber": "ABCDEFG12345",
  "clickType": "SINGLE",
  "batteryVoltage": "2000 mV"
}
```

Elija Publish to topic . El mensaje debería aparecer en el cliente MQTT de AWS IoT (elija iotbutton/+ en la columna Subscription para ver el mensaje).



## Configuración y comprobación de reglas

El motor de reglas AWS IoT escucha los mensajes MQTT de entrada que coinciden con una regla. Cuando se recibe un mensaje que coincide, la regla actúa con los datos del mensaje MQTT (por ejemplo, escribe datos en un bucket de Amazon S3, invoca una función Lambda o envía un mensaje a un tema de Amazon SNS). En este paso, creará y configurará una regla para enviar los datos recibidos desde un dispositivo a un tema de Amazon SNS. En concreto, podrá:

- Crear un tema de Amazon SNS.
- Suscribirse al tema de Amazon SNS utilizando un número de teléfono móvil.
- Crear una regla que enviará un mensaje al tema de Amazon SNS al recibir un mensaje de su dispositivo.
- Probar la regla con el botón AWS IoT o un cliente MQTT.

En la esquina superior derecha de esta página, hay una lista desplegable de Filter View. Para obtener instrucciones a fin de probar la regla con el botón AWS IoT, elija AWS IoT Button. Para obtener instrucciones a fin de probar la regla mediante el cliente MQTT de AWS IoT, elija MQTT Client.

## Creación de un tema de SNS

Se utilizará la consola de Amazon SNS para crear un tema de Amazon SNS.

### Note

Amazon SNS no está disponible en todas las regiones de AWS.

1. Abra la [consola de Amazon SNS](#).
2. En el panel de la izquierda, elija Topics.

The screenshot shows the AWS SNS dashboard. On the left, there's a sidebar with links: SNS dashboard, Topics (which is selected and highlighted in red), Applications, Subscriptions, and Text messaging (SMS). The main content area has a header "SNS dashboard". Under "Common actions", there are five items: "Create topic" (with a link to "Create a communication channel to send messages and subscribe to notifications"), "Create platform application" (with a link to "Create a platform application for mobile devices"), "Create subscription" (with a link to "Subscribe an endpoint to a topic to receive messages published to that topic"), "Publish message" (with a link to "Publish a message to a topic or as a direct publish to a platform endpoint"), and "Publish text message (SMS)" (with a link to "Publish a text message to a phone number"). To the right, under "Resources", it says "You are using the following Amazon SNS resources in the us-west-2 region:" followed by a table with four rows: Topic (0), Subscriptions (0), Applications (0), and Endpoints (0). Below that is a "More info" section with links to Getting started, Documentation, API reference, Forums, and Service health.

3. Elija Create new topic.

The screenshot shows the "Topics" page. The sidebar on the left includes: SNS dashboard, Topics (selected), Applications, Subscriptions, and Text messaging (SMS). The main area has a header "Topics" and contains several buttons at the top: "Publish to topic", "Create new topic" (which is highlighted with a red border), and "Actions". Below these are "Filter" and "Actions" dropdown menus. A table header row shows columns for "Name" and "ARN". At the bottom of the page, there's a summary: "Total Items: 0" and "Selected Items: 0".

4. Escriba un nombre de tema y otro de visualización y, a continuación, seleccione Create topic.

The screenshot shows the 'Create new topic' dialog box. It has two input fields: 'Topic name' containing 'MyIoTButtonSNSTopic' and 'Display name' containing 'IoT Button'. Below the fields are 'Cancel' and 'Create topic' buttons.

5. Anote el ARN del tema que acaba de crear.

The screenshot shows the 'Topics' list page. It displays a single topic named 'MyIoTButtonSNSTopic' with the ARN 'arn:aws:sns:us-west-2:123456789012:MyIoTButtonSNSTopic'. There are buttons for 'Publish to topic', 'Create new topic', and 'Actions'.

## Suscripción a un tema de Amazon SNS

Para recibir SMS en su teléfono móvil, debe suscribirse al tema de Amazon SNS.

1. En la consola de Amazon SNS, seleccione la casilla situada junto al tema que acaba de crear. En el menú Actions, seleccione Subscribe to topic.

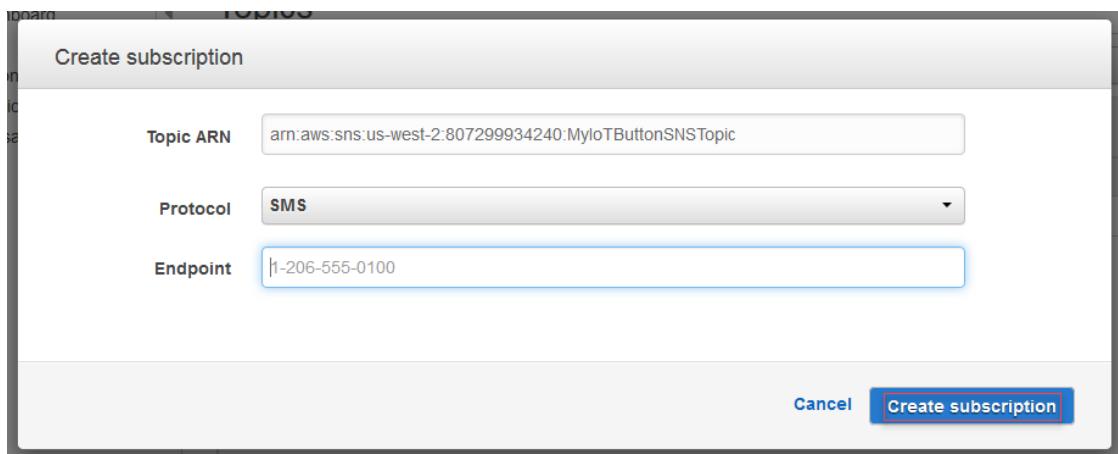
The screenshot shows the 'Topics' list page with a context menu open over the 'MyIoTButtonSNSTopic' row. The menu includes options like 'Edit topic display name', 'Subscribe to topic' (which is highlighted with a red border), 'Confirm a subscription', 'Edit topic policy', 'Edit topic delivery policy', 'Delivery status', and 'Delete topics'.

2. En Create subscription, en la lista desplegable Protocol, elija SMS.

En el campo Endpoint, escriba el número de un teléfono móvil habilitado para SMS y, a continuación, elija Create subscription.

### Note

Escriba el número de teléfono utilizando únicamente números y guiones.

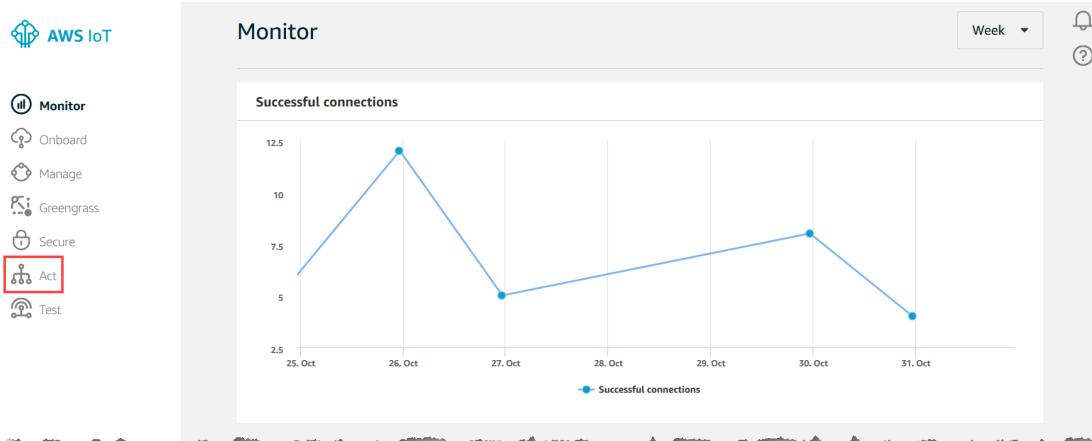


## Creación de una regla

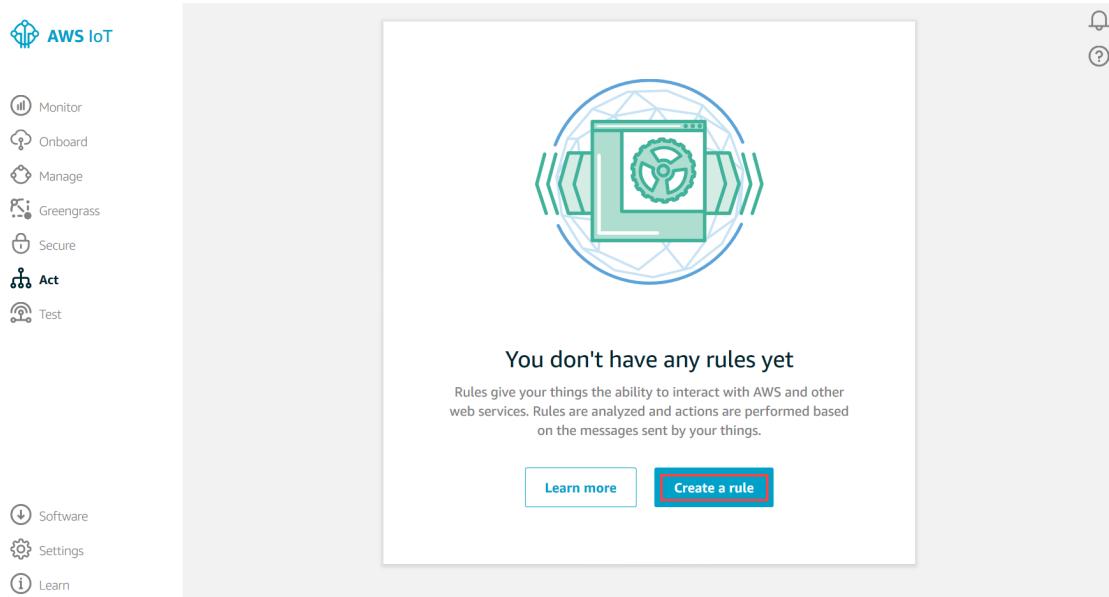
Las reglas de AWS IoT se componen de un filtro de temas, una acción de regla y, en la mayoría de los casos, un rol de IAM. Los mensajes publicados en temas que coinciden con el filtro de temas activan la regla. La acción de la regla define qué acción ejecutar cuando se activa la regla. El rol de IAM contiene una o varias políticas de IAM que determinan a qué servicios de AWS puede tener acceso la regla. Puede crear varias reglas que escuchan un único tema. Del mismo modo, puede crear una única regla que se activa con varios temas. El motor de reglas de AWS IoT procesa constantemente los mensajes publicados en temas que coinciden con los filtros de temas definidos en las reglas.

En este ejemplo, se creará una regla que utiliza Amazon SNS para enviar una notificación por SMS a un teléfono móvil.

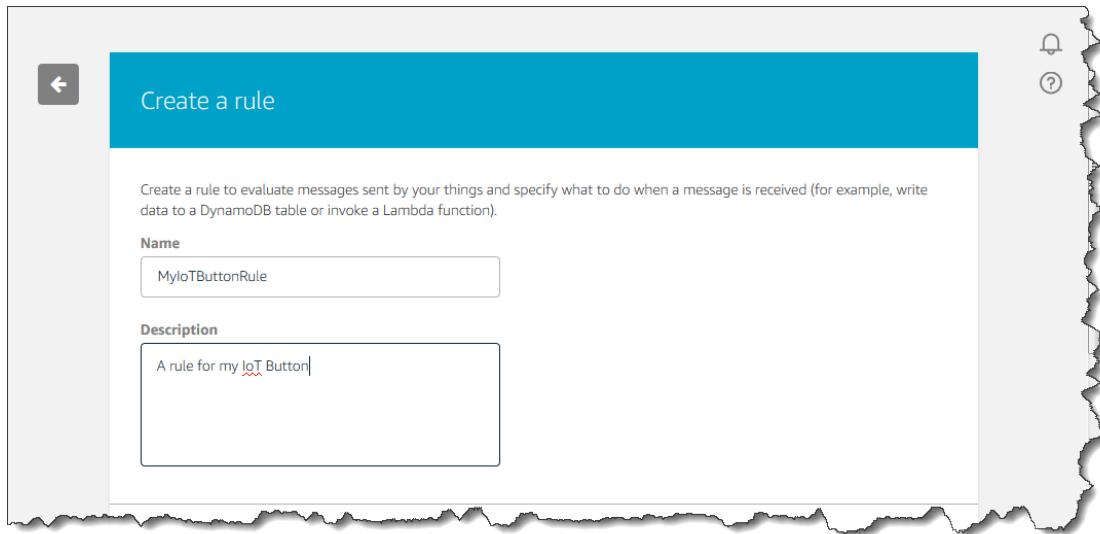
1. En la consola de AWS IoT, en el panel de navegación izquierdo, elija Act.



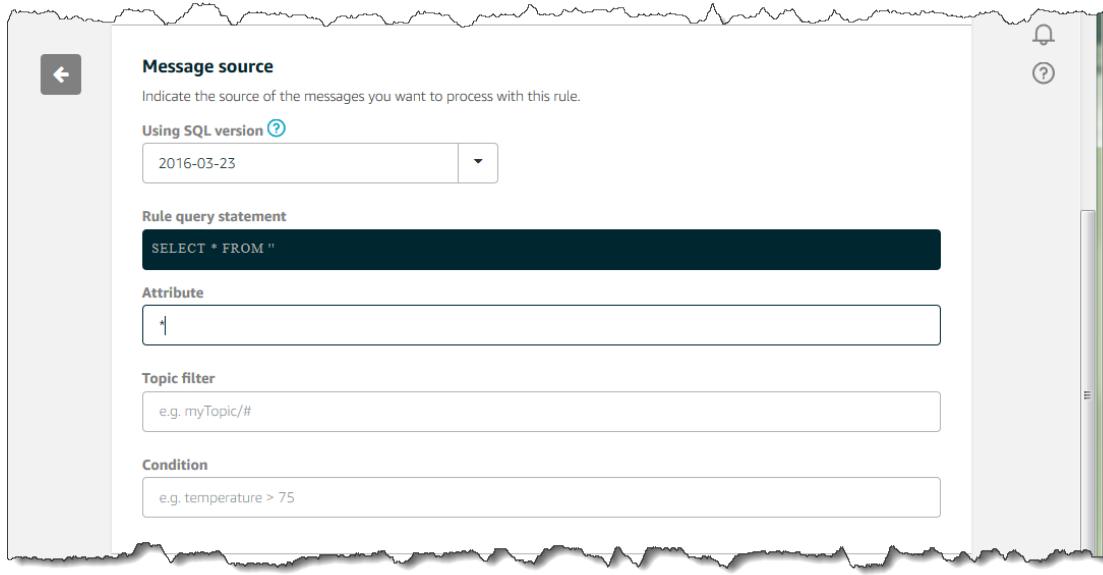
2. En la página Act, seleccione Create a rule.



3. En la página Create a rule, en el campo Name, escriba un nombre para la regla. En el campo Description, escriba una descripción para la regla.



4. Desplácese hasta Message source. Seleccione la última versión en la lista desplegable de Using SQL version. En el campo Attribute, escriba \*. Esto especifica que desea enviar todo el mensaje MQTT que ha activado la regla.

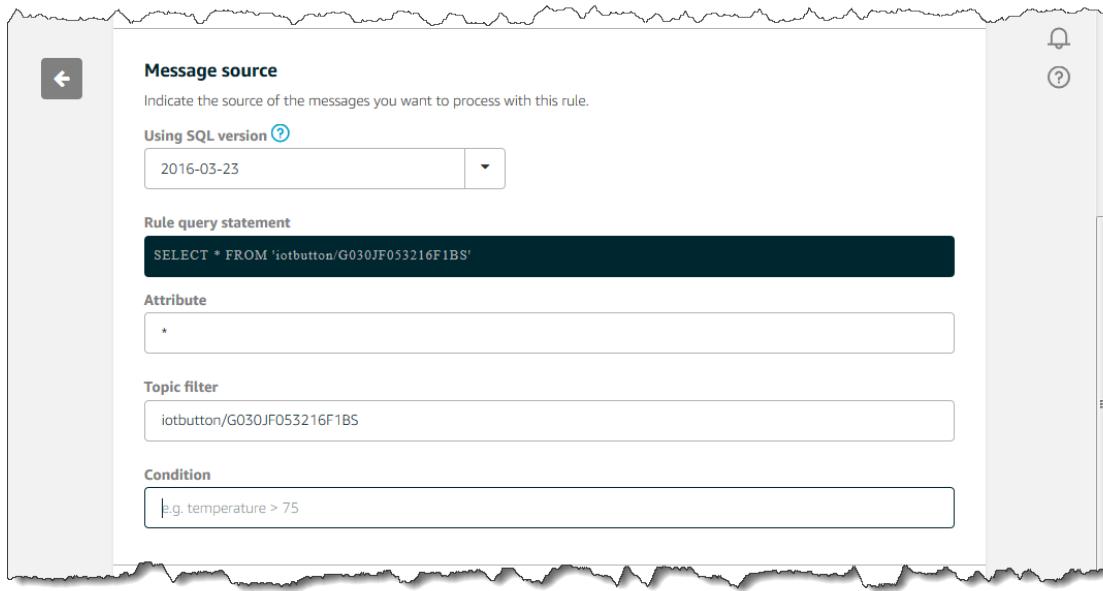


5. El motor de reglas utiliza el filtro de temas para determinar qué reglas deben activarse cuando se recibe un mensaje MQTT. En el campo Topic filter, escriba **iotbutton/DSN de su botón**. Si no utiliza un botón AWS IoT, escriba **my/topic** o el tema utilizado en la regla.

**Note**

Puede encontrar el DSN en la parte inferior del botón.

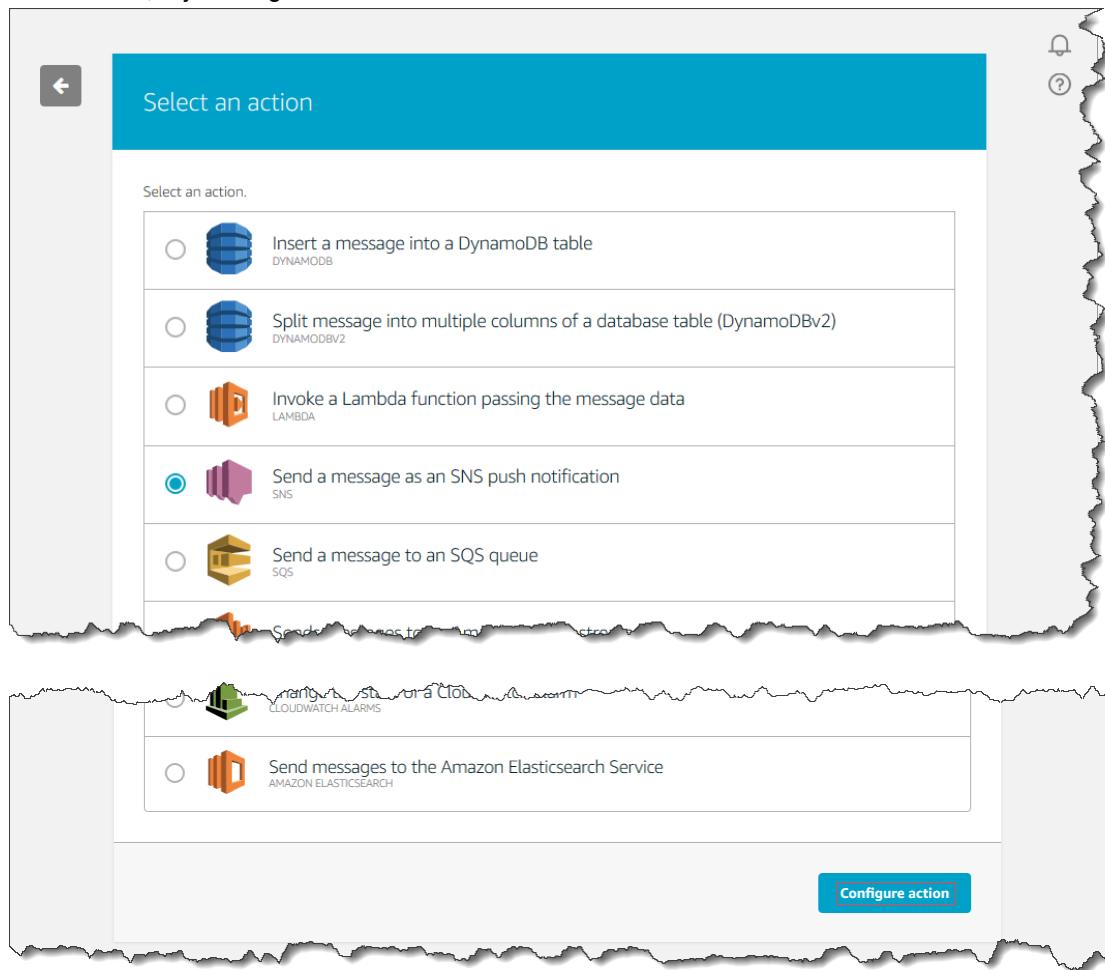
Deje Condition en blanco.



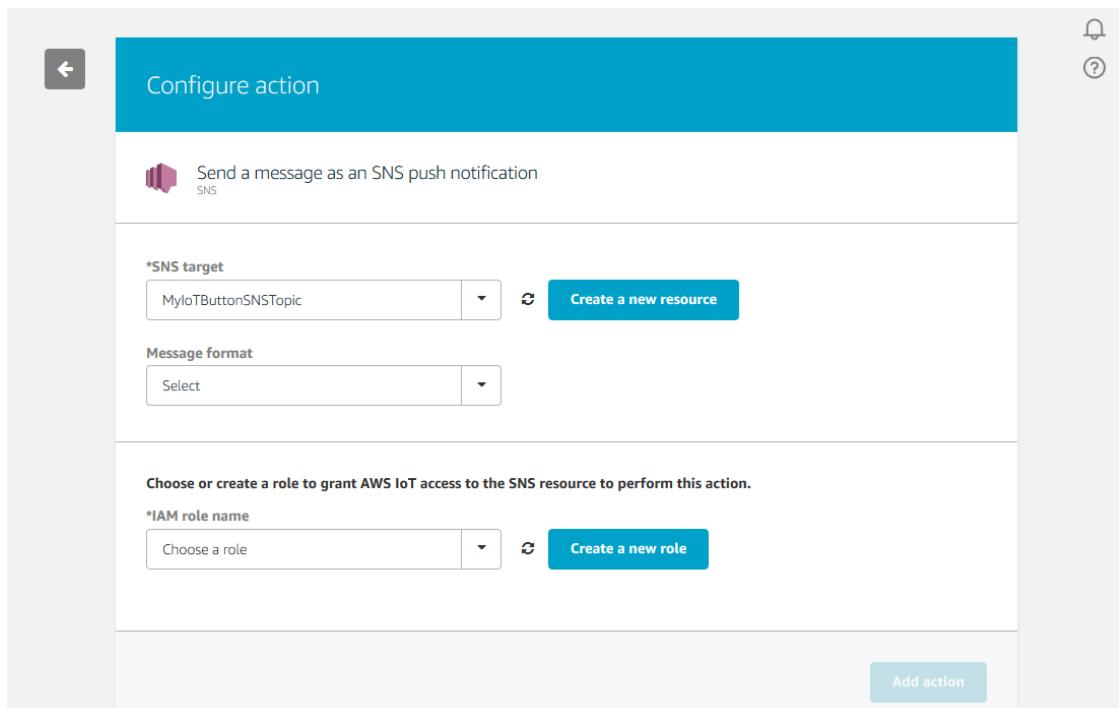
6. En Set one or more actions, elija Add action.



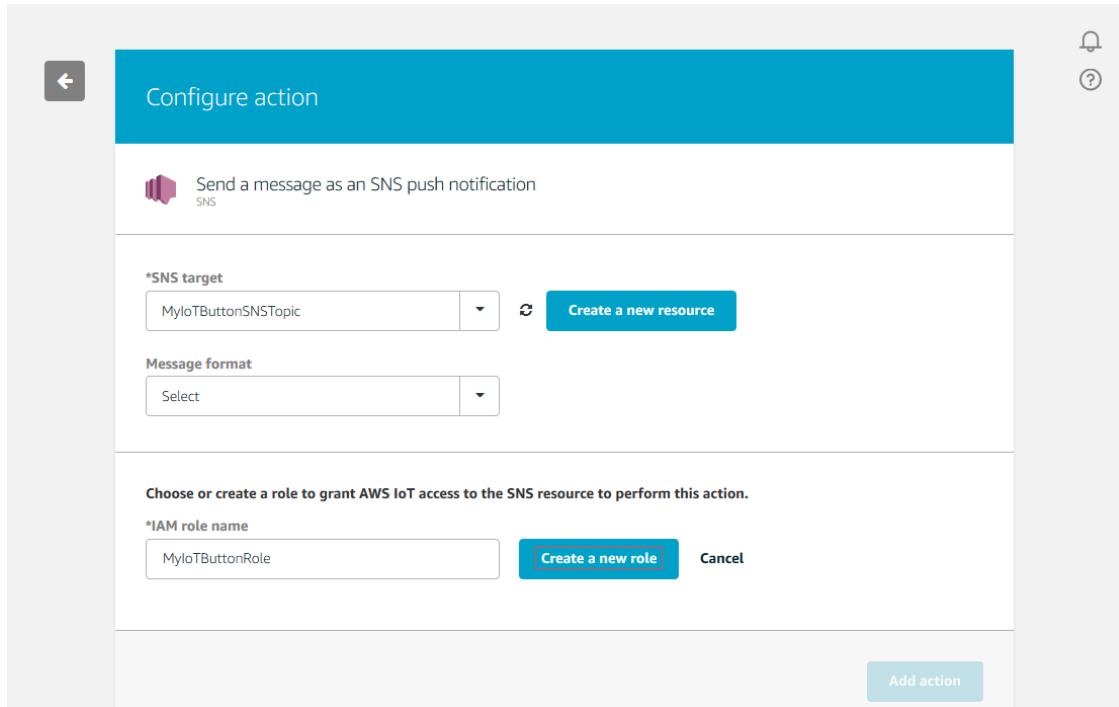
7. En la página Select an action, seleccione Send a message as an SNS push notification y, a continuación, elija Configure action.



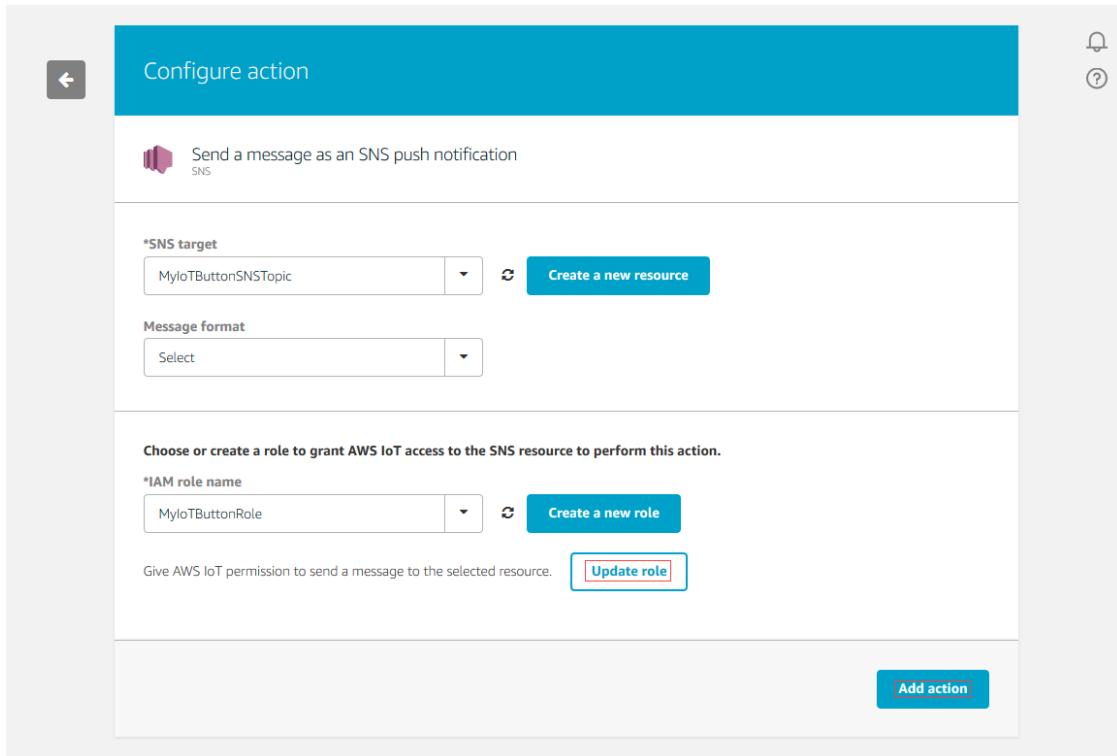
8. En la página Configure action, en la lista desplegable de SNS target, seleccione el tema de Amazon SNS que ha creado anteriormente.



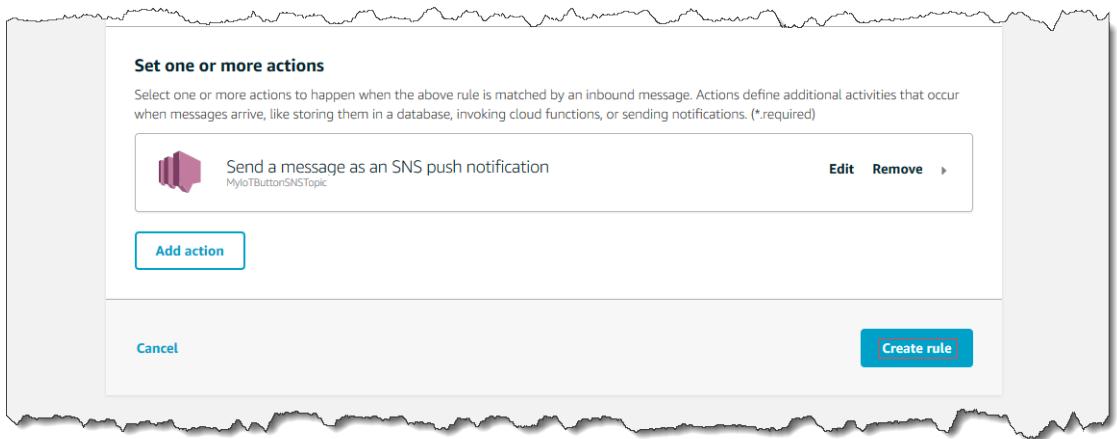
9. Ahora, debe dar permiso a AWS IoT para publicar en el tema de Amazon SNS en su nombre cuando se active la regla. Elija Create a new role. Escriba un nombre para el nuevo rol en el campo IAM role name . Una vez que haya escrito el nombre, vuelva a seleccionar Create a new role. En la lista desplegable de IAM role name , seleccione el rol que acaba de crear.



10. Elija Update role para aplicar los permisos en el rol recién creado y, a continuación, seleccione Add action.



11. En la página Create a Rule, seleccione Create rule.



Para obtener más información sobre la creación de reglas, consulte [Reglas de AWS IoT](#).

## Comprobación de la regla de Amazon SNS

Puede probar la regla utilizando un botón AWS IoT o el cliente MQTT de AWS IoT.

### Botón AWS IoT

Pulse el botón. Debería recibir un SMS que muestre el nivel de carga de la batería de su dispositivo (entre otras cosas). Mantenga pulsado (unos 2 segundos), presione rápidamente dos veces y anote los mensajes obtenidos.

## Cliente MQTT de AWS IoT

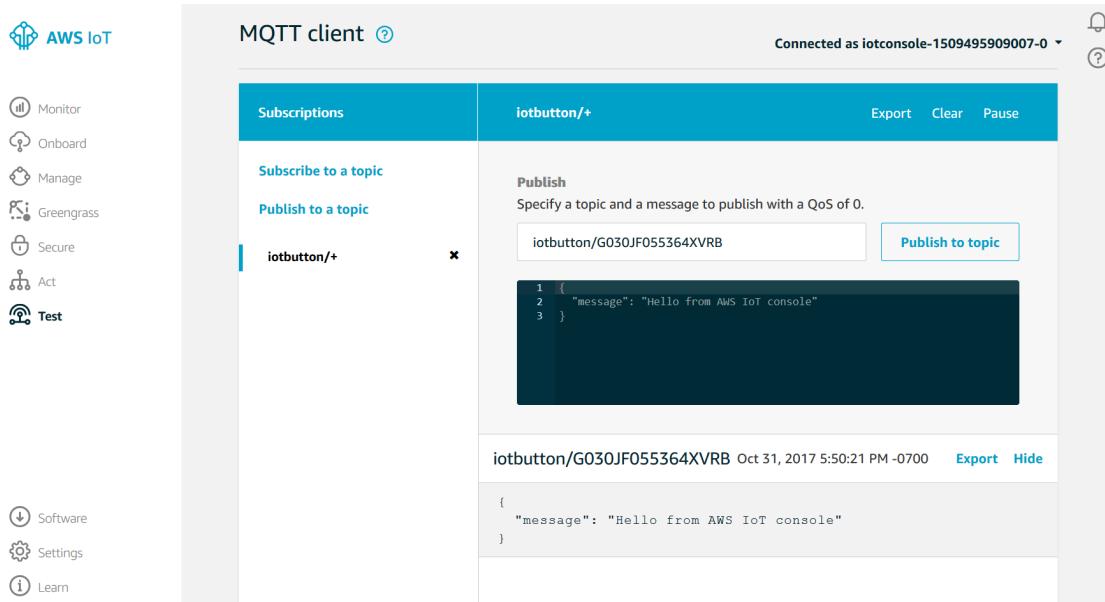
Para probar su regla con el cliente MQTT de AWS IoT:

1. En el panel de navegación izquierdo de la [consola de AWS IoT](#), elija Test.
2. En la página de cliente MQTT, en la sección Publish, campo Specify a topic and a message to publish, escriba **my/topic** o el tema que ha utilizado en la regla. En la sección de carga del mensaje, escriba el siguiente JSON:

```
{  
    "message": "Hello, from AWS IoT console"  
}
```

### Note

Si está utilizando un botón, escriba **iotbutton/DNS de su botón** en lugar de **my/topic** en el campo Specify a topic and a message to publish.



3. Elija Publish to topic . Debería recibir un mensaje de Amazon SNS en su teléfono móvil.

¡Enhorabuena! Ha creado y configurado correctamente una regla que envía los datos recibidos de un dispositivo a un tema de Amazon SNS.

## Pasos siguientes

Para obtener más información sobre las reglas de AWS IoT, consulte [Tutoriales de reglas de AWS IoT \(p. 46\)](#) y [Reglas de AWS IoT \(p. 148\)](#).

# Inicios rápidos del botón AWS IoT

Las dos inicios rápidos de esta sección muestran cómo configurar y utilizar el botón AWS IoT. Puede utilizar el asistente para el botón AWS IoT de la consola de AWS Lambda para configurar de forma rápida y sencilla su botón AWS IoT. La consola de AWS Lambda contiene un plan que automatizará el proceso de configuración de su botón AWS IoT:

- Creación y activación de un certificado X.509 y una clave privada para la autenticación en AWS IoT.
- Orientación para configurar su botón AWS IoT y conectarse a su red wifi.
- Orientación para copiar su certificado y la clave privada de su botón AWS IoT.
- Creación de una política de AWS IoT que dé al botón permiso para realizar llamadas a AWS IoT y asociación de dicha política al certificado.
- Creación de una regla de AWS IoT que invoque una función Lambda cuando se pulse el botón AWS IoT.
- Creación de un rol y una política de IAM que permita a la función Lambda enviar mensajes de correo electrónico mediante Amazon SNS.
- Creación de una función Lambda que envíe un mensaje de correo electrónico a la dirección especificada en el código de la función Lambda.

También puede configurar el botón AWS IoT mediante una plantilla AWS CloudFormation. El segundo inicio rápido le muestra cómo configurar los recursos de AWS IoT necesarios para procesar los mensajes MQTT que se envían al pulsar el botón AWS IoT, mediante una plantilla de AWS CloudFormation.



Si no dispone de un botón, puede adquirir uno [aquí](#). Para obtener más información acerca de AWS IoT, consulte [Qué es AWS IoT \(p. 1\)](#).

Temas

- [Inicio rápido del asistente para el botón AWS IoT \(p. 32\)](#)
- [Inicio rápido AWS CloudFormation del botón AWS IoT \(p. 40\)](#)
- [Pasos siguientes \(p. 45\)](#)

## Inicio rápido del asistente para el botón AWS IoT

El asistente para el botón AWS IoT es un plan de Lambda, por lo que necesita iniciar sesión en la consola de AWS Lambda para poder utilizarlo. Si no tiene una cuenta de AWS, puede crearla siguiendo los pasos que se indican a continuación.

Para crear una cuenta de AWS

1. Abra la [página de inicio de AWS](#) y seleccione Abre una cuenta gratuita.
2. Siga las instrucciones en línea. Parte del procedimiento de inscripción consiste en recibir una llamada telefónica e introducir un número PIN con su teclado de teléfono.

Para configurar el botón AWS IoT

1. Inicie sesión en la consola de administración de AWS y abra la [consola de AWS Lambda](#).
2. Si es la primera vez que utiliza la consola de AWS Lambda, verá la siguiente página. Elija el botón Get Started Now.



## AWS Lambda

AWS Lambda is a compute service that runs developers' code in response to events and automatically manages the compute resources for them, making it easy to build applications that respond quickly to new information.

[Get Started Now](#)

[Learn more about AWS Lambda](#)

Si ya ha utilizado la consola de AWS Lambda, verá la página siguiente. Elija el botón Create a Lambda function.

Lambda

board BETA

tions

Lambda > Functions

You have 32 Lambda function(s) using 1.6 MB of code storage. Choose any Lambda function to view details on invocation requests, duration, and cost. (Costs take up to 60 seconds to appear).

Create a Lambda function Actions

Function name	Description	Runtime	Code size
myButtonFunction	An AWS Lambda function that sends an email on the click of an IoT button.	Node.js 4.3	1.7 kB
michgreFunction	A starter AWS Lambda function.	Node.js 4.3	851 bytes

3. En la página Select blueprint, el menú desplegable Runtime, elija Node.js 4.3. En el cuadro de texto de filtro, escriba **button**. Para elegir el proyecto **iot-button-email**, haga doble clic en él o seleccione el botón Next.

> New function

blueprint

ure triggers

ure function

/

## Select blueprint

Blueprints are sample configurations of event sources and Lambda functions. Choose a blueprint that best aligns with your needs, or skip this step if you want to author a Lambda function and configure an event source separately. Most blueprints are open source and can be customized as needed. Otherwise noted, blueprints are licensed under [CC0](#).

Welcome to AWS Lambda! You can get started on creating your first Lambda function by choosing one of the blueprints below.

Node.js 4.3

button

Viewing

iot-button-email

An AWS Lambda function that sends an email on the click of an IoT button.

nodejs · iot · button

4. En la página Configure triggers, en el menú desplegable IoT Type, elija IoT Button.

Escriba el número de serie de su dispositivo. Encontrará el número de serie del dispositivo (DSN) en la parte posterior del botón.

Elija Generate certificate and keys.

Note

Solo tiene que generar un certificado y una clave privada una vez. A continuación, puede ir a <http://192.168.0.1/index.html> en un navegador para configurar su botón.

ia > New function using blueprint iot-button-email

t blueprint  
Configure triggers  
Configure function  
w

## Configure triggers

Configure an optional trigger to automatically invoke your function.

AWS IoT → Lambda

Warning: Altering the description or SQL statement of an existing rule will overwrite it.

IoT Type: IoT Button

Device Serial Number: [REDACTED]

Generate certificate and keys



Utilice los enlaces de la página para descargar el certificado de dispositivo y la clave privada.

[Generate certificate and keys](#)



We have created the necessary AWS IoT resources (thing, policy, certificate, private key). The remaining resources (rule and action) will be created after your function is created.

**Download these resources by clicking the links below. (NOTE: If you are using Internet Explorer or Safari, right click the links to save the files.)**

- a. [Your certificate PEM](#)
- b. [Your private key](#)

**To configure the AWS IoT Button to use your Wi-Fi and these resources to connect to AWS securely, follow these steps:**

1. Place the button into configuration mode by pressing the button down for 5 seconds until it flashes blue.
2. Connect your computer to the button's Wi-Fi network SSID "Button ConfigureMe - FFD", using "5364XVRB" (last 8 digits of device serial number) as the WPA2-PSK password.
3. Click [here](#) (opens in new tab) and use the following information to fill out the form:
  - a. Enter your local network's Wi-Fi SSID and password.
  - b. Select the certificate and private key files that you just downloaded above.
  - c. Your endpoint subdomain is **a182jd32qs965e**.
  - d. Your endpoint region is **us-east-1**.
  - e. Check the box to agree to the terms and conditions.
  - f. Click "configure".
4. Re-connect to your original Wi-Fi network.

The button should stop blinking blue and you will see a white blinking light followed by a greed solid light. Your button is now configured to connect to the internet and AWS! Continue creating your function, and your button will be connected to it automatically.

La página también incluye instrucciones para configurar su botón AWS IoT. En el paso 3, elegirá un enlace para abrir una página web que le permitirá conectar el botón AWS IoT a la red. En Wi-Fi Configuration, escriba el ID de la red (SSID) y la contraseña de la red wifi. En AWS IoT Configuration, seleccione el certificado y la clave privada que ha descargado antes. Esto copiará el certificado y la clave privada en su botón AWS IoT. Marque la casilla para aceptar los términos y condiciones del botón AWS IoT y, a continuación, seleccione el botón Configure.

## Button ConfigureMe

Enter the value for any field that you wish to change for device: [REDACTED]

### Wi-Fi Configuration:

SSID: Guest  
 Open Network(No Password)  
Password: None (unsecured)

### AWS IoT Configuration:

Certificate:  certificate.pem  
Private Key:  private.key  
Endpoint Subdomain: A3T2RR9XSNT91O  
Endpoint Region: us-west-2  
Final Endpoint: .iot.us-west-2.amazonaws.com

By clicking this box, you agree to the [AWS IoT Button Terms and Conditions](#).

Se mostrará una página de confirmación de la configuración.

## Button ConfigureMe Setup

**Thank you for configuring your device.**

**If you are unable to use your device, please enter configuration mode and try again.**

5. Cierre la pestaña Configure y vuelva a la página de la consola de AWS Lambda. Elija Enable trigger y, a continuación, seleccione Next.

En la página Configure function, escriba un nombre para la función. La descripción, el tiempo de ejecución y el código de la función Lambda ya están especificados.

da > New function using blueprint iot-button-email

ect blueprint

igure triggers

igure function

ew

## Configure function

A Lambda function consists of the custom code you want to execute. [Learn more](#) about Lambda functions.

Name\* myIoTButtonFunction

Description An AWS Lambda function that sends an em

Runtime\* Node.js 4.3

### Lambda function code

Provide the code for your function. Use the editor if your code does not require custom libraries (other than the aws-sdk). If libraries, you can upload your code and libraries as a .ZIP file. [Learn more](#) about deploying Lambda functions.

Code entry type [Edit code inline](#)

We have restored the code from your previous session. Would you like to revert to the last saved state? [Revert now](#).

```
1  /**
2   * This is a sample Lambda function that sends an Email on click of a
3   * button. It creates a SNS topic, subscribes an endpoint (EMAIL)
4   * to the topic and publishes to the topic.
5   *
6   * Follow these steps to complete the configuration of your function:
7   *
8   * 1. Update the EMAIL variable with your email address.
9   * 2. Enter a name for your execution role in the "Role name" field.
10  * Your function's execution role needs specific permissions for SNS operations
11  * to send an email. We have pre-selected the "AWS IoT Button permissions"
12  * policy template that will automatically add these permissions.
13  */
14
15 const EMAIL = 'my_email@example.com'; // TODO change me
```

En el código de la función Lambda, reemplace la dirección de correo electrónico de ejemplo por la suya propia.

```
1  /**
2   * This is a sample Lambda function that sends an Email on click of a
3   * button. It creates a SNS topic, subscribes an endpoint (EMAIL)
4   * to the topic and publishes to the topic.
5   *
6   * Follow these steps to complete the configuration of your function:
7   *
8   * 1. Update the EMAIL variable with your email address.
9   * 2. Enter a name for your execution role in the "Role name" field.
10  * Your function's execution role needs specific permissions for SNS operations
11  * to send an email. We have pre-selected the "AWS IoT Button permissions"
12  * policy template that will automatically add these permissions.
13  */
14
15 const EMAIL = 'my_email@example.com'; // TODO change me
16
17 const AWS = require('aws-sdk');
18 const SNS = new AWS.SNS({ apiVersion: '2010-03-31' });
19
20 function findExistingSubscription(topicArn, nextToken, cb) {
21   const params = {
22     TopicArn: topicArn,
23     NextToken: nextToken || null,
24   };
25   SNS.listSubscriptionsByTopic(params, (err, data) => {
26     if (err) {
```

En la sección Lambda function handler and role, en el menú desplegable Role, elija Create new role from template(s). Escriba un nombre exclusivo para el rol.

#### Lambda function handler and role

**Handler\*** index.handler

**Role\*** Create new role from template(s) 

Lambda will automatically create a role with permissions from the selected policy templates. Note that basic Lambda permissions (logging to CloudWatch) will automatically be added. If your function accesses a VPC, VPC permissions will also be added.

**Role name** myIoTButtonRole 

**Policy templates**  

En la parte inferior de la página, elija Next.

Revise la configuración de la función Lambda y, a continuación, elija Create function.

new function using blueprint iot-button-email

## Review

Please review your Lambda function details. You can go back to edit changes for each section. When you are ready, click **Create function** to complete the setup process.

### Triggers

### Lambda function

**Name** myButtonFunction

**Description** An AWS Lambda function that sends an email on the click of an IoT button.

**Runtime** Node.js 4.3

**Handler** index.handler

**Role name** myNewRole

**Policy templates** AWS IoT Button permissions

**Memory (MB)** 128

**Timeout** 3

**VPC** No VPC

**Cancel**

**Previous**

**Create**

Deberá ver una página que confirme que se ha creado la función Lambda:

The screenshot shows the AWS Lambda console interface. At the top, there are tabs for 'Services' and 'Edit'. Below that, a breadcrumb navigation shows 'Functions > myButtonFunction'. There are two buttons: 'Test' (blue) and 'Actions' (grey). The main content area displays a success message: 'Success! Your Lambda function "myButtonFunction" has been successfully created and configured with IoT: iotbutton\_! [REDACTED] as a trigger.' Below this, there are three tabs: 'Configuration' (disabled), 'Triggers' (selected, highlighted in orange), and 'Monitoring'. Under the 'Triggers' tab, it shows a single trigger named 'AWS IoT: iotbutton\_G030JF055364XVRB'. It provides details: 'arn:aws:iot:us-east-1:[REDACTED]:rule/iotbutton\_! [REDACTED]', 'Module Description: Event source for your IoT Button to Lambda', and 'SQL Statement: SELECT \* FROM 'iotbutton/! [REDACTED]''. A blue link labeled 'trigger' is visible at the bottom left.

6. Para probar la función Lambda, seleccione el botón Test. Despues de un minuto, debe recibir un mensaje de correo electrónico con AWS Notification – Subscription Confirmation en la línea de asunto. Haga clic en el enlace del mensaje de correo electrónico para confirmar la suscripción a un tema de SNS creado por la función Lambda. Cuando AWS IoT recibe un mensaje de su botón, envía un mensaje a Amazon SNS. La función Lambda crea una suscripción al tema de Amazon SNS usando la dirección de correo electrónico que ha agregado en el código. Cuando Amazon SNS recibe un mensaje sobre este tema de Amazon SNS, lo reenvía a la dirección de correo electrónico de la suscripción.

Pulse el botón para enviar un mensaje a AWS IoT. El mensaje activará la regla Lambda y, a continuación, se invocará su función Lambda. La función Lambda comprobará si el tema de SNS existe. A continuación, la función Lambda enviará el contenido del mensaje al tema de Amazon SNS. Despues, Amazon SNS reenviará el mensaje a la dirección de correo electrónico que ha especificado en el código de la función Lambda.

## Inicio rápido AWS CloudFormation del botón AWS IoT

Cuando se pulsa el botón AWS IoT, se envía información básica sobre este a un tema de Amazon SNS. A continuación, el tema le reenvía dicha información en un mensaje de correo electrónico. Este inicio rápido le mostrará cómo utilizar una plantilla de AWS CloudFormation para configurar su botón AWS IoT.

Necesitará una cuenta de AWS y un botón AWS IoT para completar los pasos que se indican en este inicio rápido.

1. Utilice la consola de AWS IoT para crear un certificado de AWS IoT:
  - a. Abra la [consola de AWS IoT](#).
  - b. Si aparece una página Welcome, elija Get started.
  - c. En el selector de regiones de AWS, elija la región de AWS en la que desea crear el certificado de AWS IoT (por ejemplo, EE. UU. Este [Norte de Virginia]). Va a crear todos los recursos de AWS compatibles (recursos de AWS IoT adicionales y un recurso de Amazon SNS) en la misma región de AWS.
  - d. En el panel de navegación izquierdo de la página Dashboard, elija Security y, a continuación, elija Certificates.
  - e. En el panel Certificates, seleccione Create.
  - f. Seleccione One-click certificate creation - Create certificate.
  - g. En la página Certificate created, elija Download para descargar el certificado, la clave privada y la CA raíz de AWS IoT, guárdelos en su equipo y, a continuación, elija Activate para continuar.
  - h. Seleccione Done.
  - i. En la página Certificates, elija el certificado que acaba de crear.
  - j. En el panel Details, anote el valor de ARN del certificado (por ejemplo, arn:aws:iot:region-ID:account-ID:cert/random-ID). Lo necesitará más tarde en este mismo procedimiento.
2. Utilice la consola de AWS CloudFormation en <https://console.aws.amazon.com/cloudformation/> para crear los recursos de AWS IoT, un recurso de Amazon SNS y un rol de IAM:
  - a. Guarde el archivo de plantilla de AWS CloudFormation denominado AWSIoTButtonQuickStart.template en su equipo.

```
{  
    "AWSTemplateFormatVersion": "2010-09-09",  
    "Description": "Creates required AWS resources to allow an AWS IoT button to send  
information through an Amazon Simple Notification Service (Amazon SNS) topic to an  
email address.",  
    "Parameters": {  
        "IoTButtonDSN": {  
            "Type": "String",  
            "AllowedPattern": "G030[A-Z][A-Z][0=9][0-9][0-9][0-5][0-9][1-7][0-9A-HJ-NP-X]  
[0-9A-HJ-NP-X][0-9A-HJ-NP-X][0-9A-HJ-NP-X]",  
            "Description": "The device serial number (DSN) of the AWS IoT Button. This can  
be found on the back of the button. The DSN must match the pattern of 'G030[A-Z]  
[A-Z][0=9][0-9][0-9][0-5][0-9][1-7][0-9A-HJ-NP-X][0-9A-HJ-NP-X][0-9A-  
HJ-NP-X]'. "  
        },  
        "CertificateARN": {  
            "Type": "String",  
            "Description": "The Amazon Resource Name (ARN) of the existing AWS IoT  
certificate."  
        },  
        "SNSTopicName": {  
            "Type": "String",  
            "Default": "aws-iot-button-sns-topic",  
            "Description": "The name of the Amazon SNS topic for AWS CloudFormation to  
create."  
        },  
        "SNSTopicRoleName": {  
            "Type": "String",  
            "Default": "aws-iot-button-sns-topic-role",  
            "Description": "The name of the IAM role for AWS CloudFormation to create. This  
IAM role allows AWS IoT to send notifications to the Amazon SNS topic."  
        },  
        "EmailAddress": {  
            "Type": "String",  
            "Description": "The email address to which notifications will be sent from the  
Amazon SNS topic."  
        }  
    },  
    "Resources": {  
        "SNSTopic": {  
            "Type": "AWS::SNS::Topic",  
            "Properties": {  
                "Name": {"Ref": "SNSTopicName"},  
                "Subscription": [{"Endpoint": {"Ref": "EmailAddress"}, "Protocol": "Email"}]  
            }  
        },  
        "SNSTopicRole": {  
            "Type": "AWS::IAM::Role",  
            "Properties": {  
                "AssumeRolePolicyDocument": {"Version": "2012-10-17", "Statement": [{"Action": "sts:AssumeRole", "Principal": "aws-iot-button-sns-topic-role", "Effect": "Allow"}]},  
                "Path": "/",  
                "Policies": [{"PolicyName": "SNSTopicPolicy", "PolicyDocument": {"Version": "2012-10-17", "Statement": [{"Action": "sns:Publish", "TopicArn": {"Ref": "SNSTopic"}, "Effect": "Allow"}]}}],  
                "RoleName": {"Ref": "SNSTopicRoleName"}  
            }  
        },  
        "IoTButton": {  
            "Type": "AWS::IoT::CustomResource",  
            "Properties": {  
                "Service": "AWSIoT",  
                "FunctionName": "aws-iot-button-sns-topic",  
                "RoleArn": {"Ref": "SNSTopicRole"},  
                "Policy": {"Version": "2012-10-17", "Statement": [{"Action": "sns:Publish", "TopicArn": {"Ref": "SNSTopic"}, "Effect": "Allow"}]},  
                "Payload": {"Type": "Text", "Value": "button press"},  
                "Region": "us-east-1",  
                "ResourceType": "button",  
                "ResourceName": {"Ref": "IoTButtonDSN"},  
                "Event": "buttonPress"  
            }  
        }  
    },  
    "Outputs": {  
        "SNSTopicArn": {  
            "Description": "The ARN of the Amazon SNS topic created by the CloudFormation  
stack.",  
            "Value": {"Fn::GetAtt": ["SNSTopic", "Arn"]}  
        },  
        "SNSTopicName": {  
            "Description": "The name of the Amazon SNS topic created by the CloudFormation  
stack.",  
            "Value": {"Fn::GetAtt": ["SNSTopic", "Name"]}  
        },  
        "SNSTopicRoleArn": {  
            "Description": "The ARN of the IAM role created by the CloudFormation stack.  
This role allows AWS IoT to publish to the Amazon SNS topic.",  
            "Value": {"Fn::GetAtt": ["SNSTopicRole", "Arn"]}  
        },  
        "SNSTopicRoleName": {  
            "Description": "The name of the IAM role created by the CloudFormation stack.  
This role allows AWS IoT to publish to the Amazon SNS topic.",  
            "Value": {"Fn::GetAtt": ["SNSTopicRole", "Name"]}  
        },  
        "IoTButtonARN": {  
            "Description": "The ARN of the AWS IoT button resource created by the CloudFormation  
stack.",  
            "Value": {"Fn::GetAtt": ["IoTButton", "Arn"]}  
        },  
        "IoTButtonName": {  
            "Description": "The name of the AWS IoT button resource created by the CloudFormation  
stack.",  
            "Value": {"Fn::GetAtt": ["IoTButton", "Name"]}  
        }  
    }  
}
```

```
        "Description": "The email address for the Amazon SNS topic to send information
to."
    }
},
"Resources": {
    "IoTThing": {
        "Type": "AWS::IoT::Thing",
        "Properties": {
            "ThingName": {
                "Fn::Join": [
                    [
                        "iotbutton_",
                        { "Ref": "IoTButtonDSN" }
                    ]
                ]
            }
        }
    },
    "IoTPolicy": {
        "Type" : "AWS::IoT::Policy",
        "Properties": {
            "PolicyDocument": {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Action": "iot:Publish",
                        "Effect": "Allow",
                        "Resource": {
                            "Fn::Join": [
                                [
                                    "arn:aws:iot:",
                                    { "Ref": "AWS::Region" },
                                    ":",
                                    { "Ref": "AWS::AccountId" },
                                    ":topic/iotbutton/",
                                    { "Ref": "IoTButtonDSN" }
                                ]
                            ]
                        }
                    }
                ]
            }
        }
    },
    "IoTPolicyPrincipalAttachment": {
        "Type": "AWS::IoT::PolicyPrincipalAttachment",
        "Properties": {
            "PolicyName": {
                "Ref": "IoTPolicy"
            },
            "Principal": {
                "Ref": "CertificateARN"
            }
        }
    },
    "IoTThingPrincipalAttachment": {
        "Type" : "AWS::IoT::ThingPrincipalAttachment",
        "Properties": {
            "Principal": {
                "Ref": "CertificateARN"
            },
            "ThingName": {
                "Ref": "IoTThing"
            }
        }
    },
}
```

```
"SNSTopic": {
    "Type": "AWS::SNS::Topic",
    "Properties": {
        "DisplayName": "AWS IoT Button Press Notification",
        "Subscription": [
            {
                "Endpoint": {
                    "Ref": "EmailAddress"
                },
                "Protocol": "email"
            }
        ],
        "TopicName": {
            "Ref": "SNSTopicName"
        }
    }
},
"SNSTopicRole": {
    "Type": "AWS::IAM::Role",
    "Properties": {
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {
                        "Service": "iot.amazonaws.com"
                    },
                    "Action": "sts:AssumeRole"
                }
            ]
        },
        "Path": "/",
        "Policies": [
            {
                "PolicyDocument": {
                    "Version": "2012-10-17",
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Action": "sns:Publish",
                            "Resource": {
                                "Fn::Join": [ "", [
                                    "arn:aws:sns:",
                                    { "Ref": "AWS::Region" },
                                    ":",
                                    { "Ref": "AWS::AccountId" },
                                    ":",
                                    { "Ref": "SNSTopicName" }
                                ] ]
                            }
                        }
                    ]
                },
                "PolicyName": {
                    "Ref": "SNSTopicRoleName"
                }
            }
        ]
    }
},
"IoTTopicRule": {
    "Type": "AWS::IoT::TopicRule",
    "Properties": {
```

```
        "RuleName": {
          "Fn::Join": [ "", [
            "iotbutton_",
            { "Ref": "IoTButtonDSN" }
          ] ]
        },
        "TopicRulePayload": {
          "Actions": [
            {
              "Sns": {
                "RoleArn": {
                  "Fn::GetAtt": [ "SNSTopicRole", "Arn" ]
                },
                "TargetArn": {
                  "Ref": "SNSTopic"
                }
              }
            ],
            "AwsIotSqlVersion": "2015-10-08",
            "RuleDisabled": false,
            "Sql": {
              "Fn::Join": [ "", [
                "SELECT * FROM 'iotbutton/'",
                { "Ref": "IoTButtonDSN" },
                "'"
              ] ]
            }
          }
        }
      }
    }
  }
}
```

- b. Abra la consola de AWS CloudFormation en <https://console.aws.amazon.com/cloudformation/>.
- c. Asegúrese de que el selector de regiones de AWS muestre la región en la que ha creado el certificado de AWS IoT (por ejemplo, EE. UU. Este [Norte de Virginia]).
- d. Elija Create Stack.
- e. En la página Select Template, elija Upload a template to Amazon S3 y, a continuación, elija Browse.
- f. Seleccione el archivo AWSIoTButtonQuickStart.template que ha guardado antes, elija Open y, a continuación, seleccione Next.
- g. En la página Specify Details, escriba un nombre para esta pila de AWS CloudFormation (por ejemplo, MyAWSIoTButtonStack) en el campo Stack name.
- h. En CertificateARN, escriba el nombre de recurso de Amazon (ARN) del certificado de AWS IoT (el valor de ARN de certificado) que ha anotado antes.
- i. En EmailAddress, escriba su dirección de correo electrónico.
- j. En IoTButtonDSN, escriba el número de serie del dispositivo (DSN). Lo encontrará en la parte posterior del botón AWS IoT (por ejemplo, G030JF051234A5BC).
- k. Puede conservar los valores predeterminados de SNSTopicName y SNSTopicRoleName, o especificar otro nombre de tema de Amazon SNS y otro nombre de rol de IAM asociado. Por ejemplo, si tiene previsto configurar más botones AWS IoT, le puede interesar cambiar estos valores. Seleccione Next.
- l. No necesita hacer nada más en la página Options. Seleccione Next.

- m. En la página Review, seleccione I acknowledge that AWS CloudFormation might create IAM resources y, a continuación, elija Create.
  - n. Cuando se muestre CREATE\_COMPLETE para MyAWSIoTButtonStack, compruebe si en su buzón de entrada hay algún mensaje con un asunto relacionado con la notificación de pulsación del botón AWS IoT. Haga clic en el enlace Confirm subscription que se encuentra en el cuerpo del mensaje de correo electrónico.
3. Con la clave privada y el certificado que ha creado anteriormente, siga los pasos que figuran en [Configure Your Device](#) para configurar su botón AWS IoT.
  4. Despues de la configuración, pulse el botón una vez. Una luz blanca parpadeará varias veces y, a continuación, se visualizará una luz verde fija durante unos segundos. Poco después, recibirá un mensaje de correo electrónico con una línea de asunto sobre la notificación de pulsación del botón AWS IoT. Verá información enviada por el botón en el cuerpo del mensaje.
  5. Una vez que haya terminado de experimentar, puede limpiar los recursos de AWS creados con la plantilla de AWS CloudFormation. Para ello, vuelva a la consola de AWS CloudFormation y elimine MyAWSIoTButtonStack. Despues de eliminar MyAWSIoTButtonStack, elimine el certificado de AWS IoT de la siguiente manera:
    - a. Vuelva a la consola de AWS IoT.
    - b. En la lista de recursos, seleccione la casilla del cuadro que representa el certificado de AWS IoT (casilla con el icono del apretón de manos).
    - c. En Actions, elija Deactivate y, a continuación, confirme la selección.
    - d. Con la casilla que representa el certificado de AWS IoT todavía marcada, en Actions, elija Delete y, a continuación, confirme la selección.
    - e. El certificado y la clave privada que ha descargado anteriormente dejarán de ser válidos, por lo que ahora puede eliminarlos de su equipo.

## Pasos siguientes

Para obtener más información sobre el proyecto Lambda utilizado para configurar su botón, consulte [Introducción a AWS IoT](#). Para obtener información sobre cómo usar AWS CloudFormation con el botón AWS IoT, consulte <http://docs.aws.amazon.com/iot/latest/developerguide/iot-button-cloud-formation.html>.

# Tutoriales de reglas de AWS IoT

Esta guía contiene tutoriales que le ayudarán a crear y probar reglas de AWS IoT. Si no ha completado el [tutorial de introducción a AWS IoT \(p. 5\)](#), le recomendamos que lo haga primero. Dicho tutorial le muestra cómo crear una cuenta de AWS y cómo conectar su dispositivo a AWS IoT.

Una regla de AWS IoT consta de una instrucción SQL SELECT, un filtro de temas y una acción de regla. Los dispositivos envían información a AWS IoT publicando mensajes en los temas MQTT. La instrucción SQL SELECT le permite extraer datos de un mensaje MQTT de entrada. El filtro de temas de una regla de AWS IoT especifica uno o varios temas MQTT. La regla se activa cuando se recibe un mensaje MQTT sobre un tema que coincide con el filtro de temas. Las acciones de regla le permiten tomar la información extraída de un mensaje MQTT y enviarla a otro servicio de AWS. Las acciones de regla se definen para servicios de AWS como Amazon DynamoDB, AWS Lambda Amazon SNS y Amazon S3. Con una regla Lambda puede llamar a otros servicios web de AWS o de terceros. Para obtener una lista completa de las acciones de regla, consulte [Acciones de las reglas de AWS IoT \(p. 156\)](#).

En estos tutoriales se presupone que utiliza el botón AWS IoT y que utilizará `iotbutton/+` como filtro de temas en las reglas. Si no dispone de botón AWS IoT, [puede comprar uno aquí](#).

De manera alternativa, puede emular el botón AWS IoT mediante un cliente MQTT, como el cliente MQTT de AWS IoT en la [consola de AWS IoT](#). Para emular el botón AWS IoT publique un mensaje similar en el tema `iotbutton/ABCDEFG12345`. El número que figura después del símbolo / es arbitrario. Se usa como número de serie para el botón.

También puede utilizar su propio dispositivo, aunque debe saber en qué tema de MQTT publica su dispositivo para especificarlo como filtro de temas en la regla. Para obtener más información, consulte [Reglas de AWS IoT \(p. 148\)](#).

El botón AWS IoT envía una carga JSON que tiene este aspecto:

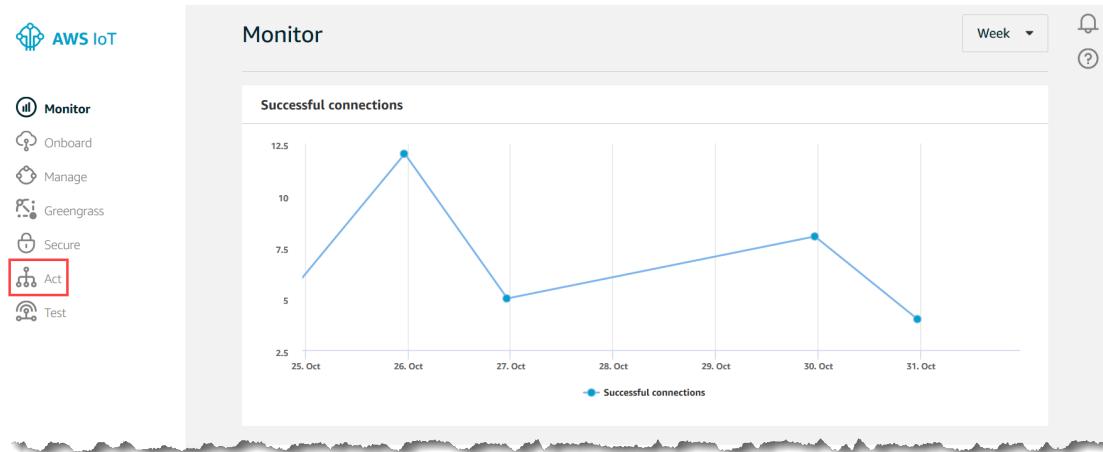
```
{  
    "serialNumber" : "ABCDEFG12345",  
    "batteryVoltage" : "2000mV",  
    "clickType" : "SINGLE"  
}
```

# Creación de un regla de DynamoDB

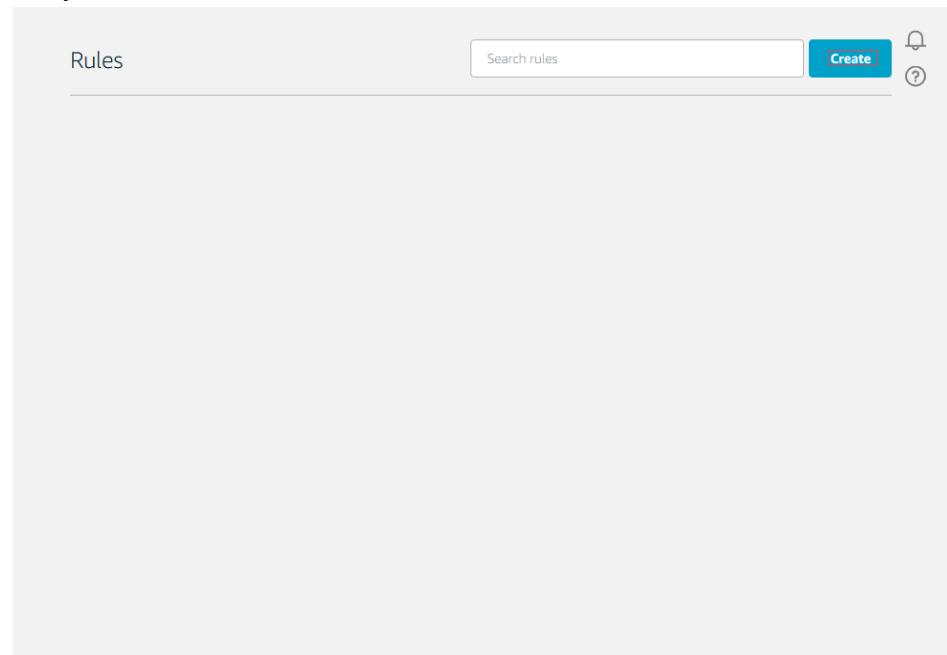
Las reglas de DynamoDB le permiten tomar información de un mensaje MQTT de entrada y escribirlo en una tabla de DynamoDB.

Para crear una regla de DynamoDB:

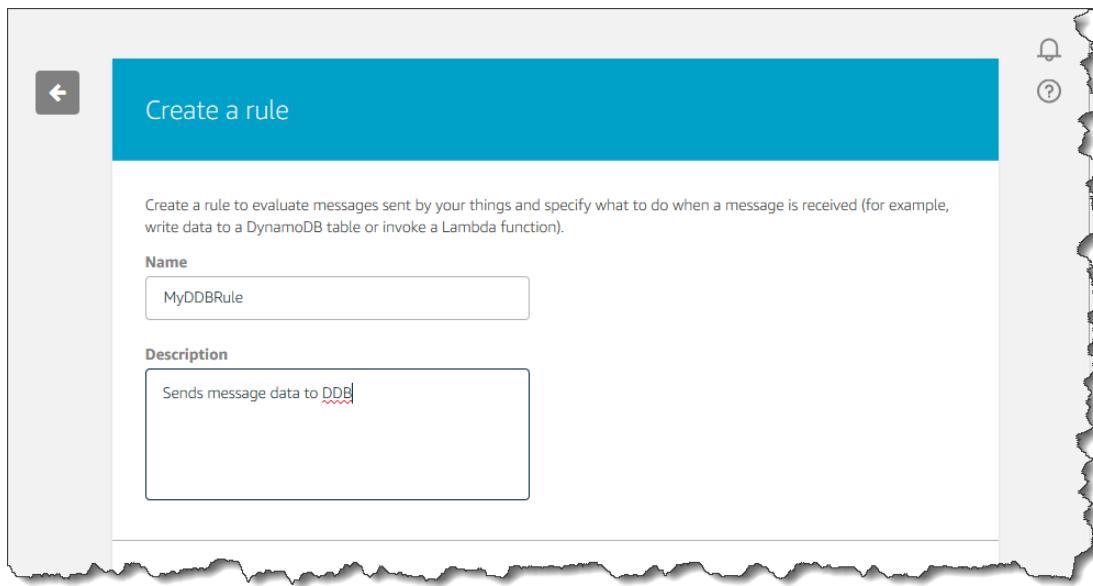
1. En el panel de navegación izquierdo de la [consola de AWS IoT](#), elija Rules.



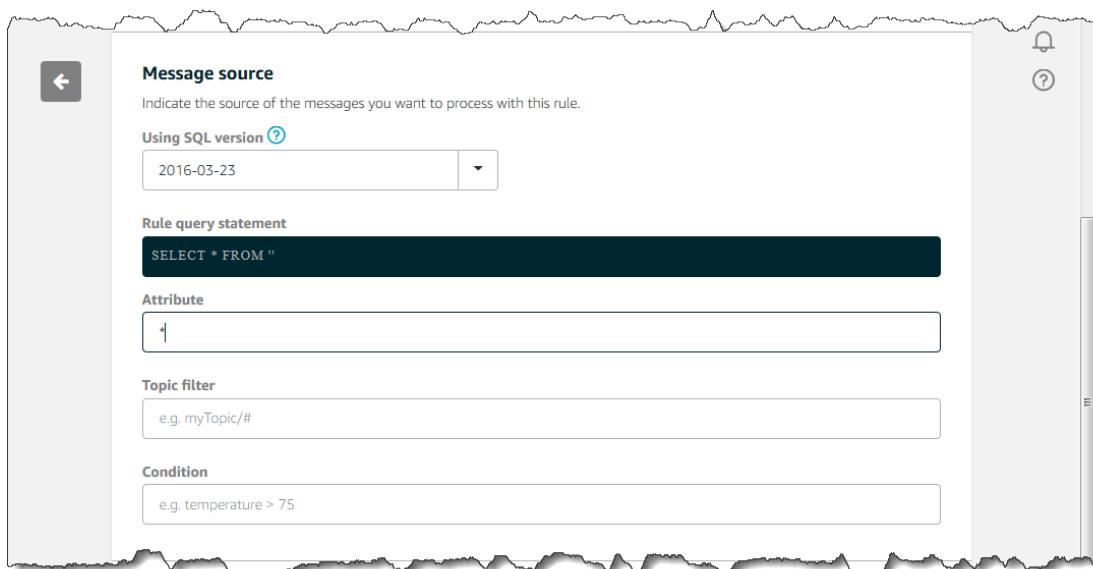
2. En la página Rules elija Create.



3. En la página Create a rule, en el campo Name, escriba un nombre para la regla. En el campo Description, escriba una descripción para la regla.



4. Desplácese hasta Message source. Seleccione la última versión en la lista desplegable de Using SQL version. En el campo Attribute, escriba \*. Esto especifica que desea enviar todo el mensaje MQTT que ha activado la regla.



5. El motor de reglas utiliza el filtro de temas para determinar qué reglas deben activarse cuando se recibe un mensaje MQTT. En el campo Topic filter, escriba **iotbutton/DNS del botón**. Si no utiliza un botón AWS IoT, escriba **my/topic** o el tema utilizado en la regla.

#### Note

Puede encontrar el DSN en la parte inferior del botón.

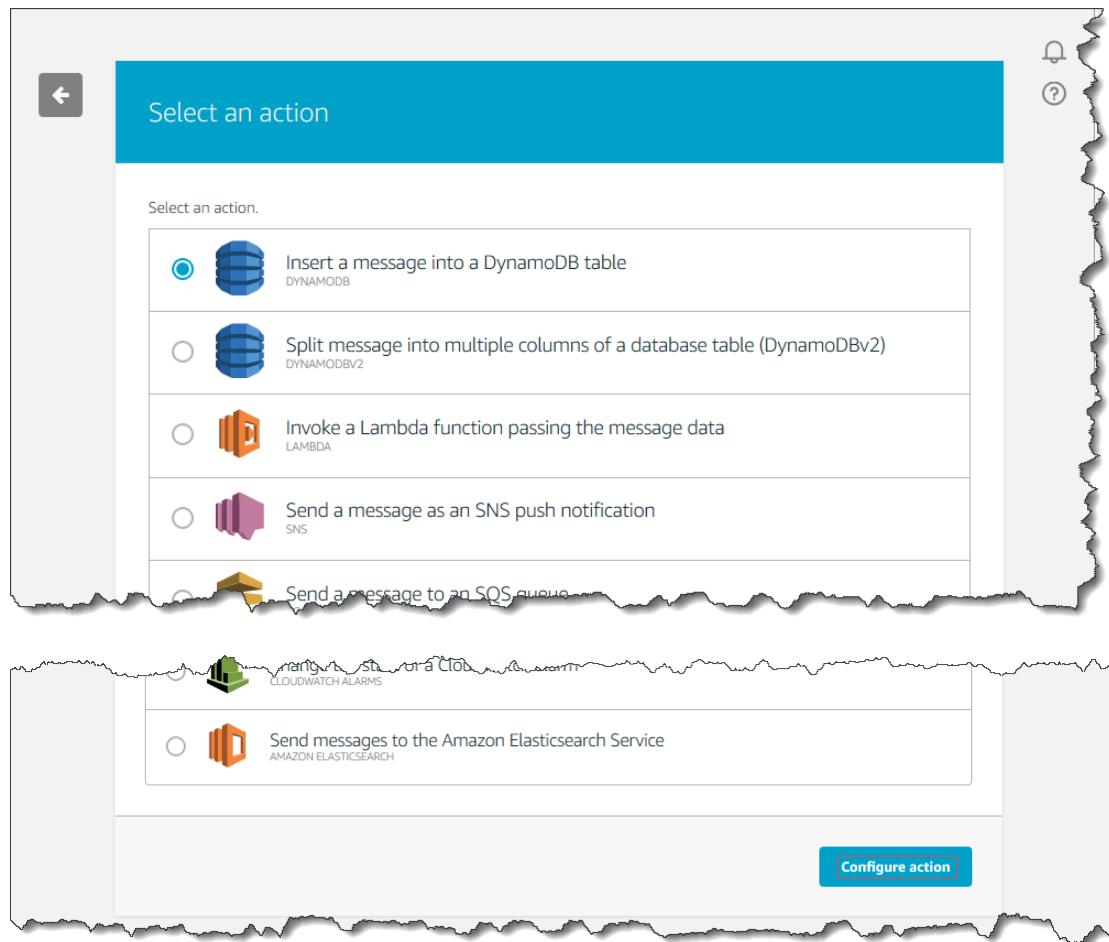
Deje Condition en blanco.

The screenshot shows the 'Message source' configuration page. It includes fields for 'Using SQL version' (set to 2016-03-23), a 'Rule query statement' containing 'SELECT \* FROM 'iotbutton/G030JF053216F1BS'', and sections for 'Attribute' (containing '\*'), 'Topic filter' (containing 'iotbutton/G030JF053216F1BS'), and 'Condition' (containing 'e.g. temperature > 75').

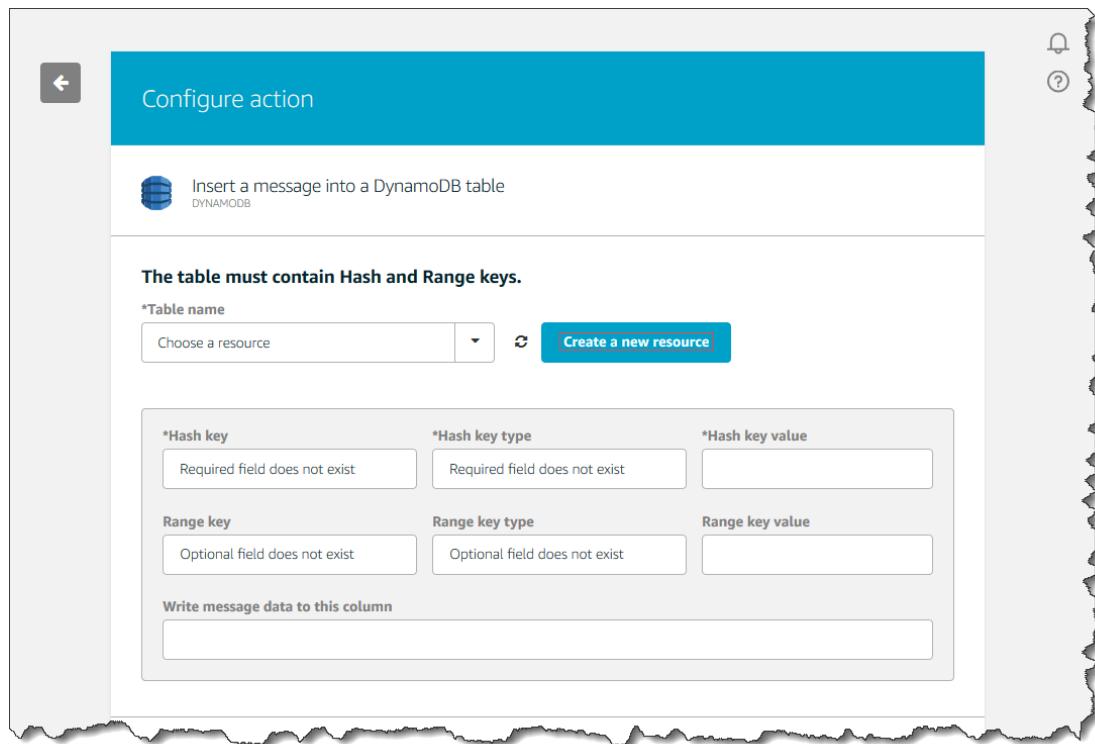
6. En Set one or more actions, elija Add action.

The screenshot shows the 'Set one or more actions' configuration page. It includes a descriptive text about selecting actions for inbound messages and a prominent 'Add action' button.

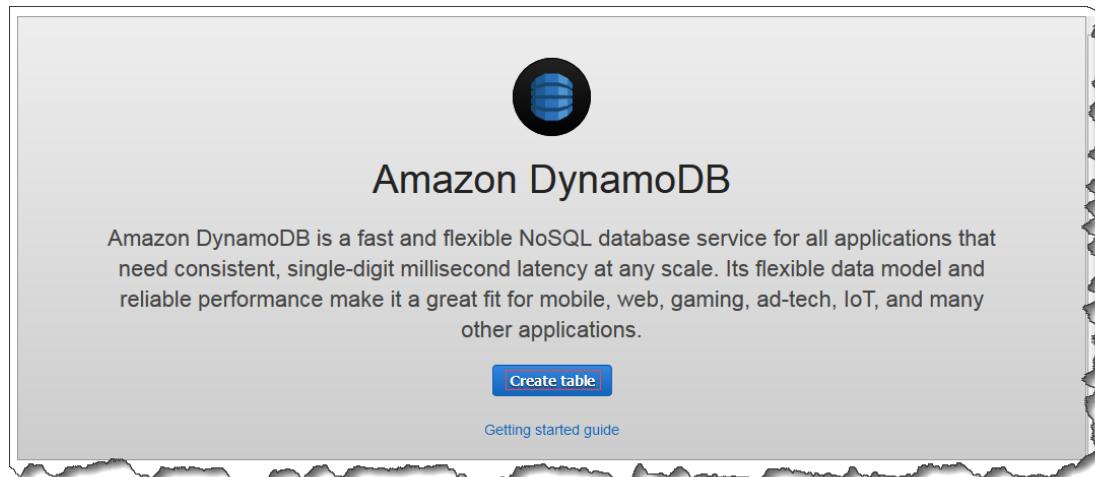
7. En la página Select an action, seleccione Insert a message into a DynamoDB table y, a continuación, elija Configure action.



8. En la página Configure action, seleccione Create a new resource.



9. En la página Amazon DynamoDB, elija Create table.



10. En la página Create DynamoDB table, escriba un nombre en el campo Table name. En Partition key, escriba **SerialNumber**. Marque la casilla Add sort key y, a continuación, escriba **clickType** en el campo Sort key. Seleccione String para la partición y las claves de ordenación.

## Create DynamoDB table

DynamoDB is a schema-less database that only requires a table name and primary key. The table's primary key is made up of one or two attributes that uniquely identify items, partition the data, and sort data within each partition.

**Table name\*** IoTButtonTable 

**Primary key\*** Partition key

SerialNumber String 

Add sort key

ClickType String 

### Table settings

Default settings provide the fastest way to get started with your table. You can modify these default settings now or after your table has been created.

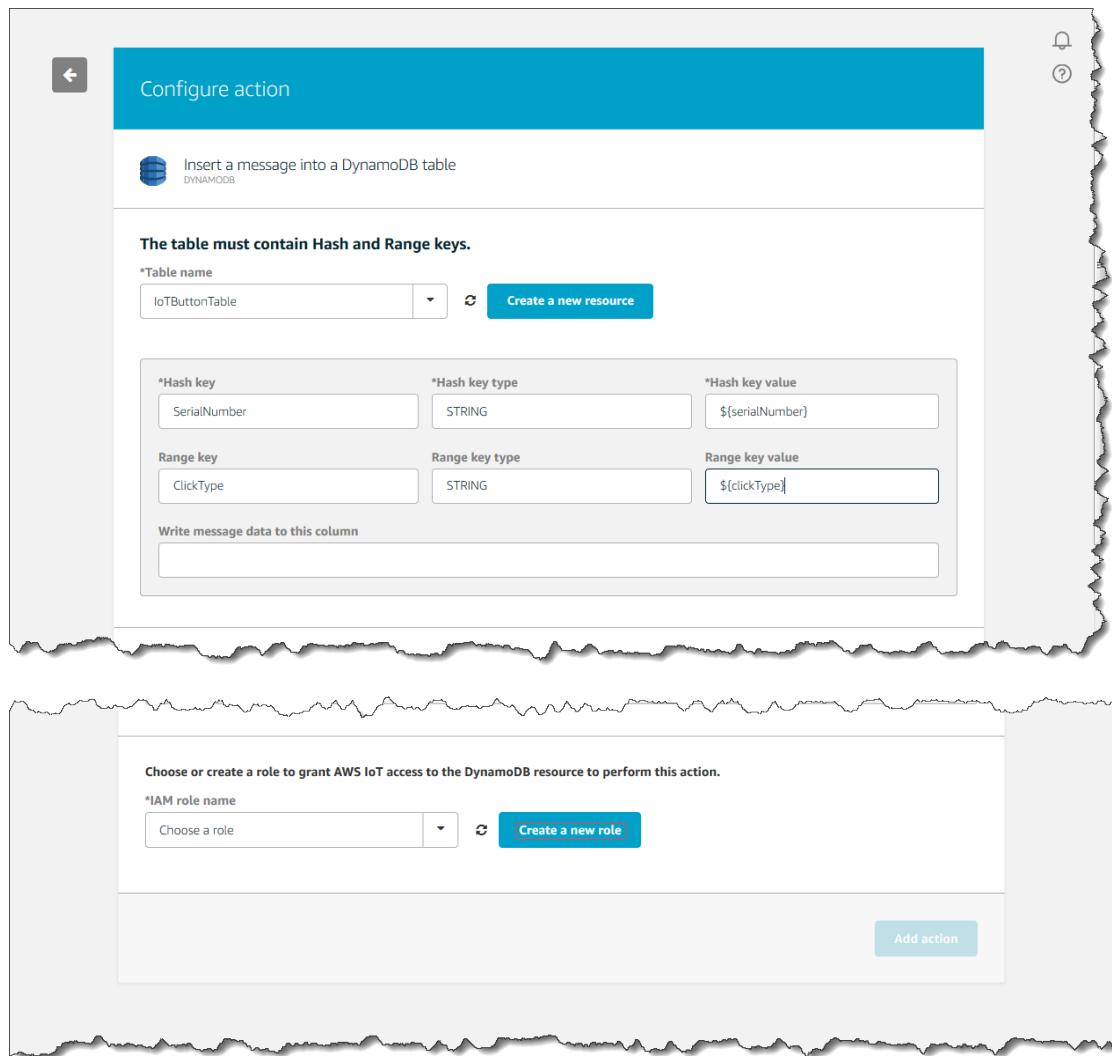
Use default settings

- No secondary indexes.
- Provisioned capacity set to 5 reads and 5 writes.
- Basic alarms with 80% upper threshold using SNS topic "dynamodb".

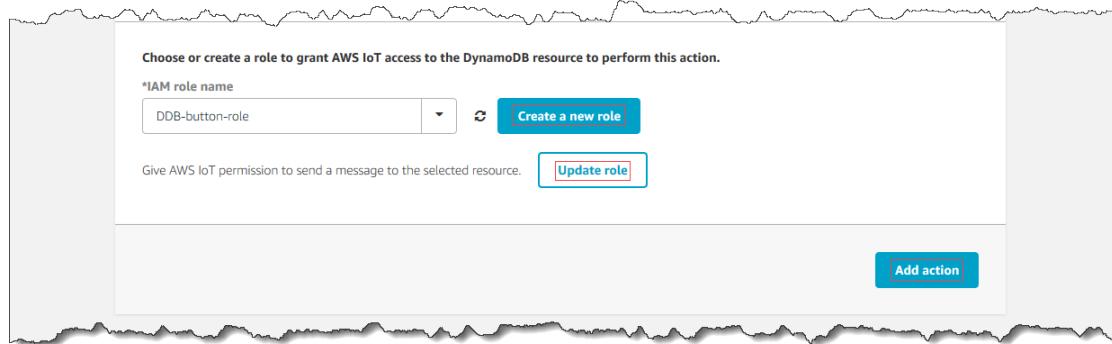
Additional charges may apply if you exceed the AWS Free Tier levels for CloudWatch or Simple Notification Service. Advanced alarm settings are available in the CloudWatch management console.

[Cancel](#) [Create](#)

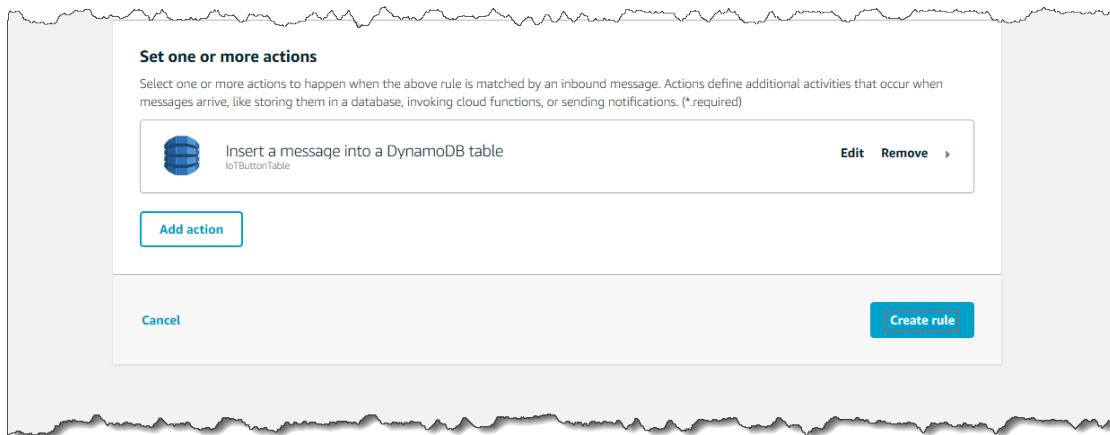
11. Seleccione Create. La tabla de DynamoDB tarda unos segundos en crearse. Cierre la pestaña del navegador donde la consola de Amazon DynamoDB está abierta. Si no cierra la pestaña, su tabla de DynamoDB no se mostrará en la lista desplegable de Table name en la página Configure action de AWS IoT.
12. En la página Configure action, elija la nueva tabla de la lista desplegable Table name. En Hash key value, escriba \${serialNumber}. Esto indica a la regla que debe tomar el valor del atributo serialNumber en el mensaje MQTT y escribirlo en la columna SerialNumber de la tabla de DynamoDB. En Range key value, escriba \${clickType}. Esto escribe el valor del atributo clickType en la columna ClickType. Deje Write message data to this column en blanco. De forma predeterminada, el mensaje completo se escribe en una columna de la tabla denominada Payload. Elija Create a new role.



13. Escriba un nombre único en IAM role name y, a continuación, elija el botón Create a new role de nuevo. Elija la función que acaba de crear, elija Update role y, a continuación, seleccione Add action.



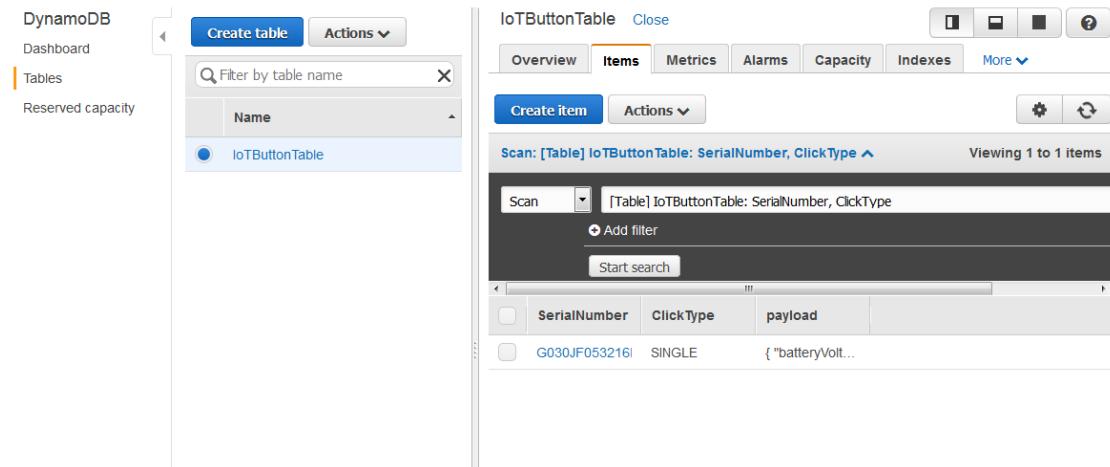
14. Elija Create rule para crear la regla.



15. Un mensaje de confirmación muestra que la regla se ha creado. Seleccione la flecha izquierda para regresar a la página Rules.

The screenshot shows the 'MyDDBRule2' rule details page. It includes sections for Overview, Description (which says 'Sends message data to DDB'), Rule query statement (containing the SQL query 'SELECT \* FROM 'iotbutton/G030JF053216F1BS''), Actions (which lists the action 'Insert a message into a DynamoDB table' (IoTButtonTable)), and an 'Edit' button for the rule itself.

16. Pruebe la regla pulsando el botón AWS IoT configurado o usando un cliente MQTT para publicar un mensaje en un tema que coincida con el filtro de temas de la regla. Por último, vuelva a la consola de DynamoDB y seleccione la tabla que ha creado para ver el tipo de pulsación o mensaje del botón.



## Creación de un regla Lambda

Puede definir una regla que llame a una función Lambda, transfiriendo datos del mensaje MQTT que ha activado la regla. Esto le permite procesar el mensaje de entrada y, a continuación, llamar a otro servicio de AWS o de terceros.

En este tutorial presuponemos que ya ha completado el [tutorial de introducción a AWS IoT \(p. 5\)](#), en el que se crea un tema de Amazon SNS y se suscribe a él usando el número de teléfono móvil. Va a crear una función Lambda que publica un mensaje en el tema de Amazon SNS que ha creado en el [tutorial de introducción a AWS IoT \(p. 5\)](#). También creará una regla de Lambda que llama a la función Lambda transfiriendo algunos datos del mensaje MQTT que ha activado la regla.

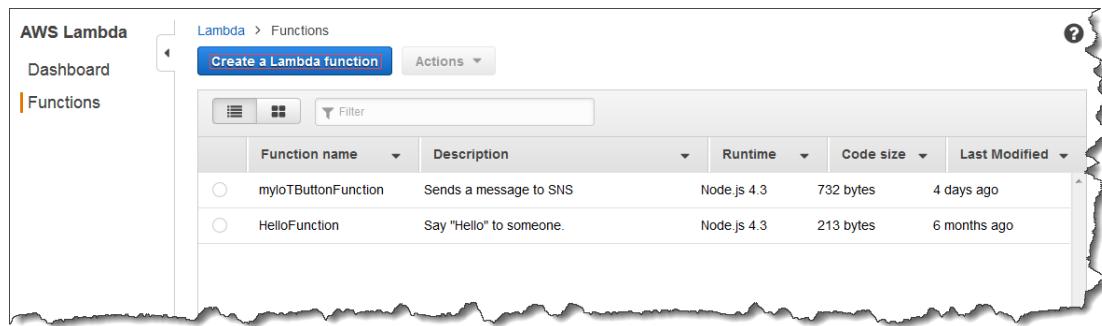
En este tutorial, también presuponemos que utiliza un botón AWS IoT para activar la regla de Lambda. Si no tiene un botón AWS IoT, [puede comprar uno aquí](#) o puede usar un cliente MQTT para enviar un mensaje MQTT que active la regla.

## Creación de la función Lambda

Para crear una función Lambda:

1. En la [consola de AWS Lambda](#), elija Get Started Now o, si ha creado una función Lambda antes, elija Create a Lambda function.





2. En la página Select blueprint, en el campo Filter, escriba **hello-world** y, a continuación, seleccione el proyecto hello-world.

Lambda > New function

Select blueprint

Configure triggers

Configure function

Review

Select blueprint

Blueprints are sample configurations of event sources and Lambda functions. Choose a blueprint that best aligns with your desired scenario and customize as needed, or skip this step if you want to author a Lambda function and configure an event source separately. Except where otherwise noted, blueprints are licensed under CC0.

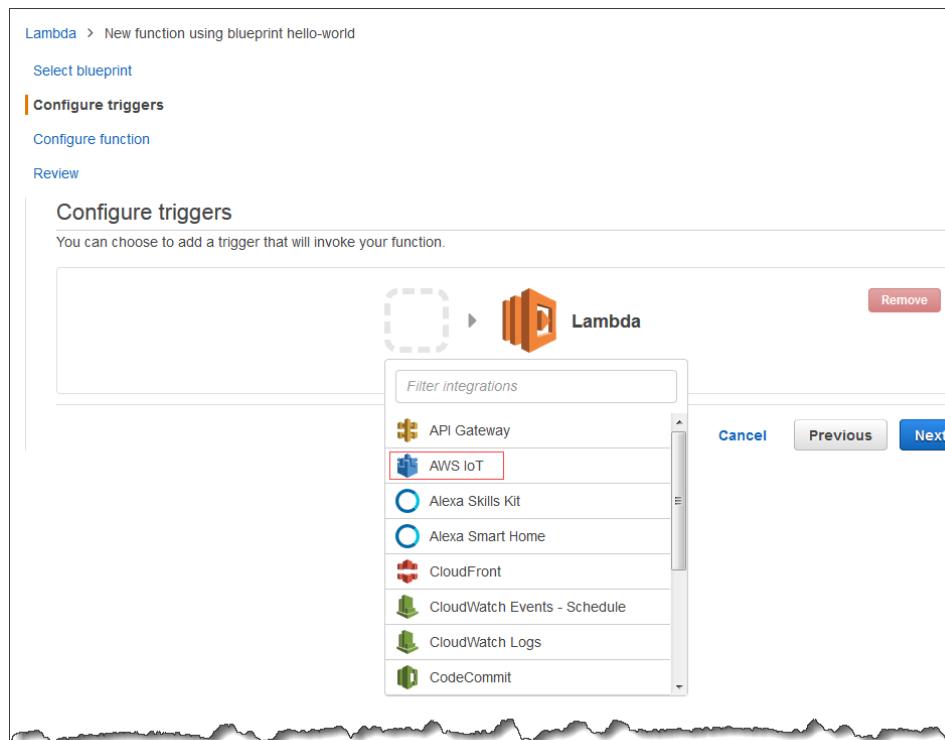
Welcome to AWS Lambda! You can get started on creating your first Lambda function by choosing one of the blueprints below.

Select runtime ▾ Filter Viewing 1-3 of 3

Blank Function	hello-world	hello-world-python
Configure your function from scratch. Define the trigger and deploy your code by stepping through our wizard. custom	A starter AWS Lambda function. nodejs	A starter AWS Lambda function. python2.7

Cancel

3. En la página Configure triggers, seleccione el cuadro situado a la izquierda del icono de Lambda y seleccione AWS IoT en el menú desplegable.



4. En el campo Device Serial Number, escriba el número de serie de dispositivo (DSN) del botón. El DSN se imprimirá en la parte posterior de su botón AWS IoT. Si todavía no ha generado un certificado ni una clave privada para su botón AWS IoT, elija Generate certificate and keys. De lo contrario, vaya al paso 6.

Lambda > New function using blueprint hello-world

Select blueprint

Configure triggers

Configure function

Review

### Configure triggers

You can choose to add a trigger that will invoke your function.

AWS IoT → Lambda

IoT Type: IoT Button

Device Serial Number: G030JF053216F1BS

Generate certificate and keys

For more information about IoT rules and SQL statements, please see the [AWS IoT documentation](#). Lambda will add the necessary permissions for AWS IoT to invoke your Lambda function. [Learn more](#) about the Lambda permissions model.

Enable trigger

Cancel Previous Next

5. Elija los enlaces para descargar el certificado PEM y la clave privada. Guarde estos archivos en un lugar seguro de su equipo.

We have created the necessary AWS IoT resources (thing, policy, certificate, private key). The remaining resources (rule and action) will be created after your function is created.

Download these resources by clicking the links below. (NOTE: If you are using Internet Explorer or Safari, right click the links to save the files.)

- a. Your certificate PEM
- b. Your private key

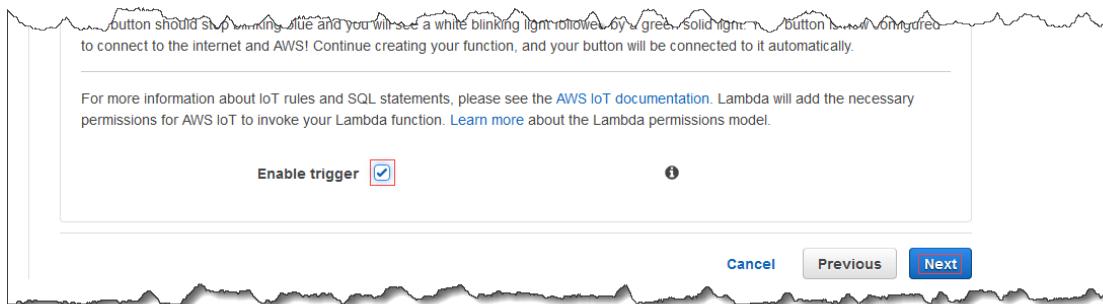
To configure the AWS IoT Button to use your Wi-Fi and these resources to connect to AWS securely, follow these steps:

1. Place the button into configuration mode by pressing the button down for 5 seconds until it flashes blue.
2. Connect your computer to the button's Wi-Fi network SSID "Button ConfigureMe - F09", using "3216F1BS" (last 8 digits of device serial number) as the WPA2-PSK password.
3. Click [here](#) (opens in new tab) and use the following information to fill out the form:
  - a. Enter your local network's Wi-Fi SSID and password.
  - b. Select the certificate and private key files that you just downloaded above.
  - c. Your endpoint subdomain is **a182jd32qs965e**.
  - d. Your endpoint region is **us-east-1**.
  - e. Check the box to agree to the terms and conditions.
  - f. Click "configure".
4. Re-connect to your original Wi-Fi network.

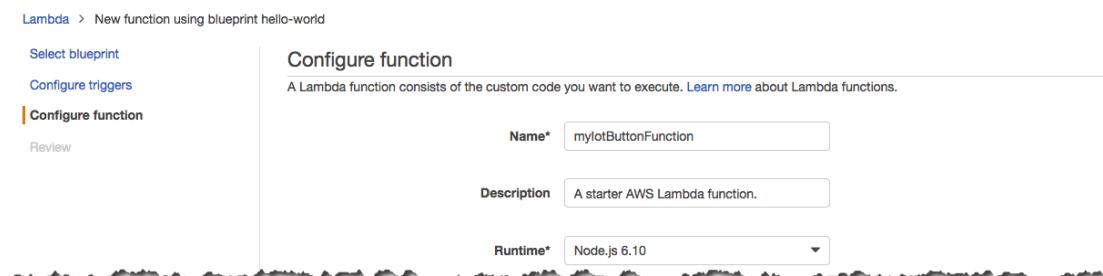
The button should stop blinking blue and you will see a white blinking light followed by a greed solid light. Your button is now configured to connect to the internet and AWS! Continue creating your function, and your button will be connected to it automatically.

Siga las instrucciones online para configurar el botón AWS IoT.

6. Asegúrese de que la casilla Enable trigger esté marcada y, a continuación, seleccione Next.



7. En la página Configure function, escriba un nombre y una descripción de la función Lambda. En Runtime, elija Node.js 6.10.



8. Desplácese hacia abajo para acceder a la sección Lambda function code de la página. Sustituya el código existente por lo siguiente:

```
console.log('Loading function');
// Load the AWS SDK
var AWS = require("aws-sdk");

// Set up the code to call when the Lambda function is invoked
exports.handler = (event, context, callback) => {
    // Load the message passed into the Lambda function into a JSON object
    var eventText = JSON.stringify(event, null, 2);

    // Log a message to the console, you can view this text in the Monitoring tab
    // in the Lambda console or in the CloudWatch Logs console
    console.log("Received event:", eventText);

    // Create a string extracting the click type and serial number from the message
    // sent by the AWS IoT button
    var messageText = "Received " + event.clickType + " message from button ID: "
    + event.serialNumber;

    // Write the string to the console
    console.log("Message to send: " + messageText);

    // Create an SNS object
    var sns = new AWS.SNS();

    // Populate the parameters for the publish operation
    // - Message : the text of the message to send
    // - TopicArn : the ARN of the Amazon SNS topic to which you want to publish
    var params = {
        Message: messageText,
        TopicArn: "arn:aws:sns:us-east-1:123456789012:MyIoTButtonSNSTopic"
    };
    sns.publish(params, context.done);
};
```

### Note

Reemplace el valor de `TopicArn` por el ARN del tema Amazon SNS que ha creado anteriormente.

9. Desplácese hacia abajo para acceder a la sección Lambda function handler and role de la página. En Role, seleccione Create a custom role. La consola de IAM se abrirá, lo que le permitirá crear un rol de IAM que Lambda pueda asumir al ejecutar la función Lambda.

Para editar la política del rol y darle permiso para publicar en su tema de Amazon SNS:

- a. Elija View Policy Document.

AWS Lambda requires access to your resources

AWS Lambda uses an IAM role that grants your custom code permissions to access AWS resources it needs.

▼ Hide Details

**Role Summary** ?

Role Lambda execution role permissions

Description

IAM Role lambda\_basic\_execution

Policy Name Create a new Role Policy

▶ [View Policy Document](#)

Don't Allow **Allow**



- b. Elija Edit para editar la política del rol.

AWS Lambda requires access to your resources

AWS Lambda uses an IAM role that grants your custom code permissions to access AWS resources it needs.

▼ Hide Details

Role Summary ?

Role Lambda execution role permissions

Description

IAM Role lambda\_basic\_execution

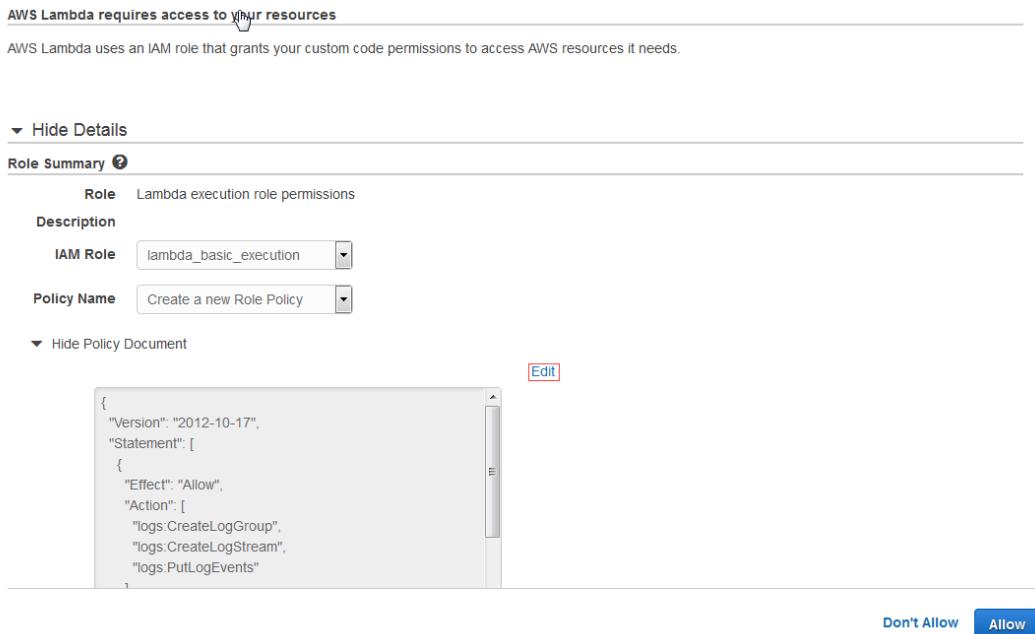
Policy Name Create a new Role Policy

▼ Hide Policy Document

Edit

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "logs:CreateLogGroup",  
                "logs:CreateLogStream",  
                "logs:PutLogEvents"  
            ],  
            "Resource": "arn:aws:logs:*:*:  
        },  
        {  
            "Effect": "Allow",  
            "Action": [  
                "sns:Publish"  
            ],  
            "Resource": "arn:aws:sns:us-east-1:123456789012:MyIoTButtonSNSTopic"  
        }  
    ]  
}
```

Don't Allow Allow



- c. Reemplace el documento de política por lo siguiente:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "logs:CreateLogGroup",  
                "logs:CreateLogStream",  
                "logs:PutLogEvents"  
            ],  
            "Resource": "arn:aws:logs:*:*:  
        },  
        {  
            "Effect": "Allow",  
            "Action": [  
                "sns:Publish"  
            ],  
            "Resource": "arn:aws:sns:us-east-1:123456789012:MyIoTButtonSNSTopic"  
        }  
    ]  
}
```

Este documento de política agrega permiso para publicar en su tema de Amazon SNS.

Note

Reemplace el valor del segundo Resource por el ARN del tema Amazon SNS que ha creado anteriormente.

10. Elija Allow.

AWS Lambda requires access to your resources

AWS Lambda uses an IAM role that grants your custom code permissions to access AWS resources it needs.

▼ Hide Details

**Role Summary**

**Role** Lambda execution role permissions

**Description**

**IAM Role** lambda\_basic\_execution

**Policy Name** Create a new Role Policy

▼ Hide Policy Document

Edit

```
"Version": "2012-10-17",
"Statement": [
{
    "Effect": "Allow",
    "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
    ],
    "Resource": "arn:aws:logs:*.:*"
},
```

Don't Allow

11. Deje la configuración de la página Advanced settings en sus valores predeterminados y seleccione Next.

**Advanced settings**

These settings allow you to control the code execution performance and costs for your Lambda function. Changing your resource settings (by selecting memory) or changing the timeout may impact your function cost. [Learn more](#) about how Lambda pricing works.

Memory (MB)\*

Timeout\*  min  sec

AWS Lambda will automatically retry failed executions for asynchronous invocations. You can additionally optionally configure Lambda to forward payloads that were not processed to a dead-letter queue (DLQ), such as an SQS queue or an SNS topic. Learn more about Lambda's [retry policy](#) and [DLQs](#). Please ensure your role has appropriate permissions to access the DLQ resource.

DLQ Resource

All AWS Lambda functions run securely inside a default system-managed VPC. However, you can optionally configure Lambda to access resources, such as databases, within your custom VPC. Learn more about accessing VPCs within Lambda. Please ensure your role has appropriate permissions to configure VPC.

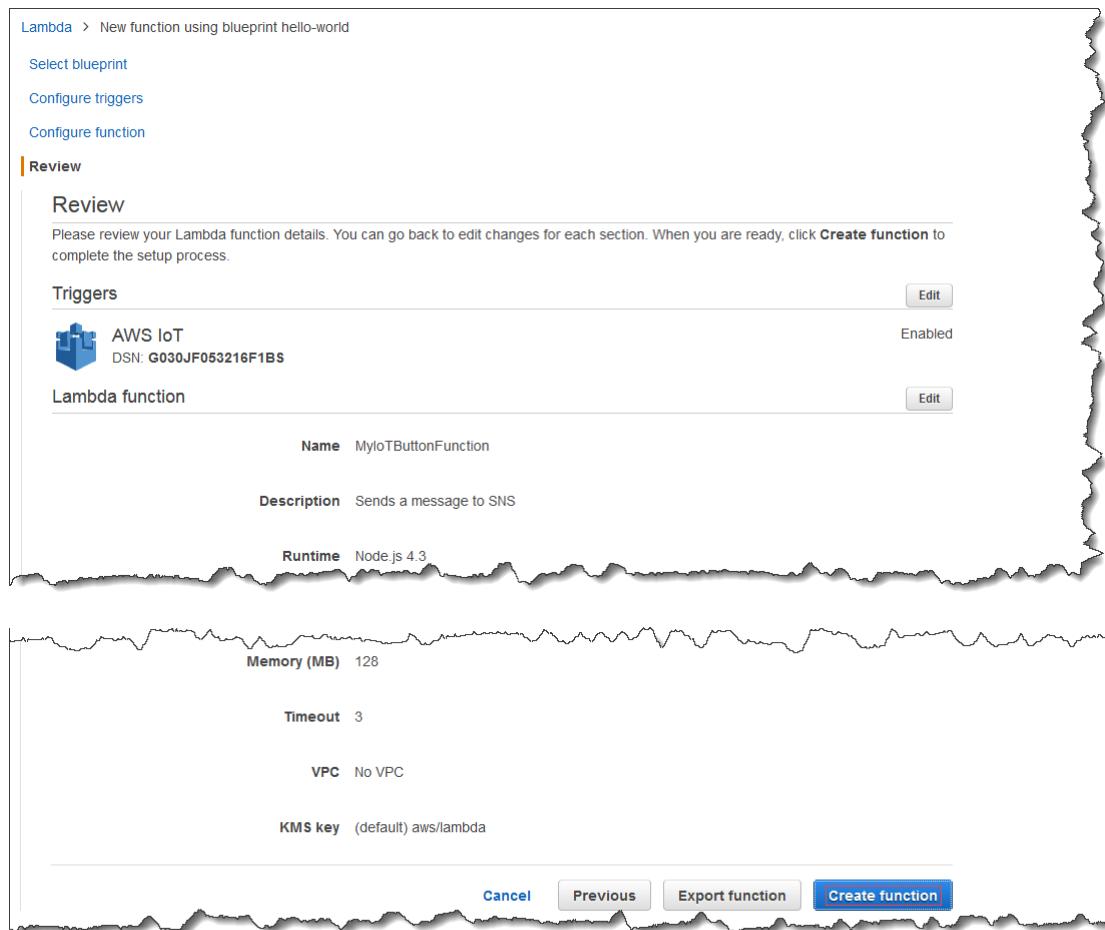
VPC

Environment variables are encrypted at rest using a default Lambda service key. You can change the key below to one of your account's keys or paste in a full KMS key ARN.

KMS key

\* These fields are required.

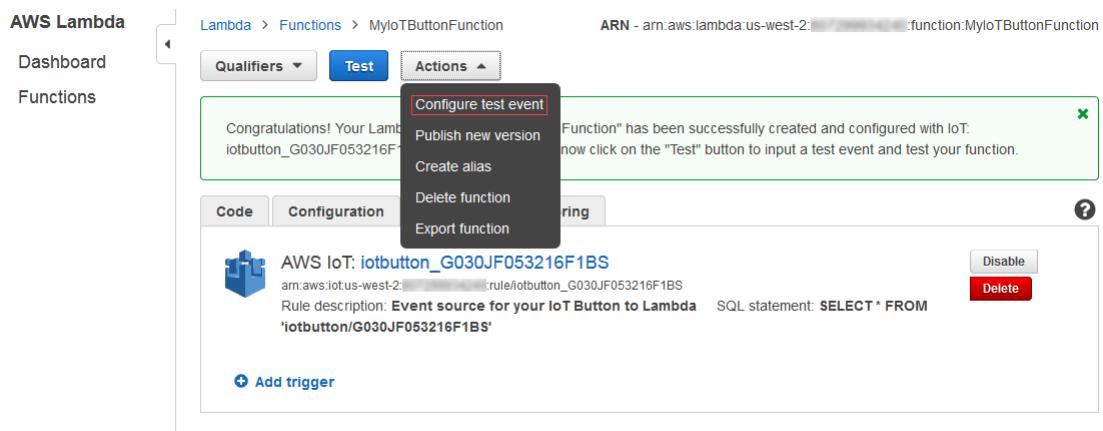
12. En la página Review, seleccione Create function.



## Comprobación de la función Lambda

Para probar la función Lambda:

1. En el menú Actions, seleccione Configure test event.



2. Copie y pegue los siguientes JSON en la página Input test event y, a continuación, seleccione Save and test.

```
{  
    "serialNumber": "ABCDEFG12345",  
    "clickType": "SINGLE",  
    "batteryVoltage": "2000 mV"  
}
```

The screenshot shows the 'Input test event' configuration page in the AWS Lambda console. At the top, there is a sample event template labeled 'Hello World'. Below it is a code editor containing the provided JSON. The 'batteryVoltage' field is highlighted with a blue selection bar. At the bottom right, there are three buttons: 'Cancel', 'Save', and a blue 'Save and test' button.

```
1 {  
2     "serialNumber": "ABCDEFG12345",  
3     "clickType": "SINGLE",  
4     "batteryVoltage": "2000 mV"  
5 }
```

3. En la consola de AWS Lambda desplácese a la parte inferior de la página. La sección Log output muestra el resultado que la función Lambda ha escrito en la consola.

### Log output

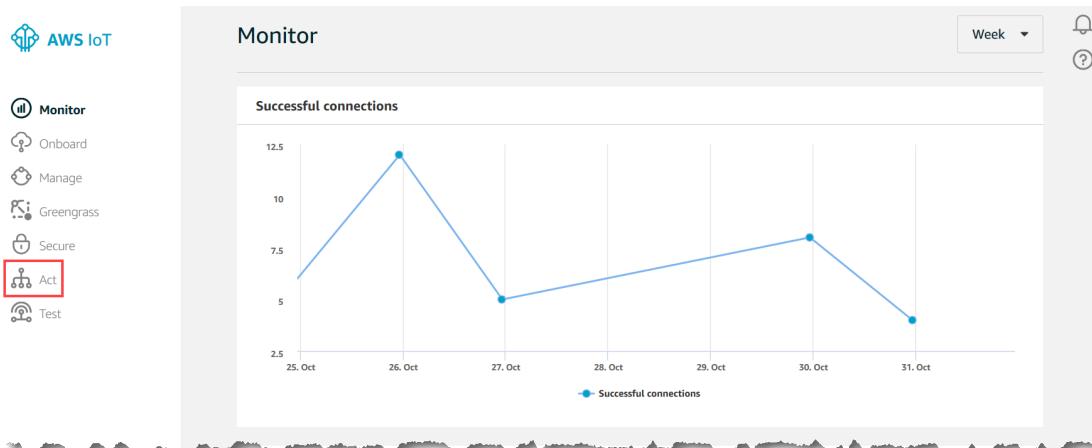
The area below shows the logging calls in your code. These correspond to a single row within the CloudWatch log group corresponding to this Lambda function. [Click here](#) to view the CloudWatch log group.

```
START RequestId: c4b5b4d1-1631-11e6-b78f-0d4d596724ad Version: $LATEST
2016-05-09T22:02:49.501Z      c4b5b4d1-1631-11e6-b78f-0d4d596724ad      Received event: {
    "serialNumber": "ABCDEFG12345",
    "clickType": "SINGLE",
    "batteryVoltage": "2000 mV"
}
2016-05-09T22:02:49.501Z      c4b5b4d1-1631-11e6-b78f-0d4d596724ad      Message to send: Received
END RequestId: c4b5b4d1-1631-11e6-b78f-0d4d596724ad
REPORT RequestId: c4b5b4d1-1631-11e6-b78f-0d4d596724ad Duration: 1215.14 ms      Billed Duration: 13
```

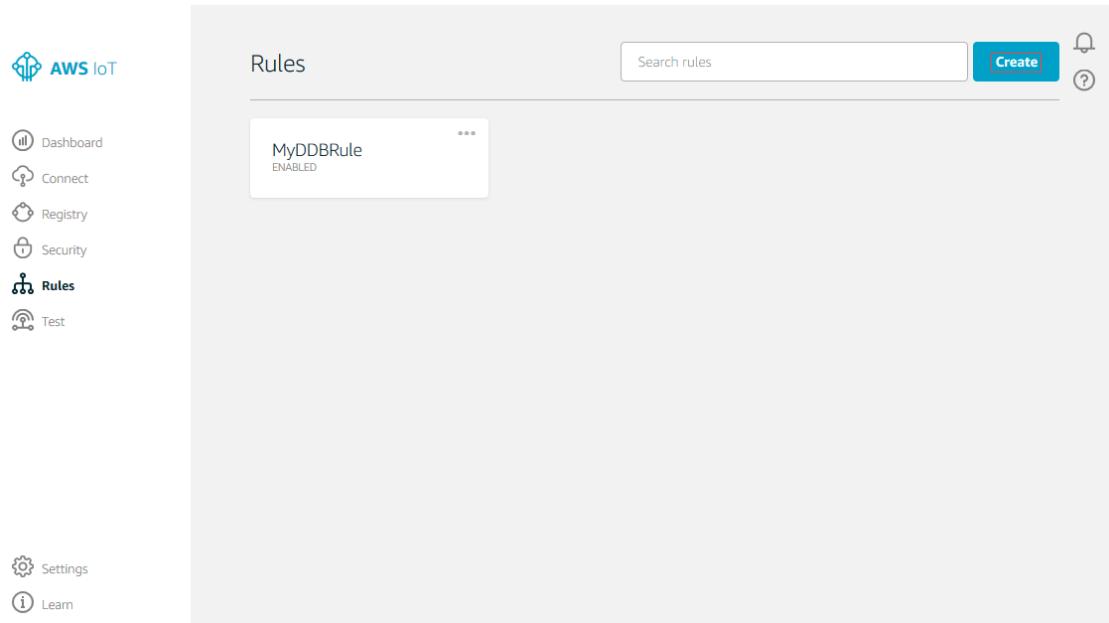
## Creación de un regla Lambda

Ahora que ha creado una función Lambda puede crear una regla que invoque la función Lambda.

1. En el panel de navegación izquierdo de la [consola de AWS IoT](#), elija Rules.



2. En la página Rules, elija Create.



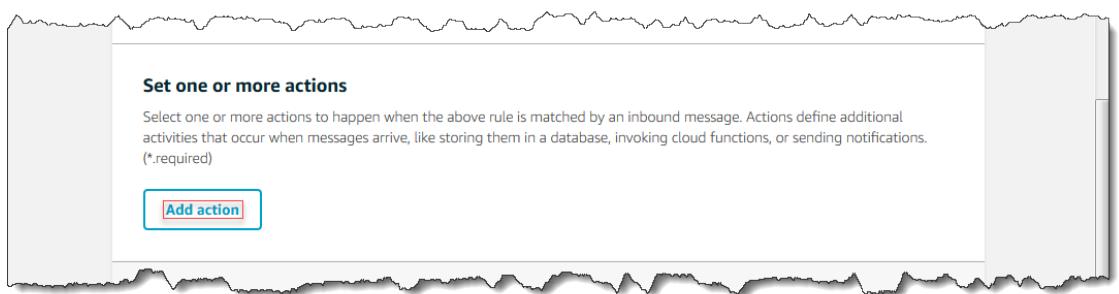
3. Escriba un nombre y la descripción de la regla.

The screenshot shows the 'Create a rule' dialog box. It has a blue header bar with the title 'Create a rule'. Below the header, there's a descriptive text: 'Create a rule to evaluate messages sent by your things and specify what to do when a message is received (for example, write data to a DynamoDB table or invoke a Lambda function.)'. There are two input fields: 'Name' containing 'MyLambdaRule' and 'Description' containing 'Invokes a Lambda function'.

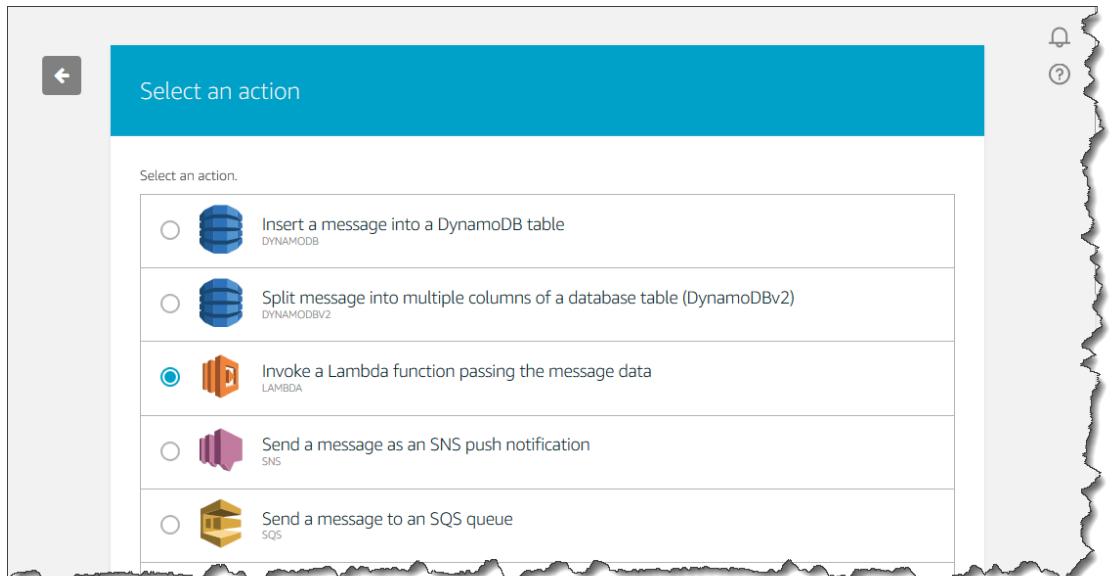
4. Especifique la siguiente configuración de la regla:

The screenshot shows the 'Message source' configuration page. It includes fields for 'Using SQL version' (set to 2016-03-23), 'Rule query statement' (containing 'SELECT \* FROM `iotbutton/+`'), 'Attribute' (containing '\*'), 'Topic filter' (containing 'iotbutton/+'), and 'Condition' (containing 'e.g. temperature > 75'). Below these fields is a section titled 'Set one or more actions'.

5. En Set one or more actions, elija Add action.

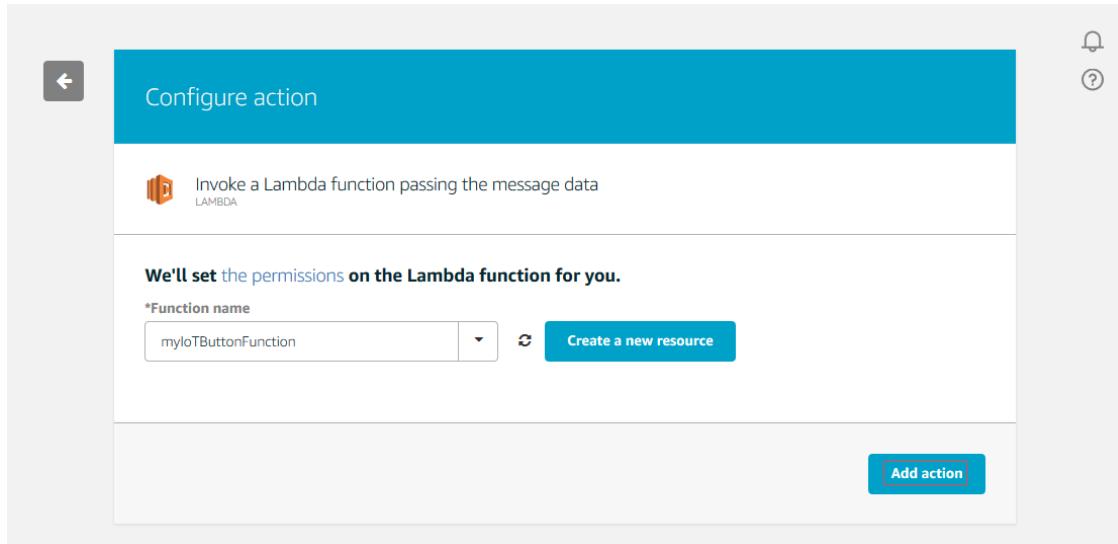


6. En la página Select an action, seleccione Invoke a Lambda function passing the message data y, a continuación, elija Configure action.

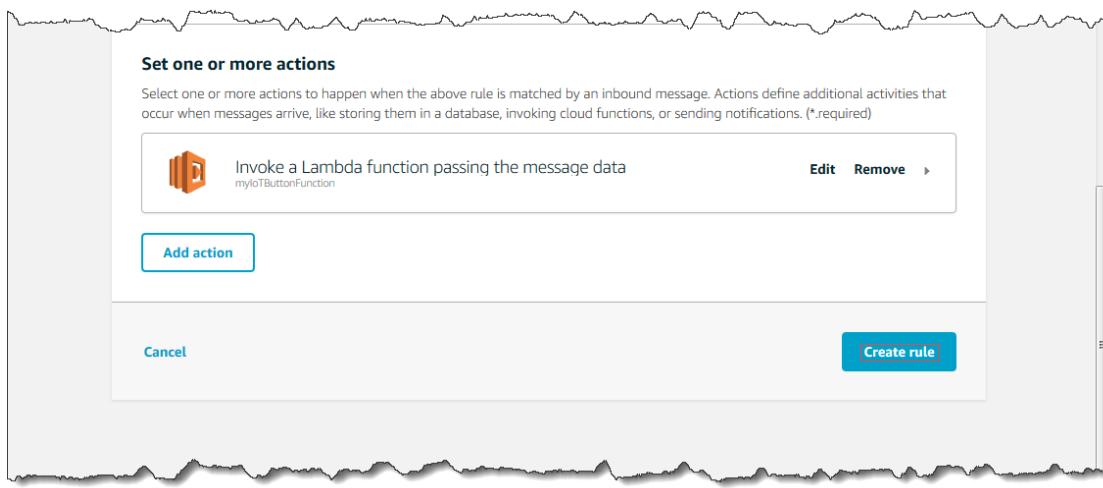




7. En la lista desplegable Function name, seleccione el nombre de la función Lambda y, a continuación, seleccione Add action.



8. Elija Create rule para crear su función Lambda.



## Comprobación de la regla Lambda

En este tutorial, presuponemos que ha completado el tutorial de introducción a AWS IoT (p. 5), que abarca:

- Configuración de un botón AWS IoT.
- Creación y suscripción de un tema de Amazon SNS con un número de teléfono móvil.

Ahora que el botón ya está configurado y conectado a la wifi y que usted ha configurado un tema de Amazon SNS, puede pulsar el botón para probar su regla Lambda. Debería recibir un mensaje de texto SMS en el teléfono que contenga:

- El número de serie del botón.
- El tipo de pulsación del botón (UNA SOLA o DOBLE).
- El voltaje de la batería.

El mensaje debe ser similar a lo siguiente:

```
IOT BUTTON> {  
    "serialNumber" : "ABCDEFG12345",  
    "clickType" : "SINGLE",  
    "batteryVoltage" : "2000 mV"  
}
```

Si no dispone de botón, [puede comprar uno aquí](#) o puede utilizar el cliente MQTT de AWS IoT en su lugar.

1. En la [consola de AWS IoT](#), seleccione Test.



2. En la página MQTT client, en la sección Publish, en Specify a topic, escriba **iotbutton/ABCDEFG12345**.

En Payload, escriba el siguiente JSON y, a continuación, elija Publish to topic.

```
{  
    "serialNumber" : "ABCDEFG12345",  
    "clickType" : "SINGLE",  
    "batteryVoltage" : "2000 mV"  
}
```



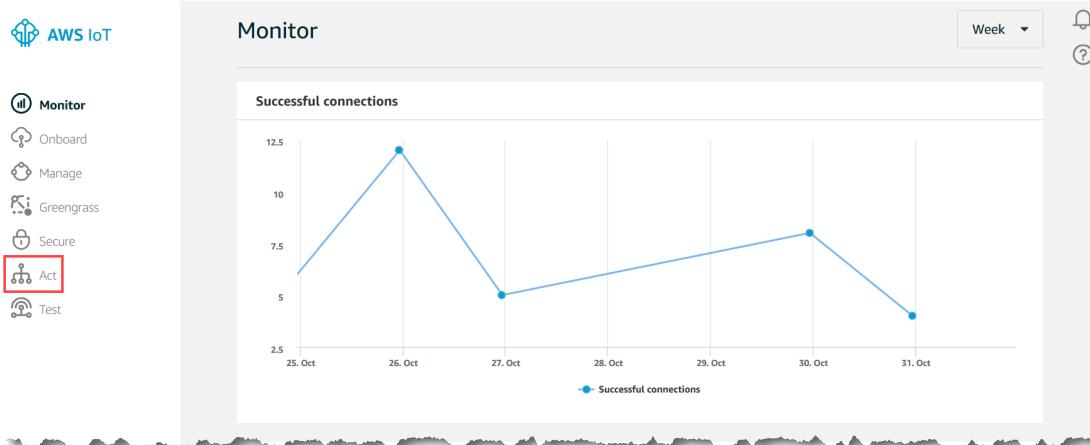
3. Debería recibir un mensaje en su teléfono móvil.

## Creación de un regla de Amazon SNS

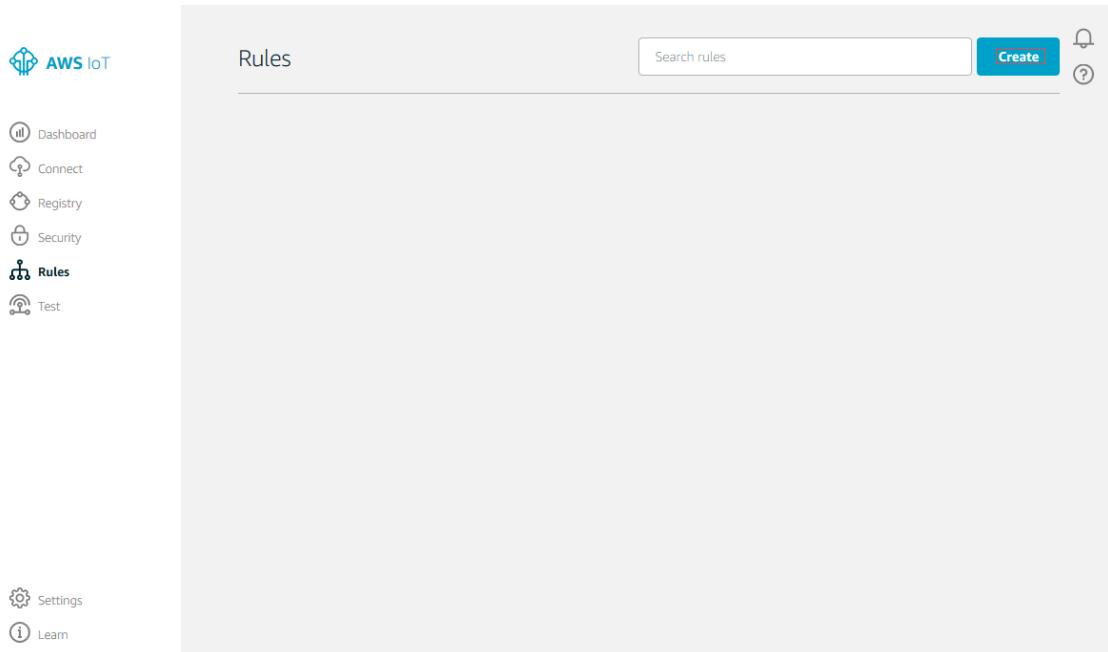
Puede definir una regla que envíe los datos de los mensajes a un tema de Amazon SNS. En este tutorial, creará una regla que envíe el nombre del objeto de AWS IoT que activó la regla a todos los suscriptores de un tema de Amazon SNS.

Para crear una regla con una acción de SNS:

1. En el panel de navegación izquierdo de la [consola de AWS IoT](#), elija Rules.



2. En la página Rules, elija Create.



3. Escriba un nombre para la regla.

The screenshot shows the 'Create a rule' wizard. It has a teal header bar with the text 'Create a rule'. Below it, there's a descriptive text: 'Create a rule to evaluate messages sent by your things and specify what to do when a message is received (for example, write data to a DynamoDB table or invoke a Lambda function)'. The 'Name' field is filled with 'MySNSRule' and is highlighted with a red border. The 'Description' field is empty.

4. En el apartado Attribute de Message source, escriba \*, topic(3). En Topic filter, escriba \$aws/things/+shadow/update/accepted. El filtro de temas especifica los temas que, cuando se publique un mensaje en ellos, activarán la acción de la regla. El signo + utilizado en el filtro de temas es un carácter comodín que coincide con cualquier nombre. El atributo añade el nombre del objeto al contenido del mensaje.

**Message source**

Indicate the source of the messages you want to process with this rule.

**Using SQL version**

2016-03-23

**Rule query statement**

```
SELECT *, topic(3) as thing FROM '$aws/things/+shadow/update/accepted'
```

**Attribute**

\*; topic(3) as thing

**Topic filter**

\$aws/things/+shadow/update/accepted

**Condition**

e.g. temperature > 75

- 
5. En la sección Set one or more actions, elija Add action.

**Set one or more actions**

Select one or more actions to happen when the above rule is matched by an inbound message. Actions define additional activities that occur when messages arrive, like storing them in a database, invoking cloud functions, or sending notifications. (\*.required)

**Add action**

6. En Select an action, seleccione Send a message as an SNS push notification y, a continuación, elija Configure action. (Este botón no se muestra en la captura de pantalla).

## Select an action

Select an action.

-  Insert a message into a DynamoDB table  
DYNAMODB
-  Split message into multiple columns of a database table (DynamoDBv2)  
DYNAMODBV2
-  Invoke a Lambda function passing the message data  
LAMBDA
-  Send a message as an SNS push notification  
SNS
-  Send a message to an SQS queue  
SQS
-  Sends messages to an Amazon Kinesis stream  
AMAZON KINESIS
-  Republish messages to an AWS IoT topic  
AWS IOT REPUBLISH
-  Send messages to an IoT metric  
IOT METRICS

7. Elija Create new topic.

Topics

Publish to topic  Actions ▾

Filter

Name	ARN
CFN-notifications	arn:aws:sns:us-west-2:803981987763:CFN-notifications
CFNEmailNotification	arn:aws:sns:us-west-2:803981987763:CFNEmailNotification
CloudFormationNotifications	arn:aws:sns:us-west-2:803981987763:CloudFormationNotifications
DDB-EXPORT-ProductCat...	arn:aws:sns:us-west-2:803981987763:DDB-EXPORT-ProductCatalog-1397236041808fBEFNVC1L
DDB-Import-Thread-1397...	arn:aws:sns:us-west-2:803981987763:DDB-Import-Thread-139721293000fKCbKloud
ElasticBeanstalkNotificati...	arn:aws:sns:us-west-2:803981987763:ElasticBeanstalkNotifications-sampleElasticBeanstalkApplication-python123-AWSEBLoadBalancer
ElasticBeanstalkNotificati...	arn:aws:sns:us-west-2:803981987763:ElasticBeanstalkNotifications-sampleElasticBeanstalkApplication-testestest-AWSEBAutoScalingGroup
ElasticBeanstalkNotificati...	arn:aws:sns:us-west-2:803981987763:ElasticBeanstalkNotifications-sampleElasticBeanstalkApplication-testestest-AWSEBLoadBalancer
ElasticBeanstalkNotificati...	arn:aws:sns:us-west-2:803981987763:ElasticBeanstalkNotifications-sdtest1-sdtest1-AWSEBAutoScalingGroup
ElasticBeanstalkNotificati...	arn:aws:sns:us-west-2:803981987763:ElasticBeanstalkNotifications-sdtest1-sdtest1-AWSEBLoadBalancer
LambdaEmailAlert	arn:aws:sns:us-west-2:803981987763:LambdaEmailAlert
MyTopic	arn:aws:sns:us-west-2:803981987763:MyTopic

Total Items: 33  
Selected Items: 0

8. Se abrirá una nueva pestaña en el navegador. Escriba un nombre y una descripción de su tema SNS y, luego, seleccione Create topic.

Create new topic

A topic name will be used to create a permanent unique identifier called an Amazon Resource Name (ARN).

Topic name	MySNSTopic	i
Display name	IoT SNS	i

Cancel **Create topic**

9. Cambie a la pestaña del navegador donde está abierta la consola de AWS IoT. En SNS target, seleccione el tema de SNS que acaba de crear. En Message format, elija JSON.

Configure action

SNS target: MySNSTopic

Message format: JSON

Choose or create a role to grant AWS IoT access to the SNS resource to perform this action.

IAM role name: Choose a role

Add action

The screenshot shows the AWS IoT Rule Configuration interface. It starts with a 'Create new topic' section where 'Topic name' is set to 'MySNSTopic' and 'Display name' is 'IoT SNS'. A red box highlights the 'Create topic' button. Next, it shows a 'Configure action' section for an 'SNS push notification' rule. The 'SNS target' dropdown is set to 'MySNSTopic' (highlighted by a red box) and the 'Create a new resource' button is visible. The 'Message format' dropdown is set to 'JSON' (highlighted by a red box). Below this, there's a role configuration section with an 'IAM role name' dropdown set to 'Choose a role', an 'Update role' button, and a 'Create a new role' button. At the bottom right of the configuration area is an 'Add action' button.

10. En IAM role name, elija Create a new role.

## Configure action

 Send a message as an SNS push notification  
SNS

\*SNS target  
MySNSTopic

Message format  
JSON

Choose or create a role to grant AWS IoT access to the SNS resource to perform this action.

\*IAM role name  
Choose a role

Add action

11. Escriba un nombre para la función y, a continuación, seleccione Create a new role.

## Configure action

SNS Send a message as an SNS push notification

\*SNS target MySNSTopic

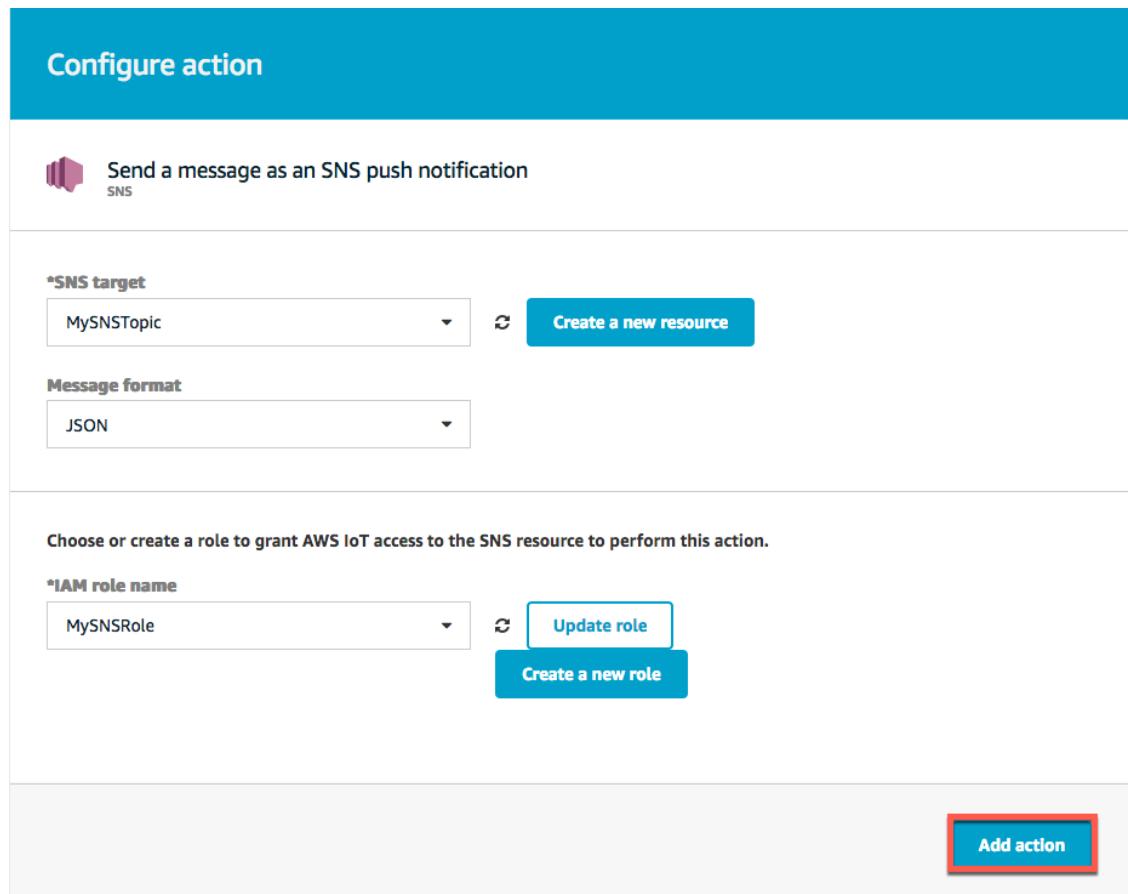
Message format JSON

Choose or create a role to grant AWS IoT access to the SNS resource to perform this action.

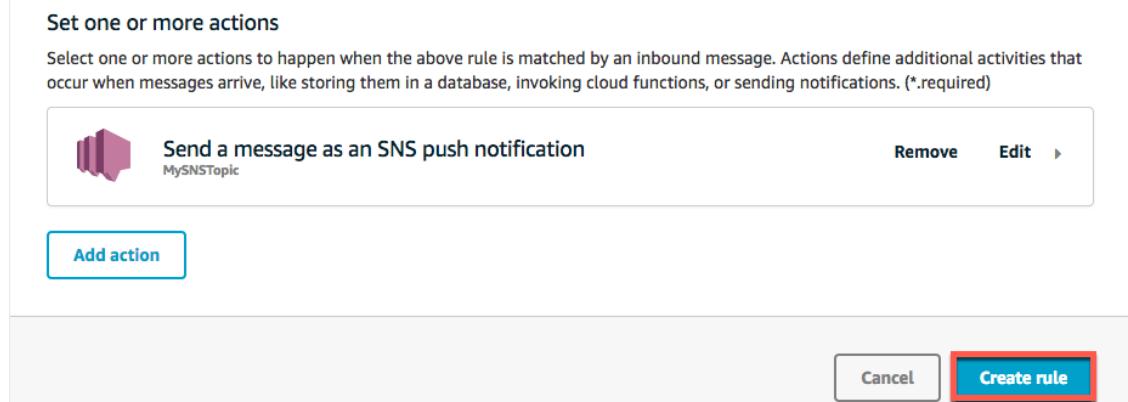
\*IAM role name MySNSRole

Add action

12. Seleccione el rol que acaba de crear y después elija Add action.



13. Elija Create rule.



Ha creado la regla. Para probarla, añada una suscripción al tema de SNS que ha creado y actualice la sombra de objeto de cualquier objeto de AWS IoT. Puede utilizar la consola de AWS IoT para buscar un objeto, abrir su página de detalles y cambiar la sombra del objeto. Cuando se notifique el cambio al servicio Thing Shadow, este servicio publicará un mensaje en \$aws/things/*MySNSThing*/shadow/update/accepted. Se activará la regla y todos los suscriptores del tema de SNS recibirán un mensaje con el nombre del objeto.

# Tutoriales de SDK de AWS IoT

Los SDK de dispositivos de AWS IoT le ayudan a conectar con facilidad y rapidez sus dispositivos a AWS IoT. Los SDK de dispositivos de AWS IoT contienen bibliotecas de código abierto, guías de desarrolladores con ejemplos y guías de migración para que pueda crear productos o soluciones de IoT innovadores en las plataformas de hardware deseadas.

En esta guía, se proporcionan instrucciones paso a paso para conectar Raspberry Pi a la plataforma de AWS IoT y configurarlo para su uso con AWS IoT SDK para Embedded C y Device SDK para Javascript. Después de seguir los pasos de esta guía, podrá conectarse a la plataforma de AWS IoT y ejecutar aplicaciones de ejemplo incluidas en estos SDK de AWS IoT.

## Contenido

- [Conexión de Raspberry Pi \(p. 78\)](#)
- [Uso de AWS IoT SDK para Embedded C \(p. 87\)](#)
- [Uso de AWS IoT Device SDK para JavaScript \(p. 90\)](#)

## Conexión de Raspberry Pi

Siga los pasos siguientes para conectar su Raspberry Pi a la plataforma de AWS IoT.

## Requisitos previos

- Placa Raspberry Pi totalmente configurada con acceso a Internet

Para obtener información acerca de cómo configurar Raspberry Pi, consulte [Raspberry Pi Quickstart Guide](#).

- Navegador Chrome o Firefox (Iceweasel)

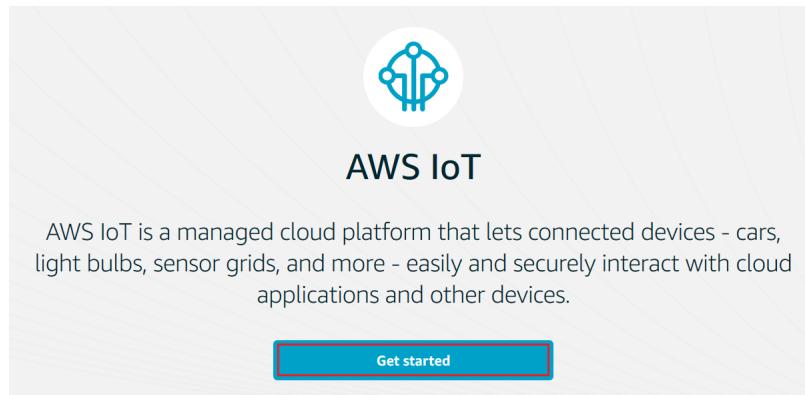
Para obtener más información acerca de la instalación de Iceweasel, consulte [las instrucciones en la wiki de Linux integrado](#).

En esta guía, se utilizan el hardware y el software siguientes:

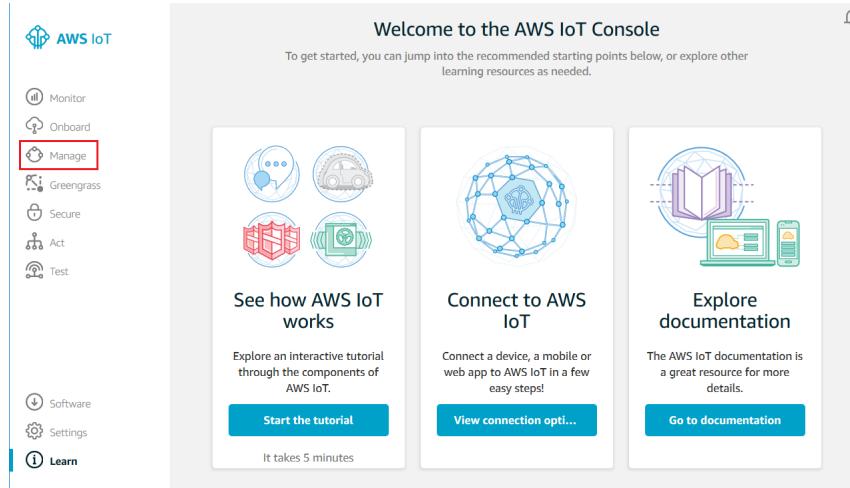
- Raspberry Pi 2 Modelo B
- Raspbian Wheezy
- Raspbian Jessie
- Navegador Iceweasel

## Inicio de sesión en la consola de AWS IoT

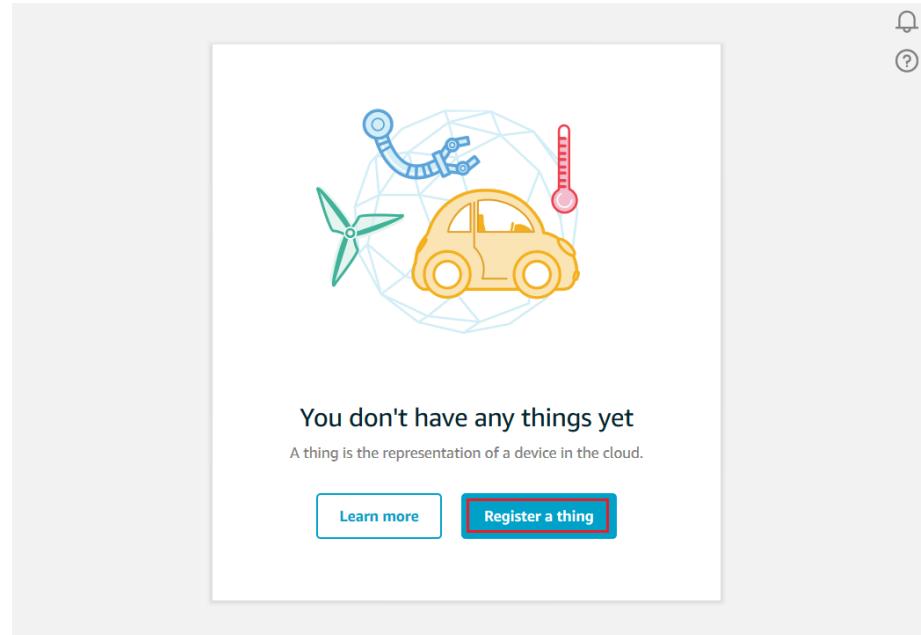
1. Encienda su Raspberry Pi y confirme que dispone de una conexión a Internet.
2. Inicie sesión en la consola de administración de AWS y abra la consola de AWS IoT en <https://aws.amazon.com/iot>. En la página Welcome, elija Get started.



3. Si es la primera vez que utiliza la consola de AWS IoT, verá la página Welcome to the AWS IoT Console. En el panel de navegación izquierdo, elija Registry para ampliar las opciones y, a continuación, elija Things.



4. En la página donde se indica You don't have any things yet, elija Register a thing. (Si ha creado un objeto antes, elija Create).



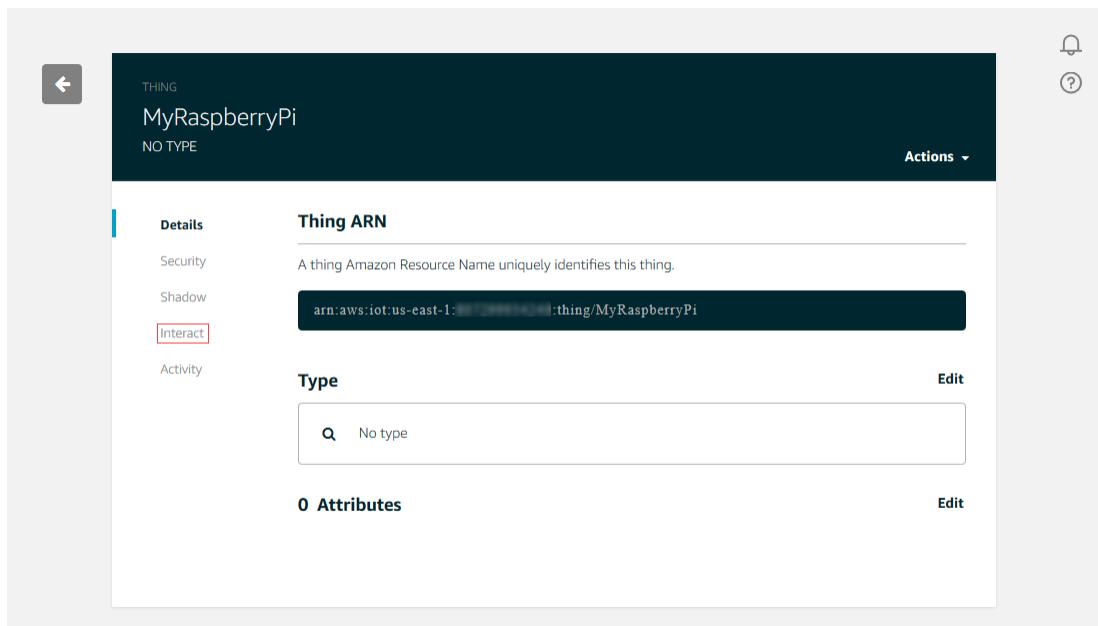
## Cree y asocie un objeto (dispositivo)

Un objeto representa un dispositivo cuyo estado o datos se almacenan en la nube de AWS. El servicio Thing Shadows mantiene una sombra de objeto por cada dispositivo conectado a AWS IoT. Thing Shadows le permite obtener acceso a los datos de estado del objeto y modificarlos.

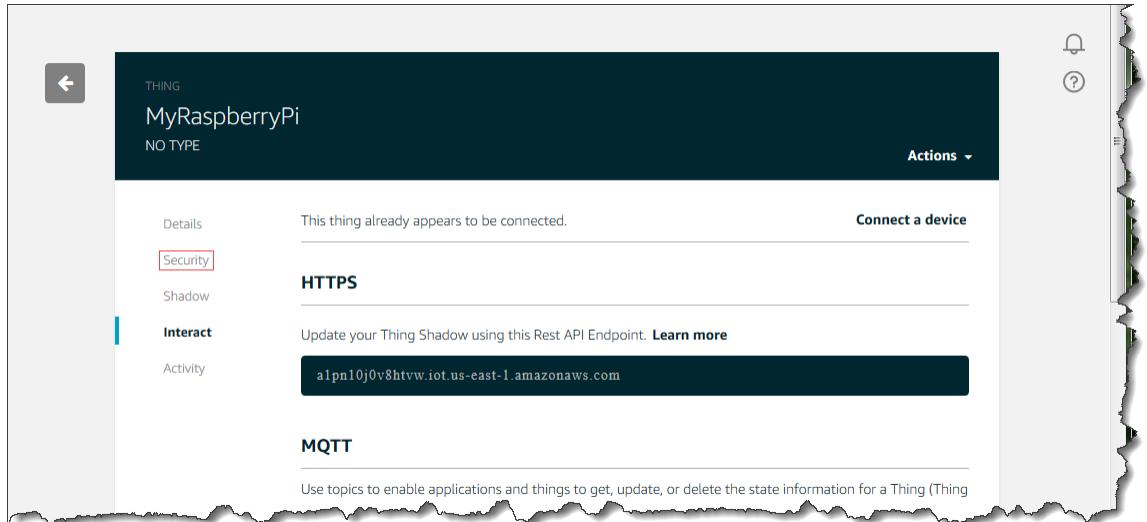
1. Escriba un nombre para el objeto y, a continuación, elija Create thing.

This screenshot shows the 'Register a thing' step in the AWS IoT console. The title bar says 'Register a thing'. Below it, a sub-instruction reads 'This step creates an entry in the thing registry and a thing shadow for your device.' A 'Name' input field contains 'MyRaspberryPi'. There is also a 'Show options ▾' button. In the bottom right corner of the form area is a blue 'Create thing' button.

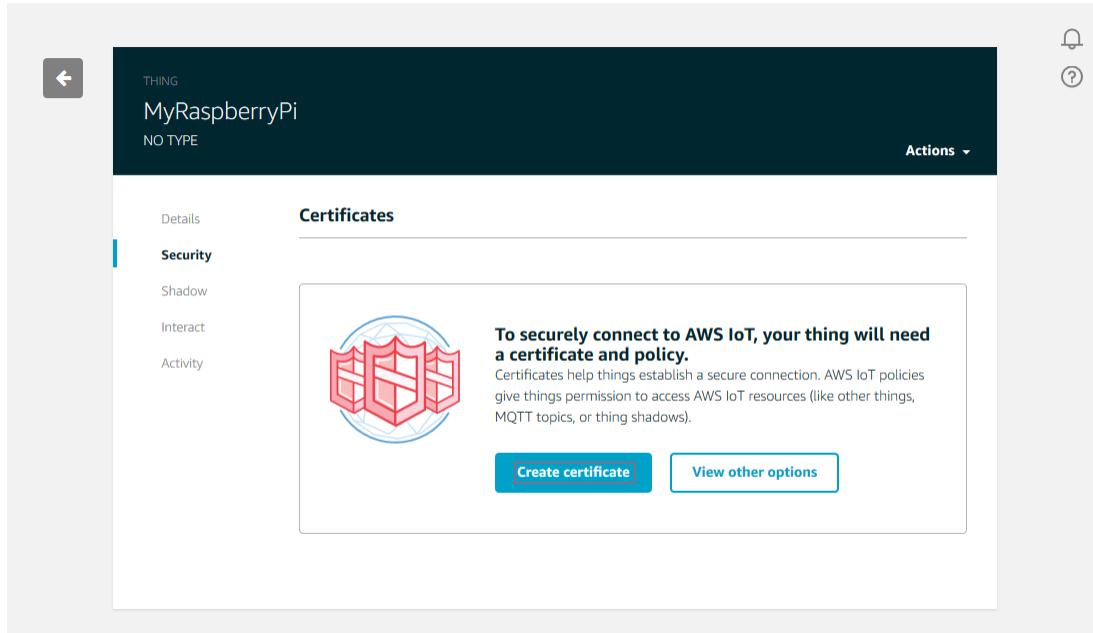
2. En la página Details, elija Interact.



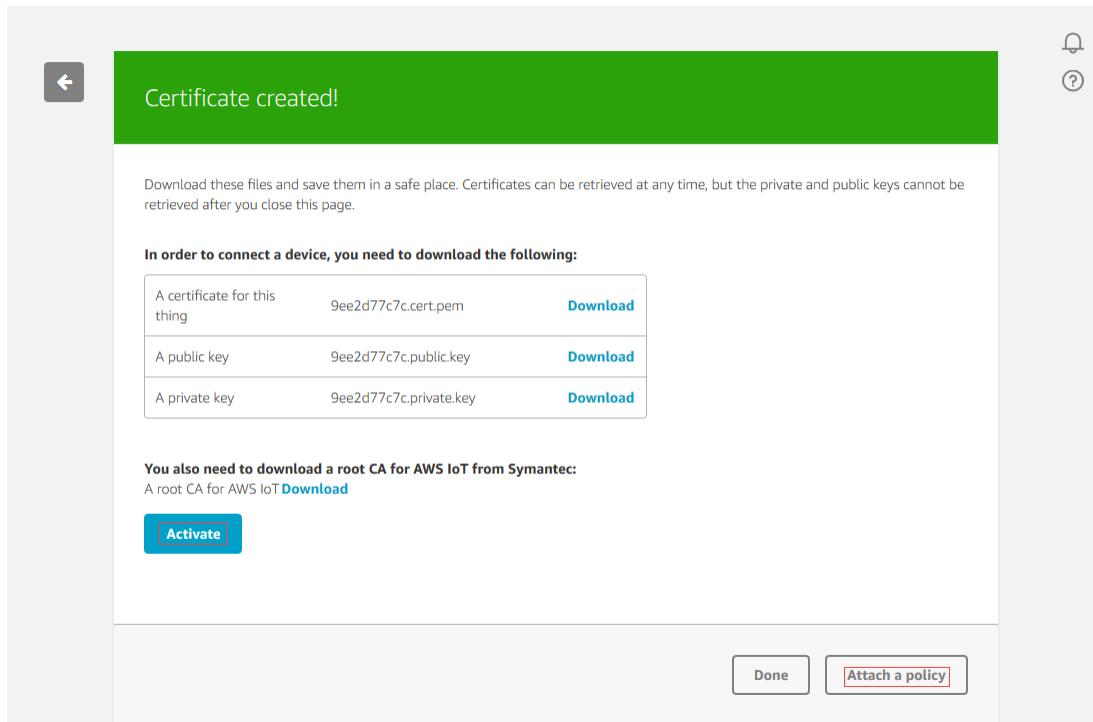
3. Tome nota del punto de enlace de la API REST. Necesitará este valor más tarde. Elija Security.



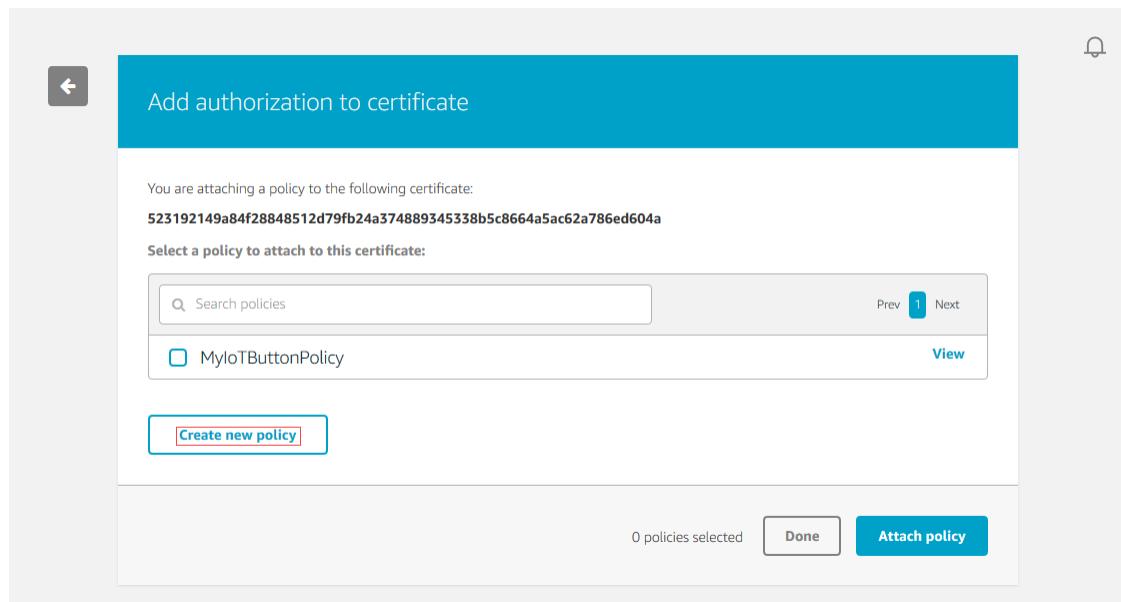
4. Elija Create certificate. Esto generará un certificado X.509 y un par de claves.



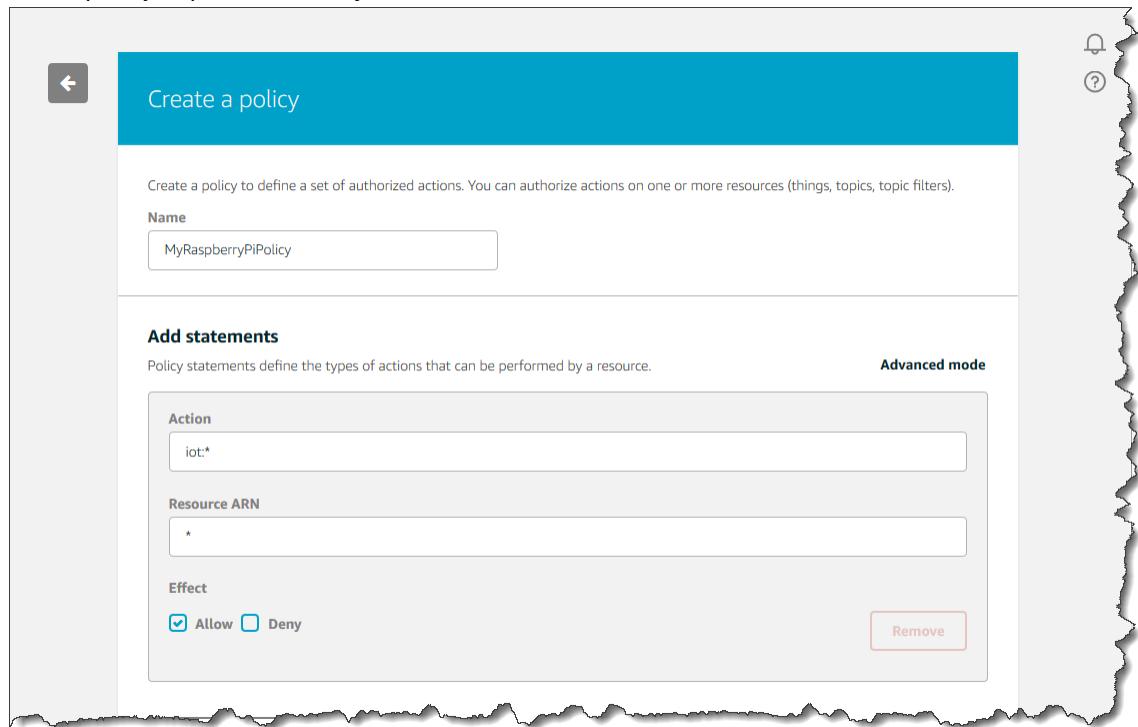
- Cree un directorio de trabajo denominado `deviceSDK` donde se almacenarán los archivos. Elija los enlaces para descargar sus claves públicas y privadas, el certificado y la CA raíz, y guárdelos en el directorio `deviceSDK`. Elija **Activate** para activar el certificado X.509 y, a continuación, elija **Attach a policy**.



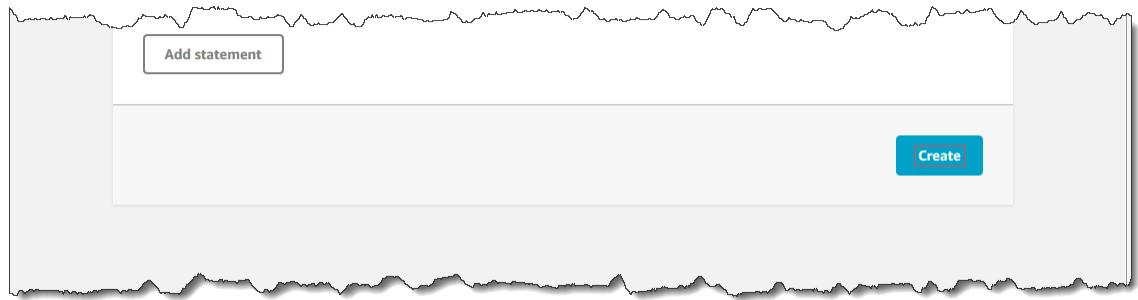
- Elija **Create new policy**.



7. En la página Create a policy, en el campo Name, escriba un nombre para la política. En el campo Action, escriba `iot:*`. En el campo Resource ARN, escriba `*`. Marque la casilla Allow. Esto permite a su Raspberry Pi publicar mensajes en AWS IoT.



8. Seleccione Create.

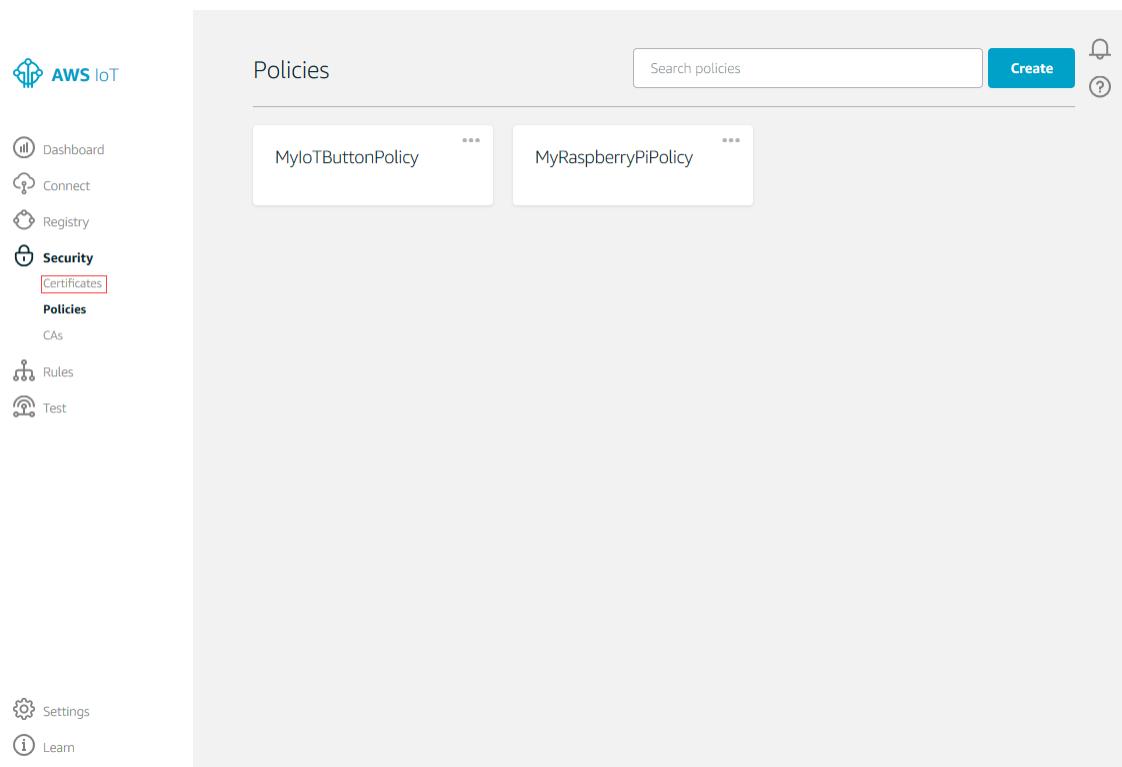


9. Seleccione la flecha izquierda para regresar a la página Policies.

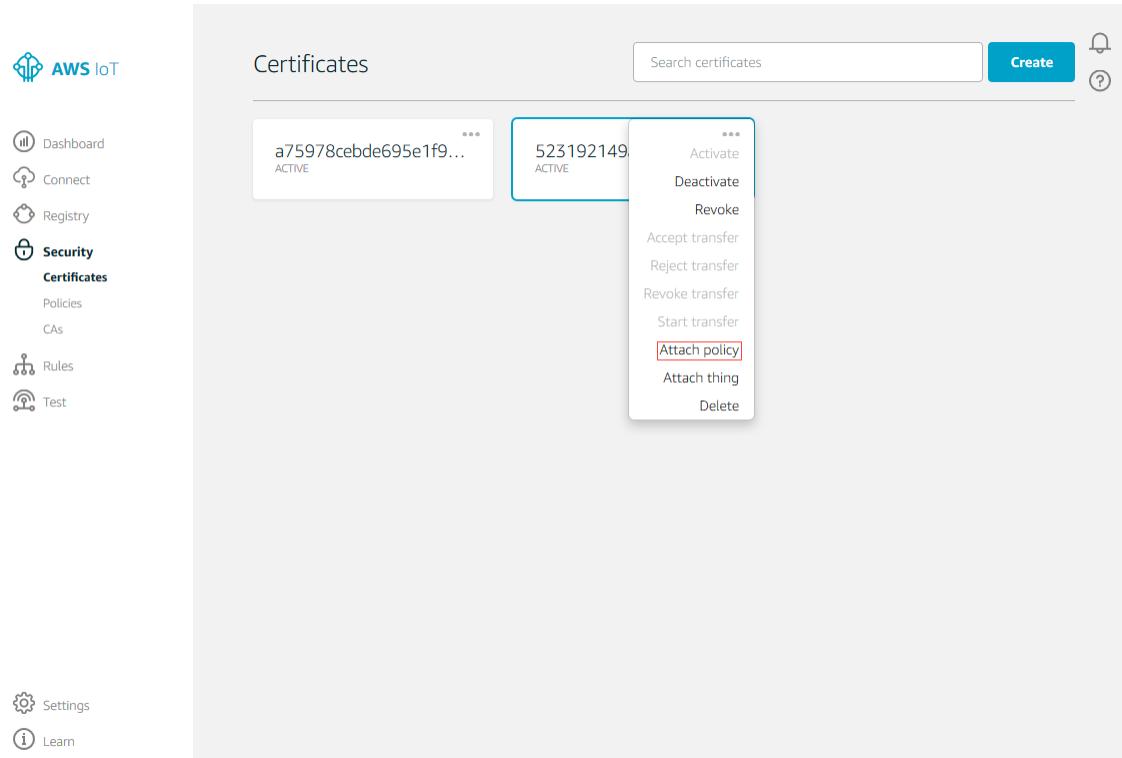
The screenshot shows the AWS IoT Policies page for the policy "MyRaspberryPiPolicy". The "Overview" tab is selected. The "Policy ARN" section displays the unique identifier: `arn:aws:iot:us-east-1:123456789012:policy/MyRaspberryPiPolicy`. The "Policy document" section shows the policy definition with a "Version 1 updated" message and an "Edit policy document" link. The JSON code for the policy is as follows:

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

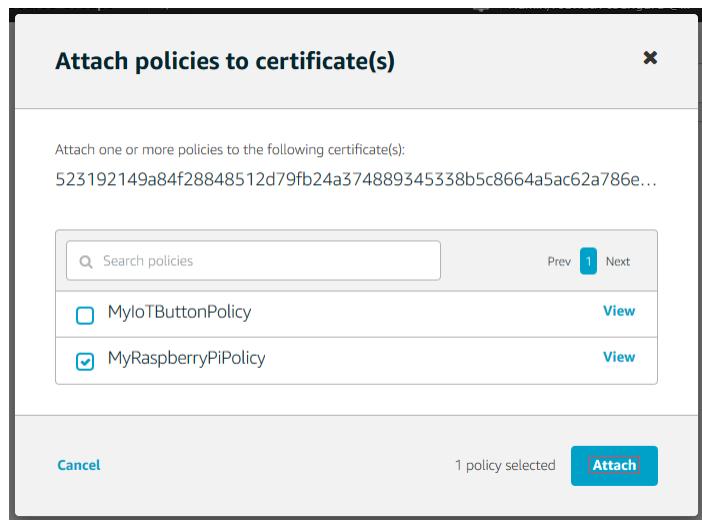
10. En la sección Security del panel de navegación izquierdo, elija Certificates.



11. En la casilla del certificado que ha creado, elija ... para abrir un menú desplegable y, a continuación, elija Attach policy.

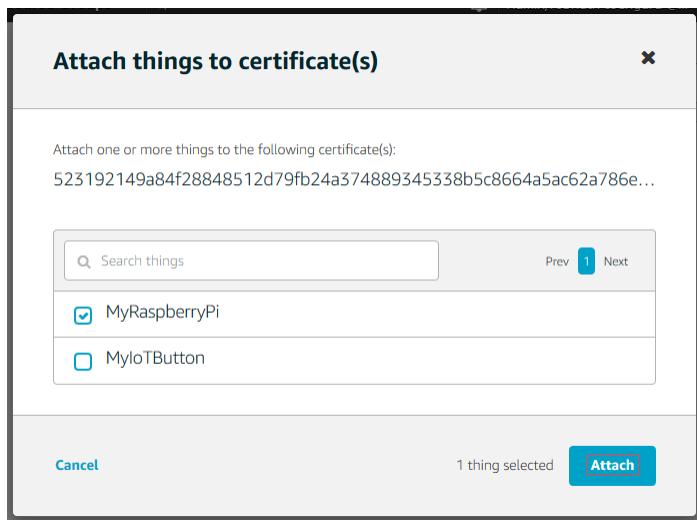


12. En el cuadro de diálogo Attach policies to certificate(s), marque la casilla de verificación junto a la política que ha creado y, a continuación, seleccione Attach.



13. En la casilla del certificado que ha creado, elija ... para abrir un menú desplegable y, a continuación, elija Attach thing.

14. En el cuadro de diálogo Attach things to certificate(s), marque la casilla situada junto al objeto que ha creado para representar su Raspberry Pi y, a continuación, elija Attach.



## Uso de AWS IoT SDK para Embedded C

Existen dos versiones de AWS IoT SDK para Embedded C: OpenSSL y mbed TLS. Usaremos la versión OpenSSL.

### Configuración del entorno de tiempo de ejecución para AWS IoT SDK para Embedded C

1. Descargue el [AWS IoT Device SDK para Embedded C](#) en un archivo tar (`linux_mqtt_openssl-latest.tar`). GUárdelo en su directorio `deviceSDK`.
2. En una ventana de la terminal, escriba el siguiente comando para extraer el archivo tar en su directorio `deviceSDK`:  
`tar -xvf linux_mqtt_openssl-latest.tar`
3. Para poder utilizar AWS IoT SDK para Embedded C, debe instalar la biblioteca de OpenSSL en Raspberry Pi. En una ventana de terminal, ejecute  
`sudo apt-get install libssl-dev.`

### Configuración de aplicación de muestra

AWS IoT SDK para Embedded C contiene aplicaciones de muestra para que las pruebe. Para simplificar las cosas, vamos a ejecutar `subscribe_publish_sample`.

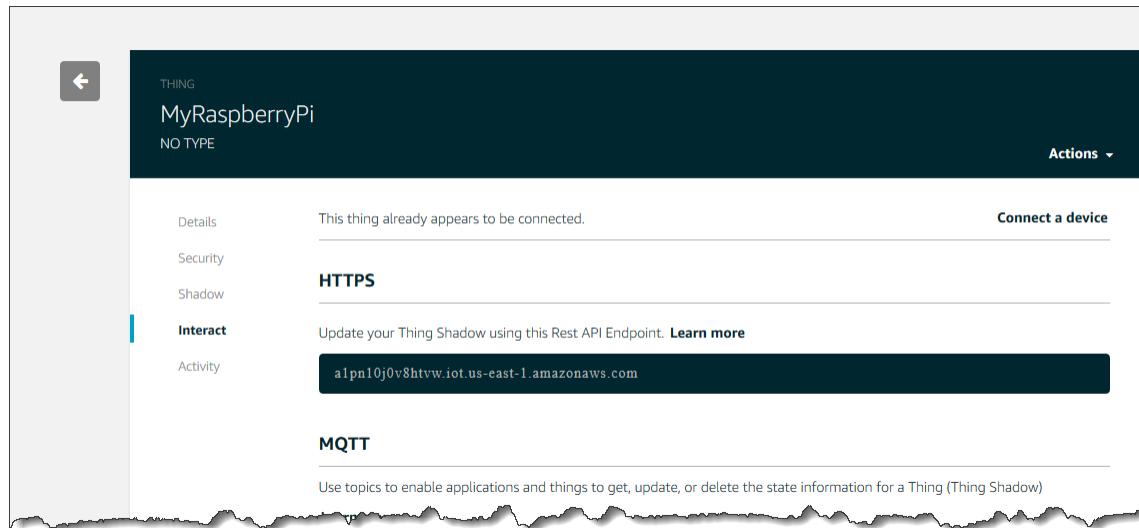
1. Copie el certificado, la clave privada y el certificado CA raíz en el directorio `deviceSDK/certs`.

Si no obtiene una copia del certificado CA raíz, puede descargarlo [aquí](#). Copie el texto de la CA raíz desde el navegador, péguelo en un archivo y, a continuación, cópielo en el directorio `deviceSDK/certs`.

#### Note

Los certificados de dispositivo y de CA raíz están sujetos a vencimiento o revocación. Si esto se produce, deberá copiar un certificado de CA nuevo o un nuevo certificado de dispositivo y clave privada en su dispositivo.

- Vaya al directorio `deviceSDK/sample_apps/subscribe_publish_sample`. Deberá configurar el punto de enlace personal, la clave privada y el certificado. El punto de enlace personal es el punto de enlace de la API REST que ha anotado antes. Si no recuerda el punto de enlace y tiene acceso a una máquina con la AWS CLI instalada, puede utilizar el comando `aws iot describe-endpoint` para encontrar la URL del punto de enlace personal. O bien, vaya a la consola de AWS IoT. Elija Registry, seleccione Things y, a continuación, elija el objeto que represente su Raspberry Pi. En la página Details del objeto, en el panel de navegación izquierdo, elija Interact. Copie todo, incluso ".com", de REST API endpoint.



- Abra el archivo `aws_iot_config.h` y, en la sección `//Get from console`, actualice los valores de los elementos siguientes:

`AWS_IOT_MQTT_HOST`

El punto de enlace personal.

`AWS_IOT_MY_THING_NAME`

El nombre del objeto.

`AWS_IOT_ROOT_CA_FILENAME`

El certificado de CA raíz.

`AWS_IOT_CERTIFICATE_FILENAME`

Su certificado.

`AWS_IOT_PRIVATE_KEY_FILENAME`

Su clave privada.

Por ejemplo:

```
// Get from console
// =====
#define AWS_IOT_MQTT_HOST      "a22j5sm6o3yzc5.iot.us-east-1.amazonaws.com"
#define AWS_IOT_MQTT_PORT      8883
#define AWS_IOT_MQTT_CLIENT_ID "MyRaspberryPi"
#define AWS_IOT_MY_THING_NAME "MyRaspberryPi"
#define AWS_IOT_ROOT_CA_FILENAME "root-CA.crt"
#define AWS_IOT_CERTIFICATE_FILENAME "4bbdc778b9-certificate.pem.crt"
#define AWS_IOT_PRIVATE_KEY_FILENAME "4bbdc778b9-private.pem.key"
```

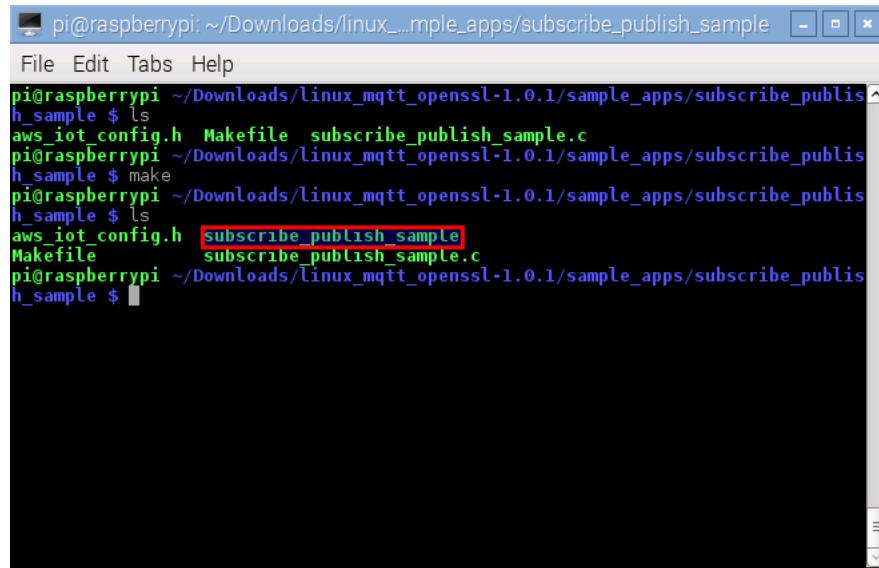
```
// =====
```

## Ejecución de las aplicaciones de muestra

1. Compile `subscribe_publish_sample_app` con el archivo Makefile incluido.

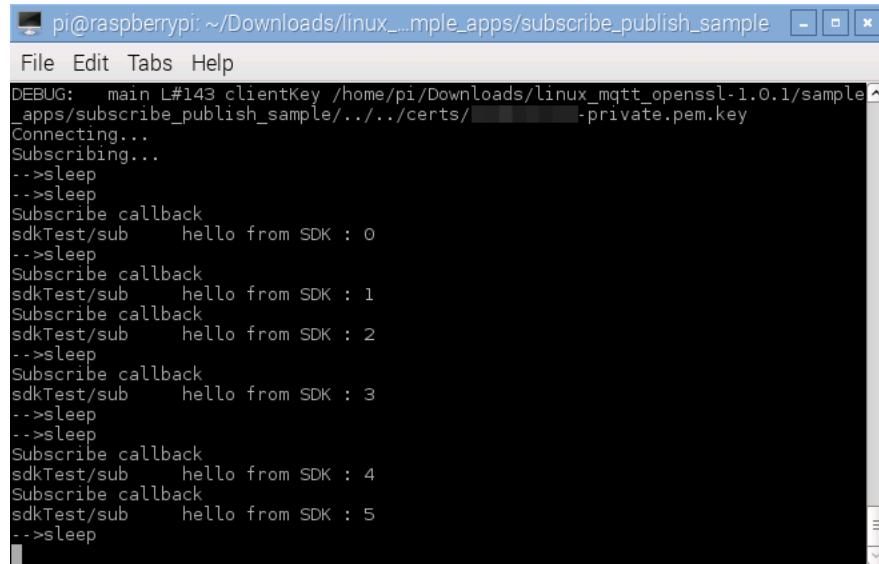
```
make -f Makefile
```

Esto generará un archivo ejecutable.



```
pi@raspberrypi:~/Downloads/linux_mqtt_openssl-1.0.1/sample_apps/subscribe_publish_sample$ make
pi@raspberrypi:~/Downloads/linux_mqtt_openssl-1.0.1/sample_apps/subscribe_publish_sample$ ls
aws_iot_config.h  Makefile  subscribe_publish_sample.c
pi@raspberrypi:~/Downloads/linux_mqtt_openssl-1.0.1/sample_apps/subscribe_publish_sample$ make
pi@raspberrypi:~/Downloads/linux_mqtt_openssl-1.0.1/sample_apps/subscribe_publish_sample$ ls
aws_iot_config.h  subscribe_publish_sample
Makefile          subscribe_publish_sample.c
pi@raspberrypi:~/Downloads/linux_mqtt_openssl-1.0.1/sample_apps/subscribe_publish_sample$
```

2. Ahora, ejecute `subscribe_publish_sample_app`. Debería ver un resultado similar a este:



```
pi@raspberrypi:~/Downloads/linux_mqtt_openssl-1.0.1/sample_apps/subscribe_publish_sample$ ./subscribe_publish_sample
DEBUG: main L#143 clientKey /home/pi/Downloads/linux_mqtt_openssl-1.0.1/sample_apps/subscribe_publish_sample/../../certs/XXXXXXXXXX-private.pem.key
Connecting...
Subscribing...
-->sleep
-->sleep
Subscribe callback
sdkTest/sub    hello from SDK : 0
-->sleep
Subscribe callback
sdkTest/sub    hello from SDK : 1
Subscribe callback
sdkTest/sub    hello from SDK : 2
-->sleep
Subscribe callback
sdkTest/sub    hello from SDK : 3
-->sleep
-->sleep
Subscribe callback
sdkTest/sub    hello from SDK : 4
Subscribe callback
sdkTest/sub    hello from SDK : 5
-->sleep
```

Su Raspberry Pi ya está conectado a AWS IoT mediante AWS IoT Device SDK para Embedded C.

# Uso de AWS IoT Device SDK para JavaScript

La forma más sencilla de instalar AWS IoT Device SDK para Node.js es utilizar npm. En esta sección, describimos cómo instalar Node y npm.

## Configuración del entorno de tiempo de ejecución para AWS IoT Device SDK para JavaScript

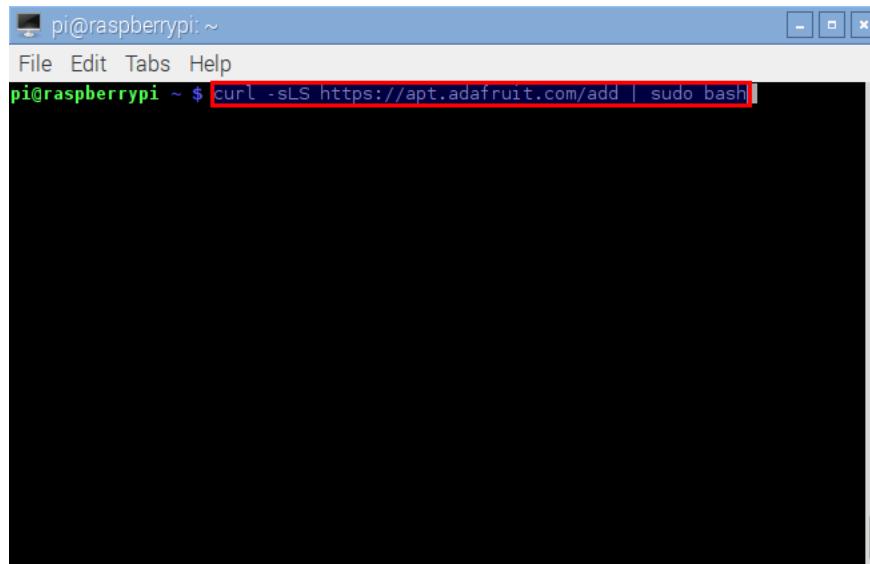
Para utilizar AWS IoT Device SDK para JavaScript, debe instalar Node y la herramienta de desarrollo npm en su Raspberry Pi. Estos paquetes no se instalan de forma predeterminada.

### Note

Antes de continuar, es posible que desee configurar el mapeo del teclado para su Raspberry Pi. Para obtener más información, consulte [Configure Raspberry Pi Keyboard Mapping](#).

1. Para agregar el repositorio Node, abra un terminal y ejecute el siguiente comando:

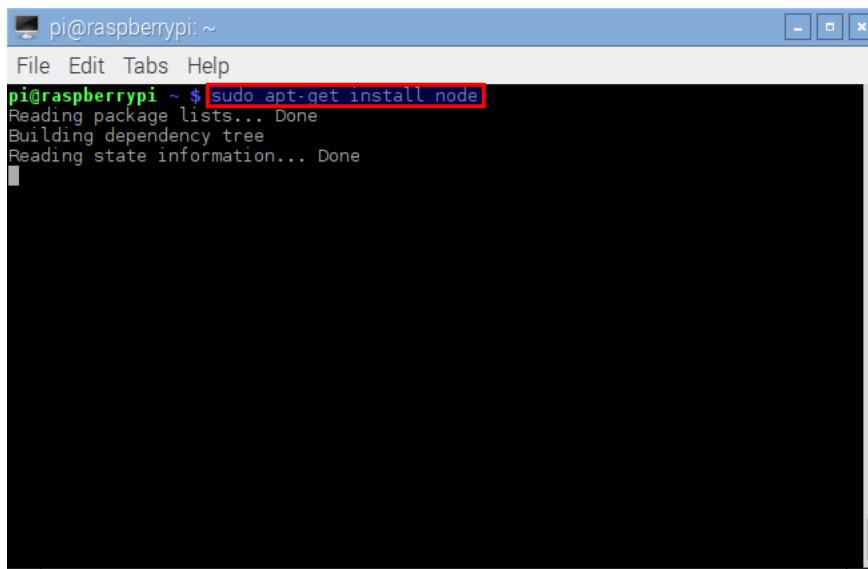
```
curl -sLS https://apt.adafruit.com/add | sudo bash
```



2. Para instalar Node, ejecute

```
sudo apt-get install node
```

Debería ver un resultado similar a este:



pi@raspberrypi: ~

File Edit Tabs Help

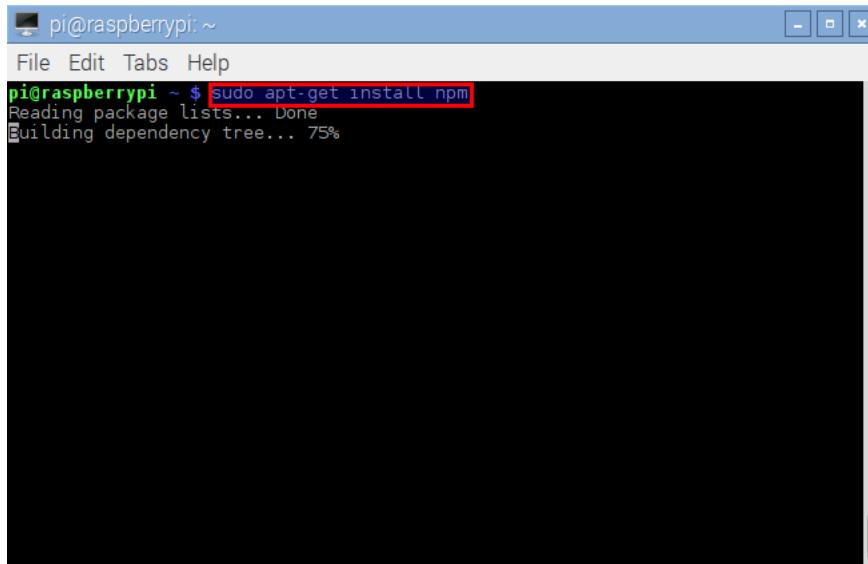
pi@raspberrypi ~ \$ sudo apt-get install node

Reading package lists... Done  
Building dependency tree  
Reading state information... Done

3. Para instalar npm, ejecute

```
sudo apt-get install npm
```

Debería ver un resultado similar a este:



pi@raspberrypi: ~

File Edit Tabs Help

pi@raspberrypi ~ \$ sudo apt-get install npm

Reading package lists... Done  
Building dependency tree... 75%

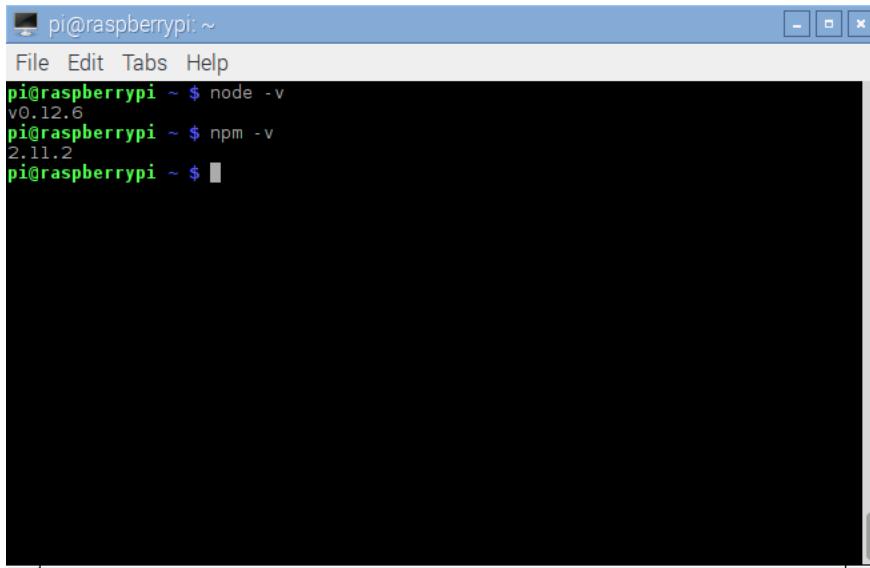
4. Para verificar la instalación de Node y npm, ejecute

```
node -v
```

y

```
npm -v
```

Debería ver un resultado similar a este:



## Instalación de AWS IoT Device SDK para JavaScript

Para instalar AWS IoT Device SDK para JavaScript/Node.js en su Raspberry Pi, abra una ventana de la consola y, desde su directorio ~/deviceSDK, use npm para instalar el SDK:

```
npm install aws-iot-device-sdk
```

Una vez realizada la instalación, debe encontrar un directorio `node_modules` en su directorio ~/deviceSDK.

## Preparación antes de ejecutar las aplicaciones de muestra

AWS IoT Device SDK para JavaScript contiene aplicaciones de muestra para que las pruebe. Para ejecutarlas, debe configurar sus certificados y la clave privada.

Edite el archivo ~/deviceSDK/node\_modules/aws-iot-device-sdk/examples/lib/cmdline.js para cambiar los nombres predeterminados de la clave privada (`privateKey`), el certificado (`clientCert`) y el certificado CA raíz (`caCert`) utilizado por los ejemplos. Por ejemplo:

```
default: {  
  region: 'us-east-1',  
  clientId: clientIdDefault,  
  privateKey: '4bbdc778b9-private.pem.key',  
  clientCert: '4bbdc778b9-certificate.pem.crt',  
  caCert: 'root-CA.crt',  
  testMode: 1,  
  reconnectPeriod: 3 * 1000, /* milliseconds */  
  delay: 4 * 1000 /* milliseconds */  
};
```

## Ejecución de las aplicaciones de muestra

Ejecute los ejemplos utilizando

```
node examples/<YourDesiredExample>.js -f <certs location>
```

Si usted se encuentra en el directorio `~/deviceSDK/node_modules/aws-iot-device-sdk/`, la ubicación de los certificados debería ser `~/deviceSDK/certs/`. Para obtener más información sobre cómo ejecutar las opciones de la línea de comandos y especificar la ubicación de los certificados y su propia dirección de host, consulte [Certificates](#).

Si quiere crear un archivo de configuración para usarlo con la opción de línea de comandos `--configuration-file (-F)`, cree un archivo (en formato JSON) con las siguientes propiedades. Por ejemplo:

```
{  
    "host": "a22j5sm6o3yzc5.iot.us-east-1.amazonaws.com"  
    "port": 8883  
    "clientId": "MyRaspberryPi"  
    "thingName": "MyRaspberryPi"  
    "caCert": "root-CA.crt"  
    "clientCert": "4bbdc778b9-certificate.pem.crt"  
    "privateKey": "4bbdc778b9-private.pem.key"  
}
```

Ahora, su Raspberry Pi ya está conectado a AWS IoT mediante AWS IoT SDK para JavaScript.

# Administración de objetos con AWS IoT

AWS IoT proporciona un registro de objetos que le ayuda a administrar sus objetos. Un objeto es una representación de un dispositivo concreto o de una entidad lógica. Puede ser un dispositivo físico o un sensor (por ejemplo, una bombilla o un interruptor en la pared). También puede ser una entidad lógica, como una instancia de una aplicación o una entidad física que no se conecta con AWS IoT, pero que está relacionada con otros dispositivos que sí lo están (por ejemplo, un automóvil con sensores de motor o un panel de control).

La información sobre un objeto se almacena en el registro de objetos en forma de datos JSON. A continuación, se muestra un ejemplo de objeto:

```
{  
    "version": 3,  
    "thingName": "MyLightBulb",  
    "defaultClientId": "MyLightBulb",  
    "thingTypeName": "LightBulb",  
    "attributes": {  
        "model": "123",  
        "wattage": "75"  
    }  
}
```

Los objetos se identifican por su nombre. También pueden tener atributos, que son pares nombre-valor que puede utilizar para almacenar información acerca del objeto, como su número de serie o su fabricante.

En un caso de uso de dispositivo típico, se utiliza el nombre del objeto como ID de cliente MQTT predeterminado. Aunque no aplicamos un mapeo entre el nombre de registro de objetos y su uso de los ID de cliente MQTT, los certificados o los estados de sombra, le recomendamos elegir un nombre de objeto y utilizarlo como ID de cliente MQTT (tanto para el registro de objetos como para el servicio Thing Shadows). De esta forma, puede organizar su flota de IoT más fácilmente, sin perder la flexibilidad del modelo de certificado de dispositivo subyacente o las sombras de objeto.

No es necesario crear un objeto en el registro de objetos para conectarlo a AWS IoT. Agregar sus objetos al registro de objetos le permite administrarlos y buscarlos con más facilidad.

# Administración de objetos con el registro de objetos

La consola de AWS IoT o la CLI de AWS se utilizan para interactuar con el registro. En las secciones siguientes se muestra cómo utilizar la CLI para trabajar con el registro de objetos.

## Creación de un objeto

El comando siguiente muestra cómo utilizar el comando de CLI `create-thing` de AWS IoT para crear un objeto:

```
$ aws iot create-thing --thing-name "MyLightBulb" --attribute-payload "{\"attributes\":{\"wattage\":\"75\", \"model\":\"123\"}}"
```

La API `create-thing` mostrará el nombre y ARN del objeto nuevo:

```
{  
    "thingArn": "arn:aws:iot:us-east-1:803981987763:thing/MyLightBulb",  
    "thingName": "MyLightBulb"  
}
```

## Lista de objetos

Puede utilizar la API `list-things` para crear una lista de todos los objetos de su cuenta:

```
$ aws iot list-things  
{  
    "things": [  
        {  
            "attributes": {  
                "model": "123",  
                "wattage": "75"  
            },  
            "version": 1,  
            "thingName": "MyLightBulb"  
        },  
        {  
            "attributes": {  
                "numOfStates": "3"  
            },  
            "version": 11,  
            "thingName": "MyWallSwitch"  
        }  
    ]  
}
```

## Buscar objetos

Puede utilizar la API `describe-thing` para crear una lista de información sobre un objeto:

```
$ aws iot describe-thing --thing-name "MyLightBulb"  
{  
    "version": 3,  
    "thingName": "MyLightBulb",  
    "defaultClientId": "MyLightBulb",  
    "thingTypeName": "StopLight",  
    "attributes": {
```

```
        "model": "123",
        "wattage": "75"
    }
}
```

Puede utilizar la API `list-things` para buscar todos los objetos asociados a un nombre de tipo de objeto:

```
$ aws iot list-things --thing-type-name "LightBulb"
```

```
{
  "things": [
    {
      "thingTypeName": "LightBulb",
      "attributes": {
        "model": "123",
        "wattage": "75"
      },
      "version": 1,
      "thingName": "MyRGBLight"
    },
    {
      "thingTypeName": "LightBulb",
      "attributes": {
        "model": "123",
        "wattage": "75"
      },
      "version": 1,
      "thingName": "MySecondLightBulb"
    }
  ]
}
```

Puede utilizar la API `list-things` para buscar todos los objetos que tienen un atributo con un valor específico:

```
$ aws iot list-things --attribute-name "wattage" --attribute-value "75"
```

```
{
  "things": [
    {
      "thingTypeName": "StopLight",
      "attributes": {
        "model": "123",
        "wattage": "75"
      },
      "version": 3,
      "thingName": "MyLightBulb"
    },
    {
      "thingTypeName": "LightBulb",
      "attributes": {
        "model": "123",
        "wattage": "75"
      },
      "version": 1,
      "thingName": "MyRGBLight"
    }
  ]
}
```

```
        "thingTypeName": "LightBulb",
        "attributes": {
            "model": "123",
            "wattage": "75"
        },
        "version": 1,
        "thingName": "MySecondLightBulb"
    }
}
```

## Actualización de un objeto

Puede utilizar la API `update-thing` para actualizar un objeto:

```
$ aws iot update-thing --thing-name "MyLightBulb" --attribute-payload "{\"attributes\": {\"wattage\":\"150\", \"model\":\"456\"}}"
```

El comando `update-thing` no genera una salida. Puede utilizar la API `describe-thing` para ver el resultado:

```
$ aws iot describe-thing --thing-name "MyLightBulb"
{
    "attributes": {
        "model": "456",
        "wattage": "150"
    },
    "version": 2,
    "thingName": "MyLightBulb"
}
```

## Eliminación de un objeto

Puede utilizar la API `delete-thing` para eliminar un objeto:

```
$ aws iot delete-thing --thing-name "MyThing"
```

## Asociar un principal a un objeto

Un dispositivo físico debe tener un certificado X.509 para comunicarse con AWS IoT. Puede asociar el certificado de su dispositivo con el objeto del registro de objetos que representa a dicho dispositivo. Para asociar un certificado a su objeto, utilice la API `attach-thing-principal`:

```
$ aws iot attach-thing-principal --thing-name "MyLightBulb" --principal "arn:aws:iot:us-east-1:123456789012:cert/a0c01f5835079de0a7514643d68ef8414ab739a1e94ee4162977b02b12842847"
```

El comando `attach-thing-principal` no genera una salida.

## Desvincular un principal de un objeto

Puede utilizar la API `detach-thing-principal` para desvincular un certificado de un objeto:

```
$ aws iot detach-thing-principal --thing-name "MyLightBulb" --principal "arn:aws:iot:us-east-1:123456789012:cert/a0c01f5835079de0a7514643d68ef8414ab739a1e94ee4162977b02b12842847"
```

El comando `detach-thing-principal` no genera una salida.

## Tipos de objeto

Los tipos de objeto le permiten almacenar información descriptiva y de configuración común a todos los objetos asociados al mismo tipo de objeto. Esto simplifica la administración de objetos en el registro de objetos. Por ejemplo, puede definir un tipo de objeto LightBulb (bombilla). Todos los objetos asociados al tipo de objeto LightBulb comparten un conjunto de atributos: número de serie, fabricante y potencia. Al crear un objeto de tipo LightBulb (o cambiar el tipo a LightBulb) puede especificar valores para cada uno de los atributos definidos en este tipo de objeto.

Aunque los tipos de objeto son opcionales, su utilización permite detectar mejor los objetos.

- Los objetos con un tipo de objeto pueden tener un máximo de 50 atributos.
- Los objetos sin tipo de objeto pueden tener un máximo de tres atributos.
- Un objeto solo se puede asociar a un tipo de objeto.
- El número de tipos de objetos que puede crear en su cuenta es ilimitado.

Los tipos de objeto son inmutables. No se puede cambiar el nombre de un tipo de objeto después de que se haya creado. Puede descartar un tipo de objeto, en cualquier momento, para evitar que se le asocien nuevos objetos. También puede eliminar tipos de objeto que no tengan objetos asociados.

## Creación de un tipo de objeto

Puede utilizar la API `create-thing-type` para crear un tipo de objeto:

```
$ aws iot create-thing-type
    --thing-type-name "LightBulb" --thing-type-properties
    "thingTypeDescription=light bulb type, searchableAttributes=wattage,model"
```

El comando `create-thing-type` devuelve una respuesta que contiene el tipo de objeto y su ARN:

```
{
  "thingTypeName": "LightBulb",
  "thingTypeArn": "arn:aws:iot:us-west-2:803981987763:thingtype/LightBulb"
}
```

## Lista de los tipos de objeto

Puede utilizar la API `list-thing-types` para crear una lista de los tipos de objeto:

```
$ aws iot list-thing-types
```

El comando `list-thing-types` devuelve una lista de los tipos de objeto definidos en su cuenta de AWS:

```
{
  "thingTypes": [
    {
      "thingTypeName": "LightBulb",
      "thingTypeProperties": {
        "deprecated": false,
        "creationDate": 1468423800950,
```

```
        "searchableAttributes": [
            "wattage",
            "model"
        ],
        "thingTypeDescription": "light bulb type"
    }
}
```

## Descripción de un tipo de objeto

Puede utilizar la API `describe-thing-type` para obtener información sobre un tipo de objeto:

```
$ aws iot describe-thing-type --thing-type-name "LightBulb"
```

La API `describe-thing-type` responde con información sobre el tipo especificado:

```
{
    "thingTypeName": "LightBulb",
    "thingTypeProperties": {
        "deprecated": false,
        "creationDate": 1468423800950,
        "searchableAttributes": [
            "wattage",
            "model"
        ],
        "thingTypeDescription": "light bulb type"
    }
}
```

## Asociación de un tipo de objeto a un objeto

Puede utilizar la API `create-thing` para especificar un tipo de objeto cuando crea un objeto:

```
$ aws iot create-thing --thing-name "MySecondLightBulb" --thing-type-name "LightBulb" --
attribute-payload "{\"attributes\": {\"wattage\":\"75\", \"model\":\"123\"}}"
```

Puede utilizar la API `update-thing` en cualquier momento para cambiar el tipo de objeto asociado a un objeto:

```
$ aws iot update-thing --thing-name "MyLightBulb" --thing-type-name "StopLight" --
attribute-payload "{\"attributes\": {\"wattage\":\"75\", \"model\":\"123\"}}"
```

También puede utilizar la API `update-thing` para desvincular un objeto de un tipo de objeto.

## Descartar un tipo de objeto

Los tipos de objeto son inmutables. No se pueden cambiar una vez que están definidos. Sin embargo, puede descartar un tipo de objeto para evitar que los usuarios le asocien nuevos objetos. Todos los objetos que están asociados al tipo de objeto permanecerán inalterados.

Para descartar un tipo de objeto, utilice la API `deprecate-thing-type`:

```
$ aws iot deprecate-thing-type --thing-type-name "myThingType"
```

Puede utilizar la API `describe-thing-type` para ver el resultado:

```
$ aws iot describe-thing --thing-type-name "StopLight":
```

```
{  
    "thingTypeName": "StopLight",  
    "thingTypeProperties": {  
        "deprecated": true,  
        "creationDate": 1468425854308,  
        "searchableAttributes": [  
            "wattage",  
            "numOfLights",  
            "model"  
        ],  
        "thingTypeDescription": "traffic light type",  
        "deprecationDate": 1468446026349  
    }  
}
```

Descartar un tipo de objeto es una operación reversible. Puede anular un descarte utilizando la marca `--undo-deprecate` con el comando de CLI `deprecate-thing-type`:

```
$ aws iot deprecate-thing-type --thing-type-name "myThingType" --undo-deprecate
```

Puede utilizar el comando de CLI `deprecate-thing-type` para ver el resultado:

```
$ aws iot deprecate-thing-type --thing-type-name "StopLight":
```

```
{  
    "thingTypeName": "StopLight",  
    "thingTypeProperties": {  
        "deprecated": false,  
        "creationDate": 1468425854308,  
        "searchableAttributes": [  
            "wattage",  
            "numOfLights",  
            "model"  
        ],  
        "thingTypeDescription": "traffic light type"  
    }  
}
```

## Eliminación de un tipo de objeto

Puede eliminar tipos de objeto solo después de que se hayan descartado. Para eliminar un tipo de objeto, utilice la API `delete-thing-type`:

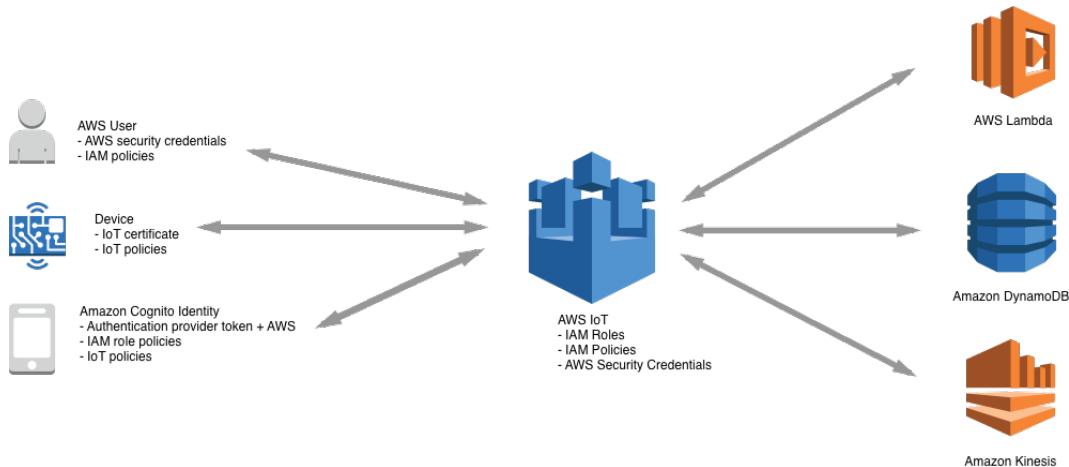
```
$ aws iot delete-thing-type --thing-type-name "StopLight"
```

### Note

Debe esperar cinco minutos después de descartar un tipo de objeto para poder eliminarlo.

# Seguridad e identidad para AWS IoT

Cada dispositivo conectado debe tener credenciales para obtener acceso al agente de mensajes o al servicio Thing Shadows. Todo el tráfico hacia o desde AWS IoT debe estar cifrado sobre el protocolo Transport Layer Security (TLS). Las credenciales de dispositivos deben guardarse en un lugar seguro para que los datos se envíen de forma segura al agente de mensajes. Los mecanismos de seguridad de la nube de AWS protegen los datos cuando estos circulan entre AWS IoT y otros dispositivos o los servicios de AWS.



- Usted es el responsable de administrar las credenciales de dispositivos (certificados X.509, credenciales de AWS) en los dispositivos, así como las políticas en AWS IoT. También es el responsable de asignar identidades exclusivas a cada uno de los dispositivos y de administrar los permisos de un dispositivo o un grupo de dispositivos.
- Los dispositivos se conectan con la identidad que haya seleccionado (certificados X.509, usuarios y grupos de IAM o identidades de Amazon Cognito) sobre una conexión segura, de acuerdo con el modelo de conexión de AWS IoT.
- El agente de mensajes de AWS IoT autentica y autoriza todas las acciones de su cuenta. Este agente es el responsable de autenticar sus dispositivos, insertar datos de dispositivos de forma segura y cumplir los permisos de acceso impuestos en los dispositivos mediante políticas.
- El motor de reglas de AWS IoT reenvía los datos de los dispositivos a otros dispositivos y otros servicios de AWS de acuerdo con las reglas que haya definido. Utiliza los sistemas de administración de acceso de AWS para transferir datos de forma segura a su destino final.

# Autenticación en AWS IoT

AWS IoT admite cuatro tipos de principales de identidad para la autenticación:

- Certificados X.509
- Usuarios, grupos y roles de IAM
- Identidades de Amazon Cognito
- Identidades federadas

Estas identidades se pueden utilizar con aplicaciones móviles, aplicaciones web o aplicaciones de escritorio. Incluso las puede utilizar un usuario que ejecute comandos de CLI de AWS IoT. Por lo general, los dispositivos AWS IoT utilizan certificados X.509, mientras que las aplicaciones móviles utilizan identidades de Amazon Cognito. Las aplicaciones web y de escritorio usan IAM o identidades federadas. Los comandos de CLI usan IAM.

## Certificados X.509

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. Los certificados X.509 se generan a través de una entidad de confianza conocida como autoridad de certificación (CA). La CA administra uno o varios certificados especiales llamados certificados CA, que utiliza para generar certificados X.509. Solo la autoridad de certificación tiene acceso a los certificados CA.

### Note

Le recomendamos que dé a cada dispositivo un certificado único para disponer de una administración precisa que incluya la revocación de certificados.

### Note

Los dispositivos deben ser compatibles con la rotación y la sustitución de certificados para garantizar un buen funcionamiento cuando los certificados caduquen.

AWS IoT es compatible con los siguientes algoritmos de firma de certificado:

- SHA256WITHRSA
- SHA384WITHRSA
- SHA384WITHRSA
- SHA512WITHRSA
- RSASSAPSS
- DSA\_WITH\_SHA256
- ECDSA-WITH-SHA256
- ECDSA-WITH-SHA384
- ECDSA-WITH-SHA512

Los certificados ofrecen varios beneficios con respecto a otros mecanismos de identificación y autenticación. Los certificados permiten usar claves asimétricas con los dispositivos. Esto significa que puede grabar claves privadas en el almacenamiento seguro de un dispositivo. De esta forma, el material criptográfico confidencial nunca sale del dispositivo. Los certificados proporcionan una autenticación del cliente más fiable que los otros sistemas, como el nombre de usuario y la contraseña o los tokens de portador, ya que la clave secreta jamás abandona el dispositivo.

AWS IoT autentica los certificados mediante el modo de autenticación de cliente del protocolo TLS. TLS está disponible en numerosos lenguajes de programación y sistemas operativos, y se utiliza generalmente para cifrar datos. En la autenticación de cliente de TLS, AWS IoT solicita un certificado X.509 de cliente y

valida el estado y la cuenta de AWS del certificado en un registro de certificados. A continuación, desafía al cliente para comprobar la propiedad de la clave privada que corresponde a la clave pública contenida en el certificado.

Para utilizar los certificados de AWS IoT, los clientes deben ser compatibles con todos los elementos siguientes en su implementación de TLS:

- TLS 1.2.
- Validación de la firma de certificado SHA-256 RSA.
- Uno de los conjuntos de cifrado de la sección de compatibilidad del conjunto de cifrado TLS.

## Certificados X.509 y AWS IoT

AWS IoT puede utilizar certificados generados por AWS IoT o certificados firmados por un certificado de CA para autenticar los dispositivos. Los certificados que genere AWS IoT no caducan. La fecha y hora de caducidad de los certificados firmados por un certificado de CA se establecen en el momento de su creación.

### Note

Le recomendamos que dé a cada dispositivo un certificado único para disponer de una administración precisa que incluya la revocación de certificados.

### Note

Los dispositivos deben ser compatibles con la rotación y la sustitución de certificados para garantizar un buen funcionamiento cuando los certificados caduquen.

Para utilizar un certificado que no se ha creado con AWS IoT, debe registrar un certificado de CA. Todos los certificados de dispositivo deben estar firmados por el certificado de CA que registre.

Puede utilizar la consola de AWS IoT o la CLI para realizar las siguientes operaciones:

- Crear y registrar un certificado de AWS IoT.
- Registrar un certificado de CA.
- Registrar un certificado de dispositivo.
- Activar o desactivar un certificado de dispositivo.
- Revocar un certificado de dispositivo.
- Transferir un certificado de dispositivo a otra cuenta de AWS.
- Generar una lista de todos los certificados de CA registrados en su cuenta de AWS.
- Generar una lista de todos los certificados de dispositivo registrados en su cuenta de AWS.

Para obtener más información acerca de qué comandos de CLI debe utilizar para estas operaciones, consulte [AWS IoT CLI Reference](#).

Para obtener más información acerca de cómo usar la consola de AWS IoT para crear los certificados, consulte [Create and Activate a Device Certificate](#).

## Autenticación del servidor

Los certificados de dispositivo permiten a AWS IoT autenticar los dispositivos. Para asegurarse de que su dispositivo se está comunicando con AWS IoT y no con otro servidor que haya suplantado a AWS IoT, copie el [certificado de CA raíz G5 principal y público de clase 3 de VeriSign](#) en su dispositivo.

### Note

Este certificado de CA es válido hasta julio de 2036, pero es posible que deba sustituirlo antes de llegar a esa fecha. Debe comprobar que puede actualizar el certificado de CA raíz en todos sus

dispositivos para asegurarse de que la conectividad se mantenga y esté al día de las prácticas recomendadas de seguridad.

Al conectarse con AWS IoT, haga referencia al certificado de CA raíz en el código de su dispositivo. Para obtener más información, consulte [SDK de AWS IoT \(p. 249\)](#).

**Note**

No puede usar su propio certificado de CA para autenticar el servidor de AWS IoT. Debe utilizar el certificado de CA raíz G5 principal y público de clase 3 de VeriSign.

## Creación y registro de un certificado de dispositivo de AWS IoT

Puede utilizar la consola de AWS IoT o la CLI de AWS IoT para crear un certificado de AWS IoT.

### Para crear un certificado (consola)

1. Inicie sesión en la consola de administración de AWS y abra la [consola de AWS IoT](#) en <https://console.aws.amazon.com/iot>.
2. En el panel de navegación izquierdo, elija Security para ampliar las opciones y, a continuación, elija Certificates. Seleccione Create.
3. Seleccione One-click certificate creation - Create certificate. O bien, para generar un certificado con una solicitud de firma de certificado (CSR), seleccione Create with CSR.
4. Utilice los enlaces a la clave pública, la clave privada y el certificado para descargarlos en una ubicación segura.
5. Elija Activate.

### Para crear un certificado (CLI)

La CLI de AWS IoT ofrece dos comandos para crear certificados:

- [create-keys-and-certificate](#)

La API [CreateKeysAndCertificate](#) crea una clave privada, una clave pública y un certificado X.509.

- [create-certificate-from-csr](#)

La API [CreateCertificateFromCSR](#) crea un certificado con una CSR.

## Uso del certificado propio

Para utilizar sus propios certificados X.509, debe registrar un certificado de CA en AWS IoT. A partir de entonces, podrá usar este certificado para firmar los certificados de dispositivo. Puede registrar hasta 10 certificados de CA con el mismo campo de asunto por cuenta de AWS y por región. Esto le permite tener varias CA para firmar los certificados de su dispositivo.

**Note**

Los certificados de dispositivo deben estar firmados por el certificado de CA registrado. Es habitual que un certificado de CA se utilice para crear un certificado de CA intermedio. Si está utilizando un certificado intermedio para firmar sus certificados de dispositivo, debe registrar el certificado de CA intermedio. Utilice el certificado de CA raíz de AWS IoT al conectarse con AWS IoT, aunque haya registrado su propio certificado de CA raíz. Los dispositivos usan el certificado de CA raíz de AWS IoT para verificar la identidad de los servidores de AWS IoT.

### Contenido

- [Registro de su certificado de CA \(p. 105\)](#)

- [Creación de un certificado de dispositivo con su certificado de CA \(p. 106\)](#)
- [Registro de un certificado de dispositivo \(p. 107\)](#)
- [Registro manual de los certificados de dispositivo \(p. 107\)](#)
- [Uso del registro automático/Just-In-Time para los certificados de dispositivo \(p. 107\)](#)
- [Desactivación del certificado de CA \(p. 108\)](#)
- [Revocación del certificado de dispositivo \(p. 108\)](#)

Si no dispone de certificado de CA, puede crear uno con herramientas de [OpenSSL](#).

#### Para crear un certificado de CA

1. Genere un par de claves.

```
openssl genrsa -out rootCA.key 2048
```

2. Utilice la clave privada del par de claves para generar un certificado de CA.

```
openssl req -x509 -new -nodes -key rootCA.key -sha256 -days 1024 -out rootCA.pem
```

#### Registro de su certificado de CA

Para registrar su certificado de CA, debe:

- Obtener un código de registro de AWS IoT.
- Firmar un certificado de verificación de clave privada con su certificado de CA.
- Transferir su certificado de CA y un certificado de verificación de clave privada al comando de CLI `register-ca-certificate`.

El campo `Common Name` del certificado de verificación de la clave privada debe establecerse en el código de registro generado por el comando de CLI `get-registration-code`. Se genera un único código de registro por cuenta de AWS. Puede ejecutar el comando `register-ca-certificate` o usar la consola de AWS IoT para registrar los certificados de CA.

#### Para registrar un certificado de CA

1. Obtenga un código de registro de AWS IoT. Este código se utilizará como `Common Name` del certificado de verificación de la clave privada.

```
aws iot get-registration-code
```

2. Genere un par de claves para el certificado de verificación de clave privada.

```
openssl genrsa -out verificationCert.key 2048
```

3. Cree una CSR para el certificado de verificación de la clave privada. Inserte su código de registro en el campo `Common Name` del certificado.

```
openssl req -new -key verificationCert.key -out verificationCert.csr
```

Se le solicitará información, incluido `Common Name` para el certificado.

```
Country Name (2 letter code) [AU]:  
State or Province Name (full name) []:
```

```
Locality Name (eg, city) []:
Organization Name (eg, company) []:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:XXXXXXXXXXXXMYREGISTRATIONCODEXXXXXX
Email Address []:
```

4. Utilice la CSR para crear un certificado de verificación de la clave privada.

```
openssl x509 -req -in verificationCert.csr -CA rootCA.pem -CAkey rootCA.key -
CAcreateserial -out verificationCert.pem -days 500 -sha256
```

5. Registre el certificado de CA en AWS IoT. Transmite el certificado de CA y el certificado de verificación de la clave privada al comando register-ca-certificate de CLI.

```
aws iot register-ca-certificate --ca-certificate file://rootCA.pem --verification-cert
file://verificationCert.pem
```

6. Ejecute el comando update-certificate de CLI para activar el certificado de CA.

```
aws iot update-ca-certificate --certificate-id XXXXXXXXXXXX --new-status ACTIVE
```

## Creación de un certificado de dispositivo con su certificado de CA

Puede utilizar un certificado de CA registrado en AWS IoT para crear un certificado de dispositivo. El certificado de dispositivo debe registrarse en AWS IoT para poder utilizarlo.

### Para crear un certificado de dispositivo

1. Genere un par de claves.

```
openssl genrsa -out deviceCert.key 2048
```

2. Cree una CSR para el certificado de dispositivo.

```
openssl req -new -key deviceCert.key -out deviceCert.csr
```

Se le solicitará que indique información, tal y como se muestra aquí.

```
Country Name (2 letter code) [AU]:
State or Province Name (full name) []:
Locality Name (eg, city) []:
Organization Name (eg, company) []:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:
```

3. Cree un certificado de dispositivo a partir de la CSR.

```
openssl x509 -req -in deviceCert.csr -CA rootCA.pem -CAkey rootCA.key -CAcreateserial -
out deviceCert.pem -days 500 -sha256
```

### Note

Debe utilizar del certificado de CA registrado en AWS IoT para crear certificados de dispositivo. Si tiene más de un certificado de CA (con el mismo campo de asunto y clave pública;) registrado en su cuenta de AWS, debe especificar el certificado de CA utilizado para crear el certificado de dispositivo al registrarlo.

4. Registrar un certificado de dispositivo.

```
aws iot register-certificate --certificate-pem file://deviceCert.pem --ca-certificate-pem file://rootCA.pem
```

5. Ejecute el comando update-certificate de CLI para activar el certificado de dispositivo.

```
aws iot update-certificate --certificate-id xxxxxxxxxxxx --new-status ACTIVE
```

## Registro de un certificado de dispositivo

Debe utilizar el certificado de CA registrado en AWS IoT para firmar los certificados de dispositivo. Si tiene más de un certificado de CA (con el mismo campo de asunto y clave pública;) registrado en su cuenta de AWS, debe especificar el certificado de CA utilizado para firmar el certificado de dispositivo al registrarlo. Puede registrar cada certificado de dispositivo manualmente, o bien puede utilizar el registro automático, que permite a los dispositivos registrar su certificado al conectarse con AWS IoT por primera vez.

### Registro manual de los certificados de dispositivo

Ejecute el comando de CLI siguiente para registrar un certificado de dispositivo:

```
aws iot register-certificate --certificate-pem file://deviceCert.crt --ca-certificate-pem file://caCert.crt
```

### Uso del registro automático/Just-In-Time para los certificados de dispositivo

Para que los certificados de dispositivo se registren automáticamente al conectarse por primera vez con AWS IoT, debe habilitar el registro automático de su certificado de CA. Esto registrará todos los certificados de dispositivo firmados por el certificado de CA cuando este se conecte con AWS IoT.

### Activación del registro automático

Utilice la API update-ca-certificate para establecer el auto-registration-status del certificado de CA en ENABLE:

```
$ aws iot update-ca-certificate --cert-id caCertificateID --new-auto-registration-status ENABLE
```

También puede establecer el auto-registration-status en ENABLE cuando utilice la API register-ca-certificate para registrar su certificado de CA:

```
aws iot register-ca-certificate --ca-certificate file://rootCA.pem --verification-cert file://privateKeyVerificationCert.crt --allow-auto-registration
```

Cuando un dispositivo intenta conectarse por primera vez con AWS IoT, como parte del protocolo de TLS, debe presentar un certificado de CA registrado y un dispositivo de certificado. AWS IoT reconoce el certificado de CA como un certificado de CA registrado, registra automáticamente el certificado de dispositivo y establece su estado en PENDING\_ACTIVATION. Esto significa que el certificado de dispositivo se registró automáticamente y que está a la espera de su activación. El certificado debe estar en el estado ACTIVE para poder conectarse con AWS IoT. Cuando AWS IoT registra automáticamente un certificado o cuando este se encuentra en estado PENDING\_ACTIVATION, AWS IoT publica un mensaje en el tema MQTT siguiente:

```
$aws/events/certificates/registered/caCertificateID
```

Donde caCertificateID es el ID del certificado de CA que generó el certificado de dispositivo.

El mensaje publicado en este tema tiene la estructura siguiente:

```
{  
    "certificateId": "certificateID",  
    "caCertificateId": "caCertificateId",  
    "timestamp": timestamp,  
    "certificateStatus": "PENDING_ACTIVATION",  
    "awsAccountId": "awsAccountId",  
    "certificateRegistrationTimestamp": "certificateRegistrationTimestamp"  
}
```

Puede crear una regla que escuche este tema y realice algunas acciones. Le recomendamos crear una regla Lambda que verifique que el certificado de dispositivo no se encuentre en una lista de revocación de certificados (CRL), active el certificado y cree una política y la asocie a este. La política determina a qué recursos puede tener acceso el dispositivo. Para obtener más información acerca de cómo crear una regla Lambda que escuche el tema \$aws/events/certificates/registered/caCertificateID y ejecute estas acciones, consulte [Just-in-Time Registration](#).

### Desactivación del certificado de CA

Cuando registra un certificado de dispositivo, AWS comprueba si el certificado de CA asociado se encuentra en estado ACTIVE. Si el certificado de CA se encuentra en estado INACTIVE, AWS IoT no permite registrar el certificado de dispositivo. Al marcar el certificado de CA como INACTIVE, evita que se registren en su cuenta los certificados de dispositivo nuevos que ha generado la CA comprometida. Puede utilizar la API update-ca-certificate para desactivar el certificado de CA:

```
$ aws iot update-ca-certificate --cert-id certificateID --new-status INACTIVE
```

#### Note

Todos los certificados de dispositivo registrados que haya firmado el certificado de CA en riesgo seguirán funcionando hasta que usted los revoque explícitamente.

Utilice la API ListCertificatesByCA para obtener una lista de todos los certificados de dispositivo registrados que firmó la CA comprometida. Por cada certificado de dispositivo firmado por el certificado de CA en riesgo, puede utilizar la API UpdateCertificate para revocar el certificado de dispositivo y evitar que este se use.

### Revocación del certificado de dispositivo

Si detecta actividad sospechosa en un certificado de dispositivo registrado, puede utilizar la API update-certificate para revocarlo:

```
$ aws iot update-certificate --cert-id certificateID  
    --new-status REVOKED
```

Si se produce algún error o excepción durante el registro automático de los certificados de dispositivo, AWS IoT envía eventos o mensajes a sus logs en CloudWatch Logs. Para obtener más información acerca de cómo configurar los logs de su cuenta, consulte la [documentación de Amazon CloudWatch](#).

## Usuarios, grupos y roles de IAM

Los usuarios, grupos y roles de IAM son los mecanismos estándar de administración de identidades y autenticación en AWS. Puede utilizarlos para conectarse con las interfaces HTTP de AWS IoT mediante el SDK y la CLI de AWS.

Los roles de IAM también permiten a AWS IoT tener acceso a otros recursos de AWS de su cuenta en su nombre. Por ejemplo, si desea que un dispositivo publique su estado en una tabla de DynamoDB, los roles

de IAM permiten a AWS IoT interactuar con Amazon DynamoDB. Para obtener más información, consulte [IAM Roles](#).

En cuanto a las conexiones del agente de mensajes sobre HTTP, AWS IoT autentica a los usuarios, grupos y roles de IAM mediante el proceso Signature Version 4. Para obtener más información, consulte [Firma de solicitudes de la API de AWS](#).

Cuando utilice AWS Signature Version 4 con AWS IoT, los clientes deben admitir lo siguiente en su implementación de TLS:

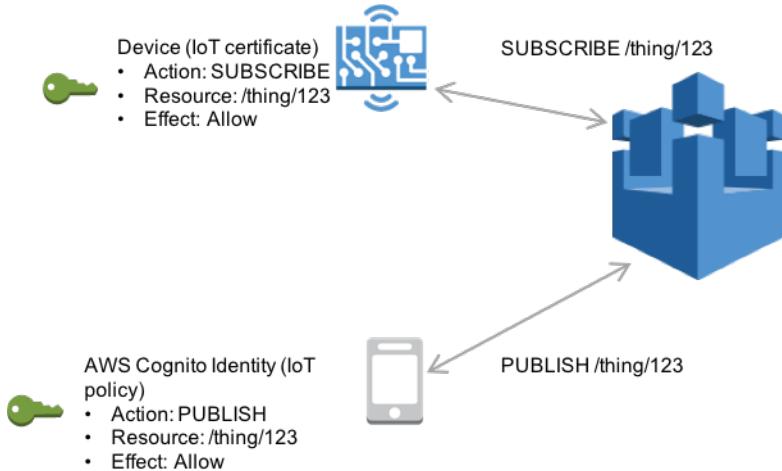
- TLS 1.2, TLS 1.1, TLS 1.0.
- Validación de la firma de certificado SHA-256 RSA.
- Uno de los conjuntos de cifrado de la sección de compatibilidad del conjunto de cifrado TLS.

Para obtener más información, consulte [IAM User Guide](#).

## Identidades de Amazon Cognito

Una identidad de Amazon Cognito le permite utilizar su propio proveedor de identidades u otros proveedores conocidos, por ejemplo, Login with Amazon, Facebook o Google. Usted intercambia un token de su proveedor de identidades por credenciales de seguridad de AWS. Las credenciales representan un rol de IAM y se pueden utilizar con AWS IoT.

AWS IoT amplía Amazon Cognito y permite asociar una política a identidades de Amazon Cognito. Puede asociar una política a una identidad de Amazon Cognito y conceder permisos muy precisos a un usuario de su aplicación AWS IoT. De esta forma, puede asignar permisos entre clientes específicos y sus dispositivos. Para obtener más información, consulte [Amazon Cognito Identity](#).



## Autorización

Las políticas establecen qué puede hacer una identidad autenticada. Los dispositivos y las aplicaciones móviles, web y de escritorio utilizan identidades autenticadas. Una identidad autenticada puede ser incluso un usuario que ejecute comandos de CLI de AWS IoT. La identidad puede ejecutar operaciones de AWS IoT solo si cuenta con una política que le conceda permiso para ello.

Tanto las políticas de AWS IoT como las de IAM se utilizan con AWS IoT para controlar las operaciones que una identidad (también denominada principal) puede realizar. El tipo de política que utilice depende del tipo de identidad que esté utilizando para autenticarse en AWS IoT. En la tabla siguiente se muestran

los tipos de identidad, los protocolos que utilizan y los tipos de políticas que se pueden utilizar para la autorización.

Las operaciones de AWS IoT se dividen en dos grupos:

- La API del plano de control le permite realizar tareas administrativas, como crear o actualizar certificados, objetos, reglas, etc.
- La API del plano de datos le permite enviar datos a AWS IoT y recibir datos de este servicio.

El tipo de política que utilice depende de si utiliza la API del plano de control o la del plano de datos.

#### API del plano de datos de AWS IoT y tipos de políticas

Protocolo y mecanismo de autenticación	SDK	Tipo de identidad	Tipo de política		
MQTT sobre autenticación mutua (puerto 8883)	SDK de dispositivos AWS IoT	Certificados X.509	Política de AWS IoT		
MQTT sobre Websocket (puerto 443)	SDK para móviles de AWS	Amazon Cognito IAM o identidad federada	Política de AWS IoT para identidades de Amazon Cognito  Política de IAM para otras identidades		
HTTP sobre autenticación de servidor (puerto 443)	AWS CLI	Amazon Cognito IAM o identidad federada	Política de AWS IoT para identidades de Amazon Cognito  Política de IAM para otras identidades		
HTTP sobre autenticación mutua (puerto 8443)	Sin compatibilidad de SDK	Certificados X.509	Política de AWS IoT		

#### API del plano de control y tipos de política de AWS IoT

Protocolo y mecanismo de autenticación	SDK	Tipo de identidad	Tipo de política		
HTTP sobre autenticación de servidor (puerto 443)	AWS CLI	Amazon Cognito IAM o identidad federada	Política de AWS IoT para identidades		

Protocolo y mecanismo de autenticación	SDK	Tipo de identidad	Tipo de política		
			de Amazon Cognito  Política de IAM para otras identidades		

Las políticas de AWS IoT están asociadas a certificados X.509 o identidades de Amazon Cognito. Las políticas de IAM están asociadas a un usuario, grupo o rol de IAM. Si utiliza la consola de AWS IoT o la CLI de AWS IoT para asociar la política (a un certificado o a Identidad de Amazon Cognito), utilice una política de AWS IoT. De lo contrario, utilice una política de IAM.

Las autorizaciones basadas en políticas son una herramienta muy eficaz. Le dan un control completo sobre lo que un dispositivo, usuario o aplicación puede hacer en AWS IoT. Por ejemplo, tomemos el caso de un dispositivo que se conecte con AWS IoT mediante un certificado. Puede permitir que el dispositivo tenga acceso a todos los temas MQTT, o bien puede restringir su acceso a un único tema. O tomemos, por ejemplo, el caso de un usuario que ejecute comandos de CLI en una línea de comandos. Si aplica una política, puede permitirle o denegarle el acceso a cualquier comando o recurso de AWS IoT. También puede controlar el acceso de una aplicación a los recursos de AWS IoT.

## Políticas de AWS IoT

Las políticas de AWS IoT son documentos JSON. Siguen las mismas convenciones que las políticas de IAM. AWS IoT admite políticas con un nombre que permite que muchas identidades puedan hacer referencia al mismo documento de política. Las políticas con nombre cuentan con varias versiones para facilitar su restauración.

AWS IoT define un conjunto de acciones de política que describen las operaciones y los recursos a los que puede conceder o denegar el acceso. Por ejemplo:

- `iot:Connect` representa el permiso para conectarse con el agente de mensajes de AWS IoT.
- `iot:Subscribe` representa el permiso para suscribirse a un tema MQTT o un filtro de temas.
- `iot:GetThingShadow` representa el permiso para obtener una sombra de objeto.

Las políticas de AWS IoT le permiten controlar el acceso al plano de datos de AWS IoT. El plano de datos de AWS IoT se compone de operaciones que le permiten conectarse con el agente de mensajes de AWS IoT, enviar y recibir mensajes de MQTT y obtener o actualizar sombras de objeto. Para obtener más información, consulte [Acciones de política de AWS IoT \(p. 112\)](#).

Una política de AWS IoT es un documento JSON que contiene una o varias declaraciones de política. Cada declaración contiene un `Effect`, una `Action` y un `Resource`. El `Effect` especifica si se permitirá o se denegará la acción. La `Action` especifica la acción que la política permite o deniega. El `Resource` especifica los recursos en los que se permite o se deniega la acción. La siguiente política concede a todos los dispositivos permiso para conectarse con el agente de mensajes de AWS IoT, pero limita las publicaciones del dispositivo a un único tema MQTT:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["iot:Publish"],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/foo/bar"]
    
```

```
},
{
  "Effect": "Allow",
  "Action": ["iot:Connect"],
  "Resource": ["*"]
}
```

## Acciones de política de AWS IoT

AWS IoT define las siguientes acciones de política:

Acciones de política MQTT

iot:Connect

Representa el permiso para conectarse con el agente de mensajes de AWS IoT. El permiso iot:Connect se comprueba cada vez que se envía una solicitud CONNECT al agente. El agente de mensajes no permite que haya dos clientes con el mismo ID de cliente conectados simultáneamente. Cuando el segundo cliente se conecta, el agente detecta este caso y desconecta uno de los clientes. El permiso iot:Connect se puede utilizar para garantizar que solo los clientes autorizados puedan conectarse mediante un ID de cliente específico.

iot:Publish

Representa el permiso de publicar en un tema MQTT. Este permiso se comprueba cada vez que se envía una solicitud PUBLISH al agente. Se puede utilizar para permitir a los clientes publicar en determinados patrones de tema.

Note

También debe conceder el permiso iot:Connect para otorgar el permiso iot:Publish.

iot:Receive

Representa el permiso para recibir un mensaje de AWS IoT. El permiso iot:Receive se comprueba cada vez que se entrega un mensaje a un cliente. Dado que este permiso se comprueba en cada entrega, se puede utilizar para revocar permisos a clientes que están suscritos en ese momento a un tema.

iot:Subscribe

Representa el permiso para suscribirse a un filtro de temas. Este permiso se comprueba cada vez que se envía una solicitud SUBSCRIBE al agente. Se puede utilizar para permitir a los clientes suscribirse a temas que coinciden con patrones de tema específicos.

Note

También debe conceder el permiso iot:Connect para otorgar el permiso iot:Subscribe.

Acciones de política de sombra de objeto

iot>DeleteThingShadow

Representa el permiso para eliminar una sombra de objeto. El permiso iot>DeleteThingShadow se comprueba cada vez que se presenta una solicitud para eliminar el documento de sombra de objeto.

iot:GetThingShadow

Representa el permiso para recuperar una sombra de objeto. El permiso iot:GetThingShadow se comprueba cada vez que se presenta una solicitud para recuperar un documento de sombra de objeto.

### iot:UpdateThingShadow

Representa el permiso para actualizar una sombra de objeto. El permiso `iot:UpdateThingShadow` se comprueba cada vez que se presenta una solicitud para actualizar el estado de un documento de sombra de objeto.

## Recursos de acción

Para especificar un recurso para una acción de política de AWS IoT, debe utilizar el ARN del recurso. Todos los ARN de recursos tienen la forma siguiente:

`arn:aws:iot:region:AWS account ID:resource type:resource name`

En la tabla siguiente se muestra el recurso que debe especificarse para cada tipo de acción:

Acción	Recurso
iot:DeleteThingShadow	Un ARN de objeto - <code>arn:aws:iot:us-east-1:123456789012:thing/thingOne</code>
iot:Connect	Un ARN de ID de cliente - <code>arn:aws:iot:us-east-1:123456789012:client/myClientId</code>
iot:Publish	Un ARN de tema - <code>arn:aws:iot:us-east-1:123456789012:topic/myTopicName</code>
iot:Subscribe	Un ARN de filtro de temas - <code>arn:aws:iot:us-east-1:123456789012:topicfilter/myTopicFilter</code>
iot:Receive	Un ARN de tema - <code>arn:aws:iot:us-east-1:123456789012:topic/myTopicName</code>
iot:UpdateThingShadow	Un ARN de objeto - <code>arn:aws:iot:us-east-1:123456789012:thing/thingOne</code>
iot:GetThingShadow	Un ARN de objeto - <code>arn:aws:iot:us-east-1:123456789012:thing/thingOne</code>

## Variables de política de AWS IoT

AWS IoT define las variables de política que se pueden utilizar en las políticas de AWS IoT dentro del bloque de condición o de recurso. Cuando se evalúa una política, las variables de política se sustituyen por valores reales. Por ejemplo, si un dispositivo se ha conectado al agente de mensajes de AWS IoT con un ID de cliente de "100-234-3456", la variable de política `iot:ClientId` se sustituirá en el documento de política por "100-234-3456". Para obtener más información acerca de las variables de política, consulte las páginas [IAM Policy Variables](#) y [Multi-Value Conditions](#).

### Variables de política básicas

AWS IoT define las siguientes variables de política básicas:

- `iot:ClientId`: el ID de cliente que se utiliza para conectarse con el agente de mensajes de AWS IoT.
- `aws:SourceIp`: la dirección IP del cliente conectado con el agente de mensajes de AWS IoT.

La política de AWS IoT siguiente muestra el uso de variables de política:

```
{  
    "Version": "2012-10-17",  
    "Statement": [{  
        "Effect": "Allow",  
        "Action": ["iot:Connect"],  
        "Resource": [  
            "arn:aws:iot:us-east-1:123451234510:client/${iot:ClientId}"  
        ]  
    },  
    {  
        "Effect": "Allow",  
        "Action": ["iot:Publish"],  
        "Resource": [  
            "arn:aws:iot:us-east-1:123451234510:topic/foo/bar/${iot:ClientId}"  
        ]  
    }]  
}
```

En los ejemplos siguientes, \${iot:ClientId} se sustituirá por el ID del cliente conectado al agente de mensajes de AWS IoT cuando se evalúe la política. Al utilizar variables de política como \${iot:ClientId}, puede abrir accidentalmente el acceso a temas que no quería incluir. Por ejemplo, si utiliza una política que utiliza \${iot:ClientId} para especificar un filtro de temas:

```
{  
    "Effect": "Allow",  
    "Action": ["iot:Subscribe"],  
    "Resource": [  
        "arn:aws:iot:us-east-1:123456789012:topicfilter/foo/${iot:ClientId}/bar"  
    ]  
}
```

Un cliente puede conectarse usando + como ID de cliente. Esto puede permitir al usuario suscribirse a cualquier tema que coincida con el filtro de temas foo/+/\*bar. Como protección contra estas deficiencias de seguridad, utilice la acción de política iot:Connect para controlar los ID de cliente que pueden conectarse. Por ejemplo, esta política permitirá el acceso únicamente a los clientes cuyo ID de cliente sea clientid1:

```
{  
    "Version": "2012-10-17",  
    "Statement": [{  
        "Effect": "Allow",  
        "Action": ["iot:Connect"],  
        "Resource": [  
            "arn:aws:iot:us-east-1:123456789012:client/clientid1"  
        ]  
    }]  
}
```

## Variables de política de certificado X.509

Las variables de política de certificado X.509 le permiten escribir políticas de AWS IoT que concedan permisos basados en los atributos de certificado X.509. En las secciones siguientes se describe cómo se pueden utilizar estas variables de política de certificado.

### Atributos de emisor

Las siguientes variables de política de AWS IoT le permiten autorizar o denegar permisos en función de atributos de certificado establecidos por el emisor del certificado.

- iot:Certificate.Issuer.DistinguishedNameQualifier

- iot:Certificate.Issuer.Country
- iot:Certificate.Issuer.Organization
- iot:Certificate.Issuer.OrganizationalUnit
- iot:Certificate.Issuer.State
- iot:Certificate.Issuer.CommonName
- iot:Certificate.Issuer.SerialNumber
- iot:Certificate.Issuer.Title
- iot:Certificate.Issuer.Surname
- iot:Certificate.Issuer.GivenName
- iot:Certificate.Issuer.Initials
- iot:Certificate.Issuer.Pseudonym
- iot:Certificate.Issuer.GenerationQualifier

### Atributos de asunto

Las siguientes variables de política de AWS IoT le permiten conceder o denegar permisos en función de atributos de asunto de certificado establecidos por el emisor del certificado.

- iot:Certificate.Subject.DistinguishedNameQualifier
- iot:Certificate.Subject.Country
- iot:Certificate.Subject.Organization
- iot:Certificate.Subject.OrganizationalUnit
- iot:Certificate.Subject.State
- iot:Certificate.Subject.CommonName
- iot:Certificate.Subject.SerialNumber
- iot:Certificate.Subject.Title
- iot:Certificate.Subject.Surname
- iot:Certificate.Subject.GivenName
- iot:Certificate.Subject.Initials
- iot:Certificate.Subject.Pseudonym
- iot:Certificate.Subject.GenerationQualifier

Los certificados X.509 permiten que estos atributos contengan uno o varios valores. De forma predeterminada, las variables de política de cada atributo de varios valores devuelven el primer valor. Por ejemplo, el atributo Certificate.Subject.Country puede contener una lista de nombres de países. Cuando se evalúan en una política, iot:Certificate.Subject.Country se sustituye por el nombre del primer país. Puede solicitar un valor de atributo específico mediante un índice de base cero. Por ejemplo, iot:Certificate.Subject.Country#1 se sustituye por el nombre del segundo país en el atributo Certificate.Subject.Country. Si especifica un valor de atributo que no existe (por ejemplo, si pide un tercer valor cuando el atributo solo tiene dos valores asignados), no se realizará ninguna sustitución y la autorización dará un resultado erróneo. Puede utilizar el sufijo .List en el nombre de la variable de política para especificar todos los valores del atributo. La política del ejemplo siguiente permite a cualquier cliente conectarse con AWS IoT, pero restringe los derechos de publicación a los clientes con certificados cuyo atributo Certificate.Subject.Organization esté establecido en "Example Corp" o "AnyCompany". Esto se consigue utilizando un atributo "Condition" que especifique una condición para la acción anterior. En este caso, la condición es que el atributo Certificate.Subject.Organization del certificado incluya uno de los valores de la lista.

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

```
"Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "iot:Connect"
        ],
        "Resource": [
            "*"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "iot:Publish"
        ],
        "Resource": [
            "*"
        ],
        "Condition": {
            "ForAllValues:StringEquals": {
                "iot:Certificate.Subject.Organization.List": [
                    "Example Corp",
                    "AnyCompany"
                ]
            }
        }
    }
]
```

### Atributos de nombre alternativo del emisor

Las variables de política de AWS IoT siguientes le permiten conceder o denegar permisos en función de los atributos de nombre alternativo del emisor establecidos por el emisor del certificado.

- `iot:Certificate.Issuer.AlternativeName.RFC822Name`
- `iot:Certificate.Issuer.AlternativeName.DNSName`
- `iot:Certificate.Issuer.AlternativeName.DirectoryName`
- `iot:Certificate.Issuer.AlternativeName.UniformResourceIdentifier`
- `iot:Certificate.Issuer.AlternativeName.IPAddress`

### Atributos de nombre alternativo de asunto

Las variables de política de AWS IoT siguientes le permiten conceder o denegar permisos en función de los atributos de nombre alternativo del tema establecidos por el emisor del certificado.

- `iot:Certificate.Subject.AlternativeName.RFC822Name`
- `iot:Certificate.Subject.AlternativeName.DNSName`
- `iot:Certificate.Subject.AlternativeName.DirectoryName`
- `iot:Certificate.Subject.AlternativeName.UniformResourceIdentifier`
- `iot:Certificate.Subject.AlternativeName.IPAddress`

### Otros atributos

Puede utilizar `iot:Certificate.SerialNumber` para permitir o denegar el acceso a recursos de AWS IoT en función del número de serie de un certificado. La variable de política `iot:Certificate.AvailableKeys` contiene el nombre de todas las variables de política de certificado que contienen valores.

## Limitaciones aplicables a las variables de política de certificado X.509

Las siguientes limitaciones se aplican a las variables de política de certificado X.509:

### Comodines

Si los atributos de certificado contienen caracteres comodín, la variable de política no se sustituirá por el valor de atributo de certificado y el texto \${policy-variable} figurará en el documento de la política. Esto puede producir un error de autorización.

### Campos de matriz

Los atributos de certificado que contienen matrices se limitan a cinco elementos. No se tendrán en cuenta los elementos adicionales.

### Longitud de cadena

Todos los valores de cadena están limitados a 1024 caracteres. Si un atributo de certificado contiene una cadena de más de 1024 caracteres, la variable de política no se sustituirá por el valor de atributo de certificado y el texto \${policy-variable} figurará en el documento de la política. Esto puede producir un error de autorización.

## Variables de la política de objeto

Las variables de política de objeto le permiten escribir políticas de AWS IoT que concedan o denieguen permisos en función de las propiedades del objeto como el nombre o el tipo de objeto, o los valores de atributo del objeto. El nombre de objeto se obtiene a partir del ID de cliente en el mensaje Connect de MQTT que se envía cuando un objeto se conecta con AWS IoT. Las variables de la política de objeto se sustituyen cuando un objeto se conecta con AWS IoT sobre MQTT empleando la autenticación mutua de TLS o MQTT sobre el protocolo WebSocket mediante identidades de Amazon Cognito autenticadas. Las variables de política de objeto también se sustituyen cuando se asocia un certificado o una identidad de Amazon Cognito autenticada a un objeto. Puede utilizar la API [AttachThingPrincipal](#) para asociar certificados e identidades de Amazon Cognito autenticadas a un objeto.

Las siguientes variables de política de objeto están disponibles:

- `iot:Connection.Thing.ThingName`
- `iot:Connection.Thing.ThingType`
- `iot:Connection.Thing.Attributes[attributeName]`
- `iot:Connection.Thing.IsAttached`

### `iot:Connection.Thing.ThingName`

Esta variable se resuelve en el nombre del objeto para el que se evalúa la política. El nombre de objeto se establece en el ID de cliente de la conexión MQTT/Websocket. Esta variable de política solo está disponible cuando la conexión se establece sobre MQTT o MQTT sobre protocolo WebSocket.

### `iot:Connection.Thing.ThingType`

Esta variable se resuelve en el tipo de objeto asociado al objeto para el que se evalúa la política. El nombre de objeto se establece en el ID de cliente de la conexión MQTT/Websocket. El nombre de tipo de objeto se obtiene a partir de una llamada a la API `DescribeThing`. Esta variable de política solo está disponible cuando la conexión se establece sobre MQTT o MQTT sobre protocolo WebSocket.

### `iot:Connection.Thing.Attributes[attributeName]`

Esta variable se resuelve en el valor del atributo especificado asociado al objeto para el que se evalúa la política. Un objeto puede tener hasta 50 atributos. Cada atributo estará disponible como variable de política: `iot:Connection.Thing.Attributes[attributeName]` donde `attributeName` es el

nombre del atributo. El nombre de objeto se establece en el ID de cliente de la conexión MQTT/Websocket. Esta variable de política solo está disponible cuando la conexión se establece sobre MQTT o MQTT sobre protocolo WebSocket.

#### iot:Connection.Thing.IsAttached

Esta variable se resuelve en `true` si el objeto para el que se evalúa la política tiene una identidad de Amazon Cognito o un certificado asociado.

## Ejemplos de políticas

Las políticas de AWS IoT se especifican en un documento JSON. Estos son los componentes de una política de AWS IoT:

Version

Debe establecerse en "2012-10-17".

Effect

Debe establecerse en "Allow" o "Deny".

Acción

Debe establecerse en "iot:`operation-name`" donde `operation-name` es uno de los valores siguientes:

"iot:Connect": conexión con AWS IoT

"iot:Receive": recepción de mensajes de AWS IoT

"iot:Publish": publicación MQTT.

"iot:Subscribe": suscripción MQTT.

"iot:UpdateThingShadow": actualizar una sombra de objeto.

"iot:GetThingShadow": recuperar una sombra de objeto.

"iot>DeleteThingShadow": eliminar una sombra de objeto.

Recurso

Debe establecerse en uno de los valores siguientes:

Cliente - arn:aws:iot:`region:account-id:client/client-id`

ARN de tema - arn:aws:iot:`region:account-id:topic/topic-name`

ARN de filtro de temas - arn:aws:iot:`region:account-id:topicfilter/topic-filter`

## Ejemplos de política de conexión

La siguiente política permite que un conjunto de ID de cliente se conecte:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "iot:Connect"  
            ],  
            "Resource": [  
                "arn:aws:iot:us-east-1:123456789012:client/*"  
            ]  
        }  
    ]  
}
```

```
        "arn:aws:iot:us-east-1:123456789012:client/clientid1",
        "arn:aws:iot:us-east-1:123456789012:client/clientid2",
        "arn:aws:iot:us-east-1:123456789012:client/clientid3"
    ],
},
{
    "Effect": "Allow",
    "Action": [
        "iot:Publish",
        "iot:Subscribe",
        "iot:Receive"
    ],
    "Resource": [
        "*"
    ]
}
]
```

La política siguiente impide que un conjunto de ID de cliente se conecte:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/clientid1",
                "arn:aws:iot:us-east-1:123456789012:client/clientid2"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

La política siguiente permite al titular de un certificado, independientemente de su ID de cliente, suscribirse a un filtro de temas foo/\*:

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

```
{  
    "Effect": "Allow",  
    "Action": [  
        "iot:Subscribe"  
    ],  
    "Resource": [  
        "arn:aws:iot:us-east-1:123456789012:topicfilter/foo/*"  
    ]  
}  
}  
]
```

## Ejemplos de política de publicación/suscripción

La política que utilice dependerá de cómo se conecte con AWS IoT. Puede conectarse con AWS IoT mediante un cliente MQTT, HTTP o WebSocket. Al conectarse con un cliente MQTT, se autenticará con un certificado X.509. Al conectarse mediante HTTP o el protocolo WebSocket, se autenticará con Signature Version 4 y Amazon Cognito.

### Políticas para clientes MQTT

Cuando especifique filtros de temas en políticas de AWS IoT para clientes MQTT, los caracteres comodín "+" y "#" de MQTT se tratarán como caracteres literales. Su uso puede dar lugar a un comportamiento inesperado. Por ejemplo, la política siguiente permitirá a un cliente suscribirse únicamente al filtro de temas `foo/+bar`:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "iot:Connect"  
            ],  
            "Resource": [  
                "*"  
            ]  
        },  
        {  
            "Effect": "Allow",  
            "Action": [  
                "iot:Subscribe"  
            ],  
            "Resource": [  
                "arn:aws:iot:us-east-1:123456789012:topicfilter/foo/+bar"  
            ]  
        }  
    ]  
}
```

#### Note

El carácter comodín de MQTT '+' no se considera un comodín dentro de una política. Los intentos de suscribirse a filtros de tema que coincidan con el patrón `foo/+bar` como `foo/baz/bar` o `foo/goo/bar` darán un error y harán que el cliente se desconecte.

Puede utilizar "\*" como comodín en el atributo de recurso de la política. Por ejemplo, la siguiente política permite al titular del certificado publicar en todos los temas y suscribirse a todos los filtros de temas de la cuenta de AWS:

```
{  
    "Version": "2012-10-17",  
}
```

```
"Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "iot:/*"
        ],
        "Resource": [
            "*"
        ]
    }
]
```

La siguiente política permite al titular del certificado publicar en todos los temas de la cuenta de AWS:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish",
                "iot:Connect"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

También puede utilizar el comodín "\*" al final del filtro de un tema. Por ejemplo, la política siguiente permite al titular del certificado suscribirse a un filtro de temas que coincide con el patrón `foo/bar/*`:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/foo/bar/*"
            ]
        }
    ]
}
```

La siguiente política permite al titular del certificado publicar en los temas `foo/bar` y `foo/baz`:

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

```
"Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "iot:Connect"
        ],
        "Resource": [
            "*"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "iot:Publish"
        ],
        "Resource": [
            "arn:aws:iot:us-east-1:123456789012:topic/foo/bar",
            "arn:aws:iot:us-east-1:123456789012:topic/foo/baz"
        ]
    }
]
```

La siguiente política impide que el titular del certificado publique en el tema `foo/bar`:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Deny",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/foo/bar"
            ]
        }
    ]
}
```

La siguiente política permite que el titular del certificado publique en el tema `foo` e impide que dicho titular publique en el tema `bar`:

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

```
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/foo"
            ]
        },
        {
            "Effect": "Deny",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/bar"
            ]
        }
    ]
}
```

La siguiente política permite que el titular del certificado se suscriba al filtro de temas `foo/bar`:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/foo/bar"
            ]
        }
    ]
}
```

La siguiente política permite que el titular del certificado publique en el tema `arn:aws:iot:us-east-1:123456789012:topic/iotmonitor/provisioning/8050373158915119971` y que dicho titular se suscriba en el filtro de temas `arn:aws:iot:us-east-1:123456789012:topicfilter/iotmonitor/provisioning/8050373158915119971`:

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

```
        ],
    },
{
    "Effect": "Allow",
    "Action": [
        "iot:Publish",
        "iot:Receive"
    ],
    "Resource": [
        "arn:aws:iot:us-east-1:123456789012:topic/iotmonitor/
provisioning/8050373158915119971"
    ]
},
{
    "Effect": "Allow",
    "Action": [
        "iot:Subscribe"
    ],
    "Resource": [
        "arn:aws:iot:us-east-1:123456789012:topicfilter/iotmonitor/
provisioning/8050373158915119971"
    ]
}
]
```

### Políticas para clientes de HTTP y WebSocket

Para las siguientes operaciones, AWS IoT utiliza políticas de AWS IoT asociadas a identidades de Amazon Cognito (mediante la API `AttachPrincipalPolicy`) para reducir el ámbito de los permisos asociados al grupo de identidades de Amazon Cognito con identidades autenticadas. Esto significa que una identidad de Amazon Cognito necesita el permiso de la política de rol de IAM asociada al grupo y la política de AWS IoT asociada a la identidad de Amazon Cognito mediante la API `AttachPrincipalPolicy` de AWS IoT.

- `iot:Connect`
- `iot:Publish`
- `iot:Subscribe`
- `iot:Receive`
- `iot:GetThingShadow`
- `iot:UpdateThingShadow`
- `iot>DeleteThingShadow`

#### Note

En el caso de otras operaciones de AWS IoT o de identidades sin autenticar, AWS IoT no reduce el ámbito de los permisos asociados al rol del grupo de identidades de Amazon Cognito. Para las identidades autenticadas y sin autenticar, esta es la política más permisiva que recomendamos asociar al rol del grupo de Amazon Cognito.

Para permitir que identidades de Amazon Cognito sin autenticar publiquen mensajes sobre HTTP en cualquier tema, asocie la política siguiente al rol de grupo de identidades de Amazon Cognito:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect",

```

```
        "iot:Publish",
        "iot:Subscribe",
        "iot:Receive",
        "iot:GetThingShadow",
        "iot:UpdateThingShadow",
        "iot:DeleteThingShadow"
    ],
    "Resource": ["*"]
}
}
```

Para permitir que identidades de Amazon Cognito sin autenticar publiquen mensajes MQTT a través de HTTP en cualquier tema de su cuenta, asocie la siguiente política al rol de grupo de identidades de Amazon Cognito:

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

#### Note

Este ejemplo se proporciona únicamente a título ilustrativo. A menos que sea absolutamente necesario para su servicio, recomendamos aplicar una política más restrictiva, que no permita a identidades de Amazon Cognito sin autenticar publicar en cualquier tema.

Para permitir que identidades de Amazon Cognito sin autenticar publiquen mensajes MQTT a través de HTTP en `topic1` en su cuenta, asocie la siguiente política a su rol de grupo de identidades de Amazon Cognito:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["iot:Publish"],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/topic1"]
        }
    ]
}
```

Para que una identidad de Amazon Cognito sin autenticar publique mensajes MQTT a través de HTTP en `topic1` en su cuenta de AWS, debe especificar dos políticas, tal y como se describe aquí. La primera política debe asociarse a un rol de grupo de identidades de Amazon Cognito. Permite que las identidades de dicho grupo realicen una llamada de publicación. La segunda política debe asociarse a un usuario de Amazon Cognito mediante la API [AttachPrincipalPolicy](#) de AWS IoT. Permite que el usuario de Amazon Cognito especificado tenga acceso al tema `topic1`.

Política de grupo de identidades de Amazon Cognito:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["iot:Publish"],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/topic1"]
        }
    ]
}
```

Política de usuario de Amazon Cognito:

```
{  
    "Version": "2012-10-17",  
    "Statement": [{  
        "Effect": "Allow",  
        "Action": ["iot:Publish"],  
        "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/topic1"]  
    }]  
}
```

Del mismo modo, el siguiente ejemplo de política permite al usuario de Amazon Cognito publicar mensajes MQTT a través de HTTP en los temas `topic1` y `topic2`. Se necesitan dos políticas. La primera política ofrece al rol de grupo de identidades de Amazon Cognito la posibilidad de realizar la llamada de publicación. La otra política ofrece al usuario de Amazon Cognito acceso a los temas `topic1` y `topic2`.

Política de grupo de identidades de Amazon Cognito:

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

Política de usuario de Amazon Cognito:

```
{  
    "Version": "2012-10-17",  
    "Statement": [{  
        "Effect": "Allow",  
        "Action": ["iot:Publish"],  
        "Resource": [  
            "arn:aws:iot:us-east-1:123456789012:topic/topic1",  
            "arn:aws:iot:us-east-1:123456789012:topic/topic2"  
        ]  
    }]  
}
```

Las siguientes políticas permiten a varios usuarios de Amazon Cognito publicar en un tema. Se necesitan dos políticas por identidad de Amazon Cognito. La primera política ofrece al rol de grupo de identidades de Amazon Cognito la posibilidad de realizar la llamada de publicación. La segunda y la tercera política dan a los usuarios de Amazon Cognito acceso a los temas `topic1` y `topic2`, respectivamente.

Política de grupo de identidades de Amazon Cognito:

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

Política de usuario 1 de Amazon Cognito:

```
{
```

```
"Version": "2012-10-17",
"Statement": [
    {
        "Effect": "Allow",
        "Action": ["iot:Publish"],
        "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/topic1"]
    }
]
```

Política de usuario 2 de Amazon Cognito:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["iot:Publish"],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/topic2"]
        }
}
```

### Ejemplos de políticas de recepción

La siguiente política impide que el titular del certificado, independientemente de su ID de cliente, reciba mensajes de un tema:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/foo/restricted"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:>"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

La siguiente política permite al titular de certificado, independientemente de su ID de cliente, suscribirse a un tema y recibir mensajes de dicho tema:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [ * ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/*"
            ]
        }
    ]
}
```

```
        "Effect": "Allow",
        "Action": [
            "iot:Subscribe"
        ],
        "Resource": [
            "arn:aws:iot:us-east-1:123456789012:topicfilter/foo/bar"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "iot:Receive"
        ],
        "Resource": [
            "arn:aws:iot:us-east-1:123456789012:topic/foo/bar"
        ]
    }
]
```

## Ejemplos de políticas de certificado

La siguiente política permite a un dispositivo publicar en un tema cuyo nombre sea igual al `certificateId` del certificado con el que se autenticó el dispositivo:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["iot:Publish"],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/${iot:CertificateId}"]
        },
        {
            "Effect": "Allow",
            "Action": ["iot:Connect"],
            "Resource": ["*"]
        }
    ]
}
```

La siguiente política permite a un dispositivo publicar en un tema cuyo nombre sea igual al campo de nombre común del asunto del certificado con el que se autenticó el dispositivo:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["iot:Publish"],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/
${iot:Certificate.Issuer.CommonName}"]
        },
        {
            "Effect": "Allow",
            "Action": ["iot:Connect"],
            "Resource": ["*"]
        }
    ]
}
```

La siguiente política permite a un dispositivo publicar en un tema que aparece prefijado con "admin /" cuando el certificado utilizado para autenticar el dispositivo tiene su campo `Subject.CommonName` establecido en "Administrator":

```
{
```

```
"Version": "2012-10-17",
"Statement": [
    {
        "Effect": "Allow",
        "Action": ["iot:Connect"],
        "Resource": ["*"]
    },
    {
        "Effect": "Allow",
        "Action": ["iot:Publish"],
        "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/admin/*"],
        "Condition": {
            "StringEquals": {
                "iot:Certificate.Subject.CommonName.2": "Administrator"
            }
        }
    }
]
```

La siguiente política permite a un dispositivo publicar en un tema que aparece prefijado con "admin /" cuando el certificado utilizado para autenticar el dispositivo tiene cualquiera de sus campos Subject.Common establecido en "Administrator":

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["iot:Connect"],
            "Resource": ["*"]
        },
        {
            "Effect": "Allow",
            "Action": ["iot:Publish"],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/admin/*"],
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "iot:Certificate.Subject.CommonName.List": "Administrator"
                }
            }
        }
    ]
}
```

## Ejemplos de políticas de objeto

La siguiente política permite a un objeto publicar en un tema específico que contenga el nombre del tipo de objeto y el nombre del objeto:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["iot:Publish"],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/
${iot:Connection.Thing.ThingType} / ${iot:Connection.Thing.ThingName}"
            ]
        }
    ]
}
```

La siguiente política permite al dispositivo conectarse si el certificado utilizado para realizar la autenticación en AWS IoT está asociado al objeto cuya política se evalúa.

```
{
```

```
"Version": "2012-10-17",
"Statement": [
    {
        "Effect": "Allow",
        "Action": ["iot:Connect"],
        "Resource": ["*"],
        "Condition": {
            "Bool": {
                "iot:Connection.Thing.IsAttached": ["true"]
            }
        }
    }
]
```

La siguiente política permite a un dispositivo publicar en un conjunto de temas ("/foo/bar" y "/foo/baz") si:

- El objeto asociado al dispositivo tiene un atributo denominado "Manufacturer" con un valor de "foo", "bar" o "baz".
- El objeto asociado al dispositivo existe en el registro de objetos y está asociado al certificado utilizado para conectarse con AWS IoT.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["iot:Publish"],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/foo/bar",
                "arn:aws:iot:us-east-1:123456789012:topic/foo/baz"
            ],
            "Condition": {
                "ForAnyValue:StringLike": {
                    "iot:Connection.Thing.Attributes[Manufacturer]": [
                        "foo",
                        "bar",
                        "baz"
                    ]
                }
            }
        }
    ]
}
```

La política siguiente permite a un dispositivo publicar en un tema si:

- El tema se compone de un nombre de tipo de objeto, un símbolo '/' y un nombre de objeto.
- El objeto existe en el registro de objetos.
- El objeto está asociado al certificado que se utiliza para conectarse con AWS IoT.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["iot:Publish"],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/
${iot:Connection.Thing.ThingType Name}/${iot:Connection.Thing.ThingName}"
            ]
        }
    ]
}
```

La siguiente política permite a un dispositivo publicar únicamente en su propio tema de sombra de objeto, si el objeto existe en el registro de objetos.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["iot:Publish"],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/
${iot:Connection.Thing.ThingName}/shadow/update"
            ]
        }
    ]
}
```

## Políticas de IoT de IAM

AWS Identity and Access Management define una acción de política para cada operación que AWS IoT defina, incluidas las API de plano de control y de plano de datos.

### Permisos de API de AWS IoT

En la siguiente tabla se muestra una lista de API de AWS IoT, los permisos de IAM necesarios y el recurso que la API manipula.

API	Permiso necesario (acciones de política)	Recursos
AcceptCertificateTransfer	iot:AcceptCertificateTransfer	arn:aws:iot: <i>region:account-id:cert/cert-id</i>
		Note  La cuenta de AWS especificada en el ARN debe ser la cuenta a la que se transfiere el certificado.
AttachPrincipalPolicy	iot:AttachPrincipalPolicy	arn:aws:iot: <i>region:account-id:cert/cert-id</i>
AttachThingPrincipal	iot:AttachThingPrincipal	arn:aws:iot: <i>region:account-id:cert/cert-id</i>
CancelCertificateTransfer	iot:CancelCertificateTransfer	arn:aws:iot: <i>region:account-id:cert/cert-id</i>
		Note  La cuenta de AWS especificada en el ARN debe ser la cuenta a la que se transfiere el certificado.
CreateCertificateFromCsr	iot:CreateCertificateFromCsr	
CreateKeysAndCertificate	iot>CreateKeysAndCertificate	
CreatePolicy	iot:CreatePolicy	*
CreatePolicyVersion	iot:CreatePolicyVersion	arn:aws:iot: <i>region:account-id:policy/policy-name</i>
		Note  Debe ser una política de AWS IoT y no una política de IAM.

API	Permiso necesario (acciones de política)	Recursos
CreateThing	iot:CreateThing	arn:aws:iot: <i>region</i> :account-id:thing/ <i>thing-name</i> .
CreateThingType	iot:CreateThingType	arn:aws:iot: <i>region</i> :account-id:thingtype/ <i>thing-type-name</i>
CreateTopicRule	iot:CreateTopicRule	arn:aws:iot: <i>region</i> :account-id:rule/ <i>rule-name</i>
DeleteCACertificate	iot>DeleteCACertificate	arn:aws:iot: <i>region</i> :account-id:cacert/ <i>cert-id</i>
DeleteCertificate	iot:DeleteCertificate	arn:aws:iot: <i>region</i> :account-id:cert/ <i>cert-id</i>
DeletePolicy	iot:DeletePolicy	arn:aws:iot: <i>region</i> :account-id:policy/ <i>policy-name</i>
DeletePolicyVersion	iot:DeletePolicyVersion	arn:aws:iot: <i>region</i> :account-id:policy/ <i>policy-name</i>
DeleteRegistrationCode	iot:DeleteRegistrationCode	
DeleteThing	iot:DeleteThing	arn:aws:iot: <i>region</i> :account-id:thing/ <i>thing-name</i>
DeleteThingType	iot:DeleteThingType	arn:aws:iot: <i>region</i> :account-id:thingtype/ <i>thing-type-name</i>
DeleteTopicRule	iot:DeleteTopicRule	arn:aws:iot: <i>region</i> :account-id:rule/ <i>rule-name</i>
DeprecateThingType	iot:DeprecateThingType	arn:aws:iot: <i>region</i> :account-id:thingtype/ <i>thing-type-name</i>
DescribeCaCertificate	iot:DescribeCaCertificate	arn:aws:iot: <i>region</i> :account-id:cacert/ <i>cert-id</i>
DescribeCertificate	iot:DescribeCertificate	arn:aws:iot: <i>region</i> :account-id:cert/ <i>cert-id</i>
DescribeEndpoint	iot:DescribeEndpoint	*
DescribeThing	iot:DescribeThing	arn:aws:iot: <i>region</i> :account-id:thing/ <i>thing-name</i>
DescribeThingType	iot:DescribeThingType	arn:aws:iot: <i>region</i> :account-id:thingtype/ <i>thing-type-name</i>
DetachPrincipalPolicy	iot:DetachPrincipalPolicy	arn:aws:iot: <i>region</i> :account-id:cert/ <i>cert-id</i>
DetachThingPrincipal	iot:DetachThingPrincipal	arn:aws:iot: <i>region</i> :account-id:cert/ <i>cert-id</i>
DisableTopicRule	iot:DisableTopicRule	arn:aws:iot: <i>region</i> :account-id:rule/ <i>rule-name</i>
EnableTopicRule	iot:EnableTopicRule	arn:aws:iot: <i>region</i> :account-id:rule/ <i>rule-name</i>
GetLoggingOptions	iot:GetLoggingOptions	\$
GetPolicy	iot:GetPolicy	arn:aws:iot: <i>region</i> :account-id:policy/ <i>policy-name</i>
GetPolicyVersion	iot:GetPolicyVersion	arn:aws:iot: <i>region</i> :account-id:policy/ <i>policy-name</i>
GetRegistrationCode	iot:GetRegistrationCode	
GetTopicRule	iot:GetTopicRule	arn:aws:iot: <i>region</i> :account-id:rule/ <i>rule-name</i>
ListCaCertificates	iot>ListCaCertificates	*
ListCertificates	iot>ListCertificates	*

API	Permiso necesario (acciones de política)	Recursos
iot>ListCertificatesByCa	iot>ListCertificatesByCa	
ListOutgoingCertificates	iot>ListOutgoingCertificates	
ListPolicies	iot>ListPolicies	*
ListPolicyPrincipals	iot>ListPolicyPrincipals	El ARN de la política: arn:aws:iot: <b>region:account-id:policy/policy-name</b>
ListPolicyVersions	iot>ListPolicyVersions	El ARN de la política: arn:aws:iot: <b>region:account-id:policy/policy-name</b>
ListPrincipalPolicies	iot>ListPrincipalPolicies	El ARN del certificado: arn:aws:iot: <b>region:account-id:cert/cert-id</b>
ListPrincipalThings	iot>ListPrincipalThings	El ARN del certificado: arn:aws:iot: <b>region:account-id:cert/cert-id</b>
ListThingPrincipals	iot>ListThingPrincipals	El ARN del objeto de AWS IoT: arn:aws:iot: <b>region:account-id:thing/thing-name</b>
ListThings	iot>ListThings	*
ListThingTypes	iot>ListThingTypes	*
ListTopicRules	iot>ListTopicRules	*
RegisterCACertificate	iot:RegisterCACertificate	
RegisterCertificate	iot:RegisterCertificate	*
RejectCertificateTransfer	iot:RejectCertificateTransfer	arn:aws:iot: <b>region:account-id:cert/cert-id</b>
ReplaceTopicRule	iot:ReplaceTopicRule	arn:aws:iot: <b>region:account-id:rule/rule-name</b>
SetDefaultPolicyVersion	iot:SetDefaultPolicyVersion	arn:aws:iot: <b>region:account-id:policy/policy-name</b>
SetLoggingOptions	iot:SetLoggingOptions	arn:aws:iot: <b>region:account-id:role/role-name</b>
TransferCertificate	iot:TransferCertificate	arn:aws:iot: <b>region:account-id:cert/cert-id</b>
UpdateCACertificate	iot:UpdateCACertificate	arn:aws:iot: <b>region:account-id:cacert/cert-id</b>
UpdateCertificate	iot:UpdateCertificate	arn:aws:iot: <b>region:account-id:cert/cert-id</b>
UpdateThing	iot:UpdateThing	arn:aws:iot: <b>region:account-id:thing/thing-name</b>

## Plantillas de política de IAM

AWS IoT proporciona un conjunto de plantillas de política de IAM que puede utilizar tal y como están, o bien como punto de partida para crear políticas de IAM personalizadas. Estas plantillas le permiten tener acceso a operaciones de configuración y de datos. Las operaciones de configuración le permiten crear objetos, certificados, políticas y reglas. Las operaciones de datos envían datos sobre protocolos MQTT y HTTP. En la tabla siguiente se describen estas plantillas.

Plantilla de política	Descripción
AWSIoTLogging	Permite a la identidad asociada configurar el registro de CloudWatch. Esta política se asocia al rol de registro de CloudWatch.
AWSIoTConfigAccess	Permite a la identidad asociada tener acceso a todas las operaciones de configuración de AWS IoT.
AWSIoTConfigReadOnlyAccess	Permite a la identidad asociada llamar a operaciones de configuración de solo lectura.
AWSIoTDataAccess	Permite a la identidad asociada tener acceso completo a todas las operaciones de datos de AWS IoT. Las operaciones de datos envían datos sobre protocolos MQTT y HTTP.
AWSIoTFullAccess	Permite a la identidad asociada tener acceso completo a todas las operaciones de datos y de configuración de AWS IoT.
AWSIoTRuleActions	Permite que la identidad asociada tenga acceso a todos los servicios de AWS admitidos en las acciones de regla de AWS IoT.

## Acceso entre cuentas

AWS IoT le permite habilitar un principal para publicar en un tema o suscribirse a un tema definido en una cuenta de AWS que no es propiedad del principal. El acceso entre cuentas se configura creando una política de IAM y un rol de IAM y, a continuación, asociando la política al rol.

En primer lugar, cree una política de IAM igual que lo haría para otros usuarios y certificados de su cuenta de AWS. Por ejemplo, la siguiente política otorga permisos para conectarse con el tema /foo/bar y publicar en él.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/foo/bar"
            ]
        }
    ]
}
```

A continuación, siga los pasos que figuran en [Creating a Role for an IAM User](#). Especifique el ID de la cuenta de AWS con la que quiere compartir el acceso. A continuación, en el último paso, asocie al rol la política que acaba de crear. Si posteriormente debe modificar el ID de cuenta de AWS al que concede acceso, puede utilizar el siguiente formato de política de confianza.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Principal": {  
                "AWS": "arn:aws:iam:us-east-1:111111111111:user/MyUser"  
            },  
            "Action": "sts:AssumeRole"  
        }  
    ]  
}
```

## Seguridad de transporte

El agente de mensajes de AWS IoT y el servicio Thing Shadows cifran toda la comunicación con TLS. TLS se utiliza para garantizar la confidencialidad de los protocolos de aplicación (MQTT, HTTP) que AWS IoT admite. TLS está disponible en una serie de lenguajes de programación y sistemas operativos.

Para MQTT, TLS cifra la conexión entre el dispositivo y el agente. AWS IoT utiliza la autenticación de cliente TLS para identificar dispositivos. Para HTTP, TLS cifra la conexión entre el dispositivo y el agente. La autenticación se delega a AWS Signature Version 4.

## Compatibilidad con el conjunto de cifrado de TLS

AWS IoT admite los conjuntos de cifrado siguientes:

- ECDHE-ECDSA-AES128-GCM-SHA256 (recomendado)
- ECDHE-RSA-AES128-GCM-SHA256 (recomendado)
- ECDHE-ECDSA-AES128-SHA256
- ECDHE-RSA-AES128-SHA256
- ECDHE-ECDSA-AES128-SHA
- ECDHE-RSA-AES128-SHA
- ECDHE-ECDSA-AES256-GCM-SHA384
- ECDHE-RSA-AES256-GCM-SHA384
- ECDHE-ECDSA-AES256-SHA384
- ECDHE-RSA-AES256-SHA384
- ECDHE-RSA-AES256-SHA
- ECDHE-ECDSA-AES256-SHA
- AES128-GCM-SHA256
- AES128-SHA256
- AES128-SHA
- AES256-GCM-SHA384
- AES256-SHA256
- AES256-SHA

# Agente de mensajes de AWS IoT

El agente de mensajes de AWS IoT es un servicio de agente de publicación/suscripción que permite enviar y recibir mensajes de AWS IoT. Durante la comunicación con AWS IoT, un cliente envía un mensaje dirigido a un tema como `Sensor/temp/room1`. A su vez, el agente de mensajes envía el mensaje a todos los clientes que se han registrado para recibir mensajes relativos a ese tema. El acto de enviar el mensaje se denomina publicación. El acto de registrarse para recibir mensajes de un filtro de temas se denomina suscripción.

El espacio de nombres del tema es independiente por cada par de cuenta y región de AWS. Por ejemplo, el tema `Sensor/temp/room1` de una cuenta de AWS no depende del tema `Sensor/temp/room1` de otra cuenta de AWS. Esto es válido también para las regiones. El tema `Sensor/temp/room1` de la misma cuenta de AWS en `us-east-1` no depende del mismo tema en `us-east-2`. AWS IoT no es compatible con el envío y la recepción de mensajes en las cuentas y las regiones de AWS.

El agente de mensajes mantiene una lista de todas las sesiones y las suscripciones de cliente de cada sesión. Cuando se publica un mensaje en un tema, el agente busca sesiones que tengan suscripciones que coincidan con el tema. A continuación, el agente publica el mensaje en todas las sesiones que tengan un cliente conectado en ese momento.

## Protocolos

El agente de mensajes admite el uso del protocolo MQTT para publicar y suscribirse, y el protocolo HTTPS para publicar. Ambos protocolos se admiten mediante IP (versiones 4 y 6). El agente de mensajes también admite MQTT sobre el protocolo WebSocket.

### Protocolo/mapeos de puerto

En la tabla siguiente se muestran todos los protocolos que AWS IoT admite, el método de autenticación y el puerto utilizado para cada protocolo.

Protocolo, autenticación y mapeos de puerto

Protocolo	Autenticación	Puerto
MQTT	Certificado de cliente	8883
HTTP	Certificado de cliente	8443
HTTP	SigV4	443
MQTT + WebSocket	SigV4	443

## MQTT

MQTT es un protocolo de mensajería ligero muy utilizado, diseñado para dispositivos limitados. Para obtener más información, consulte [MQTT](#).

Aunque la implementación del agente de mensajes de AWS IoT esté basada en MQTT (versión 3.1.1), se desvía de la especificación de esta manera:

- En AWS IoT, suscribirse a un tema con calidad de servicio (QoS) 0 significa que un mensaje se entregará cero o más veces. Es posible que un mensaje se entregue más de una vez. Los mensajes que se entreguen más de una vez pueden enviarse con otro ID de paquete. En tal caso, la marca DUP no se establece.
- AWS IoT no es compatible con la publicación y suscripción mediante QoS 2. El agente de mensajes AWS IoT no envía PUBACK o SUBACK cuando se solicita QoS 2.
- Los niveles de QoS para publicar en un tema y suscribirse a este no están relacionados entre sí. Un cliente puede suscribirse a un tema con QoS 1, mientras que otro puede publicar en el mismo tema con QoS 0.
- Cuando responde a una solicitud de conexión, el agente de mensajes envía un mensaje CONNACK. Este mensaje contiene una marca para indicar si la conexión reanuda una sesión anterior. El valor de esta marca puede ser incorrecto si dos clientes MQTT se conectan a la vez con el mismo ID de cliente.
- Cuando un cliente se suscribe a un tema, puede haber un retraso entre el momento en que el agente de mensajes envía un SUBACK y el momento en que el cliente empieza a recibir nuevos mensajes coincidentes.
- La especificación MQTT ofrece una cláusula que permite a un editor solicitar al agente que conserve el último mensaje enviado a un tema y que lo envíe a todos los futuros suscriptores de dicho tema. AWS IoT no admite mensajes conservados. Si se envía una solicitud para conservar mensajes, se interrumpe la conexión.
- El agente de mensajes utiliza el ID de cliente para identificar a cada cliente. El ID de cliente se transfiere desde el cliente al agente de mensajes como parte de la carga de MQTT. Dos clientes que tengan el mismo ID de cliente no pueden conectarse simultáneamente al agente de mensajes. Cuando un cliente se conecta al intermediario de mensajes mediante un ID de cliente que otro cliente ya está utilizando, se envía un mensaje CONNACK a ambos clientes y se interrumpe la conexión del cliente que está conectado en ese momento.
- El agente de mensajes no es compatible con las sesiones persistentes (conexiones realizadas con la marca `cleanSession` establecida en `false`). El agente de mensajes de AWS IoT presupone que todas las sesiones son válidas y que los mensajes no se almacenan entre sesiones. Si un cliente de MQTT intenta conectarse al agente de mensajes AWS IoT con `cleanSession` establecido en `false`, se desconectará.
- En raras ocasiones, el agente de mensajes reenviará el mismo mensaje PUBLISH lógico con otro ID de paquete.
- El agente de mensajes no garantiza el orden de recepción de los mensajes y ACK.

## HTTP

El agente de mensajes es compatible con los clientes que se conectan con el protocolo HTTP mediante una API REST. Los clientes pueden publicar enviando un mensaje POST a `<AWS IoT Endpoint>/topics/<url_encoded_topic_name>?qos=1`".

Por ejemplo, puede utilizar `curl` para emular una pulsación de botón. Si ha seguido el tutorial [Introducción a AWS IoT \(p. 5\)](#), en vez de utilizar el cliente MQTT de AWS IoT para publicar un mensaje como en [Cliente MQTT de AWS IoT \(p. 29\)](#), utilice un comando similar a este:

```
curl --tlsv1.2 --cacert root-CA.crt --cert 4b7828d2e5-certificate.pem.crt --key 4b7828d2e5-private.pem.key -X POST -d "{\"serialNumber\": \"G030JF053216F1BS\", \"clickType":
```

```
\" : \"SINGLE\", \"batteryVoltage\" : \"2000mV\" }" "https://a1pn10j0v8htvw.iot.us-east-1.amazonaws.com:8443/topics/iotbutton/virtualButton?qos=1"
```

--tlsv1.2

Utilice TLSv1.2 (SSL). curl debe instalarse con OpenSSL y debe utilizar la versión 1.2 de TLS.

--cacert <nombre de archivo>

El nombre de archivo del certificado de CA para verificar el homólogo.

--cert <nombre de archivo>

El nombre de archivo del certificado de cliente.

--key <nombre de archivo>

El nombre de archivo de la clave privada.

-X POST

El tipo de solicitud; en este caso, POST.

-d <datos>

Los datos de HTTP POST que quiere publicar. En este caso, emulamos los datos que se envían pulsando una única vez el botón.

"https://..."

La URL. En este caso, el punto de enlace de la API REST del objeto. (Para encontrar el punto de enlace de un objeto, elija Registry en la consola de AWS IoT para ampliar las opciones disponibles. Elija Things, seleccione el objeto y, a continuación, elija Interact). Después del punto de enlace, agregue el puerto (: 8443) seguido del tema y, por último, especifique la calidad de servicio en una cadena de consulta (?qos=1).

## MQTT sobre el protocolo WebSocket

AWS IoT admite MQTT sobre el protocolo [WebSocket](#) para permitir a las aplicaciones basadas en el navegador enviar y recibir datos a partir de dispositivos conectados a AWS IoT con credenciales de AWS. Las credenciales de AWS se especifican mediante [AWS Signature Version 4](#). El puerto TCP 443 admite WebSocket y permite a los mensajes atravesar la mayoría de firewalls y servidores proxy web.

Una conexión WebSocket se inicia en un cliente enviando una solicitud HTTP GET. La dirección URL que utilice tiene la forma siguiente:

```
wss://<endpoint>.iot.<region>.amazonaws.com/mqtt
```

wss

Especifica el protocolo WebSocket.

endpoint

Su punto de enlace de AWS IoT específico de la cuenta de AWS. Puede utilizar el comando de CLI de AWS IoT [describe-endpoint](#) para encontrar este punto de enlace.

region

La región de AWS de su cuenta de AWS.

mqtt

Especifica que enviará mensajes MQTT mediante el protocolo WebSocket.

Cuando el servidor responde, el cliente envía una solicitud de actualización para indicar al servidor que se comunicará mediante el protocolo WebSocket. Después de que el servidor reconozca la solicitud de actualización, toda la comunicación se realizará mediante el protocolo WebSocket. La implementación de WebSocket que utilice actúa como protocolo de transporte. Los datos que envíe mediante el protocolo WebSocket son mensajes MQTT.

## Uso del protocolo WebSocket en una aplicación web

La implementación WebSocket proporcionada por la mayoría de los navegadores web no permite la modificación de encabezados HTTP, por lo que debe agregar la información de Signature Version 4 a la cadena de consulta. Para obtener más información, consulte [Adición de la información de firma a la cadena de consulta](#).

El código JavaScript siguiente define algunas funciones de utilidades para generar una solicitud Signature Version 4.

```
/**  
 * utilities to do sigv4  
 * @class SigV4Utils  
 */  
function SigV4Utils() {}  
  
SigV4Utils.getSignatureKey = function (key, date, region, service) {  
    var kDate = AWS.util.crypto.hmac('AWS4' + key, date, 'buffer');  
    var kRegion = AWS.util.crypto.hmac(kDate, region, 'buffer');  
    var kService = AWS.util.crypto.hmac(kRegion, service, 'buffer');  
    var kCredentials = AWS.util.crypto.hmac(kService, 'aws4_request', 'buffer');  
    return kCredentials;  
};  
  
SigV4Utils.getSignedUrl = function(host, region, credentials) {  
    var datetime = AWS.util.date.iso8601(new Date()).replace(/[:-]|\.\d{3}/g, '');  
    var date = datetime.substr(0, 8);  
  
    var method = 'GET';  
    var protocol = 'wss';  
    var uri = '/mqtt';  
    var service = 'iotdevicegateway';  
    var algorithm = 'AWS4-HMAC-SHA256';  
  
    var credentialScope = date + '/' + region + '/' + service + '/' + 'aws4_request';  
    var canonicalQueryString = 'X-Amz-Algorithm=' + algorithm;  
    canonicalQueryString += '&X-Amz-Credential=' +  
    encodeURIComponent(credentials.accessKeyId + '/' + credentialScope);  
    canonicalQueryString += '&X-Amz-Date=' + datetime;  
    canonicalQueryString += '&X-Amz-SignedHeaders=host';  
  
    var canonicalHeaders = 'host:' + host + '\n';  
    var payloadHash = AWS.util.crypto.sha256('', 'hex')  
    var canonicalRequest = method + '\n' + uri + '\n' + canonicalQueryString + '\n' +  
    canonicalHeaders + '\nhost\n' + payloadHash;  
  
    var stringToSign = algorithm + '\n' + datetime + '\n' + credentialScope + '\n' +  
    AWS.util.crypto.sha256(canonicalRequest, 'hex');  
    var signingKey = SigV4Utils.getSignatureKey(credentials.secretAccessKey, date, region,  
    service);  
    var signature = AWS.util.crypto.hmac(signingKey, stringToSign, 'hex');  
  
    canonicalQueryString += '&X-Amz-Signature=' + signature;  
    if (credentials.sessionToken) {  
        canonicalQueryString += '&X-Amz-Security-Token=' +  
        encodeURIComponent(credentials.sessionToken);  
    }  
}
```

```
    var requestUrl = protocol + '://' + host + uri + '?' + canonicalQuerystring;
    return requestUrl;
};
```

#### Para crear una solicitud Signature Version 4

1. Cree una solicitud canónica para Signature Version 4.

El siguiente código JavaScript crea una solicitud canónica:

```
var datetime = AWS.util.date.iso8601(new Date()).replace(/[:\:-]|\.\d{3}/g, '');
var date = datetime.substr(0, 8);

var method = 'GET';
var protocol = 'wss';
var uri = '/mqtt';
var service = 'iotdevicegateway';
var algorithm = 'AWS4-HMAC-SHA256';

var credentialScope = date + '/' + region + '/' + service + '/' + 'aws4_request';
var canonicalQueryString = 'X-Amz-Algorithm=' + algorithm;
canonicalQueryString += '&X-Amz-Credential=' +
    encodeURIComponent(credentials.accessKeyId + '/' + credentialScope);
canonicalQueryString += '&X-Amz-Date=' + datetime;
canonicalQueryString += '&X-Amz-SignedHeaders=host';

var canonicalHeaders = 'host:' + host + '\n';
var payloadHash = AWS.util.crypto.sha256('', 'hex')
var canonicalRequest = method + '\n' + uri + '\n' + canonicalQueryString + '\n' +
    canonicalHeaders + '\nhost\n' + payloadHash;
```

2. Cree una cadena para firmar, genere una clave de firma y firme la cadena.

Tome la URL canónica que ha creado en el paso anterior e insértela en una cadena para firmar. Para ello, cree una cadena formada por el algoritmo de hash, la fecha, el ámbito de las credenciales y el SHA de la solicitud canónica. A continuación, genere la clave de firma y firme la cadena, tal y como se muestra en el siguiente código JavaScript.

```
var stringToSign = algorithm + '\n' + datetime + '\n' + credentialScope + '\n' +
    AWS.util.crypto.sha256(canonicalRequest, 'hex');
var signingKey = SigV4Utils.getSignatureKey(credentials.secretAccessKey, date, region,
    service);
var signature = AWS.util.crypto.hmac(signingKey, stringToSign, 'hex');
```

3. Agregue la información de firma a la solicitud.

El siguiente código JavaScript muestra cómo agregar la información de firma a la cadena de consulta.

```
canonicalQueryString += '&X-Amz-Signature=' + signature;
```

4. Si tiene credenciales de sesión (de un servidor STS, AssumeRole o Amazon Cognito), anexe el token de sesión al final de la cadena URL después de la firma:

```
canonicalQueryString += '&X-Amz-Security-Token=' +
encodeURIComponent(credentials.sessionToken);
```

5. Anexe el protocolo, el host y el URI a canonicalQueryString (al principio de la cadena):

```
var requestUrl = protocol + '://' + host + uri + '?' + canonicalQueryString;
```

6. Abra WebSocket.

El siguiente código JavaScript muestra cómo crear un cliente Paho MQTT y hacer una llamada CONNECT a AWS IoT. El argumento endpoint corresponde al punto de enlace específico de su cuenta de AWS. El clientId es un identificador de texto único entre todos los clientes conectados simultáneamente en su cuenta de AWS.

```
var client = new Paho.MQTT.Client(requestUrl, clientId);
var connectOptions = {
    onSuccess: function(){
        // connect succeeded
    },
    useSSL: true,
    timeout: 3,
    mqttVersion: 4,
    onFailure: function() {
        // connect failed
    }
};
client.connect(connectOptions);
```

## Uso del protocolo WebSocket en una aplicación móvil

Le recomendamos que utilice uno de los SDK de dispositivos de AWS IoT para conectar su dispositivo a AWS IoT cuando establezca una conexión WebSocket. Los siguientes SDK de dispositivos de AWS IoT son compatibles con las conexiones MQTT a AWS IoT basadas en WebSocket:

- [Node.js](#)
- [iOS](#)
- [Android](#)

Encontrará una implementación de referencia para conectar una aplicación web a AWS IoT mediante MQTT sobre el protocolo WebSocket en [AWS Labs WebSocket sample](#).

Si utiliza un lenguaje de programación o de script que no esté admitido actualmente, puede utilizar cualquier biblioteca WebSocket existente siempre y cuando la solicitud de actualización WebSocket inicial (HTTP POST) se conecte mediante AWS Signature Version 4. Algunos clientes MQTT, como [Eclipse Paho para JavaScript](#), admiten el protocolo WebSocket de manera nativa.

## Temas

El agente de mensajes utiliza temas para dirigir los mensajes de los clientes de publicación a los clientes de suscripción. La barra inclinada (/) se utiliza para separar la jerarquía de temas. En la tabla siguiente se muestra una lista de los comodines que se pueden utilizar en el filtro de temas al suscribirse.

### Comodines de tema

Comodín	Descripción
#	Debe ser el último carácter del tema al que se suscribe. Funciona como comodín haciendo coincidir el árbol actual con todos los subárboles. Por ejemplo, una suscripción a Sensor/# recibirá los mensajes publicados en Sensor/, Sensor/temp, Sensor/temp/room1, pero no los mensajes publicados en Sensor.
+	Correlaciona exactamente un elemento de la jerarquía de temas. Por ejemplo, una suscripción a Sensor/+/room1 recibirá los mensajes publicados en Sensor/temp/room1, Sensor/moisture/room1, y así sucesivamente.

## Temas reservados

Todos los temas que empiecen por \$ se consideran reservados y no son compatibles con operaciones de publicación ni suscripción, salvo en el caso de los temas de la lista siguiente. Cualquier otro intento de publicar en los temas que empiezan por \$ o de suscribirse a ellos interrumpirá la conexión.

Tema	Operaciones permitidas	Descripción
\$aws/events/presence/connected/ <i>clientId</i>	Subscribe	AWS IoT publica en este tema cuando un cliente MQTT con el ID de cliente especificado se conecta a AWS IoT. Para obtener más información, consulte <a href="#">Eventos de conexión/desconexión (p. 146)</a> .
\$aws/events/presence/disconnected/ <i>clientId</i>	Subscribe	AWS IoT publica en este tema cuando un cliente MQTT con el ID de cliente especificado se desconecta de AWS IoT. Para obtener más información, consulte <a href="#">Eventos de conexión/desconexión (p. 146)</a> .
\$aws/events/subscriptions/subscribed/ <i>clientId</i>	Subscribe	AWS IoT publica en este tema cuando un cliente MQTT con el ID de cliente especificado se suscribe a un tema MQTT. Para obtener más información, consulte <a href="#">Eventos de suscripción/cancelación de suscripción (p. 146)</a> .
\$aws/events/subscriptions/unsubscribed/ <i>clientId</i>	Subscribe	AWS IoT publica en este tema cuando un cliente MQTT con el ID de cliente especificado anula su suscripción a un tema

Tema	Operaciones permitidas	Descripción
		MQTT. Para obtener más información, consulte <a href="#">Eventos de suscripción/cancelación de suscripción (p. 146)</a> .
\$aws/things/ <i>thingName</i> /shadow/delete	Publicar/suscribirse	Un objeto o una aplicación publica en este tema para eliminar una sombra de objeto. Para obtener más información, consulte <a href="http://docs.aws.amazon.com/iot/latest/developerguide//thing-shadow-mqtt.html#delete-pub-sub-topic">http://docs.aws.amazon.com/iot/latest/developerguide//thing-shadow-mqtt.html#delete-pub-sub-topic</a> .
\$aws/things/ <i>thingName</i> /shadow/delete/accepted	Subscribe	El servicio Thing Shadows envía mensajes a este tema cuando se elimina una sombra de objeto. Para obtener más información, consulte <a href="http://docs.aws.amazon.com/iot/latest/developerguide//thing-shadow-mqtt.html#delete-accepted-pub-sub-topic">http://docs.aws.amazon.com/iot/latest/developerguide//thing-shadow-mqtt.html#delete-accepted-pub-sub-topic</a> .
\$aws/things/ <i>thingName</i> /shadow/delete/rejected	Subscribe	El servicio Thing Shadows envía mensajes a este tema cuando se rechaza una solicitud para eliminar una sombra de objeto. Para obtener más información, consulte <a href="http://docs.aws.amazon.com/iot/latest/developerguide//thing-shadow-mqtt.html#delete-rejected-pub-sub-topic">http://docs.aws.amazon.com/iot/latest/developerguide//thing-shadow-mqtt.html#delete-rejected-pub-sub-topic</a> .
\$aws/things/ <i>thingName</i> /shadow/get	Publicar/suscribirse	Una aplicación o un objeto publica un mensaje vacío en este tema para obtener una sombra de objeto. Para obtener más información, consulte <a href="http://docs.aws.amazon.com/iot/latest/developerguide//thing-shadow-mqtt.html">http://docs.aws.amazon.com/iot/latest/developerguide//thing-shadow-mqtt.html</a> .

Tema	Operaciones permitidas	Descripción
\$aws/things/ <i>thingName</i> /shadow/get/accepted	Subscribe	El servicio Thing Shadows envía mensajes a este tema cuando se realiza correctamente una solicitud de una sombra de objeto. Para obtener más información, consulte <a href="http://docs.aws.amazon.com/iot/latest/developerguide//thing-shadow-mqtt.html#get-accepted-pub-sub-topic">http://docs.aws.amazon.com/iot/latest/developerguide//thing-shadow-mqtt.html#get-accepted-pub-sub-topic</a> .
\$aws/things/ <i>thingName</i> /shadow/get/rejected	Subscribe	El servicio Thing Shadows envía mensajes a este tema cuando se rechaza una solicitud de una sombra de objeto. Para obtener más información, consulte <a href="http://docs.aws.amazon.com/iot/latest/developerguide//thing-shadow-mqtt.html#get-rejected-pub-sub-topic">http://docs.aws.amazon.com/iot/latest/developerguide//thing-shadow-mqtt.html#get-rejected-pub-sub-topic</a> .
\$aws/things/ <i>thingName</i> /shadow/update	Publicar/suscribirse	Un objeto o una aplicación publica en este tema para actualizar una sombra de objeto. Para obtener más información, consulte <a href="http://docs.aws.amazon.com/iot/latest/developerguide//thing-shadow-mqtt.html#update-pub-sub-topic">http://docs.aws.amazon.com/iot/latest/developerguide//thing-shadow-mqtt.html#update-pub-sub-topic</a> .
\$aws/things/ <i>thingName</i> /shadow/update/accepted	Subscribe	El servicio Thing Shadows envía mensajes a este tema cuando se realiza correctamente una actualización en una sombra de objeto. Para obtener más información, consulte <a href="http://docs.aws.amazon.com/iot/latest/developerguide//thing-shadow-mqtt.html#update-accepted-pub-sub-topic">http://docs.aws.amazon.com/iot/latest/developerguide//thing-shadow-mqtt.html#update-accepted-pub-sub-topic</a> .
\$aws/things/ <i>thingName</i> /shadow/update/rejected	Subscribe	El servicio Thing Shadows envía mensajes a este tema cuando se rechaza una actualización en una sombra de objeto. Para obtener más información, consulte <a href="http://docs.aws.amazon.com/iot/latest/developerguide//thing-shadow-mqtt.html#update-rejected-pub-sub-topic">http://docs.aws.amazon.com/iot/latest/developerguide//thing-shadow-mqtt.html#update-rejected-pub-sub-topic</a> .

Tema	Operaciones permitidas	Descripción
\$aws/things/ <i>thingName</i> /shadow/update/delta	Subscribe	El servicio Thing Shadows envía mensajes a este tema cuando se detecta una diferencia entre las secciones notificadas y las secciones deseadas de una sombra de objeto. Para obtener más información, consulte <a href="http://docs.aws.amazon.com/iot/latest/developerguide/thing-shadow-mqtt.html#update-delta-pub-sub-topic">http://docs.aws.amazon.com/iot/latest/developerguide/thing-shadow-mqtt.html#update-delta-pub-sub-topic</a> .
\$aws/things/ <i>thingName</i> /shadow/update/documents	Subscribe	AWS IoT publica un documento de estado en este tema siempre que se realiza una actualización correcta de la sombra. Para obtener más información, consulte <a href="http://docs.aws.amazon.com/iot/latest/developerguide/thing-shadow-mqtt.html#update-documents-pub-sub-topic">http://docs.aws.amazon.com/iot/latest/developerguide/thing-shadow-mqtt.html#update-documents-pub-sub-topic</a> .

## Eventos del ciclo de vida

AWS IoT publica eventos del ciclo de vida en los temas MQTT tratados en las siguientes secciones. Estos mensajes le permiten recibir notificaciones de los eventos del ciclo de vida desde el agente de mensajes.

### Note

Es posible que los mensajes de ciclo de vida se envíen de forma desordenada y que algunos de los mensajes que reciba estén duplicados.

## Política necesaria para recibir eventos de ciclo de vida

A continuación, mostramos un ejemplo de política necesaria para recibir eventos del ciclo de vida:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe",
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:region:account:topicfilter/$aws/events/*"
            ]
        }
    ]
}
```

## Eventos de conexión/desconexión

AWS IoT publica un mensaje en los siguientes temas MQTT cuando un cliente se conecta o se desconecta:

```
$aws/events/presence/connected/clientId
```

o bien

```
$aws/events/presence/disconnected/clientId
```

Donde *clientId* es el ID de cliente MQTT que se conecta con el agente de mensajes de AWS IoT o se desconecta de este.

El mensaje publicado en este tema tiene la estructura siguiente:

```
{  
    "clientId": "a1b2c3d4e5f6a7b8c9d0e1f2a3b4c5d6",  
    "timestamp": 1460065214626,  
    "eventType": "connected",  
    "sessionIdentifier": "00000000-0000-0000-0000-000000000000",  
    "principalIdentifier": "000000000000/ABCDEFGHIJKLMNPQRSTUVWXYZ:some-user/  
    ABCDEFGHIJKLMNOPQRSTUVWXYZ:some-user"  
}
```

A continuación, se muestra una lista de elementos JSON que se encuentran en los mensajes de conexión/desconexión publicados en el tema `$aws/events/presence/connected/clientId`.

**clientId**

El ID del cliente que se conecta o se desconecta.

**Note**

Los ID de cliente que contienen los símbolos # o + no recibirán eventos del ciclo de vida.

**eventType**

El tipo de evento. Los valores válidos son `connected` o `disconnected`.

**principalIdentifier**

Las credenciales que se utilizan para la autenticación. En el caso de los certificados de autenticación mutua de TLS, se trata del ID de certificado. En cuanto a las demás conexiones, se trata de las credenciales de IAM.

**sessionIdentifier**

Un identificador único global de AWS IoT que existe durante toda la vida de la sesión.

**timestamp**

Aproximación del momento en que se produjo el evento, expresada en milisegundos según la fecha de inicio Unix. La precisión de la marca de tiempo es de +/- 2 minutos.

## Eventos de suscripción/cancelación de suscripción

AWS IoT publica un mensaje en el tema MQTT siguiente cuando un cliente se suscribe a un tema MQTT o cancela su suscripción a este:

```
$aws/events/subscriptions/subscribed/clientId
```

o bien

```
$aws/events/subscriptions/unsubscribed/clientId
```

Donde *clientId* es el ID de cliente MQTT que se conecta con el agente de mensajes de AWS IoT.

El mensaje publicado en este tema tiene la estructura siguiente:

```
{  
    "clientId": "186b5",  
    "timestamp": 1460065214626,  
    "eventType": "subscribed" | "unsubscribed",  
    "sessionIdentifier": "00000000-0000-0000-0000-000000000000",  
    "principalIdentifier": "000000000000/ABCDEFGHIJKLMNPQRSTUVWXYZ:some-user/  
ABCDEFGHIJKLMNPQRSTUVWXYZ:some-user"  
    "topics" : ["foo/bar", "device/data", "dog/cat"]  
}
```

A continuación, se ofrece una lista de elementos JSON que se encuentran en los mensajes suscritos y no suscritos publicados en los temas `$aws/events/subscriptions/subscribed/clientId` y `$aws/events/subscriptions/unsubscribed/clientId`.

#### clientId

El ID del cliente que se suscribe o cancela su suscripción.

#### Note

Los ID de cliente que contienen los símbolos # o + no recibirán eventos del ciclo de vida.

#### eventType

El tipo de evento. Los valores válidos son `subscribed` o `unsubscribed`.

#### principalIdentifier

Las credenciales que se utilizan para la autenticación. En el caso de los certificados de autenticación mutua de TLS, se trata del ID de certificado. En cuanto a las demás conexiones, se trata de las credenciales de IAM.

#### sessionIdentifier

Un identificador único global de AWS IoT que existe durante toda la vida de la sesión.

#### timestamp

Aproximación del momento en que se produjo el evento, expresada en milisegundos según la fecha de inicio Unix. La precisión de la marca de tiempo es de +/- 2 minutos.

#### topics

Una matriz de los temas MQTT a los que se ha suscrito el cliente.

#### Note

Es posible que los mensajes de ciclo de vida se envíen de forma desordenada. Puede que reciba mensajes duplicados.

# Reglas de AWS IoT

Las reglas permiten a sus dispositivos interactuar con los servicios de AWS. Las reglas se analizan y las acciones se ejecutan en función del flujo de temas MQTT. Puede utilizar reglas para admitir tareas como las siguientes:

- Incrementar o filtrar los datos recibidos desde un dispositivo.
- Escribir datos recibidos desde un dispositivo en una base de datos de Amazon DynamoDB.
- Guardar un archivo en Amazon S3.
- Enviar una notificación de inserción a todos los usuarios mediante Amazon SNS.
- Publicar datos en una cola de Amazon SQS.
- Invocar una función Lambda para extraer datos.
- Procesar mensajes de un gran número de dispositivos mediante Amazon Kinesis.
- Enviar datos a Amazon Elasticsearch Service.
- Capturar una métrica de CloudWatch.
- Cambiar una alarma de CloudWatch.
- Enviar los datos de un mensaje MQTT a Amazon Machine Learning para realizar predicciones basadas en un modelo de Amazon ML.
- Enviar un mensaje a un flujo de entrada de Salesforce IoT

Para que AWS IoT pueda realizar estas acciones, debe concederle permiso de acceso a los recursos de AWS en su nombre. Cuando dichas acciones se ejecuten, se le cobrará la tarifa estándar de los servicios de AWS que utilice.

## Contenido

- [Concesión a AWS IoT del acceso requerido \(p. 149\)](#)
- [Transmisión de los permisos de rol \(p. 150\)](#)
- [Creación de una regla de AWS IoT \(p. 151\)](#)
- [Visualización de las reglas \(p. 154\)](#)
- [Versiones de SQL \(p. 154\)](#)
- [Solución de problemas de una regla \(p. 156\)](#)

- [Eliminación de una regla \(p. 156\)](#)
- [Acciones de las reglas de AWS IoT \(p. 156\)](#)
- [Referencias de SQL en AWS IoT \(p. 168\)](#)

## Concesión a AWS IoT del acceso requerido

Los roles de IAM le permiten controlar los recursos de AWS a los que cada regla tiene acceso. Para crear una regla, primero debe crear un rol de IAM con una política que le permita tener acceso a los recursos de AWS necesarios. AWS IoT asume este rol al ejecutar una regla.

Para crear un rol de IAM (AWS CLI)

1. Guarde el siguiente documento de política de confianza, que concede a AWS IoT permiso para asumir el rol, en un archivo llamado iot-role-trust.json:

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

Ejecute el comando [create-role](#) para crear un rol de IAM que especifique el archivo iot-role-trust.json:

```
aws iam create-role --role-name my-iot-role --assume-role-policy-document file://iot-role-trust.json
```

El resultado de este comando tendrá este aspecto:

```
{  
    "Role": {  
        "AssumeRolePolicyDocument": "url-encoded-json",  
        "RoleId": "AKIAIOSFODNN7EXAMPLE",  
        "CreateDate": "2015-09-30T18:43:32.821Z",  
        "RoleName": "my-iot-role",  
        "Path": "/",  
        "Arn": "arn:aws:iam::123456789012:role/my-iot-role"  
    }  
}
```

2. Guarde el siguiente JSON en un archivo llamado iot-policy.json.

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

Este JSON es un ejemplo de documento de política que concede a AWS IoT acceso de administrador a DynamoDB.

Ejecute el comando [create-policy](#) para conceder a AWS IoT acceso a sus recursos de AWS en el momento de asumir el rol y transmitir el archivo iot-policy.json:

```
aws iam create-policy --policy-name my-iot-policy --policy-document file://my-iot-policy-document.json
```

Para obtener más información acerca de cómo conceder el acceso a los servicios de AWS en las políticas de AWS IoT, consulte [Creación de una regla de AWS IoT \(p. 151\)](#).

La salida del comando [create-policy](#) contendrá el ARN de la política. Tendrá que asociar la política a un rol.

```
{  
    "Policy": {  
        "PolicyName": "my-iot-policy",  
        "CreateDate": "2015-09-30T19:31:18.620Z",  
        "AttachmentCount": 0,  
        "IsAttachable": true,  
        "PolicyId": "ZXRX6A36LTYANPAI7NJ5UV",  
        "DefaultVersionId": "v1",  
        "Path": "/",  
        "Arn": "arn:aws:iam::123456789012:policy/my-iot-policy",  
        "UpdateDate": "2015-09-30T19:31:18.620Z"  
    }  
}
```

3. Ejecute el comando [attach-role-policy](#) para asociar la política al rol:

```
aws iam attach-role-policy --role-name my-iot-role --policy-arn  
"arn:aws:iam::123456789012:policy/my-iot-policy"
```

## Transmisión de los permisos de rol

La definición de una regla implica que un rol de IAM conceda permiso para tener acceso a los recursos especificados en la acción de la regla. El motor de reglas asume dicho rol cuando se desencadena la acción de la regla. El rol tiene que estar definido en la misma cuenta de AWS que la regla.

De hecho, cuando crea o sustituye una regla, está transfiriendo un rol al motor de reglas. El usuario que realiza esta operación necesita el permiso `iam:PassRole`. Para asegurarse de que dispone de dicho permiso, cree una política que conceda el permiso `iam:PassRole` y asóciela a su usuario de IAM. La política siguiente muestra cómo conceder un permiso `iam:PassRole` para un rol.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "Stmt1",  
            "Effect": "Allow",  
            "Action": [  
                "iam:PassRole"  
            ],  
            "Resource": [  
                "arn:aws:iam::123456789012:role/myRole"  
            ]  
        }  
    ]  
}
```

En este ejemplo de política, se concede el permiso `iam:PassRole` para el rol `myRole`. El rol se especifica mediante el ARN del rol. Debe asociar esta política al usuario o al rol de IAM al que pertenezca el usuario. Para obtener más información, consulte [Working with Managed Policies](#).

**Note**

Las funciones Lambda utilizan una política basada en recursos. Dicha política está directamente asociada a la función Lambda en sí. Cuando crea una regla que invoca una función Lambda, no transmite un rol, por lo que el usuario que crea la regla no necesita el permiso `iam:PassRole`. Para obtener más información acerca de la autorización de función Lambda, consulte [Granting Permissions Using a Resource Policy](#).

## Creación de una regla de AWS IoT

Puede configurar las reglas para direccionar datos desde las cosas conectadas. Las reglas constan de los elementos siguientes:

**Nombre de la regla**

El nombre de la regla.

**Descripción opcional**

Descripción textual de la regla.

**Instrucción SQL**

Sintaxis de SQL simplificada para filtrar los mensajes recibidos sobre un tema MQTT e insertar los datos en otro punto. Para obtener más información, consulte [Referencias de SQL en AWS IoT \(p. 168\)](#).

**Versión de SQL**

Versión del motor de reglas SQL que debe utilizarse al evaluar la regla. Aunque esta propiedad es opcional, recomendamos encarecidamente que especifique la versión de SQL. Si no se define esta propiedad, se usará el valor predeterminado 2015-10-08.

**Una o varias acciones**

Las acciones que AWS IoT realiza al ejecutar la regla. Por ejemplo, puede insertar datos en una tabla de DynamoDB, escribir datos en un bucket de Amazon S3, publicar en un tema de Amazon SNS o invocar una función Lambda.

Cuando cree una regla, debe tener en cuenta la cantidad de datos que publica en los temas. Si crea reglas que contienen un patrón de tema con comodín, es posible que coincidan con una gran cantidad de mensajes y que necesite aumentar la capacidad de los recursos de AWS que utilizan las acciones de destino. Asimismo, si crea una regla para volver a publicar que contenga un patrón de tema con comodín, puede acabar teniendo una regla circular que genere un bucle infinito.

**Note**

La creación y la actualización de reglas son acciones de nivel de administrador. Todo usuario que tenga permiso para crear o actualizar reglas podrá tener acceso a los datos procesados por las reglas.

**Para crear una regla (AWS CLI)**

Ejecute el comando `create-topic-rule` para crear una regla:

```
aws iot create-topic-rule --rule-name my-rule --topic-rule-payload file://my-rule.json
```

Ejemplo de archivo de carga con una regla que inserta todos los mensajes enviados al tema `iot/test` en la tabla DynamoDB. La instrucción SQL filtra los mensajes, y el ARN del rol concede a AWS IoT permiso para escribir en la tabla DynamoDB.

```
{  
    "sql": "SELECT * FROM 'iot/test'",  
    "ruleDisabled": false,  
    "awsIotSqlVersion": "2016-03-23",  
    "actions": [  
        {  
            "dynamoDB": {  
                "tableName": "my-dynamodb-table",  
                "roleArn": "arn:aws:iam::123456789012:role/my-iot-role",  
                "hashKeyField": "topic",  
                "hashKeyValue": "${topic(2)}",  
                "rangeKeyField": "timestamp",  
                "rangeKeyValue": "${timestamp()}"  
            }  
        }  
    ]  
}
```

A continuación, se muestra un ejemplo de archivo de carga con una regla que inserta todos los mensajes enviados al tema `iot/test` en el bucket de S3 especificado. La instrucción SQL filtra los mensajes, y el ARN del rol concede a AWS IoT permiso para escribir en el bucket de Amazon S3.

```
{  
    "awsIotSqlVersion": "2016-03-23",  
    "sql": "SELECT * FROM 'iot/test'",  
    "ruleDisabled": false,  
    "actions": [  
        {  
            "s3": {  
                "roleArn": "arn:aws:iam::123456789012:role/aws_iot_s3",  
                "bucketName": "my-bucket",  
                "key": "myS3Key"  
            }  
        }  
    ]  
}
```

Ejemplo de archivo de carga con una regla que inserta datos en Amazon ES:

```
{  
    "sql": "SELECT *, timestamp() as timestamp FROM 'iot/test'",  
    "ruleDisabled": false,  
    "awsIotSqlVersion": "2016-03-23",  
    "actions": [  
        {  
            "elasticsearch": {  
                "roleArn": "arn:aws:iam::123456789012:role/aws_iot_es",  
                "endpoint": "https://my-endpoint",  
                "index": "my-index",  
                "type": "my-type",  
                "id": "${newuuid()}"  
            }  
        }  
    ]  
}
```

Ejemplo de archivo de carga con una regla que invoca una función Lambda:

```
{
```

```
{  
    "sql": "expression",  
    "ruleDisabled": false,  
    "awsIotSqlVersion": "2016-03-23",  
    "actions": [  
        {"  
            "lambda": {  
                "functionArn": "arn:aws:lambda:us-west-2:123456789012:function:my-lambda-function"  
            }  
        }  
    ]  
}
```

Ejemplo de archivo de carga con una regla que publica en un tema de Amazon SNS:

```
{  
    "sql": "expression",  
    "ruleDisabled": false,  
    "awsIotSqlVersion": "2016-03-23",  
    "actions": [  
        {"  
            "sns": {  
                "targetArn": "arn:aws:sns:us-west-2:123456789012:my-sns-topic",  
                "roleArn": "arn:aws:iam::123456789012:role/my-iot-role"  
            }  
        }  
    ]  
}
```

Ejemplo de archivo de carga con una regla que vuelve a publicar en otro tema MQTT:

```
{  
    "sql": "expression",  
    "ruleDisabled": false,  
    "awsIotSqlVersion": "2016-03-23",  
    "actions": [  
        {"  
            "republish": {  
                "topic": "my-mqtt-topic",  
                "roleArn": "arn:aws:iam::123456789012:role/my-iot-role"  
            }  
        }  
    ]  
}
```

Ejemplo de archivo de carga con una regla que inserta datos en un flujo de Amazon Kinesis Firehose:

```
{  
    "sql": "SELECT * FROM 'my-topic'",  
    "ruleDisabled": false,  
    "awsIotSqlVersion": "2016-03-23",  
    "actions": [  
        {"  
            "firehose": {  
                "roleArn": "arn:aws:iam::123456789012:role/my-iot-role",  
                "deliveryStreamName": "my-stream-name"  
            }  
        }  
    ]  
}
```

Ejemplo de archivo de carga con una regla que utiliza la función `machinelearning_predict` de Amazon Machine Learning para volver a publicar en un tema, siempre y cuando los datos de la carga de MQTT se clasifiquen como 1.

```
{  
    "sql": "SELECT * FROM 'iot/test' where machinelearning_predict('my-model',  
    'arn:aws:iam::123456789012:role/my-iot-aml-role', *).predictedLabel=1",  
    "ruleDisabled": false,  
    "awsIotSqlVersion": "2016-03-23",  
    "actions": [  
        {"  
            "sns": {  
                "targetArn": "arn:aws:sns:us-west-2:123456789012:my-sns-topic",  
                "roleArn": "arn:aws:iam::123456789012:role/my-iot-role"  
            }  
        }  
    ]  
}
```

```
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [
        {
            "republish": {
                "roleArn": "arn:aws:iam::123456789012:role/my-iot-role",
                "topic": "my-mqtt-topic"
            }
        }
    ]
}
```

A continuación se incluye un archivo de carga de ejemplo con una regla que publica mensajes en un flujo de entrada de Salesforce IoT Cloud.

```
{
    "sql": "expression",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [
        {
            "salesforce": {
                "token": "ABCDEFGHI123456789abcdefghi123456789",
                "url": "https://ingestion-cluster-id.my-env.sfdcnow.com/streams/stream-id/connection-id/my-event"
            }
        }
    ]
}
```

## Visualización de las reglas

Ejecute el comando [list-topic-rules](#) para visualizar una lista de sus reglas:

```
aws iot list-topic-rules
```

Ejecute el comando [get-topic-rule](#) para obtener información acerca de una regla:

```
aws iot get-topic-rule --rule-name my-rule
```

## Versiones de SQL

El motor de reglas de AWS IoT utiliza una sintaxis similar a SQL para seleccionar los datos de los mensajes MQTT. Las instrucciones SQL se interpretan según la versión de SQL especificada en la propiedad `awsIotSqlVersion` de un documento JSON que describe la regla. Para obtener más información acerca de la estructura de los documentos de reglas JSON, consulte [Creación de una regla \(p. 151\)](#). La propiedad `awsIotSqlVersion` le permite especificar la versión del motor de reglas AWS IoT SQL que desea utilizar. Cuando se implementa una nueva versión, puede continuar utilizando una versión anterior o cambiar la regla para utilizar la nueva versión. Las reglas actuales seguirán utilizando la versión con la que se crearon.

En el siguiente ejemplo de JSON se muestra cómo especificar la versión de SQL mediante la propiedad `awsIotSqlVersion`:

```
{
    "sql": "expression",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
```

```
    "actions": [ {
        "republish": {
            "topic": "my-mqtt-topic",
            "roleArn": "arn:aws:iam::123456789012:role/my-iot-role"
        }
    }]
}
```

Estas son las versiones compatibles actualmente:

- 2015-10-08: la versión de SQL original creada el 08/10/2015.
- 2016-03-23: la versión de SQL creada el 23/03/2016.
- beta: la versión beta de SQL más reciente. El uso de esta versión podría introducir cambios bruscos en sus reglas.

## Novedades de la versión del motor de reglas SQL del 23/03/2016

- Soluciones para seleccionar objetos JSON anidados.
- Soluciones para consultas de matriz.
- Compatibilidad con consultas entre objetos.
- Compatibilidad con la generación de una matriz como objeto de nivel superior.
- Se ha agregado la función de codificación (valor, encodingScheme), que se puede aplicar en datos con o sin formato JSON.

## Consultas entre objetos

Esta característica le permite consultar un atributo en un objeto JSON. Por ejemplo, si se recibe el mensaje MQTT siguiente:

```
{
    "e": [
        { "n": "temperature", "u": "Cel", "t": 1234, "v": 22.5 },
        { "n": "light", "u": "lm", "t": 1235, "v": 135 },
        { "n": "acidity", "u": "pH", "t": 1235, "v": 7 }
    ]
}
```

Y la regla siguiente:

```
SELECT (SELECT v FROM e WHERE n = 'temperature') as temperature FROM 'my/topic'
```

La regla generará la salida siguiente:

```
{"temperature": [{"v": 22.5}]} 
```

Si se utiliza el mismo mensaje MQTT y se aplica una regla algo más compleja, como la siguiente:

```
SELECT get((SELECT v FROM e WHERE n = 'temperature'),1).v as temperature FROM 'topic'
```

La regla generará la salida siguiente:

```
{"temperature":22.5}
```

## Generación de Array como objeto de nivel superior

Esta característica permite que una regla devuelva una matriz como objeto de nivel superior. Por ejemplo, si se recibe el mensaje MQTT siguiente:

```
{
  "a": {"b":"c"},
  "arr":[1,2,3,4]
}
```

Y la regla siguiente:

```
SELECT VALUE arr FROM 'topic'
```

La regla generará la salida siguiente:

```
[1,2,3,4]
```

## Función de codificación

Codifica la carga, que potencialmente puede estar constituida por datos sin formato JSON, en su representación de cadena (según el esquema de codificación especificado).

## Solución de problemas de una regla

Si tiene algún problema con las reglas, debe habilitar CloudWatch Logs. Si analiza los logs, puede saber si el problema es la autorización o si, por ejemplo, la condición de cláusula WHERE no ha coincidido. Para obtener más información acerca de cómo utilizar Amazon CloudWatch Logs, consulte [Setting Up CloudWatchLogs](#).

## Eliminación de una regla

Cuando ya no necesite una regla, puede eliminarla.

Para eliminar una regla (AWS CLI)

Ejecute el comando `delete-topic-rule` para eliminar una regla:

```
aws iot delete-topic-rule --rule-name my-rule
```

## Acciones de las reglas de AWS IoT

Las acciones de las reglas de AWS IoT se utilizan para especificar qué hacer cuando se activa una regla. Puede definir acciones para escribir datos en una base de datos DynamoDB o un flujo de Kinesis, o invocar una función Lambda y mucho más. Se admiten las siguientes acciones:

- `cloudwatchAlarm` para cambiar una alarma de CloudWatch.
- `cloudwatchMetric` para capturar una métrica de CloudWatch.
- `dynamoDB` para escribir datos en una base de datos DynamoDB.
- `dynamoDBv2` para escribir datos en una base de datos DynamoDB.
- `elasticsearch` para escribir datos en un dominio de Amazon Elasticsearch Service.
- `firehose` para escribir datos en un flujo de Amazon Kinesis Firehose.
- `kinesis` para escribir datos en un flujo de Kinesis.
- `lambda` para invocar una función Lambda.
- `s3` para escribir datos en un bucket de Amazon S3.
- `sns` para escribir datos como notificación de inserción.
- `sqs` para escribir datos en una cola de SQS.
- `republish` para volver a publicar el mensaje en otro tema MQTT.
- `salesforce` para escribir un mensaje en un flujo de entrada de Salesforce IoT

#### Note

Actualmente, el motor de reglas de AWS IoT no intenta volver a entregar los mensajes que no se hayan podido publicar en otro servicio.

En las secciones siguientes se explica cada acción detalladamente.

## Acción de alarma de CloudWatch

La acción de alarma de CloudWatch permite cambiar el estado de la alarma de CloudWatch. Puede especificar el motivo del cambio de estado y el valor de esta llamada. Cuando cree una regla de AWS IoT con una acción de alarma de CloudWatch, debe especificar la información siguiente:

`roleArn`

El rol de IAM que permite el acceso a la alarma de CloudWatch.

`alarmName`

El nombre de la alarma de CloudWatch.

`stateReason`

El motivo del cambio de alarma.

`stateValue`

El valor del estado de alarma. Los valores aceptables son `OK`, `ALARM` e `INSUFFICIENT_DATA`.

#### Note

Asegúrese de que el rol asociado a la regla tenga una política que conceda el permiso `cloudwatch:SetAlarmState`.

El siguiente ejemplo JSON muestra cómo definir una acción de alarma de CloudWatch en una regla de AWS IoT:

```
{  
  "rule": {  
    "sql": "SELECT * FROM 'some/topic'",  
    "state": {  
      "value": "ALARM",  
      "reason": "Temperature threshold exceeded"  
    }  
  }  
}
```

```
    "ruleDisabled": false,
    "actions": [
        "cloudwatchAlarm": {
            "roleArn": "arn:aws:iam::123456789012:role/aws_iot_cw",
            "alarmName": "IotAlarm",
            "stateReason": "Temperature stabilized.",
            "stateValue": "OK"
        }
    ]
}
```

Para obtener más información, consulte [CloudWatch Alarms](#).

## Acción de métrica de CloudWatch

La acción de métrica de CloudWatch le permite capturar una métrica de CloudWatch. Puede especificar el espacio de nombres, el nombre, el valor, la unidad y la marca de tiempo de la métrica. Cuando cree una regla de AWS IoT con una acción de métrica de CloudWatch, debe especificar la información siguiente:

**roleArn**

El rol de IAM que permite el acceso a la métrica de CloudWatch.

**metricNamespace**

El nombre del espacio de nombres de la métrica de CloudWatch.

**metricName**

El nombre de la métrica de CloudWatch.

**metricValue**

El valor de la métrica de CloudWatch.

**metricUnit**

La unidad métrica que CloudWatch admite.

**metricTimestamp**

Una marca de tiempo Unix opcional.

### Note

Asegúrese de que el rol asociado a la regla tenga una política que conceda el permiso `cloudwatch:PutMetricData`.

El siguiente ejemplo de JSON muestra cómo definir una acción de métrica de CloudWatch en una regla de AWS IoT:

```
{
    "rule": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "actions": [
            "cloudwatchMetric": {
                "roleArn": "arn:aws:iam::123456789012:role/aws_iot_cw",
                "metricNamespace": "IotNamespace",
                "metricName": "IotMetric",
                "metricValue": "1",
                "metricUnit": "Count"
            }
        ]
    }
}
```

```
        "metricUnit": "Count",
        "metricTimestamp": "1456821314"
    }
}
}
```

Para obtener más información, consulte [CloudWatch Metrics](#).

## Acción DynamoDB

La acción `dynamoDB` le permite escribir todo un mensaje MQTT o parte de él en una tabla de DynamoDB. Cuando cree una regla de DynamoDB, debe especificar la información siguiente:

`hashKeyType`

El tipo de datos de la clave hash (también denominado clave de partición). Los valores válidos son: `"STRING"` o `"NUMBER"`.

`hashKeyField`

El nombre de la clave hash (también denominada clave de partición).

`hashKeyValue`

El valor de la clave hash.

`rangeKeyType`

Opcional. El tipo de datos de la clave de rango (también denominada clave de ordenación). Los valores válidos son: `"STRING"` o `"NUMBER"`.

`rangeKeyField`

Opcional. El nombre de la clave de rango (también denominada clave de ordenación).

`rangeKeyValue`

Opcional. El valor de la clave de rango.

`operación`

Opcional. El tipo de operación que se va a realizar. Sigue la plantilla de sustitución, por lo que puede ser `#{operation}`, pero la sustitución debe tener como resultado uno de los elementos siguientes: `INSERT`, `UPDATE` o `DELETE`.

`payloadField`

Opcional. El nombre del campo donde se escribirá la carga. Si este valor no se especifica, la carga se escribe en el campo `payload`.

`tabla`

Nombre de la tabla de DynamoDB.

`roleARN`

El rol de IAM que permite tener acceso a la tabla de DynamoDB. Como mínimo, el rol debe permitir la acción de IAM `dynamoDB:PutItem`.

Los datos que se escriben en la tabla de DynamoDB son el resultado de la instrucción SQL de la regla.

Los campos `hashKeyValue` y `rangeKeyValue` se componen normalmente de expresiones (por ejemplo, `#{topic()}` o `#{timestamp()}`).

#### Note

Los datos que no son JSON se escriben en DynamoDB como datos binarios. La consola de DynamoDB mostrará los datos como texto codificado en Base64.

Asegúrese de que el rol asociado a la regla tenga una política que conceda el permiso dynamodb:PutItem.

El siguiente ejemplo de JSON muestra cómo definir una acción dynamoDB en una regla de AWS IoT:

```
{  
    "rule": {  
        "ruleDisabled": false,  
        "sql": "SELECT * AS message FROM 'some/topic'",  
        "description": "A test Dynamo DB rule",  
        "actions": [{  
            "dynamoDB": {  
                "hashKeyField": "key",  
                "roleArn": "arn:aws:iam::123456789012:role/aws_iot_dynamoDB",  
                "tableName": "my_ddb_table",  
                "hashKeyValue": "${topic()}",  
                "rangeKeyValue": "${timestamp()}",  
                "rangeKeyField": "timestamp"  
            }  
        }]  
    }  
}
```

Para obtener más información, consulte [Amazon DynamoDB Getting Started Guide](#).

## Acción DynamoDBv2

La acción dynamoDBv2 le permite escribir todo un mensaje MQTT o parte de él en una tabla de DynamoDB. Cada atributo de la carga se escribe en una columna independiente de la base de datos DynamoDB. Cuando cree una regla de DynamoDB, debe especificar la información siguiente:

#### roleARN

El rol de IAM que permite tener acceso a la tabla de DynamoDB. Como mínimo, el rol debe permitir la acción de IAM dynamoDB:PutItem.

#### tableName

Nombre de la tabla de DynamoDB.

#### Note

La carga del mensaje MQTT debe contener una clave de nivel de raíz que coincida con la clave de partición principal de la tabla y una clave de nivel de raíz que coincida con la clave de ordenación principal de la tabla, si hay una definida.

Los datos que se escriben en la tabla de DynamoDB son el resultado de la instrucción SQL de la regla.

#### Note

Asegúrese de que el rol asociado a la regla tenga una política que conceda el permiso dynamodb:PutItem.

El siguiente ejemplo de JSON muestra cómo definir una acción dynamoDB en una regla de AWS IoT:

```
{  
    "rule": {  
        "ruleDisabled": false,  
        "sql": "SELECT * AS message FROM 'some/topic'",  
        "description": "A test DynamoDBv2 rule",  
        "actions": [  
            {  
                "dynamoDBv2": {  
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_dynamoDBv2",  
                    "putItem": {  
                        "tableName": "my_ddb_table"  
                    }  
                }  
            }  
        ]  
    }  
}
```

Para obtener más información, consulte [Amazon DynamoDB Getting Started Guide](#).

## Acción Amazon ES

La acción `elasticsearch` le permite escribir datos de mensajes MQTT en un dominio de Amazon Elasticsearch Service. De ser así, los datos contenidos en Amazon ES se pueden consultar y visualizar con herramientas como Kibana. Cuando cree una regla de AWS IoT con una acción `elasticsearch`, debe especificar la información siguiente:

`endpoint`

Punto de enlace de su dominio de Amazon ES.

`índice`

Índice de Amazon ES donde desee almacenar los datos.

`type`

Tipo de documento que está almacenando.

`id`

Identificador único de cada documento.

### Note

Asegúrese de que el rol asociado a la regla tenga una política que conceda el permiso `es:ESHttpPut`.

El siguiente ejemplo de JSON muestra cómo definir una acción `elasticsearch` en una regla de AWS IoT:

```
{  
    "rule":{  
        "sql":"SELECT *, timestamp() as timestamp FROM 'iot/test'",  
        "ruleDisabled":false,  
        "actions": [  
            {  
                "elasticsearch":{  
                    "roleArn":"arn:aws:iam::123456789012:role/aws_iot_es",  
                    "endpoint":"https://my-endpoint",  
                    "index":"my-index",  
                    "type":"my-type",  
                }  
            }  
        ]  
    }  
}
```

```
        "id": "${newuuid()}"  
    }  
}  
]  
}
```

Para obtener más información, consulte [Amazon ES Developer Guide](#).

## Acción Firehose

Una acción `firehose` envía datos de un mensaje MQTT que activó la regla a un flujo de Kinesis Firehose. Cuando cree una regla con una acción `firehose`, debe especificar la información siguiente:

`deliveryStreamName`

El flujo de Kinesis Firehose en el que deben escribirse los datos del mensaje.

`roleArn`

El rol de IAM que permite obtener acceso a Kinesis Firehose.

`separator`

Un separador de caracteres que se utilizará para separar registros escritos en el flujo de firehose. Los valores válidos son: '\n' (línea nueva), '\t' (pestaña), '\r\n' (línea nueva de Windows),',' (coma).

### Note

Asegúrese de que el rol asociado a la regla tenga una política que conceda el permiso `firehose:PutRecord`.

El ejemplo de JSON siguiente muestra cómo crear una regla de AWS IoT con una acción `firehose`:

```
{  
    "rule": {  
        "sql": "SELECT * FROM 'some/topic'",  
        "ruleDisabled": false,  
        "actions": [  
            {  
                "firehose": {  
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_firehose",  
                    "deliveryStreamName": "my_firehose_stream"  
                }  
            }  
        ]  
    }  
}
```

Para obtener más información, consulte [Kinesis Firehose Developer Guide](#).

## Acción Kinesis

La acción `kinesis` le permite escribir los datos de mensajes MQTT en un flujo de Kinesis. Cuando cree una regla de AWS IoT con una acción `kinesis`, debe especificar la información siguiente:

`stream`

El flujo de Kinesis en el que se escriben los datos.

#### partitionKey

La clave de partición utilizada para determinar en qué fragmento se escriben los datos. La clave de partición se compone normalmente de una expresión (por ejemplo, "\${topic()}" o "\${timestamp()}").

#### Note

Asegúrese de que la política asociada a la regla tenga el permiso `kinesis:PutRecord`.

El siguiente ejemplo de JSON muestra cómo definir una acción `kinesis` en una regla de AWS IoT:

```
{  
    "rule": {  
        "sql": "SELECT * FROM 'some/topic'",  
        "ruleDisabled": false,  
        "actions": [{  
            "kinesis": {  
                "roleArn": "arn:aws:iam::123456789012:role/aws_iot_kinesis",  
                "streamName": "my_kinesis_stream",  
                "partitionKey": "${topic()}"  
            }  
        }]  
    }  
}
```

Para obtener más información, consulte [Kinesis Developer Guide](#).

## Acción Lambda

Una acción `lambda` llama a una función Lambda y transmite el mensaje MQTT que ha activado la regla. Para que AWS IoT llame a la función Lambda, debe configurar una política que conceda el permiso `lambda:InvokeFunction` a AWS IoT. Las funciones Lambda utilizan políticas basadas en recursos, por lo que debe asociar la política a la función Lambda en sí. Ejecute el siguiente comando de CLI para asociar una política que conceda el permiso `lambda:InvokeFunction`:

```
aws lambda add-permission --function-name "function_name" --region "region" --principal iot.amazonaws.com --source-arn arn:aws:iot:us-east-2:account_id:rule/rule_name --source-account "account_id" --statement-id "unique_id" --action "lambda:InvokeFunction"
```

A continuación, se indican los argumentos del comando `add-permission`:

`--function-name`

Nombre de la función Lambda cuya política de recursos está actualizando mediante la adición de un nuevo permiso.

`--region`

La región de AWS de su cuenta.

`--principal`

El principal que obtiene el permiso. Debe ser `iot.amazonaws.com` para que AWS IoT pueda llamar a una función Lambda.

`--source-arn`

El ARN de la regla. Puede utilizar el comando de CLI `get-topic-rule` para obtener el ARN de una regla.

--source-account

La cuenta de AWS donde está definida la regla.

--statement-id

Un identificador de instrucción único.

--action

La acción Lambda que desea permitir en esta instrucción. En este caso, queremos permitir a AWS IoT que invoque una función Lambda, por lo que especificamos `lambda:InvokeFunction`.

#### Note

Si agrega un permiso para un principal de AWS IoT sin proporcionar el ARN de origen, cualquier cuenta de AWS que cree una regla con su acción Lambda puede activar reglas para invocar su función Lambda desde AWS IoT.

Para obtener más información, consulte [Lambda Permission Model](#).

Cuando cree una regla con una acción `lambda`, debe especificar la función Lambda para invocar cuándo se activa la regla.

El ejemplo de JSON siguiente muestra una regla que llama a una función Lambda:

```
{  
  "rule": {  
    "sql": "SELECT * FROM 'some/topic'",  
    "ruleDisabled": false,  
    "actions": [{  
      "lambda": {  
        "functionArn": "arn:aws:lambda:us-  
east-2:123456789012:function:myLambdaFunction"  
      }  
    }]  
  }  
}
```

Para obtener más información, consulte [AWS Lambda Developer Guide](#).

## Acción Republish

La acción `republish` le permite volver a publicar el mensaje que activó el rol, en otro tema MQTT. Cuando cree una regla con una acción `republish`, debe especificar la información siguiente:

tema

El tema MQTT en el que se va a volver a publicar el mensaje.

roleArn

El rol de IAM que permite publicar en el tema MQTT.

#### Note

Asegúrese de que el rol asociado a la regla tenga una política que conceda el permiso `iot:Publish`.

```
{  
    "rule": {  
        "sql": "SELECT * FROM 'some/topic'",  
        "ruleDisabled": false,  
        "actions": [  
            {  
                "republish": {  
                    "topic": "another/topic",  
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_republish"  
                }  
            }  
        ]  
    }  
}
```

## Acción S3

Una acción s3 escribe los datos del mensaje MQTT que activó la regla en un bucket de Amazon S3. Cuando cree una regla de AWS IoT con una acción s3, debe especificar la información siguiente:

bucket

El bucket de Amazon S3 en el que se escribirán los datos.

cannedacl

La lista de control de acceso (ACL) predefinida de Amazon S3 que controla el acceso al objeto identificado mediante la clave de objeto. Para obtener más información, consulte [S3 Canned ACLs](#).

clave

La ruta al archivo en el que se escriben los datos. Por ejemplo, si el valor de este argumento es "\${topic()}/\${timestamp()}", el tema al que se envía el mensaje es "this/is/my/topic" y la marca de tiempo actual es 1460685389. Los datos se escribirán en un archivo llamado "1460685389" en la carpeta "this/is/my/topic" en Amazon S3.

Note

El uso de una clave estática hará que se sobrescriba únicamente un archivo en Amazon S3 por cada invocación de la regla. Lo más habitual es utilizar la marca de tiempo del mensaje u otro identificador de mensaje único, de modo que por cada mensaje recibido se guarde un archivo nuevo en Amazon S3.

roleArn

El rol de IAM que permite tener acceso al bucket de Amazon S3.

Note

Asegúrese de que el rol asociado a la regla tenga una política que conceda el permiso s3:PutObject.

El siguiente ejemplo de JSON muestra cómo definir una acción s3 en una regla de AWS IoT:

```
{  
    "rule": {  
        "sql": "SELECT * FROM 'some/topic'",  
        "ruleDisabled": false,  
        "actions": [  
            {  
                "s3": {  
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_s3",  
                    "bucket": "my-bucket",  
                    "key": "data/  
                }  
            }  
        ]  
    }  
}
```

```
        "bucketName": "my-bucket",
        "key": "${topic()}/${timestamp()}"
    }
}
}
```

Para obtener más información, consulte [Amazon S3 Developer Guide](#).

## Acción SNS

Una acción sns envía los datos del mensaje MQTT que activó la regla como notificación de inserción de SNS. Cuando cree una regla con una acción sns, debe especificar la información siguiente:

**messageFormat**

El formato del mensaje. Los valores aceptados son "JSON" y "RAW". El valor predeterminado del atributo es "RAW". SNS utiliza esta configuración para determinar si la carga debe analizarse y las partes pertinentes de la carga específicas de la plataforma deben extraerse.

**roleArn**

El rol de IAM que permite obtener acceso a SNS.

**targetArn**

El tema de SNS o el dispositivo individual al que se enviará la notificación de inserción.

**Note**

Asegúrese de que la política asociada a la regla tenga el permiso sns:Publish.

El siguiente ejemplo de JSON muestra cómo definir una acción sns en una regla de AWS IoT:

```
{
  "rule": {
    "sql": "SELECT * FROM 'some/topic'",
    "ruleDisabled": false,
    "actions": [
      {
        "sns": {
          "targetArn": "arn:aws:sns:us-east-2:123456789012:my_sns_topic",
          "roleArn": "arn:aws:iam::123456789012:role/aws_iot_sns"
        }
      }
    ]
  }
}
```

Para obtener más información, consulte [Amazon SNS Developer Guide](#).

## Acción SQS

Una acción sqs envía datos del mensaje MQTT que activó la regla a una cola SQS. Cuando cree una regla con una acción sqs, debe especificar la información siguiente:

**queueUrl**

La dirección URL de la cola SQS en la que se escriben los datos.

#### useBase64

Establezca esta opción en `true` si quiere que los datos de mensajes MQTT se codifiquen en Base64 antes de escribir en la cola SQS; de lo contrario, establezcala en `false`.

#### roleArn

El rol de IAM que permite tener acceso a la cola SQS.

#### Note

Asegúrese de que el rol asociado a la regla tenga una política que conceda el permiso `sqs:SendMessage`.

El ejemplo de JSON siguiente muestra cómo crear una regla de AWS IoT con una acción `sqs`:

```
{  
    "rule": {  
        "sql": "SELECT * FROM 'some/topic'",  
        "ruleDisabled": false,  
        "actions": [{  
            "sqs": {  
                "queueUrl": "https://sqs.us-east-2.amazonaws.com/123456789012/  
my_sqs_queue",  
                "roleArn": "arn:aws:iam::123456789012:role/aws_iot_sqs",  
                "useBase64": false  
            }  
        }]  
    }  
}
```

Para obtener más información, consulte [Amazon SQS Developer Guide](#).

## Acción Salesforce

Una acción `salesforce` envía los datos del mensaje de MQTT que activó la regla a un flujo de entrada de Salesforce IoT. Cuando cree una regla con una acción `salesforce`, debe especificar la información siguiente:

#### url

La dirección URL expuesta por el flujo de entrada de Salesforce IoT. La dirección URL está disponible en la plataforma de Salesforce IoT cuando crea un flujo de entrada. Consulte la documentación de Salesforce IoT para obtener más información.

#### token

El token que se utiliza para autenticar el acceso al flujo de entrada de Salesforce IoT especificado. El token está disponible en la plataforma de Salesforce IoT cuando crea un flujo de entrada. Consulte la documentación de Salesforce IoT para obtener más información.

#### Note

Estos parámetros no pueden sustituirse.

El ejemplo de JSON siguiente muestra cómo crear una regla de AWS IoT con una acción `salesforce`:

```
{
```

```
    "sql": "expression",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [
        "salesforce": {
            "token": "ABCDEFGHI123456789abcdefghi123456789",
            "url": "https://ingestion-cluster-id.my-env.sfdcnow.com/streams/stream-id/connection-id/my-event"
        }
    ]
}
```

Para obtener más información, consulte la documentación de Salesforce IoT

## Referencias de SQL en AWS IoT

En AWS IoT, las reglas se definen utilizando una sintaxis similar a SQL. Las instrucciones SQL se componen de tres tipos de cláusulas:

SELECT

Obligatorio. Extrae información de la carga de entrada y realiza las transformaciones.

FROM

Obligatorio. El filtro de tema MQTT desde el que la regla recibirá mensajes.

WHERE

Opcional. Agrega lógica condicional que determina si una regla se evalúa y si se ejecutan sus acciones.

Un ejemplo de instrucción SQL tiene este aspecto:

```
SELECT color AS rgb FROM 'a/b' WHERE temperature > 50
```

Un mensaje MQTT de ejemplo (también denominado carga de entrada) tiene este aspecto:

```
{
    "color": "red",
    "temperature": 100
}
```

Si este mensaje se publica en el tema 'a/b', la regla se activa y se evalúa la instrucción SQL. La instrucción SQL extrae el valor de la propiedad `rgb` si la propiedad "`temperature`" es superior a 50. La cláusula WHERE especifica la condición `temperature > 50`. La palabra clave AS cambia el nombre de la propiedad "color" a "rgb". El resultado (también denominado carga de salida) tiene este aspecto:

```
{
    "rgb": "red"
}
```

Estos datos se reenvían después a la acción de la regla, que envía los datos para seguirlos procesando. Para obtener más información sobre las acciones de las reglas, consulte [Acciones de las reglas de AWS IoT \(p. 156\)](#).

## Tipos de datos

El motor de reglas de AWS IoT admite todos los tipos de datos JSON.

### Tipos de datos admitidos

Tipo	Significado
<code>Int</code>	Un valor <code>Int</code> discreto de 34 dígitos como máximo.
<code>Decimal</code>	Un valor <code>Decimal</code> con una precisión de 34 dígitos, con una magnitud no nula mínima de 1E-999 y una magnitud máxima de 9.999... E999. <b>Note</b> Algunas funciones devuelven valores <code>Decimal</code> con doble precisión (en lugar de una precisión de 34 dígitos).
<code>Boolean</code>	<code>True</code> o bien <code>False</code> .
<code>String</code>	Una cadena UTF-8.
<code>Array</code>	Una serie de valores que no han de tener obligatoriamente el mismo tipo.
<code>Object</code>	Un valor JSON compuesto de una clave y un valor. Las claves deben ser cadenas. Los valores pueden ser de cualquier tipo.
<code>Null</code>	Valor <code>Null</code> , tal como lo define JSON. Es un valor real que representa la ausencia de valor. El usuario puede crear explícitamente un valor <code>Null</code> especificando la palabra clave <code>Null</code> en la instrucción SQL. Por ejemplo: "SELECT <code>NULL</code> AS <code>n</code> FROM 'a/b'"
<code>Undefined</code>	No es un valor. No se representa explícitamente en JSON, salvo que se omita el valor. Por ejemplo, en el objeto <code>{"foo": null}</code> , la clave "foo" devuelve <code>NULL</code> , pero la clave "bar" devuelve <code>Undefined</code> . Internamente, el lenguaje SQL trata a <code>Undefined</code> como un valor, pero este no se puede representar en JSON, por lo que cuando se serializa en JSON, los resultados son <code>Undefined</code> . <pre>{"foo":null, "bar":undefined}</pre> se serializa en JSON como: <pre>{"foo":null}</pre> Del mismo modo, <code>Undefined</code> se convierte en una cadena vacía cuando se serializa por sí mismo. Las funciones a las que se llama con argumentos no válidos (por ejemplo, tipos erróneos, número de argumentos erróneo, etc.) devolverán <code>Undefined</code> .

## Conversiones

En la tabla siguiente se muestra una lista de los resultados que se producen cuando un valor de un tipo se convierte en otro tipo (cuando se da un valor de tipo incorrecto a una función). Por ejemplo, si a la función de valor absoluto "abs" (que espera un valor Int o Decimal) se le da un valor String, esta función intentará convertir el valor String en un valor Decimal, de acuerdo con estas reglas. En este caso, 'abs ("-5.123")' se trata como 'abs(-5.123)'.

### Note

No hay ningún intento de conversión en `Array`, `Object`, `Null` o `Undefined`.

### En valor decimal

Tipo de argumento	Resultado
<code>Int</code>	Valor <code>Decimal</code> sin separador decimal.
<code>Decimal</code>	El valor de origen.
<code>Boolean</code>	<code>Undefined</code> . (Puede utilizar de forma explícita la función cast para transformar <code>true</code> = <code>1.0</code> , <code>false</code> = <code>0.0</code> ).
<code>String</code>	El motor SQL intentará analizar la cadena como <code>Decimal</code> . Intentaremos analizar las cadenas que coincidan con la expresión regular: <code>^-? \d+ (\.\d+)?((?i)E-?\d+)?\$. "0", "-1.2", "5E-12"</code> son ejemplos de cadenas que se convertirán automáticamente en valores de tipo <code>Decimal</code> .
<code>Matriz</code>	<code>Undefined</code> .
<code>Objeto</code>	<code>Undefined</code> .
<code>Null</code>	<code>Null</code> .
<code>Sin definir</code>	<code>Undefined</code> .

### En valor entero

Tipo de argumento	Resultado
<code>Int</code>	El valor de origen.
<code>Decimal</code>	El valor de origen redondeado al valor <code>Int</code> más cercano.
<code>Boolean</code>	<code>Undefined</code> . (Puede utilizar de forma explícita la función cast para transformar <code>true</code> = <code>1.0</code> , <code>false</code> = <code>0.0</code> ).
<code>String</code>	El motor SQL intentará analizar la cadena como <code>Decimal</code> . Intentaremos analizar las cadenas que coincidan con la expresión regular: <code>^-? \d+ (\.\d+)?((?i)E-?\d+)?\$. "0", "-1.2", "5E-12"</code> son ejemplos de cadenas que se convertirán automáticamente en valores <code>Decimal</code> . Intentaremos convertir <code>String</code> en un valor <code>Decimal</code> y, a continuación, eliminar los números

Tipo de argumento	Resultado
	posteriores a la coma del valor <code>Decimal</code> para obtener un valor <code>Int</code> .
Matriz	<code>Undefined</code> .
Objeto	<code>Undefined</code> .
Null	<code>Null</code> .
Sin definir	<code>Undefined</code> .

#### En valor booleano

Tipo de argumento	Resultado
<code>Int</code>	<code>Undefined</code> . (Puede utilizar explícitamente la función <code>cast</code> para transformar <code>0 = False</code> , <code>any_nonzero_value = True</code> ).
<code>Decimal</code>	<code>Undefined</code> . (Puede utilizar explícitamente la función de conversión para transformar <code>0 = False</code> , <code>any_nonzero_value = True</code> ).
<code>Boolean</code>	El valor original.
<code>String</code>	"true" = <code>True</code> y "false" = <code>False</code> (no distingue entre mayúsculas y minúsculas). Otros valores de cadena serán <code>Undefined</code> .
Matriz	<code>Undefined</code> .
Objeto	<code>Undefined</code> .
Null	<code>Undefined</code> .
Sin definir	<code>Undefined</code> .

#### En cadena

Tipo de argumento	Resultado
<code>Int</code>	Una representación de cadena del valor <code>Int</code> en notación estándar.
<code>Decimal</code>	Una cadena que representa el valor <code>Decimal</code> , posiblemente en notación científica.
<code>Boolean</code>	"true" o "false". Todos en minúscula.
<code>String</code>	El valor original.
Matriz	El valor <code>Array</code> serializado en formato JSON. La cadena obtenida será una lista separada por comas, entre corchetes. Los valores <code>String</code> se indicarán entre comillas. Los valores <code>Decimal</code> , <code>Int</code> , <code>Boolean</code> y <code>Null</code> no se indicarán entre comillas.

Tipo de argumento	Resultado
Objeto	El objeto serializado al formato JSON. La cadena obtenida será una lista separada por comas de pares clave-valor, y comenzará y terminará con llaves. Los valores String se indicarán entre comillas. Los valores Decimal, Int, Boolean y Null no se indicarán entre comillas.
Null	Undefined.
Sin definir	Sin definir.

## Operadores

Los operadores siguientes se pueden utilizar en las cláusulas SELECT, FROM y WHERE.

### Operador AND

Devuelve un resultado Boolean. Realiza una operación AND lógica. Devuelve el valor true si los operandos izquierdo y derecho son true; de lo contrario, devuelve el valor false. Se necesitan operandos de tipo Boolean u operandos de cadena "true" o "false" que no distingan entre mayúsculas y minúsculas.

Sintaxis: *expression AND expression*.

#### Operador AND

Operando izquierdo	Operando derecho	Salida
Boolean	Boolean	Boolean. True si ambos operandos son true; de lo contrario, el valor es false.
String/Boolean	String/Boolean	Si todas las cadenas son "true" o "false" (no se distingue entre mayúsculas y minúsculas), se convierten en valores de tipo Boolean y se procesan normalmente como <i>boolean AND boolean</i> .
Otro valor	Otro valor	Undefined.

### Operador OR

Devuelve un resultado Boolean. Realiza una operación OR lógica. Devuelve el valor true si el operando izquierdo o el operando derecho es true; de lo contrario, devuelve el valor false. Se necesitan operandos de tipo Boolean u operandos de cadena "true" o "false" que no distingan entre mayúsculas y minúsculas.

Sintaxis: *expression OR expression*.

#### Operador OR

Operando izquierdo	Operando derecho	Salida
Boolean	Boolean	Boolean. True si uno de los operandos es true; en caso contrario, el valor es false.

Operando izquierdo	Operando derecho	Salida
String/Boolean	String/Boolean	Si todas las cadenas son "true" o "false" (no se distingue entre mayúsculas y minúsculas), se convierten en valores Boolean y se procesan normalmente como <code>boolean OR boolean</code> .
Otro valor	Otro valor	<code>Undefined</code> .

## Operador NOT

Devuelve un resultado Boolean. Realiza una operación NOT lógica. Devuelve el valor true si el operando es false; de lo contrario, devuelve el valor true. Se necesita un operando booleano o un operando de cadena "true" o "false" que no distinga entre mayúsculas y minúsculas.

Sintaxis: NOT `expression`.

### Operador NOT

Operando	Salida
Boolean	Boolean. True si el operando es false; en caso contrario, el valor es true.
String	Si la cadena es "true" o "false" (no distingue entre mayúsculas y minúsculas), se convierte en el valor booleano correspondiente y se devuelve el valor opuesto.
Otro valor	<code>Undefined</code> .

## > operador

Devuelve un resultado Boolean. Devuelve el valor true si el operando izquierdo es superior al operando derecho. Los dos operandos se convierten en un valor Decimal y, a continuación, se comparan.

Sintaxis: `expression > expression`.

### Operador >

Operando izquierdo	Operando derecho	Salida
Int/Decimal	Int/Decimal	Boolean. True si el operando izquierdo es superior al operando derecho; de lo contrario, el valor es false.
String/Int/ Decimal	String/Int/ Decimal	Si todas las cadenas se pueden convertir en un valor Decimal y, a continuación, en un valor Boolean. Devuelve el valor true si el operando izquierdo es superior al operando derecho; de lo contrario, devuelve el valor false.
Otro valor	<code>Undefined</code> .	<code>Undefined</code> .

## >= operador

Devuelve un resultado Boolean. Devuelve el valor true si el operando izquierdo es superior o igual al operando derecho. Los dos operandos se convierten en un valor Decimal y, a continuación, se comparan.

Sintaxis: *expression >= expression*.

Operador >=

Operando izquierdo	Operando derecho	Salida
Int/Decimal	Int/Decimal	Boolean. True si el operando izquierdo es superior o igual al operando derecho; de lo contrario, el valor es false.
String/Int/ Decimal	String/Int/ Decimal	Si todas las cadenas se pueden convertir en un valor Decimal y, a continuación, en un valor Boolean. Devuelve el valor true si el operando izquierdo es superior o igual al operando derecho; de lo contrario, devuelve el valor false.
Otro valor	Undefined.	Undefined.

## < operador

Devuelve un resultado Boolean. Devuelve el valor true si el operando izquierdo es inferior al operando derecho. Los dos operandos se convierten en un valor Decimal y, a continuación, se comparan.

Sintaxis: *expression < expression*.

<"."

Operando izquierdo	Operando derecho	Salida
Int/Decimal	Int/Decimal	Boolean. True si el operando izquierdo es inferior al operando derecho; de lo contrario, el valor es false.
String/Int/ Decimal	String/Int/ Decimal	Si todas las cadenas se pueden convertir en un valor Decimal y, a continuación, en un valor Boolean. Devuelve el valor true si el operando izquierdo es inferior al operando derecho; de lo contrario, devuelve el valor false.
Otro valor	Undefined	Undefined

## <= operador

Devuelve un resultado Boolean. Devuelve el valor true si el operando izquierdo es inferior o igual al operando derecho. Los dos operandos se convierten en un valor Decimal y, a continuación, se comparan.

Sintaxis: *expression <= expression*.

### Operador >=

Operando izquierdo	Operando derecho	Salida
Int/Decimal	Int/Decimal	Boolean. True si el operando izquierdo es inferior o igual al operando derecho; de lo contrario, el valor es false.
String/Int/ Decimal	String/Int/ Decimal	Si todas las cadenas se pueden convertir en un valor Decimal y, a continuación, en un valor Boolean. Devuelve el valor true si el operando izquierdo es inferior o igual al operando derecho; de lo contrario, devuelve el valor false.
Otro valor	Undefined	Undefined

### <> operador

Devuelve un resultado Boolean. Devuelve el valor true si los operandos izquierdo y derecho no son iguales; de lo contrario, devuelve el valor false.

Sintaxis: `expression <> expression`.

### Operador <>

Operando izquierdo	Operando derecho	Salida
Int	Int	True si el operando izquierdo no es igual al operando derecho; de lo contrario, el valor es false.
Decimal	Decimal	True si el operando izquierdo no es igual al operando derecho; de lo contrario, el valor es false. Int se convierte en un valor Decimal antes de la comparación.
String	String	True si el operando izquierdo no es igual al operando derecho; de lo contrario, el valor es false.
Matriz	Matriz	True si los elementos de cada operando no son iguales y no están en el mismo orden; de lo contrario, el valor es false.
Objeto	Objeto	True si las claves y los valores de cada operando no son iguales; de lo contrario, el valor es false. El orden de las claves y los valores no tiene importancia.
Null	Null	Falso.
Cualquier valor	Undefined	Sin definir.
Undefined	Cualquier valor	Sin definir.
Tipo no coincidente	Tipo no coincidente	Verdadero.

### = operador

Devuelve un resultado Boolean. Devuelve el valor true si los operandos izquierdo y derecho son iguales; de lo contrario, devuelve el valor false.

Sintaxis: `expression = expression.`

Operador =

Operando izquierdo	Operando derecho	Salida
Int	Int	True si el operando izquierdo es igual al operando derecho; de lo contrario, el valor es false.
Decimal	Decimal	True si el operando izquierdo es igual al operando derecho; de lo contrario, el valor es false. Int se convierte en un valor Decimal antes de la comparación.
String	String	True si el operando izquierdo es igual al operando derecho; de lo contrario, el valor es false.
Matriz	Matriz	True si los elementos de cada operando son iguales y están en el mismo orden; de lo contrario, el valor es false.
Objeto	Objeto	True si las claves y los valores de cada operando son iguales; de lo contrario, el valor es false. El orden de las claves y los valores no tiene importancia.
Cualquier valor	Undefined	Undefined.
Undefined	Cualquier valor	Undefined.
Tipo no coincidente	Tipo no coincidente	Falso.

## + operador

El símbolo "+" es un operador sobrecargado. Se puede utilizar para la concatenación o la adición de cadenas.

Sintaxis: `expression + expression.`

Operador +

Operando izquierdo	Operando derecho	Salida
String	Cualquier valor	Convierte el operando derecho en una cadena que concatena al final del operando izquierdo.
Cualquier valor	String	Convierte el operando izquierdo en una cadena y concatena el operando derecho al final del operando izquierdo convertido.
Int	Int	Int value. Agrega ambos operandos.
Int/Decimal	Int/Decimal	Decimal value. Agrega ambos operandos.
Otro valor	Otro valor	Undefined.

## - operador

Resta el operando derecho del operando izquierdo.

Sintaxis: *expression - expression*.

Operador -

Operando izquierdo	Operando derecho	Salida
Int	Int	Int value. Resta el operando derecho del operando izquierdo.
Int/Decimal	Int/Decimal	Decimal value. Resta el operando derecho del operando izquierdo.
String/Int/ Decimal	String/Int/ Decimal	Si todas las cadenas se convierten en valores de tipo Decimal correctamente, se devuelve un valor Decimal. Resta el operando derecho del operando izquierdo. De lo contrario, devuelve Undefined.
Otro valor	Otro valor	Undefined.
Otro valor	Otro valor	Undefined.

## \* operador

Multiplica el operando izquierdo por el operando derecho.

Sintaxis: *expression \* expression*.

Operador \*

Operando izquierdo	Operando derecho	Salida
Int	Int	Int value. Multiplica el operando izquierdo por el operando derecho.
Int/Decimal	Int/Decimal	Decimal value. Multiplica el operando izquierdo por el operando derecho.
String/Int/ Decimal	String/Int/ Decimal	Si todas las cadenas se convierten en valores de tipo Decimal correctamente, se devuelve un valor Decimal. Multiplica el operando izquierdo por el operando derecho. De lo contrario, devuelve Undefined.
Otro valor	Otro valor	Undefined.

## / operador

Divide el operando izquierdo por el operando derecho.

Sintaxis: *expression / expression*.

## Operador /

Operando izquierdo	Operando derecho	Salida
Int	Int	Int value. Divide el operando izquierdo por el operando derecho.
Int/Decimal	Int/Decimal	Decimal value. Divide el operando izquierdo por el operando derecho.
String/Int/ Decimal	String/Int/ Decimal	Si todas las cadenas se convierten en valores de tipo Decimal correctamente, se devuelve un valor Decimal. Divide el operando izquierdo por el operando derecho. De lo contrario, devuelve Undefined.
Otro valor	Otro valor	Undefined.

## % operador

Devuelve el resto de la división del operando izquierdo por el operando derecho.

Sintaxis: `expression % expression`.

## Operador %

Operando izquierdo	Operando derecho	Salida
Int	Int	Int value. Devuelve el resto de la división del operando izquierdo por el operando derecho.
String/Int/ Decimal	String/Int/ Decimal	Si todos los valores String se convierten en valores de tipo Decimal correctamente, se devuelve un valor Decimal. Devuelve el resto de la división del operando izquierdo por el operando derecho. De lo contrario, Undefined.
Otro valor	Otro valor	Undefined.

# Funciones

Puede utilizar las funciones integradas siguientes en las cláusulas SELECT o WHERE de sus expresiones SQL.

## abs(Decimal)

Devuelve el valor absoluto de un número. Es compatible con la versión SQL 2015-10-8 y versiones posteriores.

Ejemplo: `abs(-5)` devuelve 5.

Tipo de argumento	Resultado
Int	Int, el valor absoluto del argumento.

Tipo de argumento	Resultado
Decimal	Decimal, el valor absoluto del argumento.
Boolean	Undefined.
String	Decimal. El resultado es el valor absoluto del argumento. Si la cadena no se puede convertir, el resultado es Undefined.
Matriz	Undefined.
Objeto	Undefined.
Null	Undefined.
Sin definir	Undefined.

## accountid()

Devuelve el ID de la cuenta que posee esta regla como un valor de tipo `String`. Es compatible con la versión SQL 2015-10-8 y versiones posteriores.

Ejemplo:

```
accountid() = "123456789012"
```

## acos(Decimal)

Devuelve el coseno inverso de un número en radianes. Los argumentos de tipo `Decimal` se redondean a un valor de doble precisión antes de aplicar la función. Es compatible con la versión SQL 2015-10-8 y versiones posteriores.

Ejemplo: `acos(0) = 1.5707963267948966`

Tipo de argumento	Resultado
Int	Decimal (con doble precisión), el coseno inverso del argumento. Se devuelven resultados imaginarios como Undefined.
Decimal	Decimal (con doble precisión), el coseno inverso del argumento. Se devuelven resultados imaginarios como Undefined.
Boolean	Undefined.
String	Decimal, el coseno inverso del argumento. Si la cadena no se puede convertir, el resultado es Undefined. Se devuelven resultados imaginarios como Undefined.
Matriz	Undefined.
Objeto	Undefined.
Null	Undefined.
Sin definir	Undefined.

## asin(Decimal)

Devuelve el seno inverso de un número en radianes. Los argumentos de tipo `Decimal` se redondean a un valor de doble precisión antes de aplicar la función. Es compatible con la versión SQL 2015-10-8 y versiones posteriores.

Ejemplo: `asin(0) = 0.0`

Tipo de argumento	Resultado
<code>Int</code>	<code>Decimal</code> (con doble precisión), el seno inverso del argumento. Se devuelven resultados imaginarios como <code>Undefined</code> .
<code>Decimal</code>	<code>Decimal</code> (con doble precisión), el seno inverso del argumento. Se devuelven resultados imaginarios como <code>Undefined</code> .
<code>Boolean</code>	<code>Undefined</code> .
<code>String</code>	<code>Decimal</code> (con doble precisión), el seno inverso del argumento. Si la cadena no se puede convertir, el resultado es <code>Undefined</code> . Se devuelven resultados imaginarios como <code>Undefined</code> .
<code>Matriz</code>	<code>Undefined</code> .
<code>Objeto</code>	<code>Undefined</code> .
<code>Null</code>	<code>Undefined</code> .
<code>Sin definir</code>	<code>Undefined</code> .

## atan(Decimal)

Devuelve la tangente inversa de un número en radianes. Los argumentos de tipo `Decimal` se redondean a un valor de doble precisión antes de aplicar la función. Es compatible con la versión SQL 2015-10-8 y versiones posteriores.

Ejemplo: `atan(0) = 0.0`

Tipo de argumento	Resultado
<code>Int</code>	<code>Decimal</code> (con doble precisión), la tangente inversa del argumento. Se devuelven resultados imaginarios como <code>Undefined</code> .
<code>Decimal</code>	<code>Decimal</code> (con doble precisión), la tangente inversa del argumento. Se devuelven resultados imaginarios como <code>Undefined</code> .
<code>Boolean</code>	<code>Undefined</code> .
<code>String</code>	<code>Decimal</code> , la tangente inversa del argumento. Si la cadena no se puede convertir, el resultado es <code>Undefined</code> . Se devuelven resultados imaginarios como <code>Undefined</code> .

Tipo de argumento	Resultado
Matriz	Undefined.
Objeto	Undefined.
Null	Undefined.
Sin definir	Undefined.

## atan2(Decimal, Decimal)

Devuelve el ángulo, en radianes, entre el eje x positivo y el punto (x, y) definido en los dos argumentos. El ángulo es positivo para los ángulos en sentido contrario a las agujas del reloj (plano medio superior y > 0) y es negativo para los ángulos que siguen el sentido de las agujas del reloj (plano medio inferior y < 0). Los argumentos de tipo Decimal se redondean a un valor de doble precisión antes de aplicar la función. Es compatible con la versión SQL 2015-10-8 y versiones posteriores.

Ejemplo: atan2(1, 0) = 1.5707963267948966

Tipo de argumento	Tipo de argumento	Resultado
Int/Decimal	Int/Decimal	Decimal (con doble punto (x, y) especificado)
Int/Decimal/String	Int/Decimal/String	Decimal, la tangente de la cadena no se puede calcular
Otro valor	Otro valor	Undefined.

## bitand(Int, Int)

Ejecuta una operación AND bit a bit en las representaciones de bits de los dos argumentos Int(-convertidos). Es compatible con la versión SQL 2015-10-8 y versiones posteriores.

Ejemplo: bitand(13, 5) = 5

Tipo de argumento	Tipo de argumento	Resultado
Int	Int	Int, una operación AND
Int/Decimal	Int/Decimal	Int, una operación AND. Todos los números que tienen al valor Int inferior mantienen su resultado es Undefined.
Int/Decimal/String	Int/Decimal/String	Int, una operación AND. Todas las cadenas se convierten a Decimal y se redondean al entero más cercano. Si se produce un resultado obtenido es Undefined.
Otro valor	Otro valor	Undefined.

## bitor(Int, Int)

Realiza una operación OR bit a bit de las representaciones de bit de los dos argumentos. Es compatible con la versión SQL 2015-10-8 y versiones posteriores.

Ejemplo: `bitor(8, 5) = 13`

Tipo de argumento	Tipo de argumento	Resultado
Int	Int	Int, la operación OR
Int/Decimal	Int/Decimal	Int, la operación OR Todos los números que no son enteros se redondean al valor Int inferior más cercano. Si se produce un resultado obtenido es Undefined.
Int/Decimal/String	Int/Decimal/String	Int, la operación OR Todas las cadenas se convierten a tipo Decimal y se redondean al número más cercano. Si se produce un resultado obtenido es Undefined.
Otro valor	Otro valor	Undefined.

## bitxor(Int, Int)

Ejecuta una operación XOR bit a bit en las representaciones de bits de los dos argumentos Int (-convertidos). Es compatible con la versión SQL 2015-10-8 y versiones posteriores.

Ejemplo: `bitor(13, 5) = 8`

Tipo de argumento	Tipo de argumento	Resultado
Int	Int	Int, una operación XOR
Int/Decimal	Int/Decimal	Int, una operación XOR Los números que no son enteros se redondean al número más cercano.
Int/Decimal/String	Int/Decimal/String	Int, una operación XOR Los valores de tipo String se convierten a tipo Decimal y se redondean al número más cercano. Si se produce un resultado obtenido es Undefined.
Otro valor	Otro valor	Undefined.

## bitnot(Int)

Ejecuta una operación NOT bit a bit en las representaciones del argumento Int (-convertido). Es compatible con la versión SQL 2015-10-8 y versiones posteriores.

Ejemplo: `bitnot(13) = 2`

Tipo de argumento	Resultado
Int	Int, una operación NOT bit a bit del argumento.
Decimal	Int, una operación NOT bit a bit del argumento. El valor Decimal se redondea al valor Int inferior más cercano.
String	Int, una operación NOT bit a bit del argumento. Los valores de tipo String se convierten en valores de tipo Decimal y se redondean al valor Int inferior más cercano. Si se produce un error en la conversión, el resultado obtenido es Undefined.
Otro valor	Otro valor.

## cast()

Convierte un valor de un tipo de datos a otro tipo. Cast se comporta básicamente como las conversiones estándar, salvo que puede convertir números en valores de tipo Boolean o a la inversa. Si no podemos determinar cómo convertir un tipo en otro, el resultado es Undefined. Es compatible con la versión SQL 2015-10-8 y versiones posteriores. Formato: cast(*valor* as *tipo*).

Ejemplo:

```
cast(true as Decimal) = 1.0
```

Las siguientes palabras clave pueden aparecer después de "as" cuando se llama a cast:

Palabra clave	Resultado
Decimal	Convierte un valor en Decimal.
Bool	Convierte un valor en Boolean.
Boolean	Convierte un valor en Boolean.
String	Convierte un valor en String.
Nvarchar	Convierte un valor en String.
Texto	Convierte un valor en String.
Ntext	Convierte un valor en String.
varchar	Convierte un valor en String.
Int	Convierte un valor en Int.
Int	Convierte un valor en Int.

Reglas de conversión:

Conversión en decimal

Tipo de argumento	Resultado
Int	Valor Decimal sin separador decimal.

Tipo de argumento	Resultado
Decimal	El valor de origen.
Boolean	true = 1.0, false = 0.0.
String	Intentaremos analizar la cadena como un valor Decimal. Intentaremos analizar cadenas que coincidan con el regex: ^-?\d+(.\d+)?((?i)E-?\d+)?\$. "0", "-1.2", "5E-12" son ejemplos de valores de String que se convertirían automáticamente en valores de tipo Decimal.
Matriz	Undefined.
Objeto	Undefined.
Null	Undefined.
Sin definir	Undefined.

### Conversión en entero

Tipo de argumento	Resultado
Int	El valor de origen.
Decimal	El valor de origen redondeado al valor Int inferior más cercano.
Boolean	true = 1.0, false = 0.0.
String	Intentaremos analizar la cadena como un valor Decimal. Intentaremos analizar cadenas que coincidan con el regex: ^-?\d+(.\d+)?((?i)E-?\d+)?\$. "0", "-1.2", "5E-12" son ejemplos de valores de tipo String que se convertirían automáticamente en valores de tipo Decimal. Intentaremos convertir la cadena en un valor de tipo Decimal y redondearlo al valor de tipo Int inferior más cercano.
Matriz	Undefined.
Objeto	Undefined.
Null	Undefined.
Sin definir	Undefined.

### Conversión a valor Boolean

Tipo de argumento	Resultado
Int	0 = False, any_nonzero_value = True.
Decimal	0 = False, any_nonzero_value = True.
Boolean	El valor de origen.

Tipo de argumento	Resultado
String	"true" = True y "false" = False (no distingue entre mayúsculas y minúsculas). Otros valores de cadena = <code>Undefined</code> .
Matriz	<code>Undefined</code> .
Objeto	<code>Undefined</code> .
Null	<code>Undefined</code> .
Sin definir	<code>Undefined</code> .

### Conversión en cadenas

Tipo de argumento	Resultado
Int	Una representación de cadena del valor <code>Int</code> , en notación estándar.
Decimal	Una cadena que representa el valor <code>Decimal</code> , posiblemente en notación científica.
Boolean	"true" o "false", todo en minúsculas.
String	"true" = True y "false" = False (no distingue entre mayúsculas y minúsculas). Otros valores de cadena = <code>Undefined</code> .
Matriz	La matriz serializada en formato JSON. La cadena de resultados obtenida será una lista separada por comas, entre corchetes. Los valores de tipo <code>String</code> se indican entre comillas. Los valores de tipo <code>Decimal</code> , <code>Int</code> o <code>Boolean</code> no están entre comillas.
Objeto	El objeto serializado al formato JSON. La cadena JSON será una lista separada por comas de pares clave-valor, y comenzará y terminará con llaves. Los valores de tipo <code>String</code> se indican entre comillas. Los <code>Decimals</code> o los valores de tipo <code>Int</code> , <code>Boolean</code> o <code>Null</code> no se indican entre comillas.
Null	<code>Undefined</code> .
Sin definir	<code>Undefined</code> .

## ceil(Decimal)

Redondea el valor `Decimal` indicado al valor `Int` superior más cercano. Es compatible con la versión SQL 2015-10-8 y versiones posteriores.

Ejemplos:

`ceil(1.2) = 2`

`ceil(-11.2) = -1`

Tipo de argumento	Resultado
Int	Int, el valor del argumento.
Decimal	Int, el valor de Decimal redondeado al valor de tipo Int superior más cercano.
String	Int. La cadena se convierte en un valor Decimal y se redondea al valor de tipo Int superior más cercano. Si la cadena no se puede convertir en un valor Decimal, el resultado es Undefined.
Otro valor	Undefined.

## chr(String)

Devuelve el carácter ASCII que corresponde al argumento Int determinado. Es compatible con la versión SQL 2015-10-8 y versiones posteriores.

Ejemplos:

`chr(65) = "A".`

`chr(49) = "1".`

Tipo de argumento	Resultado
Int	El carácter correspondiente al valor ASCII especificado. Si el argumento no es un valor ASCII válido, el resultado es Undefined.
Decimal	El carácter correspondiente al valor ASCII especificado. El argumento Decimal se redondea al valor Int inferior más cercano. Si el argumento no es un valor ASCII válido, el resultado es Undefined.
Boolean	Undefined.
String	Si el valor String puede convertirse en un valor Decimal, se redondea al valor Int inferior más cercano. Si el argumento no es un valor ASCII válido, el resultado es Undefined.
Matriz	Undefined.
Objeto	Undefined.
Null	Undefined.
Otro valor	Undefined.

## clientid()

Devuelve el ID del cliente MQTT que envía el mensaje o n/a si el mensaje no se ha enviado por MQTT. Es compatible con la versión SQL 2015-10-8 y versiones posteriores.

Ejemplo:

```
clientid() = "123456789012"
```

## concat()

Concatena matrices o cadenas. Esta función acepta cualquier cantidad de argumentos y devuelve un valor `String` o un valor `Array`. Es compatible con la versión SQL 2015-10-8 y versiones posteriores.

Ejemplos:

```
concat() = Undefined.  
concat(1) = "1".  
concat([1, 2, 3], 4) = [1, 2, 3, 4].  
concat([1, 2, 3], "hello") = [1, 2, 3, "hello"]  
concat("con", "cat") = "concat"  
concat(1, "hello") = "1hello"  
concat("he", "is", "man") = "heisman"  
concat([1, 2, 3], "hello", [4, 5, 6]) = [1, 2, 3, "hello", 4, 5, 6]
```

Número de argumentos	Resultado
0	<code>Undefined</code> .
1	El argumento se devuelve sin modificar.
2+	Si alguno de los argumentos es un valor <code>Array</code> , el resultado es una matriz única que contiene todos los argumentos. Si ningún argumento es un valor <code>Array</code> y al menos un argumento es un valor <code>String</code> , el resultado es la concatenación de las representaciones de <code>String</code> de todos los argumentos. Los argumentos se convertirán en valores de tipo <code>String</code> mediante las conversiones estándar indicadas arriba. ·

## cos(Decimal)

Devuelve el coseno de un número en radianes. Los argumentos de tipo `Decimal` se redondean a un valor de doble precisión antes de aplicar la función. Es compatible con la versión SQL 2015-10-8 y versiones posteriores.

Ejemplo:

```
cos(0) = 1.
```

Tipo de argumento	Resultado
<code>Int</code>	<code>Decimal</code> (con doble precisión), el coseno del argumento. Se devuelven resultados imaginarios como <code>Undefined</code> .
<code>Decimal</code>	<code>Decimal</code> (con doble precisión), el coseno del argumento. Se devuelven resultados imaginarios como <code>Undefined</code> .

Tipo de argumento	Resultado
Boolean	Undefined.
String	Decimal (con doble precisión), el coseno del argumento. Si la cadena no se puede convertir en un valor Decimal, el resultado es Undefined. Se devuelven resultados imaginarios como Undefined.
Matriz	Undefined.
Objeto	Undefined.
Null	Undefined.
Sin definir	Undefined.

## cosh(Decimal)

Devuelve el coseno hiperbólico de un número en radianes. Los argumentos de tipo Decimal se redondean a un valor de doble precisión antes de aplicar la función. Es compatible con la versión SQL 2015-10-8 y versiones posteriores.

Ejemplo: `cosh(2.3) = 5.037220649268761.`

Tipo de argumento	Resultado
Int	Decimal (con doble precisión), el coseno hiperbólico del argumento. Se devuelven resultados imaginarios como Undefined.
Decimal	Decimal (con doble precisión), el coseno hiperbólico del argumento. Se devuelven resultados imaginarios como Undefined.
Boolean	Undefined.
String	Decimal (con doble precisión), el coseno hiperbólico del argumento. Si la cadena no se puede convertir en un valor Decimal, el resultado es Undefined. Se devuelven resultados imaginarios como Undefined.
Matriz	Undefined.
Objeto	Undefined.
Null	Undefined.
Sin definir	Undefined.

## encode(value, encodingScheme)

Utilice la función encode para codificar la carga, que puede estar constituida por datos que no son JSON, en su representación de cadena basada en el esquema de codificación. Es compatible con la versión SQL 2016-03-23 y versiones posteriores.

value

Cualquiera de las expresiones válidas, tal y como se define en [Referencias de SQL en AWS IoT \(p. 168\)](#). Además, puede especificar \* para codificar toda la carga, con independencia de si está en formato JSON o no. Si suministra una expresión, el resultado de la evaluación se convertirá en una cadena antes de codificarla.

encodingScheme

Una cadena literal que representa el esquema de codificación que desea utilizar. En la actualidad, solo se admite 'base64'.

## endswith(String, String)

Devuelve un valor Booleano que indica si el primer argumento String termina con el segundo argumento String. Si alguno de los argumentos es Null o Undefined, el resultado es Undefined. Es compatible con la versión SQL 2015-10-8 y versiones posteriores.

Ejemplo: `endswith("cat", "at") = true.`

Tipo de argumento 1	Tipo de argumento 2	Resultado
String	String	True si el primer argumento es String; de lo contrario, Undefined.
Otro valor	Otro valor	Ambos argumentos son String mediante las conversiones correspondientes. True si el primer argumento es String; de lo contrario, Undefined.

## exp(Decimal)

Devuelve e a la potencia del argumento Decimal. Los argumentos de tipo Decimal se redondean a un valor de doble precisión antes de aplicar la función. Es compatible con la versión SQL 2015-10-8 y versiones posteriores.

Ejemplo: `exp(1) = e.`

Tipo de argumento	Resultado
Int	Decimal (con doble precisión), argumento potencia e.
Decimal	Decimal (con doble precisión), argumento potencia e.
String	Decimal (con doble precisión), argumento potencia e. Si el valor String no se puede convertir en un valor Decimal, el resultado es Undefined.
Otro valor	Undefined.

## get

Extrae un valor de un tipo de recopilación (matriz, cadena, objeto). No se aplicará ninguna conversión al primer argumento. La conversión se aplica tal y como se documenta en la tabla del segundo argumento. Es compatible con la versión SQL 2015-10-8 y versiones posteriores.

Ejemplos:

```
get(["a", "b", "c"], 1) = "b"
get({"a": "b"}, "a") = "b"
get("abc", 1) = "b"
```

Tipo de argumento 1	Tipo de argumento 2	Resultado
Matriz	Cualquier tipo (convertido en Int)	El elemento en el índice proporcionado por el segundo argumento (convertido en Int). Si la conversión es Undefined. Si el índice es negativo, el resultado es Undefined.
Cadena	Cualquier tipo (convertido en Int)	El carácter en el índice proporcionada por el segundo argumento (convertido en un valor Int). Si la conversión obtenido es Undefined o se supera los límites de la cadena, el resultado es Undefined.
Objeto	String (no se aplica conversión)	El valor almacenado en la clave correspondiente a la clave en el objeto como segundo argumento.
Otro valor	Cualquier valor	Undefined.

## get\_thing\_shadow(thingName, roleARN)

Muestra la sombra del elemento especificado. Es compatible con la versión SQL 2016-03-23 y versiones posteriores.

thingName

String: el nombre del elemento cuya sombra desea recuperar.

roleArn

String: un ARN de rol con permiso iot:GetThingShadow.

Ejemplo:

```
SELECT * from 'a/b'
WHERE get_thing_shadow("MyThing", "arn:aws:iam::123456789012:role/
AllowsThingShadowAccess") .state.reported.alarm = 'ON'
```

## Funciones de hash

Ofrecemos las siguientes funciones de hash:

- md2
- md5
- sha1
- sha224
- sha256
- sha384
- sha512

Todas las funciones de hash esperan un argumento de cadena. El resultado es el valor con hash de dicha cadena. Las conversiones de cadena estándar se aplican a los argumentos que no son cadenas. Todas las funciones de hash son compatibles con SQL versión 2015-10-8 y con versiones posteriores.

Ejemplos:

```
md2("hello") = "a9046c73e00331af68917d3804f70655"
```

```
md5("hello") = "5d41402abc4b2a76b9719d911017c592"
```

## indexof(String, String)

Devuelve el primer índice (basado en 0) del segundo argumento como subcadena del primer argumento. Se espera que ambos argumentos sean cadenas. Los argumentos que no sean cadenas están sujetos a las reglas de conversión estándar de cadenas. Esta función no se aplica a matrices, únicamente a cadenas. Es compatible con la versión SQL 2015-10-8 y versiones posteriores.

Ejemplos:

```
indexof("abcd", "bc") = 1
```

## isNull()

Muestra si el argumento es el valor `Null`. Es compatible con la versión SQL 2015-10-8 y versiones posteriores.

Ejemplos:

```
isNull(5) = false.
```

```
isNull(null) = true.
```

Tipo de argumento	Resultado
Int	false
Decimal	false
Boolean	false
String	false
Array	false
Object	false
Null	true
Undefined	false

## isUndefined()

Muestra si el argumento tiene el valor `undefined`. Es compatible con la versión SQL 2015-10-8 y versiones posteriores.

Ejemplos:

```
isUndefined(5) = false.  
isNull(floor([1,2,3])) = true.
```

Tipo de argumento	Resultado
Int	false
Decimal	false
Boolean	false
String	false
Array	false
Object	false
Null	false
Undefined	true

## length(String)

Devuelve el número de caracteres de la cadena suministrada. Se aplican las reglas de conversión estándar a los argumentos que no sean `String`. Es compatible con la versión SQL 2015-10-8 y versiones posteriores.

Ejemplos:

```
length("hi") = 2  
length(false) = 5
```

## ln(Decimal)

Devuelve el logaritmo natural del argumento. Los argumentos de tipo `Decimal` se redondean a un valor de doble precisión antes de aplicar la función. Es compatible con la versión SQL 2015-10-8 y versiones posteriores.

Ejemplo:`ln(e) = 1.`

Tipo de argumento	Resultado
Int	Decimal (con doble precisión), el log natural del argumento.
Decimal	Decimal (con doble precisión), el log natural del argumento.

Tipo de argumento	Resultado
Boolean	Undefined.
String	Decimal (con doble precisión), el log natural del argumento. Si la cadena no se puede convertir en un valor Decimal, el resultado es Undefined.
Matriz	Undefined.
Objeto	Undefined.
Null	Undefined.
Sin definir	Undefined.

## log(Decimal)

Devuelve el logaritmo de base 10 del argumento. Los argumentos de tipo Decimal se redondean a un valor de doble precisión antes de aplicar la función. Es compatible con la versión SQL 2015-10-8 y versiones posteriores.

Ejemplo: `log(100) = 2.0.`

Tipo de argumento	Resultado
Int	Decimal (con doble precisión), el log de base 10 del argumento.
Decimal	Decimal (con doble precisión), el log de base 10 del argumento.
Boolean	Undefined.
String	Decimal (con doble precisión), el log de base 10 del argumento. Si el valor String no se puede convertir en un valor Decimal, el resultado es Undefined.
Matriz	Undefined.
Objeto	Undefined.
Null	Undefined.
Sin definir	Undefined.

## lower(String)

Muestra la versión en minúsculas del valor String indicado. Los argumentos que no son cadenas se convierten en valores de tipo String con las reglas de conversión estándar. Es compatible con la versión SQL 2015-10-8 y versiones posteriores.

Ejemplos:

`lower("HELLO") = "hello".`

`lower(["HELLO"]) = ["hello"].`

## lpad(String, Int)

Devuelve el argumento **String**, rellenado en el lado izquierdo con el número de espacios especificado por el segundo argumento. El argumento **Int** debe estar comprendido entre 0 y 1000. Si el valor proporcionado se encuentra fuera de este rango válido, el argumento se configurará en el valor válido más cercano (0 o 1000). Es compatible con la versión SQL 2015-10-8 y versiones posteriores.

Ejemplos:

`lpad("hello", 2) = " hello".`

`lpad(1, 3) = " 1"`

Tipo de argumento 1	Tipo de argumento 2	Resultado
<b>String</b>	<b>Int</b>	<b>String</b> , el valor <b>String</b> relleno en el lado izquierdo con un espacio por cada uno de los valores <b>Int</b> proporcionado.
<b>String</b>	<b>Decimal</b>	El argumento <b>Decimal</b> más cercano y el valor <b>String</b> relleno en el lado izquierdo con el número de espacios.
<b>String</b>	<b>String</b>	El segundo argumento que se redondea al valor entero más cercano y el valor <b>String</b> se rellena en el lado izquierdo con el número de espacios especificado en la izquierda. Si no se puede convertir el valor <b>String</b> a <b>Int</b> , se devolverá <b>Undefined</b> .
Otro valor	<b>Int/Decimal/String</b>	El primer valor se convierte mediante las conversiones y si la función LPAD se aplica a un valor que no se puede convertir, el resultado es <b>Undefined</b> .
Cualquier valor	Otro valor	<b>Undefined</b> .

## ltrim(String)

Elimina todos los espacios en blanco del principio (pestañas y espacios) del valor **String** proporcionado. Es compatible con la versión SQL 2015-10-8 y versiones posteriores.

Ejemplo:

`Ltrim(" h i ") = "hi".`

Tipo de argumento	Resultado
<b>Int</b>	La representación <b>String</b> del valor <b>Int</b> con todos los espacios en blanco del principio suprimidos.
<b>Decimal</b>	La representación <b>String</b> del valor <b>Decimal</b> con todos los espacios en blanco del principio suprimidos.
<b>Boolean</b>	La representación <b>String</b> del valor booleano ("true" o "false") con todos los espacios en blanco del principio suprimidos.

Tipo de argumento	Resultado
String	El argumento con todos los espacios en blanco del principio suprimidos.
Matriz	La representación String del valor Array (mediante las reglas de conversión estándar) con todos los espacios en blanco del principio suprimidos.
Objeto	La representación String del objeto (mediante las reglas de conversión estándar) con todos los espacios en blanco del principio suprimidos.
Null	Undefined.
Sin definir	Undefined.

## machinelearning\_predict(modelId)

Utilice la función `machinelearning_predict` para realizar predicciones con los datos de un mensaje MQTT basado en un modelo Amazon Machine Learning (Amazon ML). Es compatible con la versión SQL 2015-10-8 y versiones posteriores. Los argumentos de la función `machinelearning_predict` son:

`modelId`

El ID del modelo en el que se ejecutará la predicción. El punto de enlace en tiempo real del modelo debe estar activado.

`roleArn`

El rol de IAM que tiene una política con permisos `machinelearning:Predict` y `machinelearning:GetMLModel`, y permite tener acceso al modelo en el que se ejecuta la predicción.

`record`

Los datos que van a transmitirse a la API de previsión Amazon ML. Debe representarse como un objeto JSON de capa única. Si el registro es un objeto JSON de varias capas, el registro se aplanará serializando sus valores. Por ejemplo, el JSON siguiente:

```
{ "key1": { "innerKey1": "value1"}, "key2": 0}
```

se convertiría en:

```
{ "key1": "{\"innerKey1\": \"value1\"}", "key2": 0}
```

La función devuelve un objeto JSON con los campos siguientes:

`predictedLabel`

La clasificación de la entrada en función del modelo.

`details`

Contiene los atributos siguientes:

`PredictiveModelType`

El tipo de modelo. Los valores válidos son REGRESSION, BINARY, MULTICLASS.

#### Algorithm

El algoritmo utilizado por Amazon ML para realizar las predicciones. El valor debe ser SGD.  
predictedScores

Contiene la puntuación de clasificación bruta correspondiente a cada etiqueta.  
predictedValue

El valor que Amazon ML prevé.

## mod(Decimal, Decimal)

Devuelve el resto de la división del primer argumento por el segundo argumento. Es compatible con la versión SQL 2015-10-8 y versiones posteriores. También puede utilizar "%" como un operador infijo para la misma funcionalidad modulo. Es compatible con la versión SQL 2015-10-8 y versiones posteriores.

Ejemplo: `mod(8, 3) = 2.`

Operando izquierdo	Operando derecho	Salida
Int	Int	Int, el primer y el segundo argumentos deben ser enteros para ejecutar la funcionalidad.
Int/Decimal	Int/Decimal	Decimal, el primer argumento debe ser decimal para los que quiere ejecutar la funcionalidad.
String/Int/Decimal	String/Int/Decimal	Si todas las cadenas son válidas, se devolverá el resultado decimal, la funcionalidad no se ejecutará. Si el primer argumento es undefined, el resultado es Undefined.
Otro valor	Otro valor	Undefined.

## nanvl(AnyValue, AnyValue)

Devuelve el primer argumento si es un valor Decimal válido; de lo contrario, se devuelve el segundo argumento. Es compatible con la versión SQL 2015-10-8 y versiones posteriores.

Ejemplo: `Nanvl(8, 3) = 8.`

Tipo de argumento 1	Tipo de argumento 2	Salida
Sin definir	Cualquier valor	El segundo argumento.
Null	Cualquier valor	El segundo argumento.
Decimal (NaN)	Cualquier valor	El segundo argumento.
Decimal (no NaN)	Cualquier valor	El primer argumento.
Otro valor	Cualquier valor	El primer argumento.

## newuuid()

Devuelve un UUID aleatorio de 16 bytes. Es compatible con la versión SQL 2015-10-8 y versiones posteriores.

Ejemplo: newuuid() = 123a4567-b89c-12d3-e456-789012345000

## numbytes(String)

Devuelve el número de bytes de la codificación UTF-8 de la cadena proporcionada. Se aplican las reglas de conversión estándar a los argumentos que no sean `String`. Es compatible con la versión SQL 2015-10-8 y versiones posteriores.

Ejemplos:

`numbytes("hi") = 2`

`numbytes("€") = 3`

## principal()

Devuelve la huella de certificado X.509 o el nombre de un objeto, en función del punto de enlace, MQTT o HTTP, que recibió la solicitud. Es compatible con la versión SQL 2015-10-8 y versiones posteriores.

Ejemplo:

`principal() = "ba67293af50bf2506f5f93469686da660c7c844e7b3950fb16813e0d31e9373"`

## parse\_time(String, Long, [String])

Utilice la función `parse_time` para aplicar un formato legible a una marca de fecha y hora. Es compatible con la versión SQL 2016-03-23 y versiones posteriores. Los argumentos de la función `parse_time` son:

pattern

(String) Un patrón de fecha y hora que se ajusta al formato estándar [ISO 8601](#). (En concreto, la función admite [formatos Joda-Time](#)).

timestamp

(Long) La hora que se va a formatear en milisegundos a partir del formato de hora Unix. Consulte la función [timestamp\(\)](#) (p. 209).

timezone

(String) [Opcional] La zona horaria de la fecha/hora formateada. El valor predeterminado es "UTC". La función admite [zonas horarias Joda-Time](#)

Ejemplos:

Cuando este mensaje se publique en el tema "A/B", la carga `{"ts": "1970.01.01 AD at 21:46:40 CST"}` se enviará al bucket de S3:

```
{  
    "ruleArn": "arn:aws:iot:us-east-2:ACCOUNT_ID:rule/RULE_NAME",  
    "rule": {  
        "awsIotSqlVersion": "2016-03-23",  
        "sql": "SELECT parse_time(\"yyyy.MM.dd G 'at' HH:mm:ss z\", 100000000, \"America/  
Belize\" ) as ts FROM 'A/B'",  
        "ruleDisabled": false,  
        "actions": [  
            {  
                "s3": {  
                    "bucket": "my-s3-bucket",  
                    "prefix": "logs/"  
                }  
            }  
        ]  
    }  
}
```

```

        "roleArn": "arn:aws:iam::ACCOUNT_ID:rule:role/ROLE_NAME",
        "bucketName": "BUCKET_NAME",
        "key": "KEY_NAME"
    }
},
"ruleName": "RULE_NAME"
}
}

```

Cuando este mensaje se publique en el tema "A/B", una carga similar a `{"ts": "2017.06.09 AD at 17:19:46 UTC"}` (pero con la fecha y hora actuales) se enviará al bucket de S3:

```

{
    "ruleArn": "arn:aws:iot:us-east-2:ACCOUNT_ID:rule/RULE_NAME",
    "rule": {
        "awsIotSqlVersion": "2016-03-23",
        "sql": "SELECT parse_time('yyyy.MM.dd G 'at' HH:mm:ss z", timestamp()) as ts FROM 'A/B'",
        "ruleDisabled": false,
        "actions": [
            {
                "s3": {
                    "roleArn": "arn:aws:iam::ACCOUNT_ID:rule:role/ROLE_NAME",
                    "bucketName": "BUCKET_NAME",
                    "key": "KEY_NAME"
                }
            }
        ],
        "ruleName": "RULE_NAME"
    }
}

```

`parse_time()` se puede usar también como una plantilla de sustitución. Por ejemplo, cuando este mensaje se publique en el tema "A/B", la carga se enviará al bucket de S3 con la clave = "2017":

```

{
    "ruleArn": "arn:aws:iot:us-east-2:ACCOUNT_ID:rule/RULE_NAME",
    "rule": {
        "awsIotSqlVersion": "2016-03-23",
        "sql": "SELECT * FROM 'A/B'",
        "ruleDisabled": false,
        "actions": [
            {
                "s3": {
                    "roleArn": "arn:aws:iam::ACCOUNT_ID:rule:role/ROLE_NAME",
                    "bucketName": BUCKET_NAME,
                    "key": "${parse_time('yyyy", timestamp(), "UTC")}"
                }
            }
        ],
        "ruleName": "RULE_NAME"
    }
}

```

## power(Decimal, Decimal)

Devuelve el primer argumento elevado a la potencia del segundo argumento. Los argumentos de tipo Decimal se redondean a un valor de doble precisión antes de aplicar la función. Es compatible con la versión SQL 2015-10-8 y versiones posteriores. Es compatible con la versión SQL 2015-10-8 y versiones posteriores.

Ejemplo: `power(2, 5) = 32.0.`

Tipo de argumento 1	Tipo de argumento 2	Salida
Int/Decimal	Int/Decimal	Un Decimal (con doble elevado a la potencia).
Int/Decimal/String	Int/Decimal/String	Un Decimal (con doble elevado a la potencia). Si las cadenas se convierten en algún valor String nulo, el resultado es Undefined.
Otro valor	Otro valor	Undefined.

## rand()

Devuelve un valor pseudoaleatorio, distribuido de forma uniforme entre 0,0 y 1,0. Compatible con SQL versión 2015-10-8 y versiones posteriores.

Ejemplo:

```
rand() = 0.8231909191640703
```

## regexp\_matches(String, String)

Muestra si el primer argumento contiene una coincidencia para el segundo argumento (regex).

Ejemplo:

```
Regexp_matches("aaaa", "a{2,}") = true.
```

```
Regexp_matches("aaaa", "b") = false.
```

Primer argumento:

Tipo de argumento	Resultado
Int	La representación String del valor Int.
Decimal	La representación String del valor Decimal.
Boolean	La representación String del valor booleano ("true" o "false").
String	El valor String.
Matriz	La representación String del valor Array (mediante reglas de conversión estándar).
Objeto	La representación String del objeto (mediante reglas de conversión estándar).
Null	Undefined.
Sin definir	Undefined.

Segundo argumento:

Tiene que ser una expresión regex válida. Los tipos que no son cadenas se convierten en valores de tipo `String` mediante reglas de conversión estándar. En función del tipo, la cadena obtenida puede ser o no ser una expresión regular válida. Si el argumento (convertido) no es un regex válido, el resultado es `Undefined`.

Tercer argumento:

Debe ser una cadena de sustitución de regex válida. (Puede hacer referencia a grupos de capturas). Los tipos que no sean cadenas se convertirán en valores de tipo `String` mediante reglas de conversión estándar. Si el argumento (convertido) no es una cadena de sustitución de regex válida, el resultado es `Undefined`.

## regexp\_replace(String, String, String)

Sustituye todos los segundos argumentos (expresiones regulares) que hay en el primer argumento por el tercer argumento. Hace referencia a los grupos de captura con `$`. Es compatible con la versión SQL 2015-10-8 y versiones posteriores.

Ejemplo:

```
Regexp_replace("abcd", "bc", "x") = "axd".  
Regexp_replace("abcd", "b(.*)d", "$1") = "ac".
```

Primer argumento:

Tipo de argumento	Resultado
<code>Int</code>	La representación <code>String</code> del valor <code>Int</code> .
<code>Decimal</code>	La representación <code>String</code> del valor <code>Decimal</code> .
<code>Boolean</code>	La representación <code>String</code> del valor booleano ("true" o "false").
<code>String</code>	El valor de origen.
<code>Matriz</code>	La representación <code>String</code> del valor <code>Array</code> (mediante reglas de conversión estándar).
<code>Objeto</code>	La representación <code>String</code> del objeto (mediante reglas de conversión estándar).
<code>Null</code>	<code>Undefined</code> .
<code>Sin definir</code>	<code>Undefined</code> .

Segundo argumento:

Tiene que ser una expresión regex válida. Los tipos que no son cadenas se convierten en valores de tipo `String` mediante reglas de conversión estándar. En función del tipo, la cadena obtenida puede ser o no ser una expresión regular válida. Si el argumento (convertido) no es una expresión regex válida, el resultado es `Undefined`.

Tercer argumento:

Debe ser una cadena de sustitución de regex válida. (Puede hacer referencia a grupos de capturas). Los tipos que no sean cadenas se convertirán en valores de tipo `String` mediante reglas de conversión estándar. Si el argumento (convertido) no es una cadena de sustitución de regex válida, el resultado es `Undefined`.

## regexp\_substr(String, String)

Busca la primera coincidencia del segundo parámetro (regex) en el primer parámetro. Hace referencia a los grupos de captura con "\$". Es compatible con la versión SQL 2015-10-8 y versiones posteriores.

Ejemplo:

```
regexp_substr("hihihello", "hi") => "hi"  
regexp_substr("hihihello", "(hi)*") => "hihi".
```

Primer argumento:

Tipo de argumento	Resultado
Int	La representación String del valor Int.
Decimal	La representación String del valor Decimal.
Boolean	La representación String del valor booleano ("true" o "false").
String	El argumento String.
Matriz	La representación String del valor Array (mediante reglas de conversión estándar).
Objeto	La representación String del objeto (mediante reglas de conversión estándar).
Null	Undefined.
Sin definir	Undefined.

Segundo argumento:

Tiene que ser una expresión regex válida. Los tipos que no son cadenas se convierten en valores de tipo String mediante reglas de conversión estándar. En función del tipo, la cadena obtenida puede ser o no ser una expresión regular válida. Si el argumento (convertido) no es una expresión regex válida, el resultado es Undefined.

Tercer argumento:

Debe ser una cadena de sustitución de regex válida. (Puede hacer referencia a grupos de capturas). Los tipos que no sean cadenas se convertirán en valores de tipo String mediante reglas de conversión estándar. Si el argumento no es una cadena de sustitución de regex válida, el resultado es Undefined.

## rpad(String, Int)

Devuelve el argumento de cadena, rellenado en el lado derecho con el número de espacios especificado en el segundo argumento. El argumento Int debe estar comprendido entre 0 y 1000. Si el valor proporcionado se encuentra fuera de este rango válido, el argumento se configurará en el valor válido más cercano (0 o 1000). Es compatible con la versión SQL 2015-10-8 y versiones posteriores.

Ejemplos:

```
rpad("hello", 2) = "hello ".  
rpad(1, 3) = "1 ".
```

Tipo de argumento 1	Tipo de argumento 2	Resultado
String	Int	El valor String se rellena en el lado derecho con un número de espacios igual al valor Int proporcionado.
String	Decimal	El argumento Decimal se redondea al valor Int inferior más cercano y la cadena se rellena en el lado derecho con una serie de espacios igual al valor Int proporcionado.
String	String	El segundo argumento se convierte en un valor Decimal que se redondea

Tipo de argumento 1	Tipo de argumento 2	Resultado
		al valor Int inferior más cercano. El valor String se rellena en el lado derecho con un número de espacios igual al valor Int.
Otro valor	Int/Decimal/String	El primer valor se convierte en un valor de tipo String mediante conversiones estándar y la función rpad se aplica en dicho valor de String. Si no se puede convertir, el resultado es Undefined.
Cualquier valor	Otro valor	Undefined.

## round(Decimal)

Redondea el valor `Decimal` indicado al valor `Int` más cercano. Si el valor `Decimal` está a la misma distancia de dos valores `Int`, (por ejemplo, 0,5), el valor `Decimal` se redondea al valor superior. Es compatible con la versión SQL 2015-10-8 y versiones posteriores.

Ejemplo: `Round(1.2) = 1.`

`Round(1.5) = 2.`

`Round(1.7) = 2.`

`Round(-1.1) = -1.`

`Round(-1.5) = -2.`

Tipo de argumento	Resultado
<code>Int</code>	El argumento.
<code>Decimal</code>	El valor <code>Decimal</code> se redondea al valor <code>Int</code> inferior más cercano.
<code>String</code>	El valor <code>Decimal</code> se redondea al valor <code>Int</code> inferior más cercano. Si la cadena no se puede convertir en un valor <code>Decimal</code> , el resultado es <code>Undefined</code> .
Otro valor	<code>Undefined</code> .

## rtrim(String)

Elimina todos los espacios en blanco del final (pestañas y espacios) del valor `String` proporcionado. Es compatible con la versión SQL 2015-10-8 y versiones posteriores.

Ejemplos:

`rtrim(" h i ") = " h i"`

Tipo de argumento	Resultado
<code>Int</code>	La representación <code>String</code> del valor <code>Int</code> .
<code>Decimal</code>	La representación <code>String</code> del valor <code>Decimal</code> .
<code>Boolean</code>	La representación <code>String</code> del valor booleano ("true" o "false").
<code>Matriz</code>	La representación <code>String</code> del valor <code>Array</code> (mediante reglas de conversión estándar).
<code>Objeto</code>	La representación <code>String</code> del objeto (mediante reglas de conversión estándar).
<code>Null</code>	<code>Undefined</code> .
<code>Sin definir</code>	<code>Undefined</code>

## sign(Decimal)

Devuelve el signo del número especificado. Cuando el signo del argumento es positivo, se devuelve 1. Cuando el signo del argumento es negativo, se devuelve -1. Si el argumento es 0, se devuelve 0. Es compatible con la versión SQL 2015-10-8 y versiones posteriores.

Ejemplos:

`sign(-7) = -1.`

`sign(0) = 0.`

`sign(13) = 1.`

Tipo de argumento	Resultado
Int	Int, el signo del valor Int.
Decimal	Int, el signo del valor Decimal.
String	Int, el signo del valor Decimal. La cadena se convierte en un valor Decimal y se devuelve el signo del valor Decimal. Si el valor String no se puede convertir en un valor Decimal, el resultado es Undefined. Es compatible con la versión SQL 2015-10-8 y versiones posteriores.
Otro valor	Undefined.

## sin(Decimal)

Devuelve el seno de un número en radianes. Los argumentos de tipo Decimal se redondean a un valor de doble precisión antes de aplicar la función. Es compatible con la versión SQL 2015-10-8 y versiones posteriores.

Ejemplo: `sin(0) = 0.0`

Tipo de argumento	Resultado
Int	Decimal (con doble precisión), el seno del argumento.
Decimal	Decimal (con doble precisión), el seno del argumento.
Boolean	Undefined.
String	Decimal (con doble precisión), el seno del argumento. Si la cadena no se puede convertir en un valor Decimal, el resultado es Undefined.
Matriz	Undefined.
Objeto	Undefined.
Null	Undefined.
Undefined	Undefined.

## sinh(Decimal)

Devuelve el seno hiperbólico de un número. Los valores de tipo `Decimal` se redondean a un valor de doble precisión antes de aplicar la función. El resultado es un valor `Decimal` de doble precisión. Es compatible con la versión SQL 2015-10-8 y versiones posteriores.

Ejemplo: `sinh(2.3) = 4.936961805545957`

Tipo de argumento	Resultado
<code>Int</code>	<code>Decimal</code> (con doble precisión); el seno hiperbólico del argumento.
<code>Decimal</code>	<code>Decimal</code> (con doble precisión); el seno hiperbólico del argumento.
<code>Boolean</code>	<code>Undefined</code> .
<code>String</code>	<code>Decimal</code> (con doble precisión); el seno hiperbólico del argumento. Si la cadena no se puede convertir en un valor <code>Decimal</code> , el resultado es <code>Undefined</code> .
<code>Matriz</code>	<code>Undefined</code> .
<code>Objeto</code>	<code>Undefined</code> .
<code>Null</code>	<code>Undefined</code> .
<code>Sin definir</code>	<code>Undefined</code> .

## substring(String, Int [, Int])

Espera un valor `String` seguido de uno o dos valores `Int`. Para un argumento `String` y un único argumento `Int`, esta función devuelve la subcadena del argumento `String` proporcionado que proviene del índice `Int` (de base 0 incluido) suministrado al final del argumento `String`. Para un argumento `String` y dos argumentos `Int`, esta función devuelve la subcadena del argumento `String` proporcionado que proviene del primer argumento de índice `Int` (de base 0 incluido) en el segundo argumento de índice `Int` (de base 0 no incluido). Los índices inferiores a cero se establecen en cero. Los índices superiores a la longitud de `String` se establecen en la longitud de `String`. Para la versión de tres argumentos, si el primer índice es superior (o igual) al segundo índice, el resultado es el valor `String` vacío.

Si los argumentos proporcionados no son (`String, Int`) ni (`String, Int, Int`), se les aplicarán las conversiones estándar para intentar convertirlos en los tipos adecuados. Si no es posible convertirlos, el resultado de la función será `Undefined`. Es compatible con la versión SQL 2015-10-8 y versiones posteriores.

Ejemplos:

```
substring("012345", 0) = "012345".  
substring("012345", 2) = "2345".  
substring("012345", 2.745) = "2345".  
substring(123, 2) = "3".  
substring("012345", -1) = "012345".
```

```
substring(true, 1..2) = "true".
substring(false, -2..411E247) = "false".
substring("012345", 1, 3) = "12".
substring("012345", -50, 50) = "012345".
substring("012345", 3, 1) = "".
```

## sqrt(Decimal)

Devuelve la raíz cuadrada de un número. Los argumentos de tipo `Decimal` se redondean a un valor de doble precisión antes de aplicar la función. Es compatible con la versión SQL 2015-10-8 y versiones posteriores.

Ejemplo: `sqrt(9) = 3.0.`

Tipo de argumento	Resultado
<code>Int</code>	La raíz cuadrada del argumento.
<code>Decimal</code>	La raíz cuadrada del argumento.
<code>Boolean</code>	<code>Undefined.</code>
<code>String</code>	La raíz cuadrada del argumento. Si la cadena no se puede convertir en un valor <code>Decimal</code> , el resultado es <code>Undefined.</code>
<code>Matriz</code>	<code>Undefined.</code>
<code>Objeto</code>	<code>Undefined.</code>
<code>Null</code>	<code>Undefined.</code>
<code>Sin definir</code>	<code>Undefined.</code>

## startswith(String, String)

Devuelve `Boolean`, si el primer argumento de cadena comienza con el segundo argumento de cadena. Si alguno de los argumentos es `Null` o `Undefined`, el resultado es `Undefined`. Es compatible con la versión SQL 2015-10-8 y versiones posteriores.

Ejemplo:

```
startswith("ranger", "ran") = true
```

Tipo de argumento 1	Tipo de argumento 2	Resultado
<code>String</code>	<code>String</code>	Si la primera cadena es igual que la segunda cadena.
Otro valor	Otro valor	Ambos argumentos son de tipo <code>String</code> mediante las cadenas. Indica si la primera cadena es igual que la segunda cadena. Si alguno de los argumentos es <code>Null</code> o <code>Undefined</code> , el resultado es <code>Undefined</code> .

## [tan\(Decimal\)](#)

Devuelve la tangente de un número en radianes. Los valores de tipo `Decimal` se redondean a un valor de doble precisión antes de aplicar la función. Es compatible con la versión SQL 2015-10-8 y versiones posteriores.

Ejemplo: `tan( 3 ) = -0.1425465430742778`

Tipo de argumento	Resultado
<code>Int</code>	<code>Decimal</code> (con doble precisión), la tangente del argumento.
<code>Decimal</code>	<code>Decimal</code> (con doble precisión), la tangente del argumento.
<code>Boolean</code>	<code>Undefined</code> .
<code>String</code>	<code>Decimal</code> (con doble precisión), la tangente del argumento. Si la cadena no se puede convertir en un valor <code>Decimal</code> , el resultado es <code>Undefined</code> .
<code>Matriz</code>	<code>Undefined</code> .
<code>Objeto</code>	<code>Undefined</code> .
<code>Null</code>	<code>Undefined</code> .
<code>Sin definir</code>	<code>Undefined</code> .

## [tanh\(Decimal\)](#)

Devuelve la tangente hiperbólica de un número en radianes. Los valores de tipo `Decimal` se redondean a un valor de doble precisión antes de aplicar la función. Es compatible con la versión SQL 2015-10-8 y versiones posteriores.

Ejemplo: `tanh( 2 . 3 ) = 0.9800963962661914`

Tipo de argumento	Resultado
<code>Int</code>	<code>Decimal</code> (con doble precisión), la tangente hiperbólica del argumento.
<code>Decimal</code>	<code>Decimal</code> (con doble precisión), la tangente hiperbólica del argumento.
<code>Boolean</code>	<code>Undefined</code> .
<code>String</code>	<code>Decimal</code> (con doble precisión), la tangente hiperbólica del argumento. Si la cadena no se puede convertir en un valor <code>Decimal</code> , el resultado es <code>Undefined</code> .
<code>Matriz</code>	<code>Undefined</code> .
<code>Objeto</code>	<code>Undefined</code> .
<code>Null</code>	<code>Undefined</code> .
<code>Sin definir</code>	<code>Undefined</code> .

## timestamp()

Devuelve la marca de hora actual en milisegundos desde las 00:00:00 UTC (hora universal coordinada), jueves 1 de enero de 1970, según lo observado por el motor de reglas de AWS IoT. Es compatible con la versión SQL 2015-10-8 y versiones posteriores.

Ejemplo: `timestamp() = 1481825251155`

## topic(Decimal)

Devuelve el tema al que se ha enviado el mensaje que activó la regla. Si no se especifica ningún parámetro, se devuelve todo el tema. El parámetro `Decimal` se utiliza para especificar un segmento de tema de base 1 concreto. Para el tema `foo/bar/baz`, `topic(1)` devuelve `foo`, `topic(2)` devuelve `bar` y así sucesivamente. Es compatible con la versión SQL 2015-10-8 y versiones posteriores.

Ejemplos:

`topic() = "things/myThings/thingOne"`

`topic(1) = "things"`

## traceid()

Devuelve el ID de seguimiento (UUID) del mensaje MQTT o `Undefined` si el mensaje no se ha enviado por MQTT. Es compatible con la versión SQL 2015-10-8 y versiones posteriores.

Ejemplo:

`traceid() = "12345678-1234-1234-1234-123456789012"`

## trunc(Decimal, Int)

Trunca el primer argumento según el número del valor `Decimal` especificado por el segundo argumento. Si el segundo argumento es inferior a cero, se establecerá en cero. Si el segundo argumento es superior a 34, se establecerá en 34. Los ceros del final se eliminan del resultado. Es compatible con la versión SQL 2015-10-8 y versiones posteriores.

Ejemplos:

`trunc(2.3, 0) = 2.`

`trunc(2.3123, 2 = 2.31.`

`trunc(2.888, 2) = 2.88.`

`(2.00, 5) = 2.`

Tipo de argumento 1	Tipo de argumento 2	Resultado
<code>Int</code>	<code>Int</code>	El valor de origen.
<code>Int/Decimal</code>	<code>Int/Decimal</code>	El primer argumento si el segundo argumento es un valor <code>Int</code> , se redondea

Tipo de argumento 1	Tipo de argumento 2	Resultado
Int/Decimal/String	El primer argumento se trunca en la longitud indicada por el segundo argumento. El segundo argumento, si no es un valor Int, se redondea al valor Int inferior más cercano. Los valores de tipo String se convierten en valores de tipo Decimal. Si se produce un error en la conversión de cadena, el resultado obtenido es Undefined.	
Otro valor	Undefined.	

## trim(String)

Elimina todos los espacios en blanco del principio y del final del valor String proporcionado. Es compatible con la versión SQL 2015-10-8 y versiones posteriores.

Ejemplo:

```
trim(" hi ") = "hi"
```

Tipo de argumento	Resultado
Int	La representación String del valor Int con todos los espacios en blanco de principio y fin suprimidos.
Decimal	La representación String del valor Decimal con todos los espacios en blanco de principio y fin suprimidos.
Boolean	La representación String del valor Boolean ("true" o "false") con todos los espacios en blanco de principio y fin suprimidos.
String	El argumento String con todos los espacios en blanco de principio y fin suprimidos.
Matriz	La representación String del valor Array mediante reglas de conversión estándar.
Objeto	La representación String del objeto mediante reglas de conversión estándar.
Null	Undefined.
Sin definir	Undefined.

## upper(String)

Muestra la versión en mayúsculas del valor String indicado. Los argumentos que no sean String se convierten en String mediante las reglas de conversión estándar. Es compatible con la versión SQL 2015-10-8 y versiones posteriores.

Ejemplos:

```
upper("hello") = "HELLO"  
upper(["hello"]) = ["HELLO"]
```

## Cláusula SELECT

La cláusula SELECT AWS IoT es básicamente la misma que la cláusula SELECT ANSI SQL con algunas diferencias menores.

Puede utilizar la cláusula SELECT para extraer información de los mensajes MQTT de entrada. `SELECT *` se puede utilizar para recuperar toda la carga del mensaje de entrada. Por ejemplo:

```
Incoming payload published on topic 'a/b': {"color":"red", "temperature":50}
SQL statement: SELECT * FROM 'a/b'
Outgoing payload: {"color":"red", "temperature":50}
```

Si la carga es un objeto JSON, puede hacer referencia a claves en el objeto. La carga de salida contendrá el par clave-valor. Por ejemplo:

```
Incoming payload published on topic 'a/b': {"color":"red", "temperature":50}
SQL statement: SELECT color FROM 'a/b'
Outgoing payload: {"color":"red"}
```

Puede utilizar la palabra clave AS para cambiar el nombre de las claves. Por ejemplo:

```
Incoming payload published on topic 'a/b': {"color":"red", "temperature":50}
SQL: SELECT color AS my_color FROM 'a/b'
Outgoing payload: {"my_color":"red"}
```

Puede seleccionar varios elementos separándolos con una coma. Por ejemplo:

```
Incoming payload published on topic 'a/b': {"color":"red", "temperature":50}
SQL: SELECT color as my_color, temperature as farenheit FROM 'a/b'
Outgoing payload: {"my_color":"red", "farenheit":50}
```

Puede seleccionar varios elementos incluido "\*" para agregar elementos a la carga de entrada. Por ejemplo:

```
Incoming payload published on topic 'a/b': {"color":"red", "temperature":50}
SQL: SELECT *, 15 as speed FROM 'a/b'
Outgoing payload: {"color":"red", "temperature":50, speed:15}"
```

Puede utilizar la palabra clave "VALUE" para generar cargas de salida que no sean objetos JSON. Solo puede seleccionar un elemento. Por ejemplo:

```
Incoming payload published on topic 'a/b': {"color":"red", "temperature":50}
SQL: SELECT VALUE color FROM 'a/b'
Outgoing payload: "red"
```

Puede utilizar la sintaxis '.' para explorar objetos JSON anidados en la carga de entrada. Por ejemplo:

```
Incoming payload published on topic 'a/b': {"color":{"red":255,"green":0,"blue":0},
"temperature":50}
SQL: SELECT color.red as red_value FROM 'a/b'
Outgoing payload: {"red_value":255}
```

Puede utilizar funciones (consulte [Funciones \(p. 178\)](#)) para transformar la carga de entrada. Se pueden utilizar paréntesis para agrupar. Por ejemplo:

```
Incoming payload published on topic 'a/b': {"color":"red", "temperature":50}
SQL: SELECT (temperature - 32) * 5 / 9 AS celsius, upper(color) as my_color FROM 'a/b'
```

```
Outgoing payload: {"celsius":10,"my_color":"RED"}
```

## Uso de las cargas binarias

Cuando la carga del mensaje deba tratarse como datos binarios sin procesar (en lugar de como un objeto JSON), puede utilizar el operador \* para hacer referencia a ella en una cláusula SELECT:

Se deben seguir estas reglas si se desea utilizar \* para hacer referencia a la carga del mensaje como datos binarios sin procesar:

1. La instrucción SQL y las plantillas no deben hacer referencia a nombres JSON, salvo a \*.
2. La instrucción SELECT debe incluir \* como único elemento, o debe contener solo funciones, por ejemplo:

```
SELECT * FROM 'a/b'
```

```
SELECT encode(*, 'base64') AS data, timestamp() AS ts FROM 'a/b'
```

### Ejemplos de carga binaria

La cláusula SELECT siguiente se puede utilizar con cargas binarias, ya que no se refiere a nombres JSON.

```
SELECT * FROM 'a/b'
```

La cláusula SELECT siguiente no se puede utilizar con cargas binarias porque hace referencia a device\_type en la cláusula WHERE.

```
SELECT * FROM 'a/b' WHERE device_type = 'thermostat'
```

La cláusula SELECT siguiente no se puede utilizar con cargas binarias, ya que infringe la regla n.º 2.

```
SELECT *, timestamp() AS timestamp FROM 'a/b'
```

La cláusula SELECT siguiente se puede utilizar con cargas binarias, ya que respeta la regla n.º 1 o la regla n.º 2.

```
SELECT * FROM 'a/b' WHERE timestamp() % 12 = 0
```

La regla de AWS IoT siguiente no se puede utilizar con cargas, ya que infringe la regla n.º 1.

```
{
    "sql": "SELECT * FROM 'a/b'",
    "actions": [
        {
            "republish": {
                "topic": "device/${device_id}"
            }
        }
    ]
}
```

## Cláusula FROM

La cláusula FROM suscribe a su regla a un tema o un filtro de tema. Un filtro de tema le permite suscribirse a un grupo de temas similares.

Ejemplo:

Carga de entrada publicada en el tema 'a/b': {temperature: 50}

Carga de entrada publicada en el tema 'a/c': {temperature: 50}

SQL: "SELECT temperature AS t FROM 'a/b'".

La regla está suscrita a 'a/b', por lo que la carga de entrada se transfiere a la regla, mientras que la carga de salida (transferida a las acciones de la regla) es: {t: 50}. La regla no está suscrita a 'a/c', por lo que la regla no se activa para el mensaje publicado en 'a/c'.

Puede utilizar el carácter comodín # para que coincidan todas las subruturas de un filtro de temas:

Ejemplo:

Carga de entrada publicada en el tema 'a/b': {temperature: 50}.

Carga de entrada publicada en el tema 'a/c': {temperature: 60}.

Carga de entrada publicada en el tema 'a/e/f': {temperature: 70}.

Carga de entrada publicada en el tema 'b/x': {temperature: 80}.

SQL: "SELECT temperature AS t FROM 'a/#'".

La regla está suscrita a todos los temas que empiecen por 'a', por lo que se ejecutará tres veces y enviará cargas de salida de {t: 50} (para a/b), {t: 60} (para a/c) y {t: 70} (para a/e/f) a sus acciones. No está suscrita a 'b/x', por lo que la regla no se desencadenará para el mensaje {temperature: 80}.

Puede utilizar el carácter '+' para buscar coincidencias con cualquier elemento de ruta en particular:

Ejemplo:

Carga de entrada publicada en el tema 'a/b': {temperature: 50}.

Carga de entrada publicada en el tema 'a/c': {temperature: 60}.

Carga de entrada publicada en el tema 'a/e/f': {temperature: 70}.

Carga de entrada publicada en el tema 'b/x': {temperature: 80}.

SQL: "SELECT temperature AS t FROM 'a/+'".

La regla está suscrita a todos los temas que tengan dos elementos de ruta donde el primer elemento sea 'a'. La regla se ejecuta para los mensajes enviados a 'a/b' y 'a/c', pero no para los mensajes enviados a 'a/e/f' ni 'b/x'.

Puede utilizar funciones y operadores en la cláusula WHERE. En la cláusula WHERE, no puede hacer referencia a ningún alias creado con la palabra clave AS en la cláusula SELECT. (La cláusula WHERE se evalúa en primer lugar para determinar si la cláusula SELECT debe evaluarse).

## Cláusula WHERE

La cláusula WHERE determina si se evalúa una regla para un mensaje enviado a un tema MQTT al que la regla está suscrita. Si la cláusula WHERE es true, la regla se evalúa, de lo contrario, la regla no se evalúa.

Ejemplo:

Carga de entrada publicada en a/b: {"color": "red", "temperature": 40}.

```
SQL: SELECT color AS my_color FROM 'a/b' WHERE temperature > 50 AND color <> 'red'.
```

En este caso, la regla no se evalúa; no habrá ninguna carga de salida y las acciones de las reglas no se desencadenarán.

Puede utilizar funciones y operadores en la cláusula WHERE. Sin embargo, no puede hacer referencia a ningún alias creado con la palabra clave AS en la cláusula SELECT. (La cláusula WHERE se evalúa en primer lugar para determinar si SELECT debe evaluarse).

## Literales

Puede especificar directamente objetos literales en las cláusulas SELECT y WHERE de su regla SQL, lo que puede ser útil para pasar información.

### Note

Los literales solo están disponibles cuando se utiliza SQL 2016-03-23 o versiones posteriores.

Se utiliza una sintaxis de objeto JSON (pares clave-valor separados con comas, donde las claves son cadenas y los valores son de tipo JSON escritos entre llaves {}). Por ejemplo:

Carga de entrada publicada en el tema a/b: "{lat\_long: [47.606,-122.332]}"

```
Instrucción SQL: SELECT {'latitude': get(lat_long, 0), 'longitude':get(lat_long, 1)}  
as lat_long FROM 'a/b'
```

La carga de salida obtenida sería: {'latitude':47.606, 'longitude':-122.332}.

También puede especificar directamente matrices en las cláusulas SELECT y WHERE de su regla SQL, lo que le permite agrupar información. Se utiliza una sintaxis JSON (elementos separados con comas entre corchetes [] para crear un literal de matriz). Por ejemplo:

Carga de entrada publicada en el tema a/b: {lat: 47.696, long: -122.332}

```
Instrucción SQL: SELECT [lat,long] as lat_long FROM 'a/b'
```

La carga de salida obtenida sería: {"lat\_long": [47.606,-122.332]}.

## Instrucciones case

Las instrucciones case se pueden utilizar para ejecutar bifurcaciones, como una instrucción switch o instrucciones if/else.

Sintaxis:

```
CASE v WHEN t[1] THEN r[1]  
WHEN t[2] THEN r[2] ...  
WHEN t[n] THEN r[n]  
ELSE r[e] END
```

La expresión v se evalúa y se compara con todas las expresiones t[i]. Si se encuentra una coincidencia, la expresión r[i] correspondiente se convierte en el resultado de la instrucción case. Si hay más de una posible coincidencia, se selecciona la primera de ellas. Si no hay ninguna coincidencia, se usa la expresión re de la instrucción else como resultado. Si no hay ninguna coincidencia ni instrucción else, el resultado de la instrucción case es Undefined. Por ejemplo:

Carga de entrada publicada en el tema a/b: {"color":"yellow"}

Instrucción SQL: `SELECT CASE color WHEN 'green' THEN 'go' WHEN 'yellow' THEN 'caution' WHEN 'red' THEN 'stop' ELSE 'you are not at a stop light' END as instructions FROM 'a/b'`

La carga de salida obtenida sería: `{"instructions": "caution"}`.

Las instrucciones case necesitan como mínimo una cláusula WHEN. No se necesita una cláusula ELSE.

Note

Si `v` es `Undefined`, el resultado de la instrucción case es `Undefined`.

## Extensiones JSON

Puede utilizar las extensiones siguientes de la sintaxis ANSI SQL para facilitar el trabajo con objetos JSON anidados.

" . " . "

Este operador tiene acceso a los miembros de los objetos y las funciones JSON integrados, exactamente igual que en ANSI SQL y en JavaScript. Por ejemplo:

```
SELECT foo.bar AS bar.baz FROM 'a/b'
```

\*Operador

Funciona igual que el comodín \* en ANSI SQL. Solo se utiliza en la cláusula SELECT y crea un objeto JSON nuevo que contiene los datos del mensaje. Si la carga del mensaje no tiene el formato JSON, \* devuelve toda la carga del mensaje como bytes sin procesar. Por ejemplo:

```
SELECT * FROM 'a/b'
```

Aplicación de una función a un valor de atributo

A continuación, se muestra un ejemplo de carga JSON que podría publicar un dispositivo:

```
{
    "deviceid" : "iot123",
    "temp" : 54.98,
    "humidity" : 32.43,
    "coords" : {
        "latitude" : 47.615694,
        "longitude" : -122.3359976
    }
}
```

En el ejemplo siguiente se aplica una función a un valor de atributo de una carga JSON:

```
SELECT temp, md5(deviceid) AS hashed_id FROM topic/#
```

El resultado de esta consulta es el objeto JSON siguiente:

```
{
    "temp": 54.98,
    "hashed_id": "e37f81fb397e595c4aeb5645b8cbbbd1"
}
```

## Plantillas de sustitución

Puede utilizar una plantilla de sustitución para aumentar los datos JSON que se devuelven cuando se activa una regla y AWS IoT realiza una acción. La sintaxis de una plantilla de sustitución es  `${expresión}`, donde expresión puede ser cualquier expresión compatible con AWS IoT en las cláusulas SELECT o WHERE. Para obtener más información acerca de las expresiones admitidas, consulte [Referencias de SQL en AWS IoT \(p. 168\)](#).

Las plantillas de sustitución aparecen en la cláusula SELECT dentro de una regla:

```
{  
    "sql": "SELECT *, topic() AS topic FROM 'my/iot/topic'",  
    "ruleDisabled": false,  
    "actions": [  
        {"  
            "republish": {  
                "topic": "${topic()}/republish",  
                "roleArn": "arn:aws:iam::123456789012:role/my-iot-role"  
            }  
        }  
    ]  
}
```

Si esta regla se activa mediante el JSON siguiente:

```
{  
    "deviceid" : "iot123",  
    "temp" : 54.98,  
    "humidity" : 32.43,  
    "coords" : {  
        "latitude" : 47.615694,  
        "longitude" : -122.3359976  
    }  
}
```

A continuación, se muestra la salida de la regla:

```
{  
    "coords":{  
        "longitude": -122.3359976,  
        "latitude": 47.615694  
    },  
    "humidity": 32.43,  
    "temp": 54.98,  
    "deviceid": "iot123",  
    "topic": "my/iot/topic"  
}
```

# Sombras de objeto para AWS IoT

Una sombra de objeto (en ocasiones denominada sombra de dispositivo) es un documento JSON que se utiliza para almacenar y recuperar información sobre el estado actual de un objeto (dispositivo, aplicación, etc.). El servicio Thing Shadows mantiene una sombra de objeto por cada objeto que usted conecte a AWS IoT. Las sombras de objeto se pueden utilizar para obtener y establecer el estado de un objeto sobre MQTT o HTTP, sin tener en cuenta si el objeto está conectado o no a Internet. Cada sombra de objeto se identifica de forma exclusiva por su nombre.

## Contenido

- [Flujo de datos de sombras de objeto \(p. 217\)](#)
- [Documentos de sombras de objeto \(p. 225\)](#)
- [Uso de sombras de objeto \(p. 228\)](#)
- [API RESTful de sombras de objeto \(p. 237\)](#)
- [Temas MQTT de sombras de objeto \(p. 239\)](#)
- [Sintaxis de un documento de sombra de objeto \(p. 246\)](#)
- [Mensajes de error de sombra de objeto \(p. 248\)](#)

## Flujo de datos de sombras de objeto

Los servicios Thing Shadows ejercen de intermediario; permiten a los dispositivos y las aplicaciones recuperar y actualizar las sombras de objeto.

Para ilustrar la forma en que los dispositivos y las aplicaciones se comunican con el servicio Thing Shadows, en esta sección le ayudamos a usar el cliente MQTT de AWS IoT y la CLI de AWS para simular la comunicación entre una bombilla conectada a Internet, una aplicación y el servicio Thing Shadows.

El servicio Thing Shadows utiliza temas MQTT para facilitar la comunicación entre aplicaciones y dispositivos. Para ver cómo funciona, utilice el cliente MQTT de AWS IoT para suscribirse a los temas MQTT siguientes con QoS 1:

\$aws/things/myLightBulb/shadow/update/accepted

El servicio Thing Shadows envía mensajes a este tema cuando se realiza correctamente una actualización en la sombra de objeto.

\$aws/things/myLightBulb/shadow/update/rejected

El servicio Thing Shadows envía mensajes a este tema cuando se rechaza una actualización en la sombra de objeto.

\$aws/things/myLightBulb/shadow/update/delta

El servicio Thing Shadows envía mensajes a este tema cuando se detecta una diferencia entre las secciones notificadas y las secciones deseadas de la sombra de objeto. Para obtener más información, consulte [/update/delta \(p. 242\)](#).

\$aws/things/myLightBulb/shadow/get/accepted

El servicio Thing Shadows envía mensajes a este tema cuando se realiza correctamente una solicitud para la sombra de objeto.

\$aws/things/myLightBulb/shadow/get/rejected

El servicio Thing Shadows envía mensajes a este tema cuando se rechaza una solicitud para la sombra de objeto.

\$aws/things/myLightBulb/shadow/delete/accepted

El servicio Thing Shadows envía mensajes a este tema cuando se elimina la sombra de objeto.

\$aws/things/myLightBulb/shadow/delete/rejected

El servicio Thing Shadows envía mensajes a este tema cuando se rechaza una solicitud para eliminar la sombra de objeto.

\$aws/things/myLightBulb/shadow/update/documents

El servicio Thing Shadows publica un documento de estado en este tema cuando se realiza correctamente una actualización en la sombra de objeto.

Para obtener más información acerca de todos los temas MQTT que utiliza el servicio Thing Shadows, consulte [Temas MQTT de sombras de objeto \(p. 239\)](#).

#### Note

Le recomendamos que se suscriba a los temas .../rejected para ver los errores que envía el servicio Thing Shadows.

Cuando la bombilla está online, se envía su estado actual al servicio Thing Shadows transmitiendo un mensaje MQTT al tema \$aws/things/myLightBulb/shadow/update.

Para simular esto, utilice el cliente MQTT de AWS IoT para publicar el siguiente mensaje en el tema \$aws/things/myLightbulb/shadow/update:

```
{  
    "state": {  
        "reported": {  
            "color": "red"  
        }  
    }  
}
```

Este mensaje establece el color de la bombilla en "rojo".

El servicio Thing Shadows responde enviando el siguiente mensaje al tema \$aws/things/myLightBulb/shadow/update/accepted:

```
{  
    "messageNumber": 4,  
    "payload": {  
        "state": {  
            "reported": {  
                "color": "red"  
            }  
        }  
    }  
}
```

```
        },
        "metadata": {
            "reported": {
                "color": {
                    "timestamp": 1469564492
                }
            }
        },
        "version": 1,
        "timestamp": 1469564492
    },
    "qos": 0,
    "timestamp": 1469564492848,
    "topic": "$aws/things/myLightBulb/shadow/update/accepted"
}
```

Este mensaje indica que el servicio Thing Shadows recibió la solicitud UPDATE y actualizó la sombra de objeto. Si la sombra de objeto no existe, se crea. De lo contrario, la sombra de objeto se actualiza con los datos del mensaje. Si no ve un mensaje publicado en \$aws/things/myLightBulb/shadow/update/accepted, consulte la suscripción a \$aws/things/myLightBulb/shadow/update/rejected para ver los mensajes de error.

Además, el servicio Thing Shadows publica el siguiente mensaje en el tema \$aws/things/myLightBulb/shadow/update/documents.

```
{
    "previous":null,
    "current":{
        "state":{
            "reported": {
                "color": "red"
            }
        },
        "metadata": {
            "reported": {
                "color": {
                    "timestamp": 1483467764
                }
            }
        },
        "version": 1
    },
    "timestamp": 1483467764
}
```

Los mensajes se publican en el tema /update/documents siempre que se realiza correctamente una actualización de una sombra de objeto. Para obtener más información sobre el contenido de los mensajes publicados en este tema, consulte [Temas MQTT de sombras de objeto \(p. 239\)](#).

Una aplicación que interactúa con la bombilla pasa a estar online y solicita el estado actual de la bombilla. La aplicación envía un mensaje vacío al tema \$aws/things/myLightBulb/shadow/get. Para simularlo, utilice el cliente MQTT de AWS IoT para publicar un mensaje vacío ("") en el tema \$aws/things/myLightBulb/shadow/get.

El servicio Thing Shadows responde publicando la sombra de objeto solicitada en el tema \$aws/things/myLightBulb/shadow/get/accepted:

```
{
    "messageNumber": 1,
    "payload": {
```

```
"state": {  
    "reported": {  
        "color": "red"  
    }  
},  
"metadata": {  
    "reported": {  
        "color": {  
            "timestamp": 1469564492  
        }  
    }  
},  
"version": 1,  
"timestamp": 1469564571  
},  
"qos": 0,  
"timestamp": 1469564571533,  
"topic": "$aws/things/myLightBulb/shadow/get/accepted"  
}
```

Si no ve un mensaje en el tema `$aws/things/myLightBulb/shadow/get/accepted`, busque posibles mensajes de error en el tema `$aws/things/myLightBulb/shadow/get/rejected`.

La aplicación muestra esta información para el usuario y este solicita un cambio de color de la bombilla (de rojo a verde). Para ello, la aplicación publica un mensaje en el tema `$aws/things/myLightBulb/shadow/update`:

```
{  
    "state": {  
        "desired": {  
            "color": "green"  
        }  
    }  
}
```

Para simularlo, utilice el cliente MQTT de AWS IoT para publicar el mensaje anterior en el tema `$aws/things/myLightBulb/shadow/update`.

El servicio Thing Shadows responde enviando un mensaje al tema `$aws/things/myLightBulb/shadow/update/accepted`:

```
{  
    "messageNumber": 5,  
    "payload": {  
        "state": {  
            "desired": {  
                "color": "green"  
            }  
        },  
        "metadata": {  
            "desired": {  
                "color": {  
                    "timestamp": 1469564658  
                }  
            }  
        },  
        "version": 2,  
        "timestamp": 1469564658  
    },  
    "qos": 0,  
    "timestamp": 1469564658286,  
    "topic": "$aws/things/myLightBulb/shadow/update/accepted"  
}
```

```
}
```

y al tema `$aws/things/myLightBulb/shadow/update/delta`:

```
{
  "messageNumber": 1,
  "payload": {
    "version": 2,
    "timestamp": 1469564658,
    "state": {
      "color": "green"
    },
    "metadata": {
      "color": {
        "timestamp": 1469564658
      }
    }
  },
  "qos": 0,
  "timestamp": 1469564658309,
  "topic": "$aws/things/myLightBulb/shadow/update/delta"
}
```

El servicio Thing Shadows publica un mensaje en este tema cuando acepta una actualización de la sombra de objeto y, como resultado de esta, dicha sombra contiene diferentes valores para los estados deseados y notificados.

El servicio Thing Shadows publica también un mensaje en el tema `$aws/things/myLightBulb/shadow/update/documents`:

```
{
  "previous": {
    "state": {
      "reported": {
        "color": "red"
      }
    },
    "metadata": {
      "reported": {
        "color": {
          "timestamp": 1483467764
        }
      }
    },
    "version": 1
  },
  "current": {
    "state": {
      "desired": {
        "color": "green"
      },
      "reported": {
        "color": "red"
      }
    },
    "metadata": {
      "desired": {
        "color": {
          "timestamp": 1483468612
        }
      },
      "reported": {
        "color": {
          "timestamp": 1483468612
        }
      }
    }
  }
}
```

```
        "timestamp":1483467764
    }
}
},
"version":2
},
"timestamp":1483468612
}
```

La bombilla está suscrita al tema `$aws/things/myLightBulb/shadow/update/delta`, por lo que recibe el mensaje, cambia de color y publica su nuevo estado. Para simularlo, utilice el cliente MQTT de AWS IoT para publicar el siguiente mensaje en el tema `$aws/things/myLightbulb/shadow/update` y actualizar el estado de la sombra:

```
{
  "state": {
    "reported": {
      "color": "green"
    },
    "desired": null
  }
}
```

Como respuesta, el servicio Thing Shadows envía un mensaje al tema `$aws/things/myLightBulb/shadow/update/accepted`:

```
{
  "messageNumber": 6,
  "payload": {
    "state": {
      "reported": {
        "color": "green"
      },
      "desired": null
    },
    "metadata": {
      "reported": {
        "color": {
          "timestamp": 1469564801
        }
      },
      "desired": {
        "timestamp": 1469564801
      }
    },
    "version": 3,
    "timestamp": 1469564801
  },
  "qos": 0,
  "timestamp": 1469564801673,
  "topic": "$aws/things/myLightBulb/shadow/update/accepted"
}
```

y al tema `$aws/things/myLightBulb/shadow/update/documents`:

```
{
  "previous": {
    "state": {
      "reported": {
        "color": "red"
      }
    },
  }
```

```
"metadata":{  
    "reported":{  
        "color":{  
            "timestamp":1483470355  
        }  
    }  
},  
"version":3  
},  
"current":{  
    "state":{  
        "reported":{  
            "color":"green"  
        }  
    },  
    "metadata":{  
        "reported":{  
            "color":{  
                "timestamp":1483470364  
            }  
        }  
    },  
    "version":4  
},  
"timestamp":1483470364  
}
```

La aplicación solicita el estado actual al servicio Thing Shadows y muestra los datos del estado más reciente. Para simularlo, ejecute el comando siguiente:

```
aws iot-data get-thing-shadow --thing-name "myLightBulb" "output.txt" && cat "output.txt"
```

#### Note

En Windows, omita el `&& cat "output.txt"` que muestra el contenido del archivo output.txt a la consola. Puede abrir el archivo en el Bloc de notas o en cualquier editor de texto para ver el contenido de la sombra de objeto.

El servicio Thing Shadows devuelve el documento de sombra de objeto:

```
{  
    "state":{  
        "reported":{  
            "color":"green"  
        }  
    },  
    "metadata":{  
        "reported":{  
            "color":{  
                "timestamp":1469564801  
            }  
        }  
    },  
    "version":3,  
    "timestamp":1469564864}
```

Para eliminar la sombra de objeto, publique un mensaje vacío en el tema `$aws/things/myLightBulb/shadow/delete`. AWS IoT responderá publicando un mensaje en el tema `$aws/things/myLightBulb/shadow/delete/accepted`:

```
{
```

```
    "version" : 1,  
    "timestamp" : 1488565234  
}
```

## Detección de un objeto conectado

Para determinar si un dispositivo está conectado actualmente, incluya una configuración de conexión en la sombra de objeto y utilice un mensaje MQTT Last Will and Testament (LWT), que establecerá el valor de la conexión en `false` si se desconecta un dispositivo debido a un error.

### Note

En la actualidad, el servicio Shadows de AWS IoT no tiene en cuenta los mensajes LWT enviados a los temas reservados de AWS IoT (temas que comienzan por \$), pero los clientes suscritos y el motor de reglas AWS IoT siguen procesándolos. Para que el servicio Shadows de AWS IoT reciba mensajes LWT, registre un mensaje LWT en un tema no reservado y cree una regla que vuelva a publicar el mensaje en el tema reservado. El siguiente ejemplo muestra cómo crear una regla que escuche mensajes del tema `my/things/myLightBulb/update` y los vuelva a publicar en `$aws/things/myLightBulb/shadow/update`.

```
{
  "rule": {
    "ruleDisabled": false,
    "sql": "SELECT * FROM 'my/things/myLightBulb/update'",
    "description": "Turn my/things/ into $aws/things/",
    "actions": [
      {
        "republish": {
          "topic": "$$aws/things/myLightBulb/shadow/update",
          "roleArn": "arn:aws:iam::123456789012:role/aws_iot_republish"
        }
      }
    ]
  }
}
```

Cuando un dispositivo se conecta, registra un LWT que establece la configuración de conexión en `false`:

```
{
  "state": {
    "reported": {
      "connected": "false"
    }
  }
}
```

También publica un mensaje en su tema de actualización (`$aws/things/myLightBulb/shadow/update`) y establece su estado de conexión en `true`:

```
{
  "state": {
    "reported": {
      "connected": "true"
    }
  }
}
```

Cuando el dispositivo se desconecta correctamente, publica un mensaje en su tema de actualización y establece su estado de conexión en `false`:

```
{  
    "state": {  
        "reported":{  
            "connected":"false"  
        }  
    }  
}
```

Si el dispositivo se desconecta debido a un error, su mensaje LWT se publica automáticamente en el tema de actualización.

## Documentos de sombras de objeto

El servicio Thing Shadows respeta todas las reglas de la especificación JSON. Los valores, objetos y matrices se almacenan en el documento de sombra de objeto.

### Contenido

- [Propiedades del documento \(p. 225\)](#)
- [Control de versiones de una sombra de objeto \(p. 226\)](#)
- [Token de cliente \(p. 226\)](#)
- [Ejemplo de documento \(p. 226\)](#)
- [Secciones vacías \(p. 227\)](#)
- [Matrices \(p. 227\)](#)

## Propiedades del documento

Un documento de sombra de objeto tiene las propiedades siguientes:

#### state

##### desired

El estado deseado del objeto. Las aplicaciones pueden escribir en esta parte del documento para actualizar el estado de un objeto, sin tener que conectarse directamente a un objeto.

##### reported

El estado notificado del objeto. Los objetos escriben en esta parte del documento para informar de su nuevo estado. Las aplicaciones leen esta parte del documento para determinar el estado de un objeto.

#### metadata

Información acerca de los datos almacenados en la sección state del documento. Esto incluye las marcas de tiempo, según la fecha de inicio Unix, para cada atributo de la sección state, lo que le permite determinar cuándo se actualizaron.

#### timestamp

Indica cuándo AWS IoT transmitió el mensaje. Con la marca de tiempo que figura en el mensaje y las marcas de tiempo para los atributos individuales que se muestran en la sección desired o reported, un objeto puede determinar la edad de un elemento actualizado, aunque no disponga de un reloj interno.

#### clientToken

Una cadena única del dispositivo que le permite asociar respuestas a solicitudes en un entorno de MQTT.

#### version

La versión del documento. Cada vez que el documento se actualiza, su número de versión se incrementa. Se utiliza para garantizar que la versión del documento que se actualiza sea la más reciente.

Para obtener más información, consulte [Sintaxis de un documento de sombra de objeto \(p. 246\)](#).

## Control de versiones de una sombra de objeto

El servicio Thing Shadows admite el control de versiones en cada mensaje de actualización (tanto de solicitud como de respuesta). La versión del documento JSON se incrementa con cada actualización de una sombra de objeto. Esto permite garantizar dos cosas:

- Un cliente puede recibir un error si intenta sobrescribir una sombra con una versión más antigua. Se informa al cliente de que debe volver a sincronizarlo para poder actualizar una sombra de objeto.
- Un cliente puede decidir omitir un mensaje recibido si su versión es anterior a la que tiene almacenada.

En algunos casos, un cliente puede omitir la correlación de versiones no presentando una versión.

## Token de cliente

Puede utilizar un token de cliente con mensajes basados en MQTT para verificar que el mismo token de cliente esté contenido en una solicitud y en la respuesta a dicha solicitud. De esta forma se garantiza que la respuesta y la solicitud estén asociadas.

## Ejemplo de documento

A continuación, se muestra un ejemplo de documento de sombra de objeto:

```
{  
    "state" : {  
        "desired" : {  
            "color" : "RED",  
            "sequence" : [ "RED", "GREEN", "BLUE" ]  
        },  
        "reported" : {  
            "color" : "GREEN"  
        }  
    },  
    "metadata" : {  
        "desired" : {  
            "color" : {  
                "timestamp" : 12345  
            },  
            "sequence" : {  
                "timestamp" : 12345  
            }  
        },  
        "reported" : {  
            "color" : {  
                "timestamp" : 12345  
            }  
        }  
    },  
    "version" : 10,  
    "clientToken" : "UniqueClientToken",  
    "timestamp": 123456789
```

```
}
```

## Secciones vacías

Un documento de sombra de objeto contiene una sección `desired` solo si contiene un estado deseado. Por ejemplo, el documento siguiente es un documento de estado válido sin sección `desired`:

```
{
    "reported" : { "temp": 55 }
}
```

La sección `reported` también puede estar vacía:

```
{
    "desired" : { "color" : "RED" }
}
```

Si una actualización produce la nulidad de las secciones `desired` o `reported`, la sección se elimina del documento. Para eliminar la sección `desired` de un documento (como respuesta, por ejemplo, a un dispositivo que actualiza su estado), establezca la sección deseada en `null`:

```
{
    "state": {
        "reported": {
            "color": "red"
        },
        "desired": null
    }
}
```

También es posible que un documento de sombra de objeto no contenga secciones `desired` o `reported`. En tal caso, el documento de sombra estará vacío. Por ejemplo, el documento siguiente es válido:

```
{
}
```

## Matrices

Las sombras de objeto son compatibles con las matrices, pero las tratan como valores normales, en el sentido de que la actualización de una matriz reemplaza toda la matriz. No es posible actualizar parte de una matriz.

Estado inicial:

```
{
    "desired" : { "colors" : [ "RED", "GREEN", "BLUE" ] }
}
```

Update:

```
{
    "desired" : { "colors" : [ "RED" ] }
}
```

Estado final:

```
{  
    "desired" : { "colors" : [ "RED" ] }  
}
```

Las matrices no pueden tener valores nulos. Por ejemplo, la matriz siguiente no es válida y se rechazará.

```
{  
    "desired" : {  
        "colors" : [ null, "RED", "GREEN" ]  
    }  
}
```

## Uso de sombras de objeto

AWS IoT proporciona tres métodos para trabajar con sombras de objeto:

### UPDATE

Crea una sombra de objeto si este no existe o bien la actualiza con los datos suministrados en la solicitud. Los datos se almacenan con la marca de tiempo para indicar la fecha de su última actualización. Se envían mensajes a todos los suscriptores con la diferencia entre el estado `desired` o `reported` (delta). Los objetos o las aplicaciones que reciben un mensaje pueden ejecutar una acción en función de la diferencia entre los estados `desired` o `reported`. Por ejemplo, un dispositivo puede actualizar su estado al estado deseado, o una aplicación puede actualizar su interfaz de usuario para mostrar el cambio de estado del dispositivo.

### GET

Recupera el último estado almacenado en la sombra de objeto (por ejemplo, durante el arranque de un dispositivo para recuperar la configuración y el último estado de funcionamiento). Este método devuelve todo el documento JSON, incluidos los metadatos.

### DELETE

Elimina una sombra de objeto, incluido todo su contenido. Esto elimina el documento JSON del almacén de datos. No se puede restaurar una sombra de objeto eliminada, pero se puede crear otra con el mismo nombre.

## Compatibilidad del protocolo

Estos métodos se admiten con [MQTT](#) y una API RESTful sobre HTTPS. Dado que MQTT es un modelo de comunicación de publicación/suscripción, AWS IoT implementa un conjunto de temas reservados. Los objetos o las aplicaciones se suscriben a estos temas antes de publicar en un tema de solicitud para implementar un comportamiento de solicitud-respuesta. Para obtener más información, consulte [Temas MQTT de sombras de objeto \(p. 239\)](#) y [API RESTful de sombras de objeto \(p. 237\)](#).

## Actualización de una sombra de objeto

Puede actualizar una sombra de objeto mediante la API RESTful [UpdateThingShadow \(p. 238\)](#) o publicando en el tema [/update \(p. 240\)](#). Las actualizaciones afectan únicamente a los campos especificados en la solicitud.

Estado inicial:

```
{  
    "state": {  
        "reported" : {  
            "color" : { "r" :255, "g": 255, "b": 0 }  
        }  
    }  
}
```

Se envía un mensaje de actualización:

```
{  
    "state": {  
        "desired" : {  
            "color" : { "r" : 10 },  
            "engine" : "ON"  
        }  
    }  
}
```

El dispositivo recibe el estado `desired` en el tema `/update/delta` que se desencadena mediante el mensaje `/update` anterior y, a continuación, ejecuta los cambios deseados. Cuando termina, el dispositivo debe confirmar el estado actualizado a través de la sección `reported` del documento JSON de la sombra de objeto.

Estado final:

```
{  
    "state": {  
        "reported" : {  
            "color" : { "r" : 10, "g" : 255, "b": 0 },  
            "engine" : "ON"  
        }  
    }  
}
```

## Recuperación de un documento de sombra de objeto

Puede recuperar una sombra de objeto mediante la API RESTful [GetThingShadow \(p. 237\)](#) o suscribiéndose al tema [/get \(p. 243\)](#) y publicando en él. De este modo, recupera todo el documento y el delta entre los estados `desired` o `reported`.

Ejemplo de documento:

```
{  
    "state": {  
        "desired": {  
            "lights": {  
                "color": "RED"  
            },  
            "engine": "ON"  
        },  
        "reported": {  
            "lights": {  
                "color": "GREEN"  
            },  
            "engine": "ON"  
        },  
        "metadata": {  
            "version": 1  
        }  
    }  
}
```

```
"desired": {  
    "lights": {  
        "color": {  
            "timestamp": 123456  
        },  
        "engine": {  
            "timestamp": 123456  
        }  
    }  
},  
"reported": {  
    "lights": {  
        "color": {  
            "timestamp": 789012  
        }  
    },  
    "engine": {  
        "timestamp": 789012  
    }  
},  
"version": 10,  
"timestamp": 123456789  
}  
}
```

Respuesta:

```
{  
    "state": {  
        "desired": {  
            "lights": {  
                "color": "RED"  
            },  
            "engine": "ON"  
        },  
        "reported": {  
            "lights": {  
                "color": "GREEN"  
            },  
            "engine": "ON"  
        },  
        "delta": {  
            "lights": {  
                "color": "RED"  
            }  
        }  
    },  
    "metadata": {  
        "desired": {  
            "lights": {  
                "color": {  
                    "timestamp": 123456  
                }  
            },  
            "engine": {  
                "timestamp": 123456  
            }  
        },  
        "reported": {  
            "lights": {  
                "color": {  
                    "timestamp": 789012  
                }  
            },  
            "engine": {  
                "timestamp": 789012  
            }  
        }  
    }  
}
```

```
        "timestamp": 789012
    }
},
"delta": {
    "lights": {
        "color": {
            "timestamp": 123456
        }
    }
}
},
"version": 10,
"timestamp": 123456789
}
```

## Bloqueo optimista

Puede utilizar la versión del documento de estado para asegurarse de que actualiza la versión más reciente de un documento de sombra de objeto. Cuando se suministra una versión con una solicitud de actualización, el servicio rechaza la solicitud con un código de respuesta de conflicto HTTP 409 si la versión actual del documento de estado no coincide con la versión suministrada.

Por ejemplo:

Documento inicial:

```
{
    "state" : {
        "desired" : { "colors" : [ "RED", "GREEN", "BLUE" ] }
    },
    "version" : 10
}
```

Actualización: (la versión no coincide; se rechazará la solicitud)

```
{
    "state": {
        "desired": {
            "colors": [
                "BLUE"
            ]
        }
    },
    "version": 9
}
```

Resultado:

```
409 Conflict
```

Actualización: (la versión coincide; esta solicitud se aceptará)

```
{
    "state": {
        "desired": {
            "colors": [
                "BLUE"
            ]
        }
    }
}
```

```
    },
    "version": 10
}
```

Estado final:

```
{
  "state": {
    "desired": {
      "colors": [
        "BLUE"
      ]
    }
  },
  "version": 11
}
```

## Eliminación de datos

Puede eliminar datos de una sombra de objeto publicando en el tema [/update \(p. 240\)](#) y definiendo los campos que se van a suprimir con el valor nulo. Todo campo que tenga el valor null se eliminará del documento.

Estado inicial:

```
{
  "state": {
    "desired" : {
      "lights": { "color": "RED" },
      "engine" : "ON"
    },
    "reported" : {
      "lights" : { "color": "GREEN" },
      "engine" : "OFF"
    }
  }
}
```

Se envía un mensaje de actualización:

```
{
  "state": {
    "desired": null,
    "reported": {
      "engine": null
    }
  }
}
```

Estado final:

```
{
  "state": {
    "reported" : {
      "lights" : { "color" : "GREEN" }
    }
  }
}
```

Puede eliminar todos los datos de una sombra de objeto estableciendo su estado en el valor `null`. Por ejemplo, si envía el mensaje siguiente, se eliminarán todos los datos de estado, pero la sombra de objeto permanecerá.

```
{  
    "state": null  
}
```

La sombra de objeto sigue existiendo, aunque su estado sea `null`. La versión de la sombra de objeto se incrementará cuando se produzca la siguiente actualización.

## Eliminación de una sombra de objeto

Puede eliminar un documento de sombra de objeto mediante la API RESTful

[DeleteThingShadow \(p. 239\)](#) o publicando en el tema [/delete \(p. 244\)](#).

### Note

La eliminación de una sombra de objeto no elimina el objeto y, por otra parte, eliminar el objeto no elimina la sombra de objeto.

Estado inicial:

```
{  
    "state": {  
        "desired" : {  
            "lights": { "color": "RED" },  
            "engine" : "ON"  
        },  
        "reported" : {  
            "lights" : { "color": "GREEN" },  
            "engine" : "OFF"  
        }  
    }  
}
```

Se publica un mensaje vacío en el tema `/delete`.

Estado final:

```
HTTP 404 - resource not found
```

## Estado delta

El estado delta es un tipo de estado virtual que contiene la diferencia entre los estados `desired` y `reported`. Los campos de la sección `desired` que no están incluidos en la sección `reported` se incluyen en el delta. Los campos que están en la sección `reported` y no están en la sección `desired` no se incluyen en el delta. El delta contiene metadatos, y sus valores son iguales a los metadatos del campo `desired`. Por ejemplo:

```
{  
    "state": {  
        "desired": {  
            "color": "RED",  
            "state": "STOP"  
        },  
        "reported": {  
            "color": "GREEN",  
            "state": "GO"  
        }  
    }  
}
```

```

        "engine": "ON"
    },
    "delta": {
        "color": "RED",
        "state": "STOP"
    }
},
"metadata": {
    "desired": {
        "color": {
            "timestamp": 12345
        },
        "state": {
            "timestamp": 12345
        },
        "reported": {
            "color": {
                "timestamp": 12345
            },
            "engine": {
                "timestamp": 12345
            }
        },
        "delta": {
            "color": {
                "timestamp": 12345
            },
            "state": {
                "timestamp": 12345
            }
        }
    },
    "version": 17,
    "timestamp": 123456789
}
}

```

Cuando los objetos anidados difieren, el delta contiene la ruta de acceso a la raíz.

```

{
    "state": {
        "desired": {
            "lights": {
                "color": {
                    "r": 255,
                    "g": 255,
                    "b": 255
                }
            }
        },
        "reported": {
            "lights": {
                "color": {
                    "r": 255,
                    "g": 0,
                    "b": 255
                }
            }
        }
    },
    "delta": {
        "lights": {
            "color": {
                "g": 255
            }
        }
    }
}

```

```
        },
    "version": 18,
    "timestamp": 123456789
}
```

El servicio Thing Shadows calcula el delta iterando por cada campo del estado `desired` y comparándolo con el estado `reported`.

Las matrices se tratan como valores. Si una matriz de la sección `desired` no coincide con la matriz de la sección `reported`, toda la matriz deseada se copia en el delta.

## Observación de los cambios de estado

Cuando se actualiza una sombra de objeto, los mensajes se publican en dos temas MQTT:

- \$aws/things/*nombre de objeto*/shadow/update/accepted
- \$aws/things/*nombre de objeto*/shadow/update/delta

El mensaje enviado al tema `update/delta` está destinado al objeto cuyo estado se actualiza. Este mensaje contiene únicamente la diferencia entre las secciones `desired` y `reported` del documento de sombra de objeto. Tras recibir este mensaje, el objeto decide si realizará el cambio solicitado. Si el estado del objeto cambia, publica su nuevo estado actual en el tema `$aws/things/thing-name/shadow/update`.

Los dispositivos y aplicaciones pueden suscribirse a uno de estos temas para recibir una notificación cuando el estado del documento cambie.

A continuación, se muestra un ejemplo de este flujo:

1. El dispositivo informa del estado.
2. El sistema actualiza el documento de estado en su almacén de datos persistentes.
3. El sistema publica un mensaje delta que contiene únicamente el delta y está dirigido a los dispositivos suscritos. Los dispositivos deben suscribirse a este tema para recibir actualizaciones.
4. La sombra de objeto publica un mensaje aceptado que contiene todo el documento recibido, incluidos los metadatos. Las aplicaciones deben suscribirse a este tema para recibir actualizaciones.

## Orden de los mensajes

No se garantiza que los mensajes generados por el servicio de AWS IoT lleguen al dispositivo siguiendo un orden específico.

Documento de estado inicial:

```
{
  "state" : {
    "reported" : { "color" : "blue" }
  },
  "version" : 10,
  "timestamp": 123456777
}
```

Actualización 1:

```
{
```

```
{  
    "state": { "desired" : { "color" : "RED" } },  
    "version": 10,  
    "timestamp": 123456777  
}
```

Actualización 2:

```
{  
    "state": { "desired" : { "color" : "GREEN" } },  
    "version": 11,  
    "timestamp": 123456778  
}
```

Documento de estado final:

```
{  
    "state": {  
        "reported": { "color" : "GREEN" }  
    },  
    "version": 12,  
    "timestamp": 123456779  
}
```

Se obtienen dos mensajes delta:

```
{  
    "state": {  
        "color": "RED"  
    },  
    "version": 11,  
    "timestamp": 123456778  
}
```

```
{  
    "state": { "color" : "GREEN" },  
    "version": 12,  
    "timestamp": 123456779  
}
```

El dispositivo puede recibir estos mensajes de forma desordenada. Dado que el estado de estos mensajes es acumulable, un dispositivo puede descartar con toda seguridad todos los mensajes cuyo número de versión sea anterior a la del mensaje del cual se hace un seguimiento. Si el dispositivo recibe el delta de la versión 12 antes que el de la versión 11, puede descartar sin problemas el mensaje de la versión 11.

## Supresión de mensajes de sombras de objeto

Para reducir el tamaño de los mensajes de sombra de objeto a su dispositivo, defina una regla que seleccione solo los campos que su dispositivo necesite y vuelva a publicar el mensaje en un tema MQTT al que su dispositivo escuche.

La regla se especifica en JSON y debe tener el aspecto siguiente:

```
{  
    "sql": "SELECT state, version FROM '$aws/things/+/shadow/update/delta'",  
    "ruleDisabled": false,  
    "actions": [{  
        "republish": {
```

```
        "topic": "${topic(2)}/delta",
        "roleArn": "arn:aws:iam::123456789012:role/my-iot-role"
    }]
}
```

La instrucción SELECT determina qué campos del mensaje se volverán a publicar en el tema especificado. Se usa un comodín "+" para correlacionar todos los nombres de sombra de objeto. La regla especifica que todos los mensajes coincidentes deben volver a publicarse en el tema especificado. En tal caso, la función "topic()" se utiliza para especificar el tema en el que se vuelve a publicar. topic(2) toma el valor del nombre de objeto del tema original. Para obtener más información sobre la creación de reglas, consulte [Reglas](#).

## API RESTful de sombras de objeto

Una sombra de objeto expone el siguiente URI para actualizar la información de estado:

```
https://endpoint/things/thingName/shadow
```

El punto de enlace específico con su cuenta de AWS. Para recuperar su punto de enlace, ejecute el comando [describe-endpoint](#). El formato del punto de enlace es el siguiente:

```
identifier.iot.region.amazonaws.com
```

### Acciones de API

- [GetThingShadow \(p. 237\)](#)
- [UpdateThingShadow \(p. 238\)](#)
- [DeleteThingShadow \(p. 239\)](#)

## GetThingShadow

Obtiene la sombra de objeto del objeto especificado.

El documento de estado de respuesta incluye el delta entre los estados `desired` y `reported`.

### Solicitud

La solicitud incluye los encabezados HTTP estándar y el URI siguiente:

```
HTTP GET https://endpoint/things/thingName/shadow
```

### Respuesta

En caso de éxito, la respuesta incluye encabezados HTTP estándar, así como el código y el cuerpo siguientes:

```
HTTP 200
BODY: response state document
```

Para obtener más información, consulte [Ejemplo de documento de estado de respuesta \(p. 246\)](#).

### Autorización

Para recuperar una sombra de objeto se necesita una política que permita al intermediario ejecutar la acción `iot:GetThingShadow`. El servicio Thing Shadows acepta dos formas de autenticación: Signature Version 4 con credenciales de IAM o autenticación mutua TLS con un certificado de cliente.

A continuación, se muestra una política de ejemplo que permite a un intermediario recuperar una sombra de objeto:

```
{  
    "Version": "2012-10-17",  
    "Statement": [{  
        "Effect": "Allow",  
        "Action": "iot:GetThingShadow",  
        "Resource": ["arn:aws:iot:region:account:thing/thing"]  
    }]  
}
```

## UpdateThingShadow

Actualiza la sombra del objeto especificado.

Las actualizaciones solo afectan a los campos especificados en el documento de estado de la solicitud. Todos los campos que tengan un valor `null` se eliminarán de la sombra de objeto.

### Solicitud

La solicitud incluye los encabezados HTTP estándar, así como el URI y el cuerpo siguientes:

```
HTTP POST https://endpoint/things/thingName/shadow  
BODY: request state document
```

Para obtener más información, consulte [Ejemplo de documento de estado de solicitud \(p. 246\)](#).

### Respuesta

En caso de éxito, la respuesta incluye encabezados HTTP estándar, así como el código y el cuerpo siguientes:

```
HTTP 200  
BODY: response state document
```

Para obtener más información, consulte [Ejemplo de documento de estado de respuesta \(p. 246\)](#).

### Autorización

Para actualizar una sombra de objeto se necesita una política que permita al intermediario ejecutar la acción `iot:UpdateThingShadow`. El servicio Thing Shadows acepta dos formas de autenticación: Signature Version 4 con credenciales de IAM o autenticación mutua TLS con un certificado de cliente.

A continuación, se muestra una política de ejemplo que permite a un intermediario actualizar una sombra de objeto:

```
{  
    "Version": "2012-10-17",  
    "Statement": [{  
        "Effect": "Allow",  
        "Action": "iot:UpdateThingShadow",  
        "Resource": ["arn:aws:iot:region:account:thing/thing"]  
    }]
```

}

## DeleteThingShadow

Elimina la sombra de objeto del objeto especificado.

### Solicitud

La solicitud incluye los encabezados HTTP estándar y el URI siguiente:

```
HTTP DELETE https://endpoint/things/thingName/shadow
```

### Respuesta

En caso de éxito, la respuesta incluye encabezados HTTP estándar, así como el código y el cuerpo siguientes:

```
HTTP 200
BODY: Empty response state document
```

### Autorización

Para eliminar una sombra de objeto se necesita una política que permita al intermediario ejecutar la acción `iot:DeleteThingShadow`. El servicio Thing Shadows acepta dos formas de autenticación: Signature Version 4 con credenciales de IAM o autenticación mutua TLS con un certificado de cliente.

A continuación, se muestra una política de ejemplo que permite a un intermediario eliminar una sombra de objeto:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iot:DeleteThingShadow",
            "Resource": ["arn:aws:iot:region:account:thing/thing"]
        }
    ]
}
```

## Temas MQTT de sombras de objeto

El servicio Thing Shadows utiliza temas MQTT reservados para permitir a las aplicaciones y los objetos obtener, actualizar o eliminar la información de estado de un objeto (sombra de objeto). Los nombres de estos temas comienzan por `$aws/things/nombre de objeto/shadow`. Para suscribirse a temas de sombra de objeto y publicar en ellos se necesita una autorización basada en los temas. AWS IoT se reserva el derecho a agregar nuevos temas a la estructura de temas existente. Por este motivo, le recomendamos que evite las suscripciones de tipo comodín a los temas de sombra. Por ejemplo, evite suscribirse a filtros de temas como `$aws/things/thingName/shadow/#`, ya que el número de temas que coinciden con este filtro de temas puede aumentar a medida que AWS IoT incorpora nuevos temas de sombra. Para consultar ejemplos de mensajes publicados en estos temas vaya a [Flujo de datos de sombras de objeto \(p. 217\)](#).

A continuación, mostramos temas MQTT utilizados para interactuar con sombras de objeto.

### Temas

- [/update \(p. 240\)](#)
- [/update/accepted \(p. 241\)](#)
- [/update/documents \(p. 241\)](#)
- [/update/rejected \(p. 242\)](#)
- [/update/delta \(p. 242\)](#)
- [/get \(p. 243\)](#)
- [/get/accepted \(p. 243\)](#)
- [/get/rejected \(p. 244\)](#)
- [/delete \(p. 244\)](#)
- [/delete/accepted \(p. 245\)](#)
- [/delete/rejected \(p. 245\)](#)

## /update

Publique un documento de estado de solicitud en este tema para actualizar la sombra de objeto:

```
$aws/things/thingName/shadow/update
```

Un cliente que intente actualizar el estado de un objeto debe enviar un documento de estado de solicitud JSON como el siguiente:

```
{  
    "state" : {  
        "desired" : {  
            "color" : "red",  
            "power" : "on"  
        }  
    }  
}
```

Un objeto que actualice su sombra de objeto enviará un documento de estado de solicitud JSON como el siguiente:

```
{  
    "state" : {  
        "reported" : {  
            "color" : "red",  
            "power" : "on"  
        }  
    }  
}
```

AWS IoT responde publicando en [/update/accepted \(p. 241\)](#) o en [/update/rejected \(p. 242\)](#).

Para obtener más información, consulte [Documentos de estado de la solicitud \(p. 246\)](#).

## Ejemplo de política

A continuación, mostramos un ejemplo de la política requerida:

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

```
        "Effect": "Allow",
        "Action": ["iot:Publish"],
        "Resource": ["arn:aws:iot:region:account:topic/$aws/things/thingName/shadow/
update"]
    }]
}
```

## /update/accepted

AWS IoT publica un documento de estado de respuesta en este tema cuando acepta un cambio de la sombra de objeto:

```
$aws/things/thingName/shadow/update/accepted
```

Para obtener más información, consulte [Documentos de estado de la respuesta \(p. 246\)](#).

### Ejemplo de política

A continuación, mostramos un ejemplo de la política requerida:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe",
                "iot:Receive"
            ],
            "Resource": ["arn:aws:iot:region:account:topicfilter/$aws/things/thingName/shadow/
update/accepted"]
        }
    ]
}
```

## /update/documents

AWS IoT publica un documento de estado en este tema siempre que se realiza una actualización correcta de la sombra:

```
$aws/things/thingName/shadow/update/documents
```

El documento JSON contiene dos nodos principales: `previous` y `current`. El nodo `previous` incluye el contenido del documento de sombra completo antes de realizar la actualización, mientras que `current` contiene el documento de sombra completo después de realizar la actualización sin errores. Cuando se actualiza (se crea) la sombra de objeto por primera vez, el nodo `previous` contiene `null`.

### Ejemplo de política

A continuación, mostramos un ejemplo de la política requerida:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe",
                "iot:Receive"
            ]
        }
    ]
}
```

```
        ],
        "Resource": [ "arn:aws:iot:region:account:topicfilter/$aws/things/thingName/shadow/
update/documents" ]
    }
}
```

## /update/rejected

AWS IoT publica un documento de estado de error en este tema cuando rechaza un cambio de la sombra de objeto:

```
$aws/things/thingName/shadow/update/rejected
```

Para obtener más información, consulte [Documentos de respuesta de error \(p. 247\)](#).

### Ejemplo de política

A continuación, mostramos un ejemplo de la política requerida:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe",
                "iot:Receive"
            ],
            "Resource": [ "arn:aws:iot:region:account:topicfilter/$aws/things/thingName/shadow/
update/rejected" ]
        }
    ]
}
```

## /update/delta

AWS IoT publica un documento de estado de respuesta en este tema cuando acepta un cambio de la sombra de objeto y el documento de estado de solicitud contiene diferentes valores para los estados `desired` y `reported`:

```
$aws/things/thingName/shadow/update/delta
```

Para obtener más información, consulte [Documentos de estado de la respuesta \(p. 246\)](#).

### Detalles de la publicación

- Un mensaje publicado en `update/delta` incluye únicamente los atributos deseados que difieren entre las secciones `desired` y `reported`. Contiene todos estos atributos, independientemente de si se encuentran en el mensaje de actualización actual o si ya estaban almacenados en AWS IoT. No se incluyen los atributos que no difieren entre las secciones `desired` y `reported`.
- Si un atributo se encuentra en la sección `reported`, pero no tiene equivalente en la sección `desired`, no se incluye.
- Si un atributo se encuentra en la sección `desired`, pero no tiene equivalente en la sección `reported`, se incluye.
- Si se elimina un atributo de la sección `reported`, pero sigue existiendo en la sección `desired`, se incluye.

## Ejemplo de política

A continuación, mostramos un ejemplo de la política requerida:

```
{  
    "Version": "2012-10-17",  
    "Statement": [{  
        "Effect": "Allow",  
        "Action": [  
            "iot:Subscribe",  
            "iot:Receive"  
        ],  
        "Resource": ["arn:aws:iot:region:account:topicfilter/$aws/things/thingName/shadow/  
update/delta"]  
    }]  
}
```

## /get

Publique un mensaje vacío en este tema para obtener la sombra de objeto:

```
$aws/things/thingName/shadow/get
```

AWS IoT responde publicando en [/get/accepted](#) (p. 243) o en [/get/rejected](#) (p. 244).

## Ejemplo de política

A continuación, mostramos un ejemplo de la política requerida:

```
{  
    "Version": "2012-10-17",  
    "Statement": [{  
        "Effect": "Allow",  
        "Action": [  
            "iot:Publish"  
        ],  
        "Resource": ["arn:aws:iot:region:account:topic/$aws/things/thingName/shadow/get"]  
    }]  
}
```

## /get/accepted

AWS IoT publica un documento de estado de respuesta en este tema cuando devuelve la sombra de objeto:

```
$aws/things/thingName/shadow/get/accepted
```

Para obtener más información, consulte [Documentos de estado de la respuesta](#) (p. 246).

## Ejemplo de política

A continuación, mostramos un ejemplo de la política requerida:

```
{
```

```
"Version": "2012-10-17",
"Statement": [{
    "Effect": "Allow",
    "Action": [
        "iot:Subscribe",
        "iot:Receive"
    ],
    "Resource": ["arn:aws:iot:region:account:topicfilter/$aws/things/thingName/shadow/get/accepted"]
}]}
```

## /get/rejected

AWS IoT publica un documento de respuesta de error en este tema cuando no puede devolver la sombra de objeto:

```
$aws/things/thingName/shadow/get/rejected
```

Para obtener más información, consulte [Documentos de respuesta de error \(p. 247\)](#).

## Ejemplo de política

A continuación, mostramos un ejemplo de la política requerida:

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Action": [
            "iot:Subscribe",
            "iot:Receive"
        ],
        "Resource": ["arn:aws:iot:region:account:topicfilter/$aws/things/thingName/shadow/get/rejected"]
    }]
}
```

## /delete

Para eliminar una sombra de objeto, publique un mensaje vacío para eliminar el tema:

```
$aws/things/thingName/shadow/delete
```

El contenido del mensaje no se tiene en cuenta.

AWS IoT responde publicando en [/delete/accepted \(p. 245\)](#) o en [/delete/rejected \(p. 245\)](#).

## Ejemplo de política

A continuación, mostramos un ejemplo de la política requerida:

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

```

```
        "iot:Subscribe",
        "iot:Receive"
    ],
    "Resource": ["arn:aws:iot:region:account:topic filter/$aws/things/thingName/shadow/
delete"]
}
}
```

## /delete/accepted

AWS IoT publica un mensaje en este tema cuando se elimina una sombra de objeto:

```
$aws/things/thingName/shadow/delete/accepted
```

### Ejemplo de política

A continuación, mostramos un ejemplo de la política requerida:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe",
                "iot:Receive"
            ],
            "Resource": ["arn:aws:iot:region:account:topicfilter/$aws/things/thingName/shadow/
delete/accepted"]
        }
    ]
}
```

## /delete/rejected

AWS IoT publica un documento de respuesta de error en este tema cuando no puede eliminar la sombra de objeto:

```
$aws/things/thingName/shadow/delete/rejected
```

Para obtener más información, consulte [Documentos de respuesta de error \(p. 247\)](#).

### Ejemplo de política

A continuación, mostramos un ejemplo de la política requerida:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe",
                "iot:Receive"
            ],
            "Resource": ["arn:aws:iot:region:account:topicfilter/$aws/things/thingName/shadow/
delete/rejected"]
        }
    ]
}
```

# Sintaxis de un documento de sombra de objeto

El servicio Thing Shadows utiliza los siguientes documentos en operaciones UPDATE, GET, DELETE mediante la [API RESTful](#) (p. 237) o mensajes de publicación/suscripción MQTT (p. 239). Para obtener más información, consulte [Documentos de sombras de objeto](#) (p. 225).

## Ejemplos

- [Documentos de estado de la solicitud](#) (p. 246)
- [Documentos de estado de la respuesta](#) (p. 246)
- [Documentos de respuesta de error](#) (p. 247)

## Documentos de estado de la solicitud

Los documentos de estado de la solicitud tienen el formato siguiente:

```
{  
    "state": {  
        "desired": {  
            "attribute1": integer2,  
            "attribute2": "string2",  
            ...  
            "attributeN": boolean2  
        },  
        "reported": {  
            "attribute1": integer1,  
            "attribute2": "string1",  
            ...  
            "attributeN": boolean1  
        }  
    },  
    "clientToken": "token",  
    "version": version  
}
```

- **state** - Las actualizaciones solo afectan a los campos especificados.
- **clientToken** - Si se utiliza, puede comprobar que la solicitud y la respuesta contengan el mismo token de cliente.
- **version** - Si se utiliza, el servicio Thing Shadows procesa la actualización solo si la versión especificada coincide con la versión más reciente que tiene.

## Documentos de estado de la respuesta

Los documentos de estado de la respuesta tienen el formato siguiente:

```
{  
    "state": {  
        "desired": {  
            "attribute1": integer2,  
            "attribute2": "string2",  
            ...  
            "attributeN": boolean2  
        },  
        "reported": {  
            "attribute1": integer1,  
            "attribute2": "string1",  
            ...  
            "attributeN": boolean1  
        }  
    },  
    "clientToken": "token",  
    "version": version  
}
```

```
    ...
    "attributeN": boolean1
},
"delta": {
    "attribute3": integerX,
    "attribute5": stringY
}
},
"metadata": {
    "desired": {
        "attribute1": {
            "timestamp": timestamp
        },
        "attribute2": {
            "timestamp": timestamp
        },
        ...
        "attributeN": {
            "timestamp": timestamp
        }
    },
    "reported": {
        "attribute1": {
            "timestamp": timestamp
        },
        "attribute2": {
            "timestamp": timestamp
        },
        ...
        "attributeN": {
            "timestamp": timestamp
        }
    }
},
"timestamp": timestamp,
"clientToken": "token",
"version": version
}
```

- state
  - reported- Solo está presente si un objeto ha notificado datos en la sección `reported` y contiene únicamente los campos que se encontraban en el documento de estado de la solicitud.
  - desired- Solo está presente si un objeto ha notificado datos en la sección `desired` y contiene únicamente los campos que se encontraban en el documento de estado de la solicitud.
- metadata- Contiene las marcas de tiempo de cada atributo de las secciones `desired` y `reported` para que pueda determinar cuándo se actualizó el estado.
- timestamp- Fecha y hora de inicio en que AWS IoT generó la respuesta.
- clientToken- Solo está presente si se ha utilizando un token de cliente al publicar un JSON válido en el tema `/update`.
- version- La versión actual del documento de la sombra de objeto compartido en AWS IoT. Se aumenta en una unidad con relación a la versión anterior del documento.

## Documentos de respuesta de error

Los documentos de respuesta de error tienen el formato siguiente:

```
{
    "code": error-code,
    "message": "error-message",
```

```
    "timestamp": timestamp,  
    "clientToken": "token"  
}
```

- **code**- Código de respuesta HTTP que indica el tipo de error.
- **message**- Mensaje de texto que proporciona información adicional.
- **timestamp**- Fecha y hora en que AWS IoT generó la respuesta.
- **clientToken**- Solo está presente si se ha utilizando un token de cliente al publicar un JSON válido en el tema /update.

Para obtener más información, consulte [Mensajes de error de sombra de objeto \(p. 248\)](#).

## Mensajes de error de sombra de objeto

El servicio Thing Shadows publica un mensaje en el tema de errores (a través de MQTT) si se produce un error al intentar cambiar el documento de estado. Este mensaje solo se genera como respuesta a una solicitud de publicación en uno de los temas \$aws reservados. Si el cliente actualiza el documento mediante la API REST, recibe el código de error HTTP como parte de su respuesta y no se genera un mensaje de error MQTT.

Código de error HTTP	Mensajes de error
400 (solicitud errónea)	<ul style="list-style-type: none"><li>• JSON no válido</li><li>• Falta un nodo necesario: estado</li><li>• El nodo de estado debe ser un objeto</li><li>• El nodo deseado debe ser un objeto</li><li>• El nodo notificado debe ser un objeto</li><li>• Versión no válida</li><li>• ClientToken no válido</li><li>• JSON contiene demasiados niveles de anidamiento; el máximo permitido es 6</li><li>• El estado contiene un nodo no válido</li></ul>
401 (sin autorización)	<ul style="list-style-type: none"><li>• Sin autorización</li></ul>
403 (prohibido)	<ul style="list-style-type: none"><li>• prohibido</li></ul>
404 (no encontrado)	<ul style="list-style-type: none"><li>• Objeto no encontrado</li></ul>
409 (conflicto)	<ul style="list-style-type: none"><li>• Conflicto de versiones</li></ul>
413 (carga demasiado grande)	<ul style="list-style-type: none"><li>• La carga supera el tamaño máximo permitido</li></ul>
415 (tipo de medio incompatible)	<ul style="list-style-type: none"><li>• Codificación documentada incompatible, se admite la codificación UTF-8</li></ul>
429 (demasiadas solicitudes)	<ul style="list-style-type: none"><li>• El servicio Thing Shadows generará este mensaje de error cuando haya más de diez solicitudes en tránsito.</li></ul>
500 (error de servidor interno)	<ul style="list-style-type: none"><li>• Error de servicio interno</li></ul>

# SDK de AWS IoT

## Contenido

- [SDK de AWS Mobile para Android \(p. 249\)](#)
- [Arduino Yún SDK \(p. 249\)](#)
- [AWS IoT Device SDK para Embedded C \(p. 250\)](#)
- [AWS Mobile SDK para iOS \(p. 250\)](#)
- [AWS IoT Device SDK para Java \(p. 250\)](#)
- [AWS IoT Device SDK para JavaScript \(p. 250\)](#)
- [AWS IoT Device SDK para Python \(p. 251\)](#)

Los SDK de dispositivos de AWS IoT le ayudan a conectar con facilidad y rapidez sus dispositivos a AWS IoT. Los SDK de dispositivos de AWS IoT contienen bibliotecas de código abierto, guías de desarrolladores con ejemplos y guías de migración para que pueda crear productos o soluciones de IoT innovadores en las plataformas de hardware deseadas.

## SDK de AWS Mobile para Android

AWS SDK para Android contiene una biblioteca, ejemplos y documentación para que los desarrolladores creen aplicaciones móviles conectadas mediante AWS. Este SDK también permite llamar a las API de AWS IoT. Para obtener más información, consulte los siguientes temas:

- [AWS Mobile SDK for Android on GitHub](#)
- [AWS Mobile SDK for Android Readme](#)
- [AWS Mobile SDK for Android Samples](#)

## Arduino Yún SDK

AWS IoT Arduino Yún SDK permite a los desarrolladores conectar sus tarjetas compatibles Arduino Yún con AWS IoT. Al conectar un dispositivo a AWS IoT, los usuarios pueden trabajar de forma segura con el agente de mensajes, las reglas y las sombras de objeto que AWS IoT proporciona, y con otros servicios de AWS como AWS Lambda, Kinesis y Amazon S3. Para obtener más información, consulte los siguientes temas:

- [Arduino Yún SDK on GitHub](#)
- [Arduino Yún SDK Readme](#)

## AWS IoT Device SDK para Embedded C

AWS IoT Device SDK para Embedded C es un conjunto de archivos de origen C, que se puede utilizar en aplicaciones insertadas para establecer conexiones seguras con la plataforma de AWS IoT. Contiene clientes de transporte, implementaciones TLS y ejemplos de uso. También es compatible con características específicas de AWS IoT como una API para tener acceso al servicio Thing Shadows. Se distribuye como código fuente y está diseñado para integrarse en el firmware cliente junto con código de aplicación, otras bibliotecas y RTOS. Para obtener más información, consulte los siguientes temas:

- [AWS IoT Device SDK for Embedded C GitHub](#)
- [AWS IoT Device SDK for Embedded C Readme](#)
- [AWS IoT Device SDK for Embedded C Porting Guide](#)

## AWS Mobile SDK para iOS

AWS SDK para iOS es un kit de desarrollo de software de código abierto, distribuido con licencia de Apache Open Source. El SDK para iOS contiene una biblioteca, ejemplos de código y documentación para ayudar a los desarrolladores a crear aplicaciones móviles conectadas mediante AWS. Este SDK también permite llamar a la API de AWS IoT.

- [AWS SDK for iOS on GitHub](#)
- [AWS SDK for iOS Readme](#)
- [AWS SDK for iOS Samples](#)

## AWS IoT Device SDK para Java

AWS IoT Device SDK para Java permite a los desarrolladores de Java tener acceso a la plataforma de AWS IoT mediante MQTT o MQTT sobre protocolo WebSocket. El SDK es compatible con la sombra de objeto de AWS IoT. Puede tener acceso a las sombras de objeto mediante métodos de HTTP, incluidos GET, UPDATE y DELETE. El SDK es también compatible con un modelo de acceso de sombra de objeto simplificado, lo que permite a los desarrolladores intercambiar datos con sombras de objeto utilizando únicamente métodos getter y setter, sin tener que serializar o deserializar documentos JSON. Para obtener más información, consulte los siguientes temas:

- [AWS IoT Device SDK for Java on GitHub](#)
- [AWS IoT Device SDK for Java readme](#)

## AWS IoT Device SDK para JavaScript

El paquete aws-iot-device-sdk.js permite a los desarrolladores escribir aplicaciones JavaScript que tengan acceso a AWS IoT mediante MQTT o MQTT sobre protocolo WebSocket. Se puede utilizar en entornos de Node.js y aplicaciones de navegador. Para obtener más información, consulte los siguientes temas:

- [AWS IoT Device SDK for JavaScript on GitHub](#)
- [AWS IoT Device SDK for JavaScript readme](#)

## AWS IoT Device SDK para Python

AWS IoT Device SDK para Python permite a los desarrolladores escribir scripts de Python para tener acceso con sus dispositivos a la plataforma de AWS IoT mediante MQTT o MQTT sobre protocolo WebSocket. Al conectar sus dispositivos a AWS IoT, los usuarios pueden trabajar de forma segura con el agente de mensajes, las reglas y las sombras de objeto que AWS IoT proporciona, y con otros servicios de AWS como AWS Lambda, Kinesis, Amazon S3 y mucho más.

- [AWS IoT Device SDK for Python on GitHub](#)
- [AWS IoT Device SDK for Python readme](#)

# Monitorización de AWS IoT

La monitorización es una parte importante del mantenimiento de la fiabilidad, la disponibilidad y el desempeño de AWS IoT 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 un error que se produce en distintas partes del código, en caso de que ocurra. Antes de empezar a monitorizar AWS IoT, debe crear un plan de monitorización 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 monitorización?
- ¿Quién debería recibir una notificación cuando surjan problemas?

El siguiente paso consiste en establecer un punto de referencia del desempeño normal de AWS IoT en su entorno. Para ello, se mide el desempeño en distintos momentos y bajo distintas condiciones de carga. A medida que monitoree AWS IoT, almacene los datos de monitorización históricos para que pueda compararlos con los datos de desempeño actual, identificar los patrones de desempeño normal y las anomalías en el desempeño, así como desarrollar métodos para la resolución de problemas.

Por ejemplo, si está utilizando Amazon EC2, puede supervisar el uso de la CPU, la E/S de disco y el uso de la red en relación con las instancias. Si el desempeño no alcanza los valores del punto de referencia establecido, es posible que deba volver a configurar u optimizar la instancia para reducir la utilización de la CPU, mejorar la E/S de disco o reducir el tráfico de red.

Para establecer un punto de referencia debe, como mínimo, monitorizar los elementos siguientes:

- PublishIn.Success
- PublishOut.Success
- Subscribe.Success
- Ping.Success
- Connect.Success
- GetThingShadow.Accepted
- UpdateThingShadow.Accepted

- DeleteThingShadow.Accepted
- RulesExecuted

#### Temas

- [Herramientas de monitorización \(p. 253\)](#)
- [Monitorización con Amazon CloudWatch \(p. 254\)](#)
- [Registro de llamadas a la API en AWS IoT con AWS CloudTrail \(p. 261\)](#)

## Herramientas de monitorización

AWS proporciona varias herramientas que puede utilizar para monitorizar AWS IoT. Puede configurar algunas de estas herramientas para que monitoricen por usted, pero otras herramientas requieren intervención manual. Le recomendamos que automatice las tareas de monitorización en la medida de lo posible.

### Herramientas de monitorización automatizadas

Puede utilizar las siguientes herramientas de monitorización automatizada para monitorizar AWS IoT e informar cuando haya algún problema:

- Amazon CloudWatch Alarms – Watch a single metric over a time period that you specify, and perform one or more actions based on the value of the metric relative to a given threshold over a number of time periods. The action is a notification sent to an Amazon Simple Notification Service (Amazon SNS) topic or Auto Scaling policy. CloudWatch alarms do not invoke actions simply because they are in a particular state; the state must have changed and been maintained for a specified number of periods. For more information, see [Monitorización con Amazon CloudWatch \(p. 254\)](#).
- Amazon CloudWatch Logs – Monitor, store, and access your log files from AWS CloudTrail or other sources. For more information, see [Monitoring Log Files](#) in the Guía del usuario de Amazon CloudWatch.
- Amazon CloudWatch Events – Match events and route them to one or more target functions or streams to make changes, capture state information, and take corrective action. For more information, see [What is Amazon CloudWatch Events](#) in the Guía del usuario de Amazon CloudWatch.
- AWS CloudTrail Log Monitoring – Share log files between accounts, monitor CloudTrail log files in real time by sending them to CloudWatch Logs, write log processing applications in Java, and validate that your log files have not changed after delivery by CloudTrail. For more information, see [Working with CloudTrail Log Files](#) in the AWS CloudTrail User Guide.

### Herramientas de monitorización manual

Otra parte importante de la monitorización de AWS IoT implica la monitorización manual de los elementos que no cubren las alarmas de CloudWatch. AWS IoT, CloudWatch y otros paneles de la consola de AWS proporcionan una vista rápida del entorno de AWS. Le recomendamos que también compruebe los archivos de log de AWS IoT.

- El panel de AWS IoT muestra:
  - Certificados de CA
  - Certificados
  - Políticas
  - Reglas
  - Objetos
- La página de inicio de CloudWatch muestra:

- Alarmas y estado actual
- Gráficos de alarmas y recursos
- Estado de los servicios

Además, puede utilizar CloudWatch para hacer lo siguiente:

- Crear [paneles personalizados](#) para monitorizar los servicios que le interesan.
- Realizar un gráfico con los datos de las métricas para resolver problemas y descubrir tendencias
- Buscar y examinar todas sus métricas de recursos de AWS.
- Crear y editar las alarmas de notificación de problemas

## Monitorización con Amazon CloudWatch

Puede monitorizar AWS IoT mediante CloudWatch, que recopila y procesa los datos sin formato de AWS IoT en métricas legibles y casi en tiempo real. Estas estadísticas se registran durante un periodo de dos semanas, de forma que pueda acceder a información histórica y obtener una mejor perspectiva sobre el rendimiento de su aplicación web o servicio. De forma predeterminada, los datos de las métricas de AWS IoT se envían automáticamente a CloudWatch en períodos de 1 minuto. Para obtener más información, consulte [qué son Amazon CloudWatch, Amazon CloudWatch Events y Amazon CloudWatch Logs](#) en la Guía del usuario de Amazon CloudWatch.

### Temas

- [Métricas y dimensiones de AWS IoT \(p. 254\)](#)
- [¿Cómo utilizo las métricas de AWS IoT? \(p. 258\)](#)
- [Creación de alarmas de CloudWatch para monitorizar AWS IoT \(p. 259\)](#)

## Métricas y dimensiones de AWS IoT

Cuando interactúa con AWS IoT, este envía las siguientes métricas y dimensiones a CloudWatch todos los minutos. Puede seguir los siguientes procedimientos para ver las métricas de AWS IoT.

Para consultar las métricas mediante la consola de CloudWatch

Las métricas se agrupan en primer lugar por el espacio de nombres de servicio y, a continuación, por las diversas combinaciones de dimensiones dentro de cada espacio de nombres.

1. Abra la consola de CloudWatch en <https://console.aws.amazon.com/cloudwatch/>.
2. En el panel de navegación, seleccione Metrics.
3. En el panel CloudWatch Metrics by Category, en la categoría de métricas de AWS IoT, seleccione una categoría de métricas y, a continuación, en el panel superior, desplácese hasta ver la lista completa de métricas.

Para ver métricas mediante la CLI de AWS

- En el símbolo del sistema, ejecute el siguiente comando:

```
aws cloudwatch list-metrics --namespace "AWS/IoT"
```

CloudWatch muestra las siguientes métricas para AWS IoT:

## AWS IoT Metrics

AWS IoT sends the following metrics to CloudWatch once per received request.

### IoT Metrics

Metric	Description
RulesExecuted	The number of AWS IoT rules executed.

### Rule Metrics

Metric	Description
TopicMatch	The number of incoming messages published on a topic on which a rule is listening. The <code>RuleName</code> dimension contains the name of the rule.
ParseError	The number of JSON parse errors that occurred in messages published on a topic on which a rule is listening. The <code>RuleName</code> dimension contains the name of the rule.

### Rule Action Metrics

Metric	Description
Success	The number of successful rule action invocations. The <code>RuleName</code> dimension contains the name of the rule that specifies the action. The <code>ActionType</code> dimension contains the type of action that was invoked.
Failure	The number of failed rule action invocations. The <code>RuleName</code> dimension contains the name of the rule that specifies the action. The <code>RuleName</code> dimension contains the name of the rule that specifies the action. The <code>ActionType</code> dimension contains the type of action that was invoked.

### Message Broker Metrics

Metric	Description
Connect.AuthError	The number of connection requests that could not be authorized by the message broker. The <code>Protocol</code> dimension contains the protocol used to send the CONNECT message.
Connect.ClientError	The number of connection requests rejected because the MQTT message did not meet the requirements defined in <a href="#">AWS IoT Limits</a> . The <code>Protocol</code> dimension contains the protocol used to send the CONNECT message.
Connect.ServerError	The number of connection requests that failed because an internal error occurred. The <code>Protocol</code> dimension

Metric	Description
	contains the protocol used to send the CONNECT message.
Connect.Success	The number of successful connections to the message broker. The <code>Protocol</code> dimension contains the protocol used to send the CONNECT message.
Connect.Throttle	The number of connection requests that were throttled because the client exceeded the allowed connect request rate. The <code>Protocol</code> dimension contains the protocol used to send the CONNECT message.
Ping.Success	The number of ping messages received by the message broker. The <code>Protocol</code> dimension contains the protocol used to send the ping message.
PublishIn.AuthError	The number of publish requests the message broker was unable to authorize. The <code>Protocol</code> dimension contains the protocol used to publish the message.
PublishIn.ClientError	The number of publish requests rejected by the message broker because the message did not meet the requirements defined in <a href="#">AWS IoT Limits</a> . The <code>Protocol</code> dimension contains the protocol used to publish the message.
PublishIn.ServerError	The number of publish requests the message broker failed to process because an internal error occurred. The <code>Protocol</code> dimension contains the protocol used to send the PUBLISH message.
PublishIn.Success	The number of publish requests successfully processed by the message broker. The <code>Protocol</code> dimension contains the protocol used to send the PUBLISH message.
PublishIn.Throttle	The number of publish request that were throttled because the client exceeded the allowed inbound message rate. The <code>Protocol</code> dimension contains the protocol used to send the PUBLISH message.
PublishOut.AuthError	The number of publish requests made by the message broker that could not be authorized by AWS IoT. The <code>Protocol</code> dimension contains the protocol used to send the PUBLISH message.
PublishOut.ClientError	The number of publish requests made by the message broker that were rejected because the message did not meet the requirements defined in <a href="#">AWS IoT Limits</a> . The <code>Protocol</code> dimension contains the protocol used to send the PUBLISH message.
PublishOut.Success	The number of publish requests successfully made by the message broker. The <code>Protocol</code> dimension contains the protocol used to send the PUBLISH message.

Metric	Description
Subscribe.AuthError	The number of subscription requests made by a client that could not be authorized. The <code>Protocol</code> dimension contains the protocol used to send the <code>SUBSCRIBE</code> message.
Subscribe.ClientError	The number of subscribe requests that were rejected because the <code>SUBSCRIBE</code> message did not meet the requirements defined in <a href="#">AWS IoT Limits</a> . The <code>Protocol</code> dimension contains the protocol used to send the <code>SUBSCRIBE</code> message.
Subscribe.ServerError	The number of subscribe requests that were rejected because an internal error occurred. The <code>Protocol</code> dimension contains the protocol used to send the <code>SUBSCRIBE</code> message.
Subscribe.Success	The number of subscribe requests that were successfully processed by the message broker. The <code>Protocol</code> dimension contains the protocol used to send the <code>SUBSCRIBE</code> message.
Subscribe.Throttle	The number of subscribe requests that were throttled because the client exceeded the allowed subscribe request rate. The <code>Protocol</code> dimension contains the protocol used to send the <code>SUBSCRIBE</code> message.
Unsubscribe.ClientError	The number of unsubscribe requests that were rejected because the <code>UNSUBSCRIBE</code> message did not meet the requirements defined in <a href="#">AWS IoT Limits</a> . The <code>Protocol</code> dimension contains the protocol used to send the <code>UNSUBSCRIBE</code> message.
Unsubscribe.ServerError	The number of unsubscribe requests that were rejected because an internal error occurred. The <code>Protocol</code> dimension contains the protocol used to send the <code>UNSUBSCRIBE</code> message.
Unsubscribe.Success	The number of unsubscribe requests that were successfully processed by the message broker. The <code>Protocol</code> dimension contains the protocol used to send the <code>UNSUBSCRIBE</code> message.
Unsubscribe.Throttle	The number of unsubscribe requests that were rejected because the client exceeded the allowed unsubscribe request rate. The <code>Protocol</code> dimension contains the protocol used to send the <code>UNSUBSCRIBE</code> message.

#### Note

The message broker metrics are displayed in the AWS IoT console under Protocol Metrics.

### Thing Shadow Metrics

Metric	Description
DeleteThingShadow.Accepted	The number of DeleteThingShadow requests processed successfully. The <code>Protocol</code> dimension contains the protocol used to make the request.
GetThingShadow.Accepted	The number of GetThingShadow requests processed successfully. The <code>Protocol</code> dimension contains the protocol used to make the request.
UpdateThingShadow.Accepted	The number of UpdateThingShadow requests processed successfully. The <code>Protocol</code> dimension contains the protocol used to make the request.

#### Note

The thing shadow metrics are displayed in the AWS IoT console under Protocol Metrics.

### Dimensions for Metrics

Metrics use the namespace and provide metrics for the following dimension(s):

Dimension	Description
ActionType	The <a href="#">action type</a> specified by the rule that triggered by the request.
Protocol	The protocol used to make the request. Valid values are: MQTT or HTTP
RuleName	The name of the rule triggered by the request.

## ¿Cómo utilizo las métricas de AWS IoT?

Las métricas mostradas por AWS IoT proporcionan información que puede analizar de diferentes maneras. Los siguientes casos de uso se basan en una situación en la que tiene diez objetos que se conectan a Internet una vez al día. Cada día:

- Diez objetos se conectan con AWS IoT casi al mismo tiempo.
- Cada objeto se suscribe a un filtro de temas y, a continuación, espera una hora antes de desconectarse. Durante este período, los objetos se comunican entre sí y obtienen más información sobre el estado del mundo.
- Cada objeto publica alguna percepción que tenga según los datos que acaba de encontrar utilizando `UpdateThingShadow`.
- Cada objeto se desconecta de AWS IoT.

Se trata de sugerencias que puede usar como punto de partida y no de una lista completa.

- [¿Cómo se me puede enviar una notificación si mis objetos no se conectan correctamente cada día? \(p. 259\)](#)
- [¿Cómo se me puede enviar una notificación si mis objetos no publican datos cada día? \(p. 260\)](#)

- ¿Cómo se me puede enviar una notificación si las actualizaciones de la sombra de mi objeto se rechazan cada día? (p. 260)

## Creación de alarmas de CloudWatch para monitorizar AWS IoT

Puede crear una alarma de CloudWatch que envíe un mensaje de Amazon SNS cuando la alarma cambie de estado. Una alarma vigila una única métrica durante el período especificado y realiza una o varias acciones según el valor de la métrica relativo a un determinado umbral durante varios períodos de tiempo. La acción es una notificación que se envía a un tema de Amazon SNS o a una política de Auto Scaling. Las alarmas invocan acciones únicamente para los cambios de estado prolongados. Las alarmas de CloudWatch no invocan acciones simplemente por tener un estado determinado. Es necesario que el estado haya cambiado y se mantenga durante un número especificado de períodos.

### ¿Cómo se me puede enviar una notificación si mis objetos no se conectan correctamente cada día?

1. Cree un tema de Amazon SNS, arn:aws:sns:us-east-1:123456789012:things-not-connecting-successfully.

Para obtener más información, consulte [Configuración de Amazon Simple Notification Service](#).

2. Cree la alarma.

```
Prompt>aws cloudwatch put-metric-alarm \
    --alarm-name ConnectSuccessAlarm \
    --alarm-description "Alarm when my Things don't connect successfully" \
    --namespace AWS/IoT \
    --metric-name Connect.Success \
    --dimensions Name=Protocol,Value=MQTT \
    --statistic Sum \
    --threshold 10 \
    --comparison-operator LessThanThreshold \
    --period 86400 \
    --unit Count \
    --evaluation-periods 1 \
    --alarm-actions arn:aws:sns:us-east-1:1234567890:things-not-connecting-successfully
```

```
Prompt>aws cloudwatch put-metric-alarm \
    --alarm-name ConnectSuccessAlarm \
    --alarm-description "Alarm when my Things don't connect successfully" \
    --namespace AWS/IoT \
    --metric-name Connect.Success \
    --dimensions Name=Protocol,Value=MQTT \
    --statistic Sum \
    --threshold 10 \
    --comparison-operator LessThanThreshold \
    --period 86400 \
    --unit Count \
    --evaluation-periods 1 \
    --alarm-actions arn:aws:sns:us-east-1:1234567890:things-not-connecting-successfully
```

3. Pruebe la alarma.

```
Prompt>aws cloudwatch set-alarm-state --alarm-name ConnectSuccessAlarm --state-reason
"initializing" --state-value OK
```

```
Prompt>aws cloudwatch set-alarm-state --alarm-name ConnectSuccessAlarm --state-reason  
"initializing" --state-value ALARM
```

## ¿Cómo se me puede enviar una notificación si mis objetos no publican datos cada día?

1. Cree un tema de Amazon SNS, arn:aws:sns:us-east-1:123456789012:things-not-publishing-data.

Para obtener más información, consulte [Configuración de Amazon Simple Notification Service](#).

2. Cree la alarma.

```
Prompt>aws cloudwatch put-metric-alarm \  
    --alarm-name PublishInSuccessAlarm\  
    --alarm-description "Alarm when my Things don't publish their data \  
    --namespace AWS/IoT \  
    --metric-name PublishIn.Success \  
    --dimensions Name=Protocol,Value=MQTT \  
    --statistic Sum \  
    --threshold 10 \  
    --comparison-operator LessThanThreshold \  
    --period 86400 \  
    --unit Count \  
    --evaluation-periods 1 \  
    --alarm-actions arn:aws:sns:us-east-1:1234567890:things-not-publishing-data
```

3. Pruebe la alarma.

```
Prompt>aws cloudwatch set-alarm-state --alarm-name PublishInSuccessAlarm --state-reason  
"initializing" --state-value OK
```

```
Prompt>aws cloudwatch set-alarm-state --alarm-name PublishInSuccessAlarm --state-reason  
"initializing" --state-value ALARM
```

## ¿Cómo se me puede enviar una notificación si las actualizaciones de la sombra de mi objeto se rechazan cada día?

1. Crear un tema de Amazon SNS, arn:aws:sns:us-east-1:1234567890:things-shadow-updates-rejected.

Para obtener más información, consulte [Configuración de Amazon Simple Notification Service](#).

2. Cree la alarma.

```
Prompt>aws cloudwatch put-metric-alarm \  
    --alarm-name UpdateThingShadowSuccessAlarm \  
    --alarm-description "Alarm when my Things Shadow updates are getting rejected" \  
    --namespace AWS/IoT \  
    --metric-name UpdateThingShadow.Success \  
    --dimensions Name=Protocol,Value=MQTT \  
    --statistic Sum \  
    --threshold 10 \  
    --comparison-operator LessThanThreshold \  
    --period 86400 \  
    --unit Count \  
    --alarm-actions arn:aws:sns:us-east-1:1234567890:things-shadow-updates-rejected
```

```
--evaluation-periods 1 \
--alarm-actions arn:aws:sns:us-east-1:1234567890:things-shadow-updates-rejected
```

3. Pruebe la alarma.

```
Prompt>aws cloudwatch set-alarm-state --alarm-name UpdateThingShadowSuccessAlarm --
state-reason "initializing" --state-value OK
```

```
Prompt>aws cloudwatch set-alarm-state --alarm-name UpdateThingShadowSuccessAlarm --
state-reason "initializing" --state-value ALARM
```

## Registro de llamadas a la API en AWS IoT con AWS CloudTrail

AWS IoT está integrado con CloudTrail, un servicio que captura todas las llamadas a la API de AWS IoT y envía los archivos de log a un bucket de Amazon S3 especificado. CloudTrail captura las llamadas a la API desde la consola de AWS IoT o desde su código a las API de AWS IoT. Mediante la información recopilada por CloudTrail puede determinar la solicitud que se realizó a AWS IoT, la dirección IP de origen desde la que se realizó la solicitud, quién realizó la solicitud, cuándo la realizó, etcétera.

Para obtener más información sobre CloudTrail, incluido cómo configurarlo y habilitarlo, consulte la [AWS CloudTrail User Guide](#).

### Información de AWS IoT en CloudTrail

Cuando el registro de CloudTrail está habilitado en su cuenta de AWS, las llamadas a la API realizadas en acciones de AWS IoT se escriben en los archivos de log de CloudTrail junto con otros registros del servicio de AWS. CloudTrail determina cuándo crear y escribir en un nuevo archivo en función del período de tiempo y del tamaño del archivo.

Todas las acciones de AWS IoT se registran por parte de CloudTrail y se documentan en la [referencia de la API de AWS IoT](#). Por ejemplo, las llamadas a las secciones CreateThing, ListThings y ListTopicRules generan entradas en los archivos de log de CloudTrail.

Cada entrada de log contiene información sobre quién generó la solicitud. La información de identidad del usuario en la entrada de log le ayuda a determinar lo siguiente:

- Si la solicitud se realizó con las credenciales del nodo raíz o del usuario de 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 servicio de AWS.

Para obtener más información, consulte el [elemento userIdentity de CloudTrail](#).

Puede almacenar los archivos log en su bucket de Amazon S3 durante todo el tiempo que desee, pero también puede definir reglas de ciclo de vida de Amazon S3 para archivar o eliminar archivos log automáticamente. De forma predeterminada, los archivos log se cifran con cifrado de servidor de Amazon S3 (SSE).

Si desea recibir notificaciones sobre la entrega de archivos log, puede configurar CloudTrail para que publique las notificaciones de Amazon SNS cuando se envíen nuevos archivos log. Para obtener más información, consulte [Configuring Amazon SNS Notifications for CloudTrail](#).

También puede agregar archivos de log de AWS IoT de varias regiones de AWS y de varias cuentas de AWS en un solo bucket de Amazon S3.

Para obtener más información, consulte [Receiving CloudTrail Log Files from Multiple Regions](#) y [Receiving CloudTrail Log Files from Multiple Accounts](#).

## Entradas de archivos de log de AWS IoT

Los archivos log de CloudTrail puede contener una o varias entradas de log. Cada entrada muestra varios eventos con formato JSON. Una entrada de log representa una única solicitud de cualquier origen e incluye información sobre la acción solicitada, la fecha y la hora de la acción, los parámetros de la solicitud, etcétera. Las entradas de log no son un rastro de la pila ordenada de las llamadas API públicas, por lo que no aparecen en ningún orden específico.

En el ejemplo siguiente se muestra una entrada de registro de CloudTrail que ilustra la acción `AttachPrincipalPolicy`.

```
{  
    "timestamp": "1460159496",  
    "AdditionalEventData": "",  
    "Annotation": "",  
    "ApiVersion": "",  
    "ErrorCode": "",  
    "ErrorMessage": "",  
    "EventID": "8bfff4fed-c229-4d2d-8264-4ab28a487505",  
    "EventName": "AttachPrincipalPolicy",  
    "EventTime": "2016-04-08T23:51:36Z",  
    "EventType": "AwsApiCall",  
    "ReadOnly": "",  
    "RecipientAccountList": "",  
    "RequestId": "d4875df2-fde4-11e5-b829-23bf9b56cbcd",  
    "RequestParamters": {  
        "principal": "arn:aws:iot:us-  
east-1:123456789012:cert/528ce36e8047f6a75ee51ab7beddb4eb268ad41d2ea881a10b67e8e76924d894",  
        "policyName": "ExamplePolicyForIoT"  
    },  
    "Resources": "",  
    "ResponseElements": "",  
    "SourceIpAddress": "52.90.213.26",  
    "UserAgent": "aws-internal/3",  
    "UserIdentity": {  
        "type": "AssumedRole",  
        "principalId": "AKIAI44QH8DHBEXAMPLE",  
        "arn": "arn:aws:sts::12345678912:assumed-role/iotmonitor-us-east-1-beta-  
InstanceRole-1C5T1YCYMHPYT/i-35d0a4b6",  
        "accountId": "222222222222",  
        "accessKeyId": "access-key-id",  
        "sessionContext": {  
            "attributes": {  
                "mfaAuthenticated": "false",  
                "creationDate": "Fri Apr 08 23:51:10 UTC 2016"  
            },  
            "sessionIssuer": {  
                "type": "Role",  
                "principalId": "AKIAI44QH8DHBEXAMPLE",  
                "arn": "arn:aws:iam::123456789012:role/executionServiceEC2Role/iotmonitor-  
us-east-1-beta-InstanceRole-1C5T1YCYMHPYT",  
                "accountId": "222222222222",  
                "userName": "iotmonitor-us-east-1-InstanceRole-1C5T1YCYMHPYT"  
            }  
        },  
        "invokedBy": {  
    }}
```

```
        "serviceAccountId": "111111111111"
    },
    "VpcEndpointId": ""
}
```

# Solución de problemas de AWS IoT

La siguiente información puede ayudarle a solucionar problemas habituales en AWS IoT.

## Tareas

- [Diagnóstico de problemas de conectividad \(p. 264\)](#)
- [Configuración de CloudWatch Logs \(p. 265\)](#)
- [Diagnóstico de problemas de las reglas \(p. 270\)](#)
- [Diagnóstico de problemas en las sombras de objeto \(p. 270\)](#)
- [Diagnosticar problemas con acciones del flujo de entrada de Salesforce IoT \(p. 271\)](#)

## Diagnóstico de problemas de conectividad

### Autenticación

¿Cómo autentican mis dispositivos los puntos de enlace de AWS IoT?

Agregue el certificado de autoridad de certificación (CA) de AWS IoT al almacén de confianza de su cliente. Puede descargar el certificado de CA desde [aquí](#).

¿Cómo puedo validar un certificado configurado correctamente?

Ejecute el comando `s_client` de OpenSSL para probar una conexión con el punto de enlace de AWS IoT:

```
openssl s_client -connect custom_endpoint.iot.us-east-1.amazonaws.com:8443 -  
CAfile CA.pem -cert cert.pem -key privateKey.pem
```

### Autorización

He recibido una respuesta PUBNACK o SUBNACK del agente. ¿Qué tengo que hacer?

Asegúrese de que el certificado que utilice para llamar a AWS IoT tenga una política asociada. De forma predeterminada, todas las operaciones de publicación o suscripción se deniegan.

# Configuración de CloudWatch Logs

Como los mensajes de sus dispositivos pasan por el agente de mensajes y el motor de reglas, AWS IoT envía eventos de progresión sobre cada mensaje. Puede inscribirse para ver estos eventos en CloudWatch Logs. Para obtener más información, consulte [CloudWatch Logs](#).

## Note

Antes de habilitar el registro de AWS IoT, asegúrese de comprender bien los permisos de acceso a CloudWatch Logs en su cuenta de AWS. Los usuarios con acceso a CloudWatch Logs podrán consultar la información de depuración de sus dispositivos.

## Configuración de un rol de IAM para registro

Utilice la consola de IAM para crear un rol de registro.

### Creación de un rol de IAM de registro

Los documentos siguientes proporcionan la política de confianza y la política de roles que permiten a AWS IoT enviar logs a CloudWatch en su nombre.

Política de roles:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "logs:CreateLogGroup",  
                "logs:CreateLogStream",  
                "logs:PutLogEvents",  
                "logs:PutMetricFilter",  
                "logs:PutRetentionPolicy"  
            ],  
            "Resource": [  
                "*"  
            ]  
        }  
    ]  
}
```

Política de confianza:

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

## Registro del rol de registro en AWS IoT

Utilice la consola de AWS IoT o el comando de CLI siguiente para registrar el rol de registro en AWS IoT.

```
aws iot setLoggingOptions --logging-options-payload  
roleArn="arn:aws:iam::<your-aws-account-num>:role/  
IoTLoggingRole",logLevel="INFO"
```

El nivel de log puede ser DEBUG, INFO, ERROR o DISABLED:

- DEBUG ofrece la información más detallada de la actividad de AWS IoT.
- INFO proporciona un resumen de la mayoría de acciones. Esto es suficiente para la mayoría de los usuarios.
- ERROR proporciona únicamente casos de error.
- DISABLED elimina el registro en su conjunto, aunque mantiene intacto el rol de registro.

## Formato de entrada de los logs de CloudWatch

Cada entrada de log tiene la información siguiente:

Evento

Describe las acciones que tienen lugar en AWS IoT.

LogLevel

El nivel de registro. Puede ser DEBUG, INFO, ERROR, WARN o DISABLED.

TimeStamp

La hora en que se generó el log.

Traceld

Identificador generado de forma aleatoria para una solicitud de entrada. Se puede utilizar para filtrar todos los logs correspondientes a un mensaje de entrada.

PrincipalId

Huella de certificado o nombre de objeto, en función del punto de enlace (MQTT o HTTP) que ha recibido la solicitud de un dispositivo.

Según el contexto del mensaje, los campos siguientes también pueden incluirse en los mensajes de log:

Topic Name

El nombre de tema MQTT que se agrega a una entrada al recibir un mensaje MQTT de publicación o de suscripción.

ClientId

El ID del cliente que envió un mensaje MQTT.

ThingName

El nombre de objeto agregado a una entrada cuando se envía una solicitud a un punto de enlace HTTP para actualizar o suprimir el estado de un objeto.

RuleId

El identificador de la regla que contiene el ID de una regla cuando esta se activa.

## Log Level

El nivel de log especifica los tipos de logs que se generarán.

### ERROR

Cualquier error que provoque el fracaso de una operación.

Los logs solo incluirán información de ERROR.

### WARN

Todo lo que pueda llegar a producir incoherencias en el sistema, aunque no obligatoriamente el fracaso de la operación.

Los logs incluirán información de ERROR y de WARN.

### INFO

Información general acerca del flujo de objetos.

Los logs incluirán información de INFO, ERROR y WARN.

### DEBUG

Información que puede ser útil para depurar un problema.

Los logs incluirán información de DEBUG, INFO, ERROR y WARN.

### DISABLED

Todos los registros están desactivados.

## Eventos de registro y códigos de error

En esta sección se presentan los eventos de registro y los códigos de error que AWS IoT envía.

### Identidad y seguridad

Nombre de operación/evento	Descripción
Éxito de la autenticación	Certificado autenticado correctamente.
Error de autenticación	No se pudo autenticar un certificado.

### Identidad y códigos de error de seguridad

Código de error	Descripción del error
401	Sin autorización

### Agente de mensajes

Nombre de operación/evento	Descripción
Publicación MQTT	Publicación MQTT recibida.
Suscripción a MQTT	Suscripción a MQTT recibida.

Nombre de operación/evento	Descripción
Conexión MQTT	Conexión MQTT recibida.
Desconexión MQTT	Desconexión MQTT recibida.
HTTP/1.1 POST	MHTTP/1.1 POST recibido.
HTTP/1.1 GET	HTTP/1.1 GET recibido.
Método HTTP/1.1 no admitido	Se utiliza cuando un mensaje contiene un error de sintaxis o la acción (HTTP PUT/DELETE/) está prohibida.
Mensaje HTTP incorrecto	La conexión se ha interrumpido debido a un mensaje HTTP incorrecto.
Mensaje MQTT incorrecto	La conexión se ha interrumpido debido a un mensaje MQTT incorrecto.
Error de autorización	Este cliente ha intentado publicar en un tema o suscribirse a este sin autorización.
El paquete supera el tamaño máximo de carga	Este cliente ha intentado publicar una carga que supera el límite superior del agente de mensajes.

#### Códigos de error del agente de mensajes

Código de error	Descripción del error
400	solicitud errónea
401	Sin autorización
403	prohibido
503	Service Unavailable

#### Eventos del motor de reglas

Nombre de operación/evento	Descripción
MessageReceived	Se ha recibido la solicitud de un tema.
DynamoActionSuccess	Registro de DynamoDB colocado correctamente.
DynamoActionFailure	No se pudo colocar el registro de DynamoDB.
KinesisActionSuccess	Mensaje de Kinesis publicado correctamente.
KinesisActionFailure	No se pudo publicar un mensaje de Kinesis.
LambdaActionSuccess	Función Lambda invocada correctamente.
LambdaActionFailure	No se pudo invocar la función Lambda.
RepublishActionSuccess	Mensaje publicado de nuevo correctamente.
MessageReceived	Solicitud de tema recibida.

Nombre de operación/evento	Descripción
RepublishActionFailure	No se pudo volver a publicar un mensaje.
S3ActionSuccess	Objeto de Amazon S3 colocado correctamente.
S3ActionFailure	No se pudo colocar un objeto de Amazon S3.
SNSActionSuccess	Publicación correcta en tema Amazon SNS.
SNSActionFailure	No se pudo publicar en tema Amazon SNS.
SQSActionSuccess	Mensaje enviado correctamente a Amazon SQS.
SQSActionFailure	No se pudo enviar un mensaje a Amazon SQS.
SalesforceActionSuccess	Mensaje enviado correctamente a un flujo de entrada de Salesforce.
SalesforceActionFailure	No se pudo enviar un mensaje a un flujo de entrada de Salesforce.

#### Eventos de sombra de objeto

Nombre de operación/evento	Descripción
UpdateThingState	El estado de un objeto se actualiza en HTTP o MQTT.
DeleteThing	Se elimina un objeto.

#### Códigos de error de sombra de objeto

Código de error	Descripción del error
400	Solicitud errónea.
401	Sin autorización.
403	prohibido.
404	No encontrado.
409	Conflicto.
413	Solicitud demasiado grande.
422	No se pudo procesar una solicitud.
429	Demasiadas solicitudes.
500	Error interno.
503	Servicio no disponible.

## Diagnóstico de problemas de las reglas

CloudWatch Logs es el mejor lugar para solucionar los problemas con las reglas. Cuando habilita CloudWatch Logs para AWS IoT obtiene información sobre qué reglas se activan, así como su éxito o fracaso. También obtiene información sobre si las condiciones de la cláusula WHERE coinciden.

El problema más habitual es la autorización. En este caso, los logs le indicarán que su rol no está autorizado a realizar operaciones AssumeRole en el recurso.

Para consultar los logs de CloudWatch (consola)

1. En la consola de administración de AWS, vaya a la consola de CloudWatch.
2. Elija Logs y, a continuación, elija el grupo de logs AWSIoTLogs de la lista.
3. En la página Streams for AWSIoTLogs, encontrará un flujo de log por cada principal (certificado X.509, usuario de IAM o identidad de Amazon Cognito) que haya llamado a AWS IoT con su cuenta.

Para obtener más información, consulte [CloudWatch Logs](#).

El usuario final controla los servicios externos. Antes de ejecutar la regla, asegúrese de que los servicios externos estén configurados con suficientes unidades de capacidad y procesamiento.

## Diagnóstico de problemas en las sombras de objeto

### Diagnóstico de sombras de objeto

Problema	Directrices para solucionar problemas
Un documento de sombra de objeto se rechaza con el error "documento JSON no válido".	Si no está familiarizado con JSON, modifique los ejemplos proporcionados en esta guía y adáptelos a sus necesidades. Para obtener más información, consulte <a href="#">Sintaxis de los documentos de sombra de objeto</a> .
He enviado un JSON correcto, pero el documento de sombra de objeto no tiene nada almacenado o solo fragmentos de JSON.	Compruebe que ha seguido las directrices de formato JSON. Solo los campos JSON de las secciones <code>desired</code> y <code>reported</code> se almacenarán. No se tendrá en cuenta el contenido JSON (aunque sea formalmente correcto) que no esté en estas secciones.
He recibido un error que indica que la sombra de objeto supera el tamaño máximo permitido.	La sombra de objeto admite únicamente 8 KB de datos. Intente acortar los nombres de los campos que están dentro de su documento JSON o cree más sombras de objeto. Un dispositivo puede tener un número ilimitado de sombras de objeto. El único requisito es que el nombre de objeto sea único en su cuenta.
Cuando recibo una sombra de objeto, esta supera los 8 KB. ¿Por qué pasa esto?	En la recepción, el servicio AWS IoT agrega metadatos a la sombra de objeto. El servicio incluye estos datos en su respuesta, pero no cuentan para el límite de 8 KB. Solo los datos de estado <code>desired</code> y <code>reported</code> del documento de

Problema	Directrices para solucionar problemas
	estado enviado a la sombra de objeto cuentan para calcular el límite.
Mi solicitud se ha rechazado porque la versión es incorrecta. ¿Qué tengo que hacer?	Realice una operación GET para sincronizarse con la última versión del documento. Cuando utilice MQTT, suscríbase al tema <code>/update/accepted</code> para recibir notificaciones sobre cambios de estado y la última versión del documento JSON.
La marca de tiempo está desajustada en varios segundos.	El servicio AWS IoT actualiza la marca de tiempo de algunos campos individuales y de todo el documento JSON cuando los recibe; la marca de tiempo también se actualiza cuando el documento de estado se publica en el mensaje/ <code>update/accepted</code> y/ <code>update/delta</code> . Los mensajes pueden retrasarse en la red, lo que puede provocar una demora de varios segundos en la marca de tiempo.
Mi dispositivo puede publicar y suscribirse en los temas de sombra de objeto correspondientes, pero cuando intento actualizar el documento de sombra de objeto en la API HTTP REST, recibo un mensaje HTTP 403.	Compruebe que haya creado políticas en IAM que permitan a sus credenciales tener acceso a estos temas y a su acción correspondiente (UPDATE/GET/DELETE). Las políticas de IAM y las de certificado son independientes.
Otros problemas.	El servicio Thing Shadows registra los errores en CloudWatch Logs. Para identificar los problemas de configuración y de dispositivo, active CloudWatch Logs y consulte los logs para obtener información de depuración.

## Diagnosticar problemas con acciones del flujo de entrada de Salesforce IoT

### Registro de seguimiento de ejecución

¿Cómo puedo ver el registro de seguimiento de ejecución de una acción de Salesforce?

Si CloudWatch Logs no está instalado, consulte la sección [Configuración de CloudWatch Logs \(p. 265\)](#). Una vez que haya activado los logs, podrá ver el registro de seguimiento de ejecución de la acción de Salesforce.

### Éxito y error de una acción

¿Cómo puedo saber que los mensajes se han enviado correctamente a un flujo de entrada de Salesforce IoT?

Consulte los logs generados por la ejecución de la acción de Salesforce en CloudWatch Logs. Si aparece "Action executed successfully", eso significa que el motor de reglas de AWS IoT recibió la confirmación de Salesforce IoT de que el mensaje se envió correctamente al flujo de entrada de destino.

Si tiene problemas con la plataforma de Salesforce IoT , consulte la ayuda de Salesforce IoT

¿Qué hago si los mensajes no se han enviado correctamente a un flujo de entrada de Salesforce IoT

Consulte los logs generados por la ejecución de la acción de Salesforce en CloudWatch Logs. En función de la entrada de log, puede realizar las siguientes operaciones:

**Failed to locate the host**

Compruebe que el parámetro `url` de la acción es correcto y que el flujo de entrada de Salesforce IoT existe.

**Received Internal Server Error from Salesforce**

Reintentar. Si el problema continúa, póngase en contacto con el equipo de soporte de Salesforce IoT

**Received Bad Request Exception from Salesforce**

Compruebe la carga que envía para los errores.

**Received Unsupported Media Type Exception from Salesforce**

Salesforce IoT no admite una carga binaria en este momento. Compruebe que está enviando una carga JSON.

**Received Unauthorized Exception from Salesforce**

Compruebe que el parámetro `token` de la acción es correcto y que su token sigue siendo válido.

**Received Not Found Exception from Salesforce**

Compruebe que el parámetro `url` de la acción es correcto y que el flujo de entrada de Salesforce IoT existe.

Si surge algún problema que no figura en la lista, póngase en contacto con AWS Support.

Para obtener más información, consulte [CloudWatch Logs](#).