aws

Guía para desarrolladores

AWS IoT Core



Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS IoT Core: Guía para desarrolladores

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Las marcas comerciales y la imagen comercial de Amazon no se pueden utilizar en relación con ningún producto o servicio que no sea de Amazon de ninguna manera que pueda causar confusión entre los clientes y que menosprecie o desacredite a Amazon. Todas las demás marcas registradas que no son propiedad de Amazon son propiedad de sus respectivos propietarios, que pueden o no estar afiliados, conectados o patrocinados por Amazon.

Table of Contents

¿Qué es AWS IoT? 1
Cómo acceden tus dispositivos y aplicaciones AWS IoT 2
¿Qué AWS IoT puede hacer 3
El IoT en la industria 3
El IoT en la domótica 3
¿Cómo AWS IoT funciona 4
El universo del IoT 4
AWS IoT descripción general de los servicios 7
AWS IoT Core servicios 12
Obtenga más información sobre AWS IoT 16
Recursos de formación para AWS IoT 16
AWS IoT recursos y guías 17
AWS IoT en las redes sociales 18
AWS servicios utilizados por el motor de AWS IoT Core reglas 18
Protocolos de comunicación compatibles con AWS IoT Core 19
Novedades de la consola de AWS IoT 20
Leyenda 24
Trabajando con AWS SDKs 24
Tutoriales de introducción 26
Conecta tu primer dispositivo a AWS IoT Core 26
Configurar Cuenta de AWS 28
Inscríbase en una Cuenta de AWS 28
Creación de un usuario con acceso administrativo 29
Abre la AWS IoT consola 30
Tutorial interactivo 31
Conexión de dispositivos IoT 32
Guardar el estado del dispositivo desconectado 33
Enrutamiento de los datos del dispositivo a los servicios 34
Tutorial de conexión rápida 35
Paso 1. Comenzar el tutorial 36
Paso 2. Crear un objeto 37
Paso 3. Descargar archivos en su dispositivo 41
Paso 4. Ejecutar la muestra 44
Paso 5. Seguir explorando 47

Prueba de conectividad 48
Tutorial de conexión avanzado 54
¿Qué dispositivo se adapta mejor a sus necesidades? 55
Crea AWS IoT recursos 56
Configuración del dispositivo 61
Vea los mensajes MQTT con el cliente AWS IoT MQTT 100
Visualización de mensajes MQTT en el cliente MQTT 101
Publicación de mensajes MQTT desde el cliente MQTT 104
Probar las suscripciones compartidas en el cliente MQTT 106
AWS IoT tutoriales 108
Creación de demostraciones con el cliente de AWS IoT dispositivos 108
Requisitos previos para crear demostraciones con el AWS IoT Device Client 109
Preparación para usar IoT Device Client 112
Instalación y configuración del cliente de dispositivo de IoT 127
Comunicación con Device Client mediante MQTT 140
Ejecución de tareas de IoT con Device Client 160
Limpieza 175
Creación de soluciones con el dispositivo AWS IoT SDKs 185
Comience a crear soluciones con el AWS IoT dispositivo SDKs 186
Conexión de un dispositivo a AWS IoT Core mediante el SDK del AWS IoT dispositivo 186
Crear AWS IoT reglas para enrutar los datos del dispositivo a otros servicios 211
Retener el estado del dispositivo mientras el dispositivo está fuera de línea con Device Shadows 256
Crear un autorizador personalizado para AWS IoT Core 288
Monitorización de la humedad del suelo con una AWS IoT Raspberry Pi 306
Connect to AWS IoT Core 320
AWS IoT Core: puntos de conexión del plano de control 320
AWS IoT puntos finales del dispositivo 322
AWS IoT Core para dispositivos y puertas de enlace WAN LoRa 323
Conéctese a puntos finales AWS IoT Core de servicio 324
AWS CLI para AWS IoT Core 325
AWS SDKs 326
AWS Aplicaciones SDKs 331
RESTO APIs de los AWS IoT Core servicios 332
Connect dispositivos a AWS IoT 333
AWS IoT datos del dispositivo y puntos finales de servicio 334

AWS IoT Dispositivo SDKs 337
Protocolos de comunicación de dispositivos 339
Temas de MQTT 390
Configuraciones de dominio 417
Conéctese a puntos AWS IoT finales FIPS 446
puntos de conexión del plano de control de AWS IoT Core 447
puntos de conexión del plano de datos de AWS IoT Core 447
Puntos de conexión de proveedores de credenciales de AWS IoT Core 448
puntos de conexión de datos de trabajos de AWS IoT Device Management 448
puntos de conexión de AWS IoT Device Management - Fleet Hub 449
puntos de conexión de tunelización segura de AWS IoT Device Management 449
AWS IoT Device Management - Puntos finales de integraciones gestionadas 449
Administre dispositivos 451
Registro 452
Creación de un objeto 452
Lista de objetos 453
Describir las cosas 455
Actualización de un objeto 456
Eliminación de un objeto 456
Asociar un principal a un objeto 457
Enumeración de los objetos asociados a una entidad principal 458
Enumeración de las entidades principales asociadas a un objeto 458
Enumeración de los objetos asociados a una entidad principal V2 459
Enumeración de las entidades principales asociadas a un objeto V2 460
Desvincular un principal de un objeto 460
Tipos de cosas 461
Creación de un tipo de objeto 462
Lista de los tipos de objeto 462
Descripción de un tipo de objeto 463
Asociación de un tipo de objeto a un objeto 463
Actualización de un tipo de objeto 464
Descartar un tipo de objeto 464
Eliminación de un tipo de objeto 466
Grupos de objetos estáticos 466
Crear un grupo de objetos estático 468
Descripción de un grupo de objetos 469

Agregar un objeto a un grupo de objetos estático ... 470
Eliminar un objeto de un grupo de objetos estático ... 471
Enumerar los objetos en un grupo de objetos ... 471
Enumeración de grupos de objetos ... 472
Enumerar grupos para un objeto ... 474
Actualizar un grupo de objetos estático ... 475
Eliminación de un grupo de objetos ... 475
Asociar una política a un grupo de objetos estático ... 476
Desconectar una política de un grupo de objetos estático ... 477
Mostrar las políticas asociadas a un grupo de objetos estático ... 477
Enumeración de los grupos para una política ... 478
Obtención de políticas en vigor para un objeto ... 478
Prueba de autorización para acciones de MQTT ... 479
Grupos de objetos dinámicos ... 481
Uso de casos de uso de grupos de objetos dinámicos ... 482
Creación de un grupo de objetos dinámico ... 483
Describir un grupo de objetos dinámico ... 484
Actualizar un grupo de objetos dinámico ... 486
Eliminar un grupo de objetos dinámico ... 486
Limitaciones de los grupos de objetos dinámicos y estáticos ... 486
Limitaciones de los grupos de objetos dinámicos ... 487
Asociación de un objeto a una conexión ... 489
Casos de uso ... 490
Cómo asociar un objeto a una conexión ... 491
Agregación de atributos de propagación ... 494
Consola de administración de AWS ... 495
AWS CLI ... 495
Etiquetar recursos ... 498
Conceptos básicos de etiquetas ... 498
Restricciones y limitaciones en las etiquetas ... 499
Etiqueta con políticas de IAM ... 500
Grupos de facturación ... 503
Visualización de datos de uso y asignación de costos ... 503
Seguridad ... 506
Seguridad en AWS IoT ... 507
Autenticación ... 508

Información general del certificado X.509 508
Autenticación del servidor 508
Autenticación del cliente 512
Autenticación y autorización personalizada 554
Autorización 584
AWS formación y certificación 587
AWS IoT Core políticas 587
Autorizar llamadas directas a los AWS servicios mediante el proveedor de AWS IoT Core credenciales 669
Acceso entre cuentas con IAM 677
Protección de datos 679
Cifrado de datos en AWS IoT 680
Seguridad del transporte en AWS IoT Core 681
Cifrado de datos 687
Identity and Access Management 703
Público 703
Autenticación con identidades de IAM 704
Administración del acceso con políticas 705
¿Cómo AWS IoT funciona con IAM 707
Ejemplos de políticas basadas en identidades 739
AWS políticas gestionadas 743
Resolución de problemas 759
Registro y supervisión 761
Herramientas de monitorización 761
Validación de conformidad 763
Resiliencia 764
Uso AWS IoT Core con puntos finales de VPC 765
Creación de puntos finales de VPC para el plano de control AWS IoT Core 765
Creación de puntos finales de VPC para el plano de datos AWS IoT Core 766
Creación de puntos de enlace de VPC para el proveedor de credenciales AWS IoT Core ... 767
Creación de un punto de conexión de interfaz de Amazon VPC 768
Configure una zona alojada privada 770
Control del acceso a AWS IoT Core más de puntos finales de VPC 772
Limitaciones 773
Escalar los puntos finales de VPC con AWS IoT Core 775
Uso de dominios personalizados con puntos de conexión de VPC 775

Disponibilidad de puntos finales de VPC para AWS IoT Core ... 775
Uso AWS IoT Device Management con puntos finales de VPC ... 775
Seguridad de la infraestructura ... 781
Monitorización de la seguridad ... 782
Prácticas recomendadas de seguridad ... 783
Protección de las conexiones MQTT en AWS IoT ... 783
Mantener sincronizado el reloj del dispositivo ... 786
Validar el certificado de servidor ... 786
Usar una identidad única por dispositivo ... 787
Utilice un segundo Región de AWS como respaldo ... 787
Usar aprovisionamiento justo a tiempo ... 788
Permisos para ejecutar pruebas de AWS IoT Device Advisor ... 788
Prevención de la sustitución confusa entre servicios para Device Advisor ... 789
AWS formación y certificación ... 791
Supervisar AWS IoT ... 792
Configure el AWS IoT registro ... 793
Configurar el inicio de sesión en V2 AWS IoT ... 794
Configuración del rol y la política de registro ... 794
Configure el registro en la AWS IoT (consola) ... 797
Configurar el inicio de sesión a nivel de cuenta y evento (AWS IoT CLI) ... 799
Configurar anulaciones específicas de recursos en (CLI) AWS IoT ... 802
Niveles de registro ... 804
Supervisa AWS IoT las alarmas y las métricas con Amazon CloudWatch ... 805
Uso de AWS IoT métricas ... 806
Crea CloudWatch alarmas ... 806
Métricas y dimensiones ... 811
Supervise AWS IoT mediante registros CloudWatch ... 837
Visualización AWS IoT de los registros en la CloudWatch consola ... 837
CloudWatch Registra las entradas de AWS IoT registro ... 838
Sube registros del lado del dispositivo a Amazon CloudWatch ... 878
Funcionamiento ... 879
Carga de registros del lado del dispositivo mediante reglas AWS IoT ... 880
Registra las llamadas a AWS IoT la API ... 890
AWS IoT información en CloudTrail ... 890
Descripción de las entradas de los archivos de AWS IoT registro ... 891
Reglas ... 894

Concesión de acceso a ... 895
Revocación del acceso al motor de reglas ... 898
Transmisión de los permisos de rol ... 898
Creación de una regla ... 899
Creación de una regla (consola) ... 901
Creación de una regla (CLI) ... 902
Administración de una regla ... 906
Etiquetado de una regla ... 906
Visualización de una regla ... 908
Eliminación de una regla ... 908
AWS IoT acciones de reglas ... 908
Apache Kafka ... 912
CloudWatch alarmas ... 925
CloudWatch Registros ... 926
CloudWatch métricas ... 929
DynamoDB ... 931
Dinamo DBv2 ... 934
Elasticsearch ... 937
HTTP ... 939
AWS IoT Events ... 989
AWS IoT SiteWise ... 991
Firehose ... 997
Kinesis Data Streams ... 999
Lambda ... 1002
Ubicación ... 1005
OpenSearch ... 1009
Volver a publicar ... 1012
S3 ... 1015
IoT de Salesforce ... 1017
SNS ... 1018
SQS ... 1021
Step Functions ... 1023
Timestream ... 1025
Solución de problemas de las reglas ... 1033
Acceso a los recursos entre cuentas ... 1033
Requisitos previos ... 1034

Configuración de varias cuentas para Amazon SQS 1034
Configuración de varias cuentas para Amazon SNS 1036
Configuración de varias cuentas para Amazon S3 1038
Configuración multicuenta para AWS Lambda 1040
Control de errores (acción de error) 1043
Formato de mensaje de acción de error 1043
Ejemplo de acción de error 1045
Basic Ingest 1046
Uso de Basic Ingest 1046
AWS IoT Referencia de SQL 1047
Cláusula SELECT 1049
Cláusula FROM 1051
Cláusula SET 1052
Cláusula WHERE 1056
Tipos de datos 1057
Operadores 1063
Funciones 1074
Literales 1151
Instrucciones case 1152
Extensiones JSON 1153
Plantillas de sustitución 1155
Consultas de objetos anidados 1157
Cargas binarias 1159
Versiones de SQL 1166
Sombras 1168
Uso de sombras 1168
Elegir utilizar sombras con nombre o sin nombre 1169
Acceso a sombras 1170
Uso de sombras en dispositivos, aplicaciones y otros servicios en la nube 1170
Orden de los mensajes 1171
Recorte de mensajes de sombra 1173
Uso de sombras en dispositivos 1174
Inicializar el dispositivo en la primera conexión a AWS IoT 1175
Procesar los mensajes mientras el dispositivo está conectado a AWS IoT 1178
Procesar los mensajes cuando el dispositivo se vuelve a conectar a AWS IoT 1179
Uso de sombras en aplicaciones y servicios 1179

Inicializar la aplicación o el servicio al conectarse a AWS IoT 1180
El estado del procesamiento cambia mientras la aplicación o el servicio están conectados a AWS IoT 1181
Detección de un dispositivo conectado 1181
Simulación de comunicaciones del servicio Device Shadow 1183
Configuración de la simulación 1183
Inicializar el dispositivo 1184
Enviar una actualización desde la aplicación 1188
Responder a la actualización en el dispositivo 1190
Ver la actualización en la aplicación 1195
Más allá de la simulación 1197
Interacción con sombras 1197
Compatibilidad del protocolo 1198
Estado de solicitud y notificación 1198
Actualización de la sombra 1199
Recuperación de un documento de sombra 1203
Eliminación de datos de sombra 1204
API REST de sombra de dispositivo 1207
GetThingShadow 1209
UpdateThingShadow 1210
DeleteThingShadow 1211
ListNamedShadowsForThing 1212
Temas MQTT de sombra de dispositivo 1214
/get 1215
/get/accepted 1216
/get/rejected 1217
/update 1218
/update/delta 1219
/update/accepted 1220
/update/documents 1221
/update/rejected 1222
/delete 1223
/delete/accepted 1224
/delete/rejected 1225
Documentos del servicio Device Shadow 1226
Ejemplos de documento de sombra 1226

Propiedades del documento ... 1233
Estado delta ... 1234
Control de versiones de documentos de sombra ... 1236
Tokens de cliente en documentos de sombra ... 1237
Propiedades de documento de sombra vacío ... 1237
Valores de matriz en documentos sombra ... 1238
Mensajes de error de Device Shadow ... 1239
Catálogo de paquetes de software ... 1241
Preparación para utilizar el Catálogo de paquetes de software ... 1242
Ciclo de vida de la versión de paquete ... 1242
Convenciones de nomenclatura de versiones de paquetes ... 1244
Versión predeterminada ... 1244
Atributos de la versión ... 1244
Lista de materiales de software ... 1245
AWS IoT Habilitar la indexación de la flota ... 1249
Sombra con nombre reservado ... 1249
Eliminar un paquete de software ... 1251
Preparación de la seguridad ... 1252
Autenticación basada en recursos ... 1252
AWS IoT Job: derechos para implementar versiones de paquetes ... 1254
AWS IoT Job: derechos para actualizar la sombra reservada con nombre ... 1255
AWS IoT Permisos de tareas para descargar desde Amazon S3 ... 1257
Permisos para actualizar la lista de materiales de software para una versión de paquete ... 1257
Preparación para la indexación de flotas ... 1260
Establecer la sombra `$package` como origen de datos ... 1261
Las métricas se muestran en la consola ... 1261
Patrones de consulta ... 1262
Recopilación de la distribución de las versiones del paquete mediante `getBucketsAggregation` ... 1265
Preparando AWS IoT trabajos ... 1265
Parámetros de sustitución de trabajos AWS IoT ... 1266
Preparación del documento de trabajo y la versión del paquete para su implementación ... 1270
Asignar un nombre a los paquetes y las versiones al desplegarlos ... 1274
Segmentar los trabajos mediante grupos de cosas AWS IoT dinámicos ... 1275
Versiones reservadas de paquetes y sombras con nombre ... 1275
Desinstalar un paquete de software ... 1276

Introducción 1276
Crear un paquete y una versión 1277
Desplegar una versión del paquete 1280
Asociar una versión de paquete 1282
Tareas 1284
Acceder a AWS IoT los trabajos 1284
AWS IoT Puestos de trabajo, regiones y puntos finales 1284
¿Qué es una operación remota? 1285
Ventajas de usar AWS IoT Device Management Jobs para operaciones remotas 1285
¿Qué es Jobs? AWS IoT 1288
Conceptos clave de trabajos 1289
Trabajos y estados de ejecución de los trabajos 1293
Administración de trabajos 1298
Firma de código para trabajos 1299
Documento de trabajo 1299
Prefirmado URLs 1299
URL prefirmada para la carga de archivos 1302
URL prefirmada mediante el control de versiones de Amazon S3 1303
Creación y administración de trabajos mediante la consola 1305
Creación y administración de trabajos mediante la CLI 1308
Plantillas de tarea 1320
Plantillas personalizadas y AWS gestionadas 1320
Utilice plantillas AWS gestionadas 1321
Creación de plantillas de trabajo personalizadas 1341
Configuraciones de trabajos 1350
Cómo funcionan las configuraciones de trabajos 1351
Especificación de configuraciones adicionales 1367
Dispositivos y trabajos 1377
Programación de dispositivos para trabajar con trabajos 1380
Flujo de trabajo del dispositivo 1380
Flujo de trabajo 1382
Notificaciones de trabajos 1387
AWS IoT operaciones de API de trabajos 1395
API de administración y control de trabajos y tipos de datos 1398
Operaciones y tipos de datos de las API MQTT y HTTPS de dispositivos de Jobs 1417
Protección de los usuarios y los dispositivos de Jobs 1433

Tipo de política obligatorio para AWS IoT Jobs 1433
Autorización de los usuarios y los servicios en la nube de Jobs 1434
Autorización del uso de trabajos por parte de los dispositivos 1447
AWS IoT Límites de puestos 1452
Límites de ejecución de trabajos 1452
Límites de trabajos activos y simultáneos 1454
Comandos 1458
Inicio rápido 1459
Estado y conceptos de los comandos 1460
Conceptos clave de los comandos 1460
Estados de comando 1465
Estado de la ejecución del comando 1465
Flujo de trabajo de comandos 1469
Creación y administración de comandos 1470
Selección de los dispositivos de destino y suscripción a los temas 1470
Inicio y supervisión de las ejecuciones de comandos 1472
(Opcional) Habilitación de las notificaciones para los eventos de comandos 1473
Creación y administración de comandos 1475
Creación de un recurso de comandos 1475
Recuperación de información sobre un comando 1481
Enumere los comandos de su Cuenta de AWS 1483
Actualización de un recurso de comandos 1485
Obsolescencia o restauración de un recurso de comandos 1487
Eliminación de un recurso de comandos 1487
Inicio y supervisión de las ejecuciones de comandos 1489
Inicio de la ejecución de un comando 1490
Actualización del resultado de la ejecución de un comando 1497
Recuperación de la ejecución de un comando 1503
Visualización de actualizaciones de comandos mediante el cliente de prueba de MQTT ... 1506
Enumera las ejecuciones de comandos en tu Cuenta de AWS 1508
Eliminación de una ejecución de comandos 1511
Obsolescencia de un recurso de comandos 1512
Consideraciones clave 1513
Obsolescencia de un recurso de comandos (consola) 1513
Obsolescencia de un recurso de comandos (CLI) 1513
Comprobación del tiempo y el estado de obsolescencia 1514

Restauración de un recurso de comando 1514
Tunelización segura 1516
¿Qué es la tunelización segura? 1516
Conceptos de tunelización segura 1517
Cómo funciona la tunelización segura 1518
Ciclo de vida del túnel seguro 1519
Tutoriales de tunelización segura 1520
Tutoriales en esta sección 1520
Abra un túnel e inicie una sesión SSH en el dispositivo remoto 1521
Abra un túnel para el dispositivo remoto y utilice SSH basado en navegador 1539
Proxy local 1544
Cómo usar el proxy local 1544
Configure el proxy local para los dispositivos que utilizan el proxy web 1551
Multiplexación y conexiones TCP simultáneas 1559
Multiplexación de múltiples flujos de datos 1560
Uso de conexiones TCP simultáneas 1564
Configuración de un dispositivo remoto y uso de un agente de IoT 1567
Fragmento de agente de IoT 1567
Control del acceso a los túneles 1569
Requisitos previos de acceso al túnel 1569
Políticas de acceso a túnel 1569
Resolver los problemas de conectividad de los túneles seguros 1577
Error de token de acceso al cliente no válido 1577
Error de discordancia del token del cliente 1578
Problemas de conectividad de dispositivos remotos 1579
Aprovisionamiento de dispositivos 1582
Aprovisionamiento de dispositivos en AWS IoT 1583
Aprovisionamiento de flota APIs 1585
Aprovisionamiento de dispositivos que no tienen certificados de dispositivo mediante el aprovisionamiento de flotas 1585
Aprovisionamiento por reclamación 1586
Aprovisionamiento por usuario de confianza 1589
Uso de ganchos de aprovisionamiento previo con la CLI AWS 1591
Aprovisionamiento de dispositivos que tienen certificados de dispositivo 1595
Aprovisionamiento de un solo objeto 1596
Just-in-time aprovisionamiento 1596

Registro masivo 1603
Aprovisionamiento de plantillas 1604
Sección de parámetros 1604
Sección de recursos 1605
Ejemplo de plantilla para el registro masivo 1611
Ejemplo de plantilla para just-in-time aprovisionamiento (JITP) 1612
Aprovisionamiento de flotas 1614
Enlaces de preaprovisionamiento 1618
Preaprovisionamiento de entrada de enlace 1619
Valor de retorno del enlace previo a la provisión 1619
Ejemplo de enlace Lambda de preaprovisionamiento 1620
Firma de certificados autogestionada mediante un proveedor de certificados AWS IoT Core .. 1622
Funcionamiento de la firma de certificados autoadministrada en el aprovisionamiento de flotas 1624
Entrada de la función de Lambda del proveedor de certificados 1625
Valor devuelto por la función de Lambda del proveedor de certificados 1626
Ejemplo de función de Lambda 1626
Firma de certificados autoadministrada para el aprovisionamiento de flotas 1628
AWS CLI comandos para el proveedor de certificados 1630
Creación de políticas y roles de IAM para un usuario que instala un dispositivo 1632
Creación de una política de IAM para el usuario que va a instalar un dispositivo 1633
Creación de un rol de IAM para el usuario que instalará un dispositivo 1634
Actualización de una política existente para autorizar una plantilla nueva 1635
API de MQTT de aprovisionamiento de dispositivos 1636
CreateCertificateFromCsr 1637
CreateKeysAndCertificate 1639
RegisterThing 1641
Indexación de flota 1645
Administración de actualizaciones de índices 1645
Consulta de estado de conectividad de un dispositivo específico 1645
Buscar en todos los orígenes de datos 1645
Consulta de datos agregados 1646
Monitorización de datos agregados y creación alarmas mediante métricas de flota 1646
Administración de la indexación de flotas 1646
Indexación de objetos 1646
Indexación de grupos de objetos 1648

Campos administrados ... 1648
Campos personalizados ... 1650
Administración de la indexación de objetos ... 1651
Administración de la indexación de grupos de objetos ... 1667
Estado de conectividad de los dispositivos ... 1669
Funcionamiento ... 1670
Características ... 1670
Ventajas ... 1670
Requisitos previos ... 1671
Ejemplos ... 1671
Consulta de datos agregados ... 1673
GetStatistics ... 1673
GetCardinality ... 1676
GetPercentiles ... 1677
GetBucketsAggregation ... 1679
Autorización ... 1681
Sintaxis de la consulta ... 1681
Características admitidas ... 1681
Características no admitidas ... 1682
Notas ... 1682
Ejemplo de consultas de objetos ... 1683
Ejemplo de consultas de grupo de objetos ... 1688
Indexación de datos de ubicación ... 1689
Formatos de datos admitidos ... 1690
Cómo indexar los datos de ubicación ... 1691
Actualización de la configuración de indexación de objetos ... 1692
Ejemplo de geoconsultas ... 1694
Tutorial introductorio ... 1695
Métricas de flota ... 1700
Tutorial introductorio ... 1701
Administración de métricas de flota ... 1708
Entrega de archivos basada en MQTT ... 1716
¿Qué es una transmisión? ... 1716
Administración de un flujo ... 1717
Conceder permisos a sus dispositivos ... 1718
Conecta tus dispositivos a AWS IoT ... 1719

TagResource Uso 1720
Utilice la entrega de archivos AWS IoT basada en MQTT en los dispositivos 1721
Se usa DescribeStream para obtener datos de transmisión 1721
Obtener bloques de datos de un archivo de transmisión 1723
Gestión de errores derivados de la entrega de archivos AWS IoT basada en MQTT 1730
Un ejemplo de caso de uso en Freertos OTA 1732
Asesor de dispositivos 1733
Configuración 1735
Cree un objeto de &IoT 1735
Cree un rol de IAM para utilizarlo como su rol de dispositivo 1735
Cree una política administrada de forma personalizada para que un usuario de IAM utilice Device Advisor 1738
Cree un usuario de IAM para utilizar Device Advisor 1739
Configuración del dispositivo 1742
Introducción a Device Advisor en la consola 1743
Flujo de trabajo de Device Advisor 1753
Requisitos previos 1753
Crear una definición de conjunto de pruebas 1753
Obtener una definición del conjunto de pruebas 1756
Obtener un punto de conexión de prueba 1757
Iniciar la ejecución de un conjunto de pruebas 1757
Obtener una ejecución del conjunto de pruebas 1758
Detener la ejecución de un conjunto de pruebas 1758
Obtener un informe de cualificación para una ejecución correcta del conjunto de pruebas de cualificación 1759
Flujo de trabajo detallado de la consola de Device Advisor 1759
Requisitos previos 1760
Crear una definición de conjunto de pruebas 1760
Iniciar la ejecución de un conjunto de pruebas 1767
Detener la ejecución de un conjunto de pruebas (opcional) 1769
Ver los detalles y los registros de las ejecuciones del conjunto de pruebas 1771
Descarga un informe AWS IoT de calificación 1773
Flujo de trabajo de la consola de pruebas de larga duración 1773
puntos de conexión de VPC de Device Advisor (AWS PrivateLink) 1782
Consideraciones sobre los puntos AWS IoT Core Device Advisor finales de VPC 1783
Cree un punto final de VPC de interfaz para AWS IoT Core Device Advisor 1784

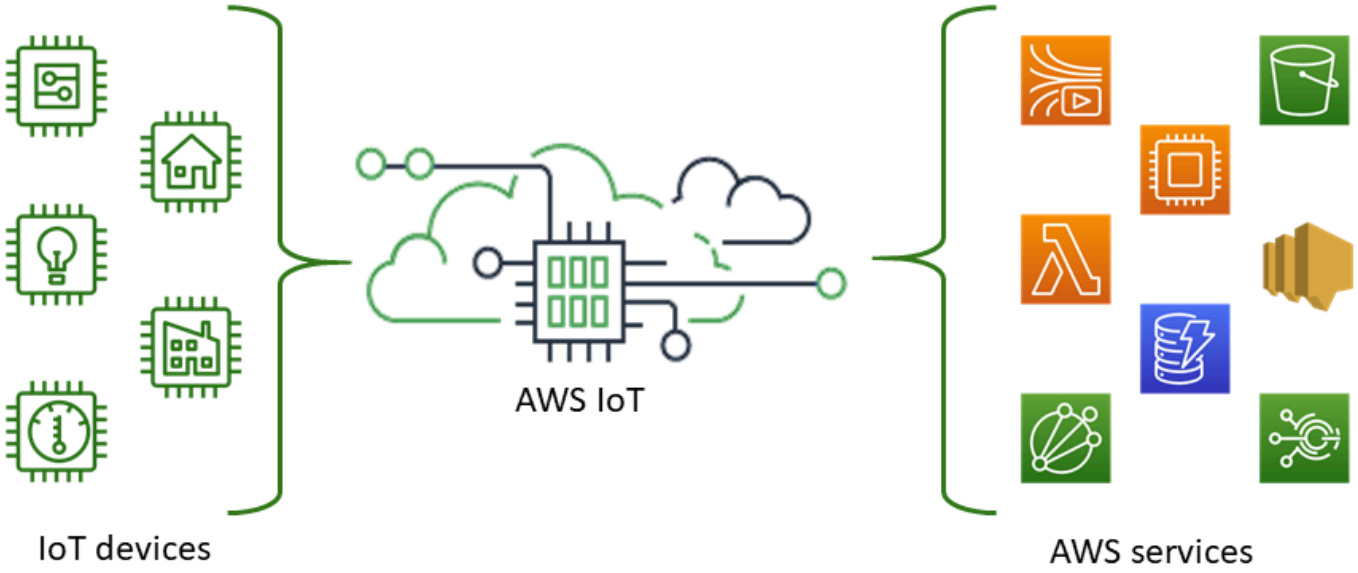
Control del acceso a AWS IoT Core Device Advisor más de puntos finales de VPC 1784
Casos de prueba de Device Advisor 1786
Casos de prueba de Device Advisor para poder optar al programa de calificación de AWS dispositivos. 1786
TLS 1787
MQTT 1794
Sombra 1809
Ejecución de trabajo 1811
Permisos y políticas 1813
Pruebas de larga duración 1814
Device Location 1832
Tipos de mediciones y solucionadores 1833
Cómo funciona la ubicación del AWS IoT Core dispositivo 1834
Cómo utilizar la ubicación del dispositivo AWS IoT Core 1835
Resolver la ubicación de los dispositivos IoT 1836
Resolver la ubicación del dispositivo (consola) 1837
Resolver la ubicación del dispositivo (API) 1840
Solución de errores al resolver la ubicación 1842
Resolución de la ubicación del dispositivo mediante los temas de MQTT 1843
Formato de los temas de MQTT de ubicación de dispositivos 1843
Política de los temas de MQTT de ubicación de dispositivos 1845
Temas y carga de Device Location 1846
Solucionadores de ubicación y carga útil del dispositivo 1851
Solucionador basado en Wi-Fi 1851
Solucionador basado en dispositivos móviles 1852
Solucionador de búsqueda inversa de IP 1857
Solucionador GNSS 1858
Solucionador BLE para dispositivos compatibles con Amazon Sidewalk 1860
Mensajes de los eventos 1861
Cómo se generan los mensajes de eventos 1861
Política de recepción de mensajes de eventos 1861
Habilite los eventos para AWS IoT 1862
Eventos de registro 1867
Eventos de objeto 1867
Eventos de tipo de objeto 1869
Eventos de grupo de objetos 1873

Eventos de trabajos 1878
Eventos del ciclo de vida 1883
Eventos de conexión/desconexión 1884
Evento de error de intento de conexión 1888
Eventos de suscripción/cancelación de suscripción 1890
Resolución de problemas 1892
AWS IoT Core guía de solución de problemas 1892
Diagnóstico de problemas de conectividad 1893
Diagnóstico de problemas de las reglas 1896
Diagnóstico de problemas relacionados con las sombras 1899
Diagnosticar problemas con acciones de Salesforce 1901
Diagnóstico de los límites de flujo 1902
Resolución para las desconexiones en una flota de dispositivos 1903
AWS IoT Device Management guía de solución de problemas 1904
AWS IoT Solución de problemas de trabajos 1904
Resolución de problemas de la indexación de flotas 1909
AWS IoT Solución de problemas del catálogo de paquetes de software de administración de dispositivos 1912
AWS IoT Guía de solución de problemas de Device Advisor 1920
AWS IoT errores 1923
SDK de dispositivos, SDK para móviles y cliente de dispositivo de AWS IoT 1925
SDK de dispositivos de AWS IoT 1925
SDK de dispositivos de AWS IoT para Embedded C 1928
AWS Mobile SDK 1928
Cliente de dispositivo de AWS IoT 1929
Versiones anteriores de los SDK de dispositivos de AWS IoT 1930
Ejemplos de código 1931
Conceptos básicos 1932
Hola AWS IoT 1932
Conceptos básicos 1941
Acciones 2052
AWS IoTCuotas de 2150
AWS IoT CorePrecios de 2151
..... mmclii

¿Qué es AWS IoT?

AWS IoT proporciona los servicios en la nube que conectan sus dispositivos de IoT a otros dispositivos y servicios AWS en la nube. AWS IoT proporciona software para dispositivos que puede ayudarlo a integrar sus dispositivos de IoT en soluciones AWS IoT basadas en datos. Si sus dispositivos se pueden conectar AWS IoT, AWS IoT puede conectarlos a los servicios en la nube que AWS proporcionan.

Para obtener una introducción práctica AWS IoT, visiteTutoriales de introducción.



Con AWS IoTél, puede seleccionar las up-to-date tecnologías más adecuadas para su solución. Para ayudarle a gestionar y dar soporte a sus dispositivos de IoT sobre el terreno, AWS IoT Core es compatible con los siguientes protocolos:

- MQTT (Message Queuing and Telemetry Transport)
- MQTT sobre WSS (Websockets Secure)
- HTTPS (Hypertext Transfer Protocol - Secure)
- LoRaWAN (red de área amplia y largo alcance)

El agente de AWS IoT Core mensajes admite dispositivos y clientes que utilizan los protocolos MQTT y MQTT sobre WSS para publicar mensajes y suscribirse a ellos. También es compatible con dispositivos y clientes que utilizan el protocolo HTTPS para publicar mensajes.

AWS IoT Core para LoRa WAN le ayuda a conectar y administrar dispositivos LoRa WAN (red de área amplia de largo alcance y bajo consumo) inalámbricos. AWS IoT Core para LoRa WAN reemplaza la necesidad de desarrollar y operar un servidor de red LoRa WAN (LNS).

Si no necesita AWS IoT funciones como las comunicaciones, las reglas o los trabajos del dispositivo, consulte AWS Mensajería para obtener información sobre otros servicios de AWS IoT mensajería que podrían adaptarse mejor a sus necesidades.

Cómo acceden tus dispositivos y aplicaciones AWS IoT

AWS IoT proporciona las siguientes interfaces paraAWS IoT tutoriales:

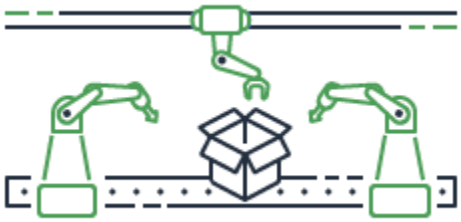
- AWS IoT Dispositivo SDKs: cree aplicaciones en sus dispositivos que envíen y reciban mensajes. AWS IoT Para obtener más información, consulte SDK de dispositivos, SDK para móviles y cliente de dispositivo de AWS IoT.
- AWS IoT Core para LoRa WAN: conecte y administre sus dispositivos y puertas de enlace LoRa WAN de largo alcance (WAN) mediante AWS IoT Core WAN. LoRa
- AWS Command Line Interface (AWS CLI): ejecuta comandos para AWS IoT Windows, macOS y Linux. Estos comandos le permiten crear y administrar objetos, certificados, reglas, trabajos y políticas. Para empezar, consulte la AWS Command Line Interface Guía del usuario de . Para obtener más información sobre los comandos de AWS IoT, consulte iot en la Referencia de AWS CLI comandos.
- AWS IoT API: cree sus aplicaciones de IoT mediante solicitudes HTTP o HTTPS. Estas acciones de API le permiten crear y administrar mediante programación objetos, certificados, reglas y políticas. Para obtener más información sobre las acciones de la API AWS IoT, consulte Acciones en la referencia de la AWS IoT API.
- AWS SDKs—Cree sus aplicaciones de IoT utilizando un idioma APIs específico. Estas SDKs incluyen la HTTP/HTTPS API y permiten programar en cualquiera de los lenguajes compatibles. Para obtener más información, consulte AWS SDKs Herramientas.

También puede acceder a AWS IoT través de la AWS IoT consola, que proporciona una interfaz gráfica de usuario (GUI) a través de la cual puede configurar y administrar los objetos, los certificados, las reglas, los trabajos, las políticas y otros elementos de sus soluciones de IoT.

¿Qué AWS IoT puede hacer

En este tema, se describen algunas de las soluciones que quizá necesite y que son posibles con AWS IoT .

El IoT en la industria



Estos son algunos ejemplos de AWS IoT soluciones para casos de uso industrial que aplican tecnologías de IoT para mejorar el rendimiento y la productividad de los procesos industriales.

Soluciones para casos de uso industriales

- Se utilizan AWS IoT para crear modelos predictivos de calidad en operaciones industriales

  Descubra cómo AWS IoT puede recopilar y analizar datos de las operaciones industriales para crear modelos predictivos de calidad. Más información
- Úselo AWS IoT para respaldar el mantenimiento predictivo en las operaciones industriales

  Descubra cómo AWS IoT puede ayudar a planificar el mantenimiento preventivo para reducir el tiempo de inactividad no planificado. Más información

El IoT en la domótica



Estos son algunos ejemplos de AWS IoT soluciones para casos de uso de automatización del hogar que aplican tecnologías de IoT para crear aplicaciones de IoT escalables que automatizan las actividades del hogar mediante dispositivos domésticos conectados.

Soluciones de domótica

- Úsalo AWS IoT en tu hogar conectado

  Vea cómo AWS IoT puede proporcionar soluciones integradas de automatización del hogar.
- Úselo AWS IoT para proporcionar seguridad y monitoreo en el hogar

  Descubra cómo AWS IoT puede aplicar el aprendizaje automático y la computación perimetral a su solución de automatización del hogar.

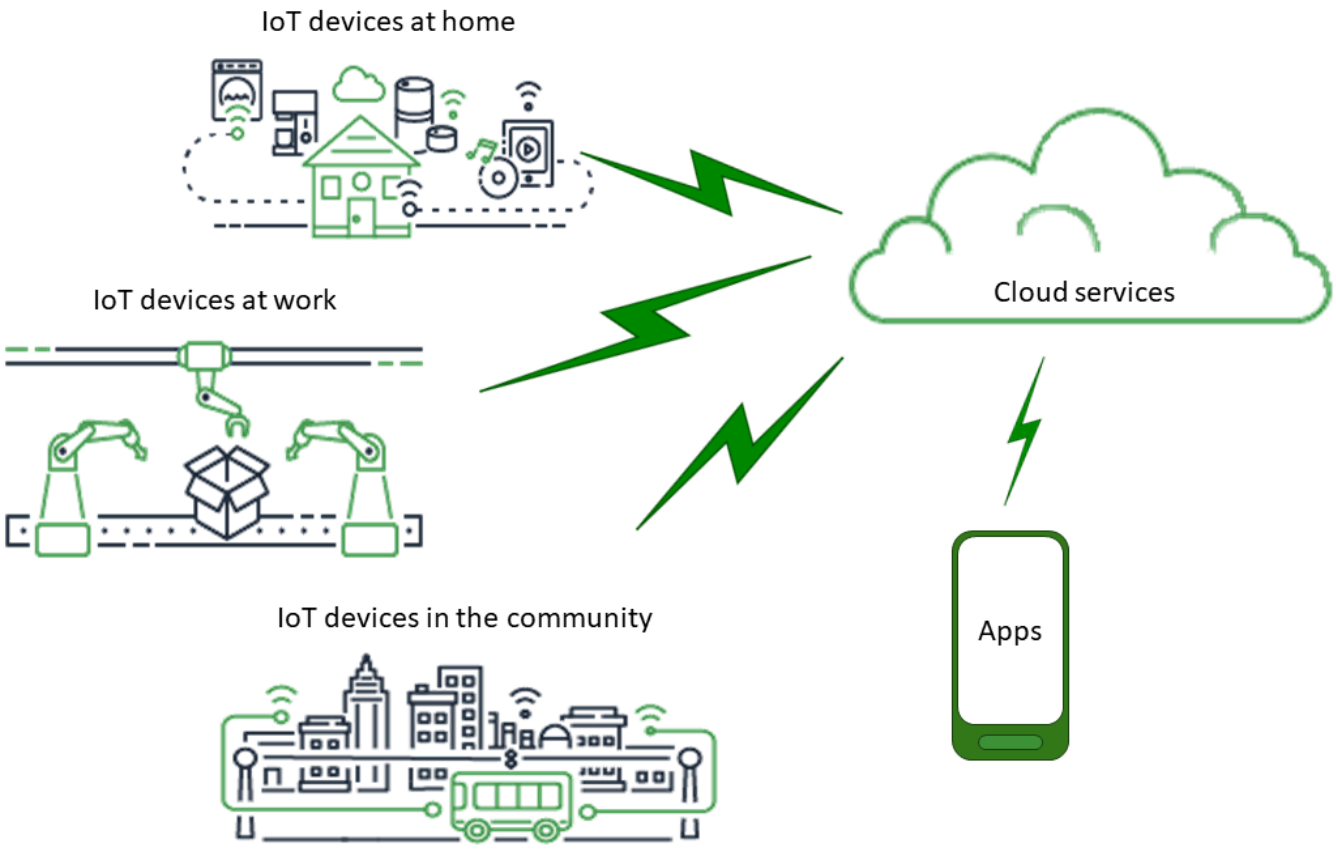
Para obtener una lista de soluciones para casos de uso industrial, de consumo y comercial, consulte el repositorio de soluciones AWS IoT.

¿Cómo AWS IoT funciona

AWS IoT proporciona servicios en la nube y soporte para dispositivos que puede usar para implementar soluciones de IoT. AWS proporciona muchos servicios en la nube para dar soporte a las aplicaciones basadas en el IoT. Para saber por dónde empezar, en esta sección se proporciona un diagrama y una definición de los conceptos esenciales en el universo del IoT.

El universo del IoT

En general, el Internet de las cosas (IoT) consta de los componentes clave que se muestran en este diagrama.



Aplicaciones

Las aplicaciones le dan al usuario final acceso a los dispositivos de IoT y a las características de los servicios en la nube a los que están conectados esos dispositivos.

Servicio en la nube

Los servicios en la nube son servicios distribuidos de procesamiento y almacenamiento de datos a gran escala que están conectados a Internet. Entre los ejemplos se incluyen:

- Servicios de conexión y administración de IoT

  AWS IoT es un ejemplo de servicio de conexión y administración de IoT.
- Servicios de cómputo, como Amazon Elastic Compute Cloud y AWS Lambda
- Servicios de bases de datos, como Amazon DynamoDB

Comunicación

Los dispositivos se comunican con los servicios en la nube mediante diversas tecnologías y protocolos. Entre los ejemplos se incluyen:

- Wifi/Internet de banda ancha
- Datos móviles de banda ancha
- Datos móviles de banda estrecha
- Red de área amplia (LoRaWAN) de largo alcance
- Comunicaciones RF privadas

dispositivos

Un dispositivo es un tipo de hardware que administra las interfaces y las comunicaciones. Los dispositivos suelen estar ubicados muy cerca de las interfaces del mundo real que supervisan y controlan. Los dispositivos pueden incluir recursos informáticos y de almacenamiento, como microcontroladores, CPU o memorias. Entre los ejemplos se incluyen:

- Raspberry Pi
- Arduino
- Asistentes con interfaz de voz
- LoRaWAN y dispositivos
- Dispositivos Amazon Sidewalk
- Dispositivos IoT personalizados

Interfaces

Una interfaz es un componente que conecta un dispositivo al mundo físico.

- Interfaces de usuario

  Componentes que permiten a dispositivos y usuarios comunicarse entre sí.

  - Interfaces de entrada

    Permiten que un usuario se comunique con un dispositivo

Ejemplos: teclado, botón

- Interfaces de salida

  Permiten que un dispositivo se comunique con un usuario

  Ejemplos: pantalla alfanumérica, pantalla gráfica, luz indicadora, timbre de alarma

- Sensores

  Componentes de entrada que miden o detectan algo en el mundo exterior de forma que un dispositivo pueda entenderlo. Entre los ejemplos se incluyen:

  - Sensor de temperatura (convierte la temperatura en una señal analógica o digital)
  - Sensor de humedad (convierte la humedad relativa en una señal analógica o digital)
  - Convertidor analógico a digital (convierte una tensión analógica en un valor numérico)
  - Unidad de medición de distancia por ultrasonidos (convierte una distancia en un valor numérico)
  - Sensor óptico (convierte el nivel de luz en un valor numérico)
  - Cámara (convierte los datos de imagen en datos digitales)

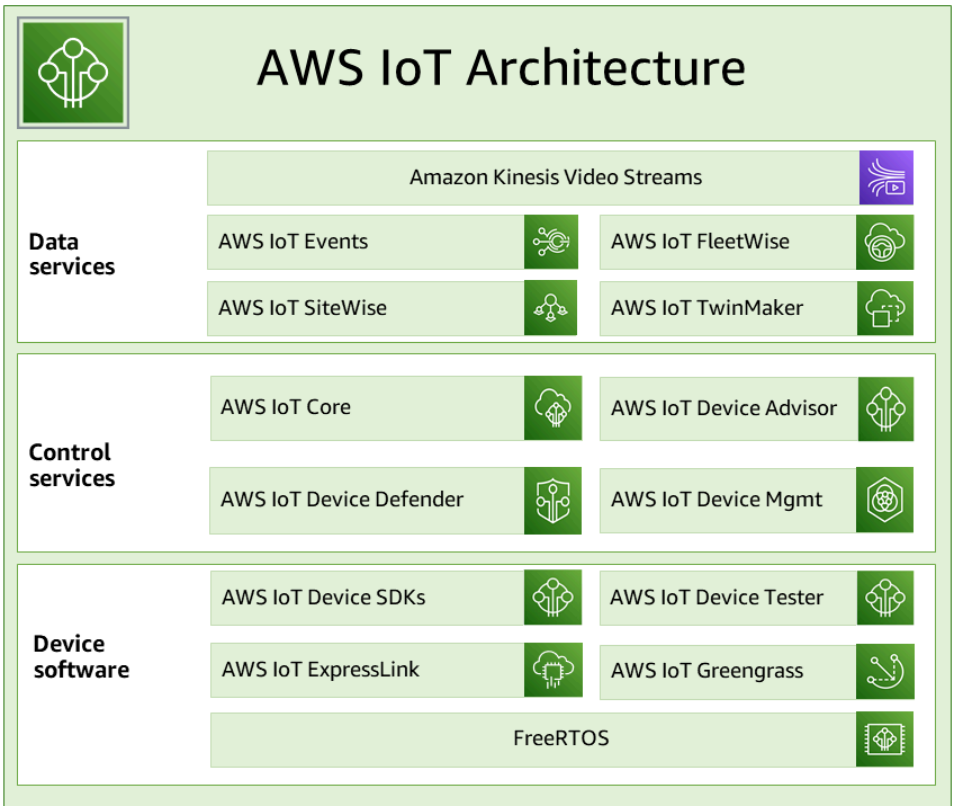
- Actuadores

  Componentes de salida que el dispositivo puede usar para controlar algo en el mundo exterior. Entre los ejemplos se incluyen:

  - Motores paso a paso (convierten las señales eléctricas en movimiento)
  - Relés (controlan altas tensiones y corrientes eléctricas)

AWS IoT descripción general de los servicios

En el universo de IoT, AWS IoT proporciona los servicios que respaldan los dispositivos que interactúan con el mundo y los datos que pasan entre ellos y AWS IoT. AWS IoT se compone de los servicios que se muestran en esta ilustración para respaldar su solución de IoT.



AWS IoT software del dispositivo

AWS IoT proporciona este software para dar soporte a sus dispositivos de IoT.

AWS IoT Dispositivo SDKs

El AWS IoT dispositivo y el móvil le SDKs ayudan a conectar sus dispositivos de manera eficiente a AWS IoT. AWS IoT Device y Mobile SDKs incluyen bibliotecas de código abierto, guías para desarrolladores con ejemplos y guías de portabilidad para que pueda crear productos o soluciones de IoT innovadores en las plataformas de hardware que elija.

AWS IoT Device Tester

AWS IoT Device Testerpara FreeRTOS y AWS IoT Greengrass es una herramienta de automatización de pruebas para microcontroladores. AWS IoT Device Tester prueba el dispositivo para determinar si ejecutará FreeRTOS o AWS IoT Greengrass si interoperará con servicios. AWS IoT

AWS IoT ExpressLink

AWS IoT ExpressLink alimenta una gama de módulos de hardware desarrollados y ofrecidos por AWS los socios. Los módulos de conectividad incluyen software AWS validado, lo que facilita

y agiliza la conexión segura de los dispositivos a la nube y se integran sin problemas con una gama de AWS servicios. Para obtener más información, visite la página de información AWS IoT ExpressLinkgeneral o consulte la Guía del AWS IoT ExpressLink programador.

AWS IoT Greengrass

AWS IoT Greengrassse extiende AWS IoT a los dispositivos periféricos para que puedan actuar localmente a partir de los datos que generan, ejecutar predicciones basadas en modelos de aprendizaje automático y filtrar y agregar los datos de los dispositivos. AWS IoT Greengrass permite que sus dispositivos recopilen y analicen datos más cerca de donde se generan, reaccionen de forma autónoma ante los eventos locales y se comuniquen de forma segura con otros dispositivos de la red local. Puede utilizarlas AWS IoT Greengrass para crear aplicaciones perimetrales mediante módulos de software prediseñados, denominados componentes, que pueden conectar sus dispositivos perimetrales a AWS servicios o servicios de terceros.

FreeRTOS

FreeRTOS es un sistema operativo de código abierto y en tiempo real para microcontroladores que le permite incluir dispositivos periféricos pequeños y de bajo consumo en su solución de IoT. FreeRTOS incluye un núcleo y un conjunto cada vez mayor de bibliotecas de software que admiten muchas aplicaciones. Los sistemas FreeRTOS pueden conectar de forma segura sus dispositivos pequeños y de bajo consumo a AWS IoT y permiten la ejecución de dispositivos periféricos más potentes con AWS IoT Greengrass.

AWS IoT servicios de control

Conéctese a los siguientes AWS IoT servicios para administrar los dispositivos de su solución de IoT.

AWS IoT Core

AWS IoT Corees un servicio en la nube gestionado que permite a los dispositivos conectados interactuar de forma segura con las aplicaciones en la nube y otros dispositivos. AWS IoT Core puede admitir muchos dispositivos y mensajes, y puede procesar y enrutar esos mensajes a AWS IoT puntos finales y otros dispositivos. Con AWS IoT Core esto, sus aplicaciones pueden interactuar con todos sus dispositivos incluso cuando no están conectados.

AWS IoT Core Asesor de dispositivos

AWS IoT Core Device Advisor es una capacidad de prueba basada en la nube y totalmente administrada para validar dispositivos IoT durante el desarrollo del software del dispositivo.

Device Advisor proporciona pruebas prediseñadas que puede usar para validar los dispositivos de IoT y lograr una conectividad confiable y segura antes de implementar los dispositivos en producción. AWS IoT Core

AWS IoT Device Defender

AWS IoT Device Defender le ayuda a proteger su flota de dispositivos de IoT. AWS IoT Device Defender audita continuamente sus configuraciones de IoT para asegurarse de que no se desvíen de las mejores prácticas de seguridad. AWS IoT Device Defender envía una alerta cuando detecta cualquier brecha en la configuración de IoT que pueda suponer un riesgo para la seguridad, como certificados de identidad que se comparten entre varios dispositivos o un dispositivo con un certificado de identidad revocado que intenta AWS IoT Coreconectarse.

AWS IoT Administración de dispositivos

AWS IoT Los servicios de administración de dispositivos le ayudan a rastrear, monitorear y administrar la gran cantidad de dispositivos conectados que componen sus flotas de dispositivos. AWS IoT Los servicios de administración de dispositivos lo ayudan a garantizar que sus dispositivos de IoT funcionen de manera adecuada y segura después de su implementación. También proporciona un túnel seguro para acceder a sus dispositivos, supervisar su estado y detectar y solucionar problemas de forma remota; además, ofrece servicios para gestionar las actualizaciones de software y firmware de los dispositivos.

AWS IoT servicios de datos

Analice los datos de los dispositivos de su solución de IoT y tome las medidas adecuadas mediante los siguientes AWS IoT servicios.

Amazon Kinesis Video Streams

Amazon Kinesis Video Streams le permite transmitir vídeo en directo desde los dispositivos a AWS la nube, donde se almacena, cifra e indexa de forma duradera, lo que le permite acceder a sus datos. easy-to-use APIs Puede utilizar Amazon Kinesis Video Streams para capturar grandes cantidades de datos de vídeo en vivo de millones de fuentes, como smartphones, cámaras de seguridad, webcams, cámaras integradas en vehículos, drones y otras fuentes. Amazon Kinesis Video Streams le permite reproducir vídeo para su visualización en directo y bajo demanda, así como crear rápidamente aplicaciones que aprovechen la visión por ordenador y el análisis de vídeo mediante la integración con Amazon Rekognition Video y bibliotecas para marcos ML. También puede enviar datos serializados en el tiempo que no sean de vídeo, como, por ejemplo, datos de audio, imágenes térmicas, datos de profundidad, datos de radar y mucho más.

Amazon Kinesis Video Streams con WebRTC

Amazon Kinesis Video Streams con WebRTC proporciona una implementación de WebRTC que cumple con los estándares como una capacidad totalmente administrada. Puede utilizar Amazon Kinesis Video Streams con WebRTC para transmitir contenido multimedia en directo de forma segura o realizar interacciones bidireccionales de audio o vídeo entre cualquier cámara, dispositivo IoT y reproductores web o móviles compatibles con WebRTC. Al tratarse de una capacidad totalmente gestionada, no es necesario crear, operar ni escalar ninguna infraestructura de nube relacionada con WebRTC, como servidores de señalización o retransmisión de contenido multimedia, para transmitir contenido multimedia de forma segura entre aplicaciones y dispositivos. Con Amazon Kinesis Video Streams con WebRTC, puede crear fácilmente aplicaciones para la transmisión de contenido multimedia peer-to-peer en directo o la interactividad de audio o vídeo en tiempo real entre cámaras, dispositivos IoT, navegadores web y dispositivos móviles para diversos casos de uso.

AWS IoT Eventos

AWS IoT Events detecta y responde a los eventos de los sensores y aplicaciones de IoT. Los eventos son patrones de datos que identifican circunstancias más complicadas de lo esperado, como los detectores de movimiento que utilizan señales de movimiento para activar las luces y las cámaras de seguridad. AWS IoT Events monitorea continuamente los datos de varios sensores y aplicaciones de IoT y se integra con otros servicios, como IoT AWS IoT Core SiteWise, DynamoDB y otros, para permitir la detección temprana y obtener información única.

AWS IoT FleetWise

AWS IoT FleetWisees un servicio gestionado que puede utilizar para recopilar y transferir datos de vehículos a la nube prácticamente en tiempo real. Con AWS IoT FleetWise él, puede recopilar y organizar fácilmente los datos de los vehículos que utilizan diferentes protocolos y formatos de datos. AWS IoT FleetWise ayuda a transformar los mensajes de bajo nivel en valores legibles para las personas y a estandarizar el formato de los datos en la nube para el análisis de los datos. También puede definir esquemas de recopilación de datos para controlar qué datos recopilar en los vehículos y cuándo transferirlos a la nube.

AWS IoT SiteWise

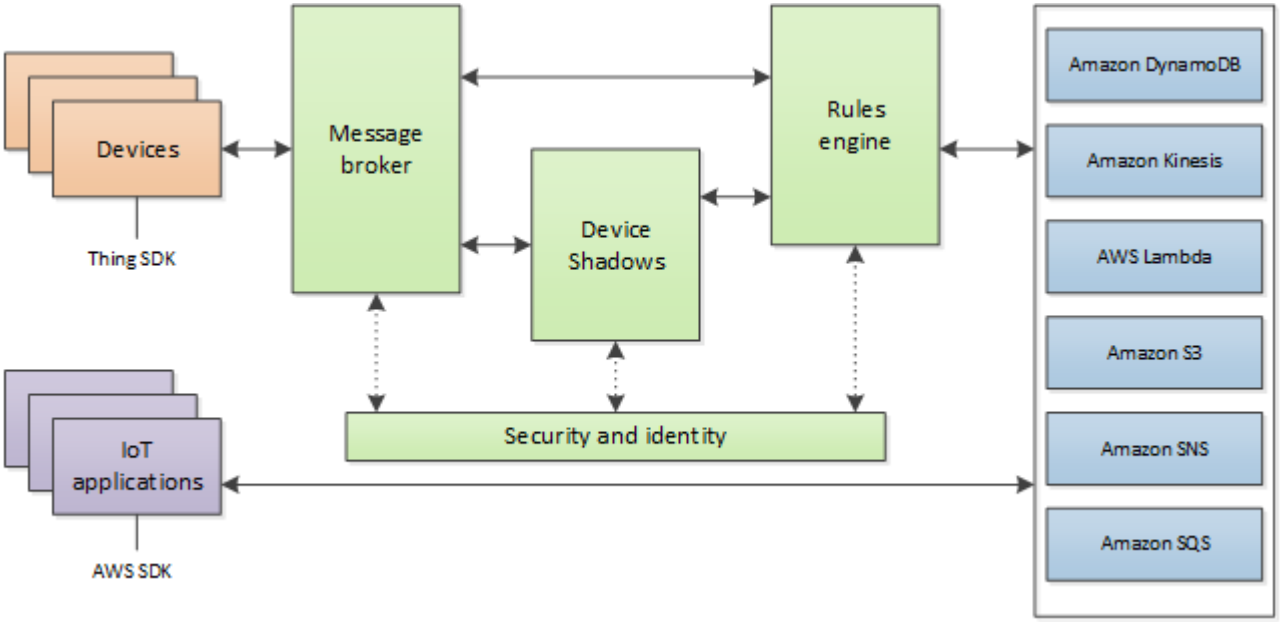
AWS IoT SiteWiserecopila, almacena, organiza y supervisa los datos transmitidos desde los equipos industriales mediante mensajes MQTT o APIs a escala, proporcionando un software que se ejecuta en una puerta de enlace en sus instalaciones. La pasarela se conecta de forma segura a los servidores de datos locales y automatiza el proceso de recopilación y organización de los datos y su envío a la nube. AWS

AWS IoT TwinMaker

AWS IoT TwinMakercrea gemelos digitales operativos de sistemas físicos y digitales. AWS IoT TwinMaker crea visualizaciones digitales mediante mediciones y análisis de una variedad de sensores, cámaras y aplicaciones empresariales del mundo real para ayudarlo a realizar un seguimiento de su fábrica, edificio o planta industrial física. Puede utilizar estos datos del mundo real para supervisar las operaciones, diagnosticar y corregir errores y optimizar las operaciones.

AWS IoT Core servicios

AWS IoT Core proporciona los servicios que conectan sus dispositivos de IoT a la AWS nube para que otros servicios y aplicaciones en la nube puedan interactuar con sus dispositivos conectados a Internet.



La siguiente sección describe cada uno de los AWS IoT Core servicios que se muestran en la ilustración.

AWS IoT Core servicios de mensajería

Los servicios de AWS IoT Core conectividad proporcionan una comunicación segura con los dispositivos de IoT y gestionan los mensajes que pasan entre ellos y AWS IoT.

Gateway de dispositivos

Permite a los dispositivos comunicarse de forma segura y eficaz con AWS IoT. La comunicación de los dispositivos está protegida por protocolos seguros con certificados X.509.

Agente de mensajes

Proporciona un mecanismo seguro para que los dispositivos y AWS IoT las aplicaciones publiquen y reciban mensajes entre sí. Puede utilizar el protocolo MQTT directamente o MQTT Over WebSocket para publicar y suscribirse. Para obtener más información acerca de los protocolos compatibles con AWS IoT , consulte the section called “Protocolos de comunicación de dispositivos”. Los dispositivos y los clientes también pueden usar la interfaz HTTP REST para publicar datos en el agente de mensajes.

El agente de mensajes distribuye los datos del dispositivo a los dispositivos que se han suscrito a él y a otros AWS IoT Core servicios, como el servicio Device Shadow y el motor de reglas.

AWS IoT Core para WAN LoRa

AWS IoT Core para LoRa WAN permite configurar una red LoRa WAN privada conectando sus dispositivos LoRa WAN y puertas de enlace AWS sin necesidad de desarrollar y operar un servidor de red LoRa WAN (LNS). Los mensajes recibidos desde los dispositivos LoRa WAN se envían al motor de reglas, donde se pueden formatear y enviar a otros servicios. AWS IoT

Motor de reglas

El motor de reglas conecta los datos del agente de mensajes con otros servicios AWS IoT para guardarlos y seguir procesándolos. Por ejemplo, puede insertar, actualizar o consultar una tabla de DynamoDB o invocar una función de Lambda con base en una expresión que haya definido en el motor de reglas. Puede utilizar un lenguaje basado en SQL para seleccionar datos de cargas de mensajes y procesar y enviar datos a otros servicios, como Amazon Simple Storage Service (Amazon S3), Amazon DynamoDB y AWS Lambda. También puede crear reglas que vuelvan a publicar mensajes en el agente de mensajes y en otros suscriptores. Para obtener más información, consulte Reglas para AWS IoT.

AWS IoT Core servicios de control

Los servicios AWS IoT Core de control proporcionan funciones de seguridad, administración y registro de los dispositivos.

Servicio de autenticación personalizado

Puede definir autorizadores personalizados para permitirle administrar su propia estrategia de autenticación y autorización con un servicio de autenticación personalizado y una función de Lambda. Los autorizadores personalizados permiten AWS IoT autenticar los dispositivos y autorizar las operaciones mediante estrategias de autenticación y autorización mediante un token portador.

Los autorizadores personalizados pueden implementar varias estrategias de autenticación; por ejemplo, la verificación con el token web JSON o la llamada al proveedor. OAuth Deben devolver los documentos de política empleados por la puerta de enlace del dispositivo para autorizar las operaciones de MQTT. Para obtener más información, consulte Autenticación y autorización personalizada.

Servicio de aprovisionamiento de dispositivos

Le permite aprovisionar dispositivos mediante una plantilla que describe los recursos necesarios para el dispositivo: un objeto, un certificado y una o varias políticas. Un objeto es una entrada en el registro que contiene atributos que describen un dispositivo. Los dispositivos utilizan certificados para autenticarse. AWS IoT Las políticas determinan qué operaciones pueden realizar los dispositivos en AWS IoT.

Las plantillas contienen variables que se sustituirán por los valores en un diccionario (mapa). Puede usar la misma plantilla para aprovisionar varios dispositivos tan solo pasando diferentes valores para las variables de la plantilla en el diccionario. Para obtener más información, consulte Aprovisionamiento de dispositivos.

Registro de grupos

Los grupos permiten administrar varios dispositivos a la vez clasificándolos en grupos. Los grupos también pueden contener otros grupos; puede crear una jerarquía de grupos. Cualquier acción que realice en un grupo principal se aplicará a sus grupos secundarios. La misma acción también se aplica a todos los dispositivos del grupo principal y a todos los dispositivos de los grupos secundarios. Los permisos otorgados a un grupo se aplicarán a todos los dispositivos del grupo y a todos sus grupos secundarios. Para obtener más información, consulte Administrar dispositivos con AWS IoT.

Servicio de Jobs

Permite definir un conjunto de operaciones remotas que se envían a uno o más dispositivos conectados a AWS IoT o que se ejecutan en uno o más de esos dispositivos. Por ejemplo, puede definir un trabajo que indique a un conjunto de dispositivos descargar e instalar actualizaciones

de firmware y aplicaciones, reiniciar, rotar certificados o realizar operaciones remotas de solución de problemas.

Para crear un trabajo, especifique una descripción de las operaciones remotas que se van a realizar y una lista de destinos que deben realizarlas. Los destinos pueden ser dispositivos individuales, grupos o ambos. Para obtener más información, consulte AWS IoT Empleos.

Registro

Organiza los recursos asociados a cada dispositivo en la nube de AWS . Es necesario registrar los dispositivos y asociar hasta tres atributos personalizados a cada uno. También puede asociar los certificados y el cliente MQTT IDs a cada dispositivo para mejorar su capacidad de administrarlos y solucionar sus problemas. Para obtener más información, consulte Administrar dispositivos con AWS IoT.

Servicio de seguridad e identidad

Proporciona una responsabilidad compartida en materia de seguridad en la AWS nube. Los dispositivos deben proteger sus 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 . Para obtener más información, consulte Autenticación.

AWS IoT Core servicios de datos

AWS IoT Core Los servicios de datos ayudan a sus soluciones de IoT a ofrecer una experiencia de aplicación fiable incluso con dispositivos que no siempre están conectados.

Sombra del dispositivo

Documento JSON utilizado para almacenar y recuperar información del estado actual de un dispositivo.

Servicio Device Shadow

El servicio de sombra de dispositivo mantiene el estado de un dispositivo para que las aplicaciones puedan comunicarse con él, ya sea que el dispositivo esté en línea o no. Cuando un dispositivo está desconectado, el servicio de sombra de dispositivo administra los datos de las aplicaciones conectadas. Cuando el dispositivo se vuelve a conectar, sincroniza su estado con el de su sombra en el servicio de sombra de dispositivo. Los dispositivos también pueden publicar su estado actual en una sombra para que lo usen las aplicaciones o los demás dispositivos que

quizá no estén siempre conectados. Para obtener más información, consulte AWS IoT Servicio Device Shadow.

AWS IoT Core servicio de soporte

Integración de Amazon Sidewalk para AWS IoT Core

Amazon Sidewalk es una red compartida que mejora las opciones de conectividad para que los dispositivos funcionen mejor juntos. Amazon Sidewalk es compatible con una amplia gama de dispositivos de clientes para distintas aplicaciones, como localizar mascotas u objetos de valor, ofrecer seguridad inteligente y control de la iluminación para el hogar o hacer diagnósticos remotos de electrodomésticos y herramientas. La integración de Amazon Sidewalk for AWS IoT Core permite a los fabricantes de dispositivos añadir su flota de dispositivos Sidewalk a la AWS IoT nube.

Para obtener más información, consulte AWS IoT Core para Amazon Sidewalk.

Obtenga más información sobre AWS IoT

Este tema le ayuda a familiarizarse con el mundo de AWS IoT. Puede obtener información general sobre cómo se aplican las soluciones de IoT en diversos casos de uso, recursos de formación, enlaces a redes sociales AWS IoT y todos los demás AWS servicios, y una lista de los servicios y protocolos de comunicación que AWS IoT utiliza.

Recursos de formación para AWS IoT

Ofrecemos estos cursos de formación para ayudarle a aprender sobre ellos AWS IoT y cómo aplicarlos al diseño de su solución.

- Introducción a AWS IoT

  Un resumen en vídeo AWS IoT de sus servicios principales.
- Profundice en la AWS IoT autenticación y la autorización

  Un curso avanzado que explora los conceptos de AWS IoT autenticación y autorización. Aprenderá cómo autenticar y autorizar a los clientes a acceder al plano de AWS IoT control y al plano de datos. APIs
- Internet of Things Foundation Series

Una ruta de aprendizaje electrónico con módulos sobre distintas tecnologías y características de IoT.

AWS IoT recursos y guías

Estos son recursos técnicos detallados sobre aspectos específicos de AWS IoT.

- IoT Lens — AWS IoT Well-Architected Framework

  Un documento que describe las mejores prácticas para diseñar la arquitectura de sus aplicaciones de IoT en AWS.
- Diseño de temas de MQTT para AWS IoT Core

  Un documento técnico que describe las mejores prácticas para diseñar temas de MQTT AWS IoT Core y aprovechar AWS IoT Core las funciones con MQTT.
- Resumen e introducción

  Un documento PDF que describe las diferentes formas de aprovisionar grandes AWS IoT flotas de dispositivos.
- AWS IoT Core Asesor de dispositivos

  AWS IoT Core Device Advisor proporciona pruebas prediseñadas que puede usar para validar los dispositivos de IoT y así aplicar las mejores prácticas de conectividad confiables y seguras antes de implementar los dispositivos en producción. AWS IoT Core
- AWS IoT Recursos

  Los recursos específicos del IoT, como guías técnicas, arquitecturas de referencia, libros electrónicos y publicaciones de blog seleccionadas, se presentan en un índice con capacidad de búsqueda.
- IoT Atlas

  Información general sobre cómo resolver problemas comunes de diseño de IoT. El IoT Atlas analiza en profundidad los desafíos de diseño a los que probablemente se enfrentará al desarrollar su solución de IoT.
- AWS Documentos técnicos y guías

  Nuestra colección actual de libros blancos y guías sobre AWS IoT y otras AWS tecnologías.

AWS IoT en las redes sociales

Estos canales de redes sociales proporcionan información sobre AWS IoT temas AWS relacionados.

- El Internet de las cosas está en AWS IoT marcha: blog oficial
- AWS IoT vídeos en el canal Amazon Web Services en YouTube

Estas cuentas de redes sociales cubren todos los AWS servicios, incluidos AWS IoT

- El canal Amazon Web Services en YouTube
- Amazon Web Services en Twitter
- Amazon Web Services en Facebook
- Amazon Web Services en Instagram
- Amazon Web Services en LinkedIn

AWS servicios utilizados por el motor de AWS IoT Core reglas

El motor de AWS IoT Core reglas puede conectarse a estos AWS servicios.

- Amazon DynamoDB

  Amazon DynamoDB es un servicio de base de datos de NoSQL escalable que ofrece un rendimiento rápido y predecible.
- Amazon Kinesis

  Amazon Kinesis facilita la recopilación, el procesamiento y el análisis de datos de streaming en tiempo real para que pueda obtener información oportuna y reaccionar rápidamente ante la nueva información. Amazon Kinesis puede ingerir datos en tiempo real, como vídeo, audio, registros de aplicaciones, secuencias de clics de sitios web y datos de telemetría de IoT para aplicaciones de machine learning, análisis, etc.
- AWS Lambda

  AWS Lambda le permite ejecutar código sin aprovisionar ni administrar servidores. Puedes configurar tu código para que se active automáticamente a partir de AWS IoT datos y eventos o llamarlo directamente desde una aplicación web o móvil.
- Amazon Simple Storage Service

Amazon Simple Storage Service (Amazon S3) puede almacenar y recuperar cualquier cantidad de datos en cualquier momento y desde cualquier lugar de la web. AWS IoT las reglas pueden enviar datos a Amazon S3 para su almacenamiento.

- Amazon Simple Notification Service

  Amazon Simple Notification Service (Amazon SNS) es un servicio web que permite a las aplicaciones, los usuarios finales y los dispositivos enviar y recibir notificaciones desde la nube.

- Amazon Simple Queue Service

  Amazon Simple Queue Service (Amazon SQS) es un servicio de colas de mensajes que desacopla y escala microservicios, sistemas distribuidos y aplicaciones sin servidor.

- OpenSearch Servicio Amazon

  Amazon OpenSearch Service (OpenSearch Service) es un servicio gestionado que facilita la implementación, el funcionamiento y el escalado OpenSearch, un popular motor de búsqueda y análisis de código abierto.

- Amazon SageMaker AI

  Amazon SageMaker AI puede crear modelos de aprendizaje automático (ML) mediante la búsqueda de patrones en sus datos de IoT. El servicio utiliza estos modelos para procesar nuevos datos y generar predicciones para su aplicación.

- Amazon CloudWatch

  Amazon CloudWatch proporciona una solución de monitorización fiable, escalable y flexible para ayudarte a configurar, gestionar y escalar tus propios sistemas e infraestructuras de monitorización.

Protocolos de comunicación compatibles con AWS IoT Core

En estos temas, se proporciona más información sobre los protocolos de comunicación que utiliza AWS IoT. Para obtener más información sobre los protocolos que utilizan los dispositivos AWS IoT y servicios y a los que se conectan AWS IoT, consulteConnect to AWS IoT Core.

- MQTT (Message Queuing Telemetry Transport)

La página principal del sitio MQTT.org donde puede encontrar las especificaciones del protocolo MQTT. Para obtener más información sobre cómo AWS IoT es compatible con MQTT, consulteMQTT.

- HTTPS (Hypertext Transfer Protocol - Secure)

  Los dispositivos y las aplicaciones pueden acceder a AWS IoT los servicios mediante HTTPS.

- LoRaWAN (red de área amplia y largo alcance)

  LoRaLos dispositivos WAN y las puertas de enlace se pueden conectar AWS IoT Core mediante AWS IoT Core LoRa WAN.

- TLS (Transport Layer Security) v1.3

  La especificación del TLS v1.3 (RFC 5246). AWS IoT usa TLS v1.3 para establecer conexiones seguras entre dispositivos y. AWS IoT

Novedades de la consola de AWS IoT

Estamos actualizando la interfaz de usuario de la consola AWS IoT para adaptarla a una nueva experiencia. Estamos actualizando la interfaz de usuario por etapas, de modo que algunas páginas de la consola incluyan una nueva experiencia, otras tengan la experiencia original y la nueva y otras solo tengan la experiencia original.

En esta tabla, se muestra el estado de las distintas áreas de la interfaz de usuario de la consola AWS IoT con fecha de 27 de enero de 2022.

Estado de la interfaz de usuario de la consola AWS IoT

Página de la consola	Experiencia original	Nueva experiencia	Comentarios
Supervisión de	No disponible	Disponible	
Actividad	No disponible	Disponible	
Incorporación (Introducción)	No disponible	Disponible	No disponible en regiones CN

Página de la consola	Experiencia original	Nueva experiencia	Comentarios
Incorporación (Plantillas de aprovisionamiento de flotas)	Disponible	Disponible	
Administración (Objetos)	Disponible	Disponible	
Administración (Tipos)	Disponible	Disponible	
Administración (Grupos de objetos)	Disponible	Disponible	
Administración (Grupos de facturación)	Disponible	Disponible	
Administración (Trabajos)	Disponible	Disponible	
Administración (Plantillas de trabajo)	No disponible	Disponible	
Administración (Túneles)	No disponible	Disponible	
Fleet Hub (Introducción)	No disponible	Disponible	No disponible en todas las Regiones de AWS
Fleet Hub (Aplicaciones)	No disponible	Disponible	No disponible en todas las Regiones de AWS
Greengrass (Introducción)	No disponible	Disponible	No disponible en todas las Regiones de AWS

Página de la consola	Experiencia original	Nueva experiencia	Comentarios
Greengrass (Dispositivos de núcleo)	No disponible	Disponible	No disponible en todas las Regiones de AWS
Greengrass (Componentes)	No disponible	Disponible	No disponible en todas las Regiones de AWS
Greengrass (Despliegues)	No disponible	Disponible	No disponible en todas las Regiones de AWS
Greengrass (Clásico (V1))	Disponible	Disponible	
Conectividad inalámbrica (Introducción)	No disponible	Disponible	No disponible en todas las Regiones de AWS
Conectividad inalámbrica (Gateways)	No disponible	Disponible	No disponible en todas las Regiones de AWS
Conectividad inalámbrica (Dispositivos)	No disponible	Disponible	No disponible en todas las Regiones de AWS
Conectividad inalámbrica (Perfiles)	No disponible	Disponible	No disponible en todas las Regiones de AWS
Conectividad inalámbrica (Destinos)	No disponible	Disponible	No disponible en todas las Regiones de AWS
Seguridad (Certificados)	Disponible	Disponible	

Página de la consola	Experiencia original	Nueva experiencia	Comentarios
Seguridad (Políticas)	Disponible	Disponible	
Seguridad (Entidades de certificación)	Disponible	Disponible	
Seguridad (Alias de rol)	Disponible	Disponible	
Seguridad (Autorizadores)	Disponible	Disponible	
Defender (Introducción)	No disponible	Disponible	
Defender (Auditoría)	No disponible	Disponible	
Defender (Detectar)	No disponible	Disponible	
Defender (Acciones de mitigación)	No disponible	Disponible	
Defender (Configuración)	No disponible	Disponible	
Actuar (Reglas)	Disponible	Disponible	
Actuar (Destinos)	Disponible	Disponible	
Probar (Device Advisor)	Disponible	Disponible	No disponible en todas las Regiones de AWS
Probar (Cliente de prueba de MQTT)	Disponible	Disponible	
Software	Disponible	Disponible	
Configuración	No disponible	Disponible	

Página de la consola	Experiencia original	Nueva experiencia	Comentarios
Aprender	Disponible	Aún no está disponible	

Leyenda

Valores de los estados

- Disponible

  Esta experiencia de interfaz de usuario se puede utilizar.
- No disponible

  Esta experiencia de interfaz de usuario no se puede utilizar.
- Aún no está disponible

  La nueva experiencia de interfaz de usuario se está desarrollando, pero aún no está lista.
- En curso

  La nueva experiencia de interfaz de usuario está en proceso de actualización. Sin embargo, es posible que algunas páginas sigan teniendo la experiencia de usuario original.

Se utiliza AWS IoT con un SDK AWS

AWS Los kits de desarrollo de software (SDKs) están disponibles para muchos lenguajes de programación populares. Cada SDK proporciona una API, ejemplos de código y documentación que facilitan a los desarrolladores la creación de aplicaciones en su lenguaje preferido.

Documentación de SDK	Ejemplos de código
AWS SDK para C++	AWS SDK para C++ ejemplos de código
AWS CLI	AWS CLI ejemplos de código
AWS SDK for Go	AWS SDK for Go ejemplos de código

Documentación de SDK	Ejemplos de código
AWS SDK for Java	AWS SDK for Java ejemplos de código
AWS SDK para JavaScript	AWS SDK para JavaScript ejemplos de código
AWS SDK para Kotlin	AWS SDK para Kotlin ejemplos de código
AWS SDK for .NET	AWS SDK for .NET ejemplos de código
AWS SDK for PHP	AWS SDK for PHP ejemplos de código
AWS Tools for PowerShell	AWS Tools for PowerShell ejemplos de código
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) ejemplos de código
AWS SDK for Ruby	AWS SDK for Ruby ejemplos de código
AWS SDK para Rust	AWS SDK para Rust ejemplos de código
AWS SDK para SAP ABAP	AWS SDK para SAP ABAP ejemplos de código
AWS SDK para Swift	AWS SDK para Swift ejemplos de código

Ejemplo de disponibilidad

¿No encuentra lo que necesita? Solicite un ejemplo de código a través del enlace de Enviar comentarios que se encuentra al final de esta página.

Cómo empezar con AWS IoT Core los tutoriales

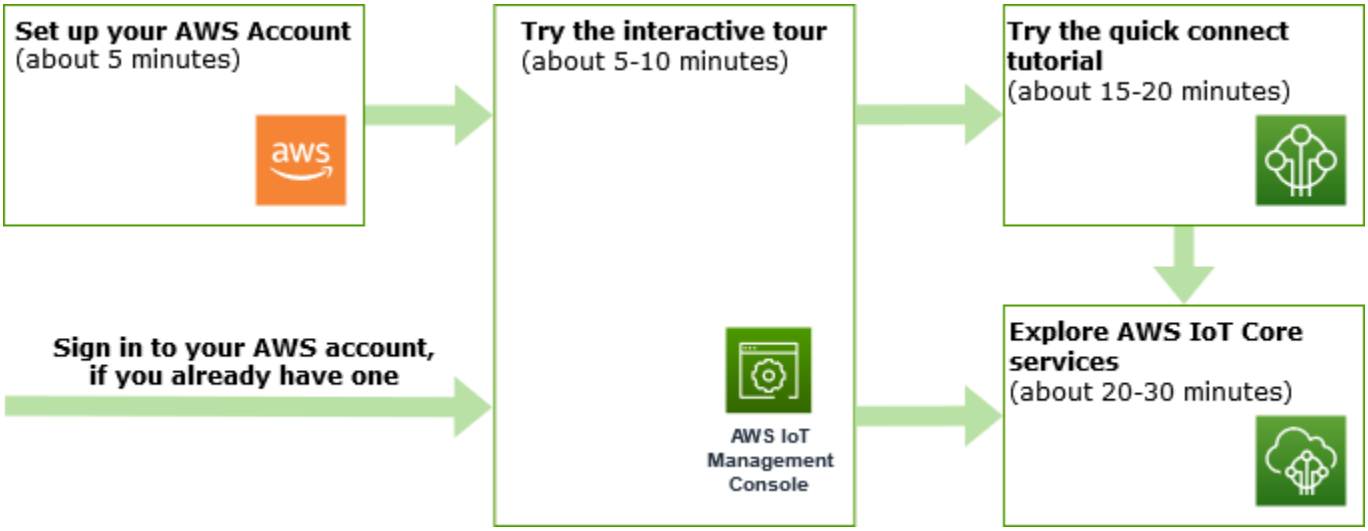
Tanto si eres nuevo en el IoT como si tienes años de experiencia, estos recursos presentan los AWS IoT conceptos y términos que te ayudarán a empezar a utilizarlos AWS IoT.

- Mire por dentro AWS IoT y por dentro sus componentes¿Cómo AWS IoT funciona.
- Obtenga más información sobre AWS IoT en nuestra colección de materiales y vídeos de formación. Este tema también incluye una lista de los servicios a los que AWS IoT se puede conectar, enlaces a redes sociales y enlaces a las especificaciones de los protocolos de comunicación.
- the section called “Conecta tu primer dispositivo a AWS IoT Core”.
- Desarrolle sus soluciones de IoT mediante la Connect to AWS IoT Core y explore los AWS IoT tutoriales.
- Pruebe y valide sus dispositivos IoT para una comunicación segura y fiable mediante con Asesor de dispositivos.
- Administre la solución mediante servicios de administración de AWS IoT Core , como, por ejemplo, Indexación de flotas, AWS IoT Empleos y AWS IoT Device Defender.
- Analice los datos de sus dispositivos mediante los AWS IoT servicios de datos.

Conecta tu primer dispositivo a AWS IoT Core

AWS IoT Core los servicios conectan los dispositivos de IoT a AWS IoT servicios y otros AWS servicios. AWS IoT Core incluye la puerta de enlace del dispositivo y el intermediario de mensajes, que conectan y procesan los mensajes entre sus dispositivos de IoT y la nube.

A continuación te explicamos cómo puedes empezar con AWS IoT Core y AWS IoT.



En esta sección se presenta un recorrido por sus principales servicios y se proporcionan varios ejemplos de cómo conectar un dispositivo AWS IoT Core y pasar mensajes entre ellos. AWS IoT Core Pasar mensajes entre los dispositivos y la nube es fundamental para todas las soluciones de IoT y es la forma en que sus dispositivos pueden interactuar con otros AWS servicios.

- Configurar Cuenta de AWS

  Antes de poder utilizar AWS IoT los servicios, debe configurar un Cuenta de AWS. Si ya tiene un usuario de IAM Cuenta de AWS y uno para usted, puede usarlos y omitir este paso.

- Pruebe el tutorial de conexión rápida

  Este tutorial es el mejor si quieres empezar rápidamente AWS IoT y ver cómo funciona en un escenario limitado. En este tutorial, necesitarás un dispositivo e instalarás algún AWS IoT software en él. Si no tiene un dispositivo IoT, puede usar un ordenador personal Windows, Linux o macOS como dispositivo para este tutorial. Si quieres intentarlo AWS IoT, pero no tienes un dispositivo, prueba la siguiente opción.

- Probar el tutorial interactivo

  Esta demostración es la mejor si quiere ver lo que puede hacer una AWS IoT solución básica sin conectar un dispositivo ni descargar ningún software. El tutorial interactivo presenta una solución simulada basada en AWS IoT Core servicios que ilustra cómo interactúan.

- Explore AWS IoT Core los servicios con un tutorial práctico

  Este tutorial es ideal para los desarrolladores que quieren empezar con él, ya AWS IoT que pueden seguir explorando otras AWS IoT Core funciones, como el motor de reglas y las sombras.

Este tutorial sigue un proceso similar al tutorial de conexión rápida, pero proporciona más detalles sobre cada paso para permitir una transición más fluida a los tutoriales más avanzados.

- Vea los mensajes MQTT con el cliente AWS IoT MQTT

  Aprenda a usar el cliente de pruebas MQTT para ver sus primeros mensajes MQTT de publicación de dispositivo en AWS IoT. El cliente de pruebas MQTT es una herramienta útil para supervisar y solucionar problemas de las conexiones de los dispositivos.

> **Note**
>
> Si desea probar más de uno de estos tutoriales de iniciación o repetir el mismo tutorial, deberá eliminar el objeto que creó en un tutorial anterior antes de empezar otro. Si no elimina el objeto de un tutorial anterior, tendrá que usar un nombre diferente para los siguientes tutoriales. Esto se debe a que el nombre del objeto debe ser exclusivo de su cuenta y su Región de AWS.

Para obtener más información al respecto AWS IoT Core, consulte ¿Qué es? AWS IoT Core

Configurar Cuenta de AWS

Antes de usarlo AWS IoT Core por primera vez, complete las siguientes tareas:

Temas

- Inscríbase en una Cuenta de AWS
- Creación de un usuario con acceso administrativo
- Abre la AWS IoT consola

Inscríbase en una Cuenta de AWS

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

Para suscribirte a una Cuenta de AWS

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

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

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

AWS te envía un correo electrónico de confirmación una vez finalizado el proceso de registro. En cualquier momento, puedes ver la actividad actual de tu cuenta y administrarla accediendo a https://aws.amazon.com/y seleccionando Mi cuenta.

Creación de un usuario con acceso administrativo

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

Proteja su Usuario raíz de la cuenta de AWS

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

   Para obtener ayuda para iniciar sesión con el usuario raíz, consulte Iniciar sesión como usuario raíz en la Guía del usuario de AWS Sign-In .
2. Active la autenticación multifactor (MFA) para el usuario raíz.

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

Creación de un usuario con acceso administrativo

1. Activar IAM Identity Center.

   Consulte las instrucciones en Activar AWS IAM Identity Center en la Guía del usuario de AWS IAM Identity Center .
2. En IAM Identity Center, conceda acceso administrativo a un usuario.

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

Inicio de sesión como usuario con acceso de administrador

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

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

Concesión de acceso a usuarios adicionales

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

   Para conocer las instrucciones, consulte Create a permission set en la Guía del usuario de AWS IAM Identity Center .
2. Asigne usuarios a un grupo y, a continuación, asigne el acceso de inicio de sesión único al grupo.

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

- Abre la AWS IoT consola

Si ya tiene un usuario Cuenta de AWS y un usuario para usted, puede usarlos y pasar athe section called “Abre la AWS IoT consola”.

Abre la AWS IoT consola

La mayoría de los temas orientados a la consola de esta sección comienzan desde la AWS IoT consola. Si aún no has iniciado sesión en la tuya Cuenta de AWS, inicia sesión, abre la AWS IoT consola y continúa con la siguiente sección para empezar a usarla. AWS IoT

Tutorial interactivo

El tutorial interactivo muestra los componentes de una solución de IoT sencilla basada en AWS IoT. El tutorial muestra cómo los dispositivos de IoT interactúan con AWS IoT Core los servicios. En este tema se proporciona una vista previa del tutorial AWS IoT Core interactivo.

> **Note**
>
> Las imágenes de la consola incluyen animaciones que no aparecen en las imágenes de este tutorial.

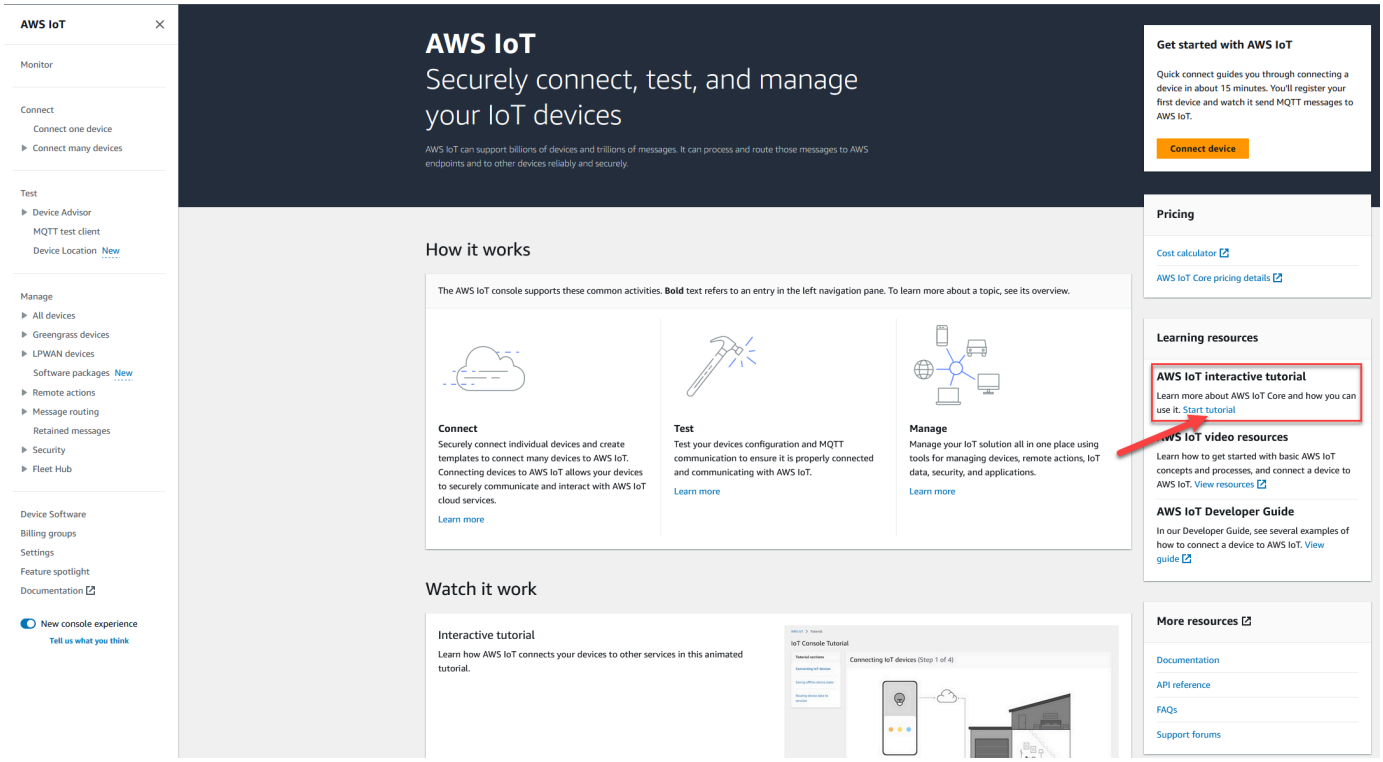
Para ejecutar la demostración, primero debe the section called “Configurar Cuenta de AWS”. Sin embargo, el tutorial no requiere AWS IoT recursos, software adicional ni programación.

Calcule dedicar unos 5-10 minutos a esta demostración. Si le dedica 10 minutos, le dará más tiempo para comprender cada uno de los pasos.

Para ejecutar el tutorial AWS IoT Core interactivo

1. Abra la página de AWS IoT inicio en la AWS IoT consola.

   En la página de inicio de AWS IoT , en el panel de la ventana Recursos de aprendizaje, seleccione Iniciar tutorial.



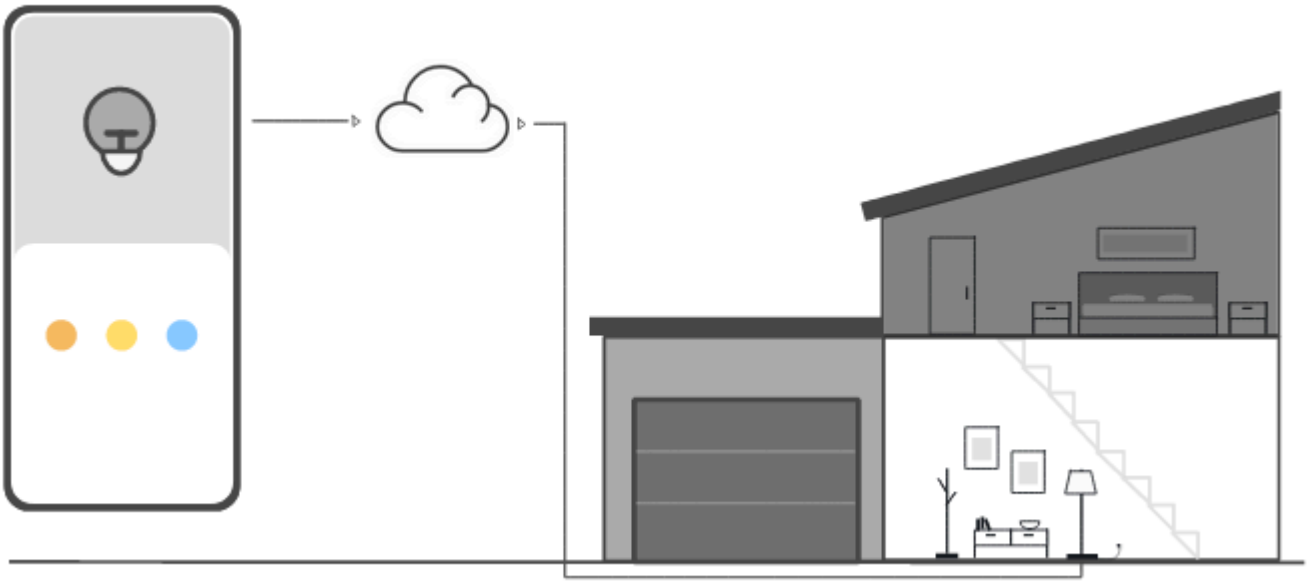
2. En la página Tutorial de la consola de AWS IoT , revise las secciones del tutorial y seleccione Comenzar sección cuando esté listo para continuar.

En las siguientes secciones se describe cómo el tutorial de la AWS IoT consola presenta estas AWS IoT Core funciones:

- Conexión de dispositivos IoT
- Guardar el estado del dispositivo desconectado
- Enrutamiento de los datos del dispositivo a los servicios

Conexión de dispositivos IoT

Descubra cómo se comunican los dispositivos de IoT con AWS IoT Core.

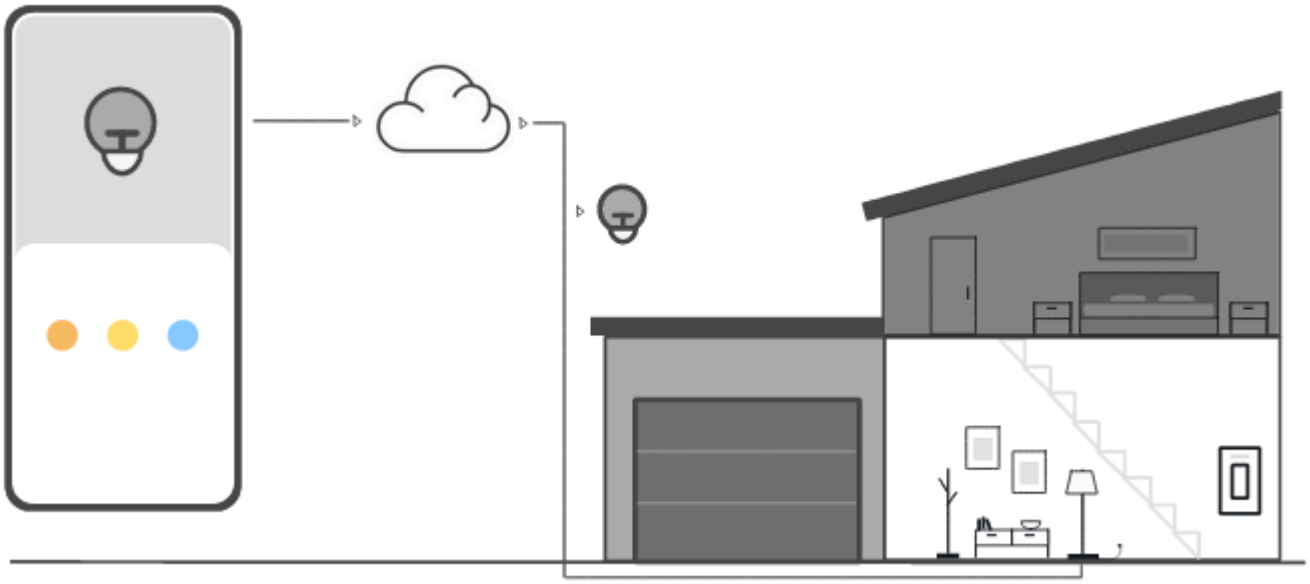


La animación de este paso muestra cómo dos dispositivos, el dispositivo de control a la izquierda y una lámpara inteligente doméstica a la derecha, se conectan y se comunican con AWS IoT Core en la nube. La animación muestra los dispositivos comunicándose con los mensajes que reciben AWS IoT Core y reaccionando ante ellos.

Para obtener más información sobre cómo conectar dispositivos a AWS IoT Core, consulte Connect to AWS IoT Core.

Guardar el estado del dispositivo desconectado

Obtenga información sobre cómo AWS IoT Core guardar el estado del dispositivo mientras un dispositivo o una aplicación están desconectados.



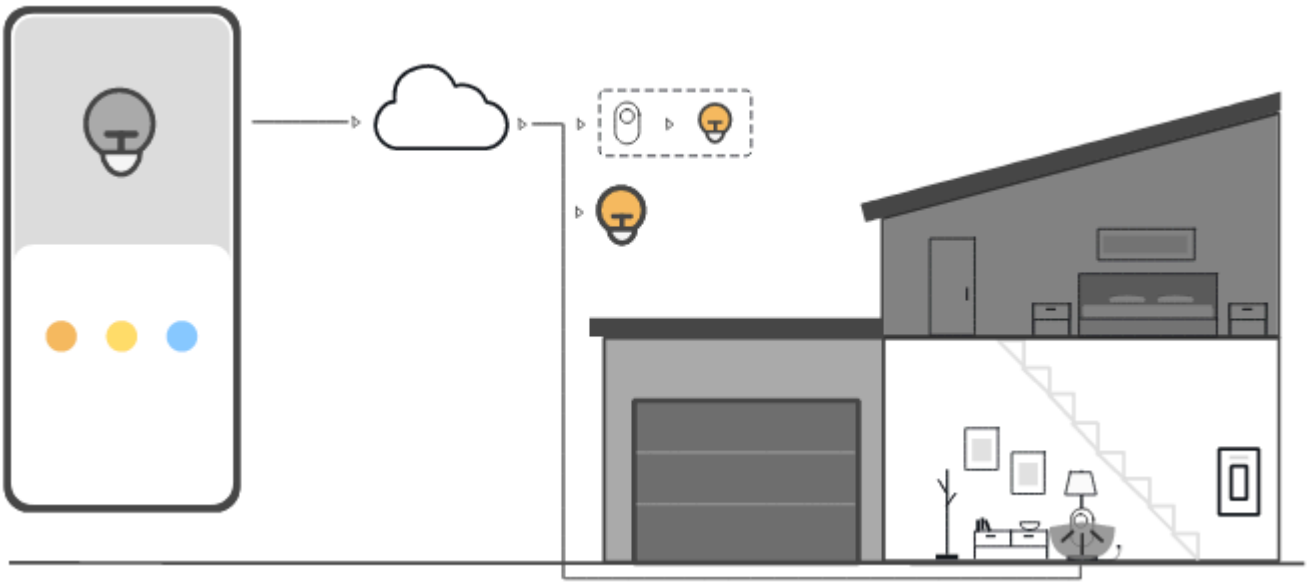
La animación de este paso muestra cómo el servicio Device Shadow AWS IoT Core guarda la información de estado del dispositivo para el dispositivo de control y la lámpara inteligente. Mientras la lámpara inteligente está apagada, Device Shadow guarda los comandos del dispositivo de control.

Cuando la lámpara inteligente se vuelve a conectar AWS IoT Core, recupera esos comandos. Cuando el dispositivo de control está desconectado, Device Shadow guarda la información de estado de la lámpara inteligente. Cuando el dispositivo de control se vuelve a conectar, recupera el estado actual de la lámpara inteligente para actualizar su estado.

Si desea más información sobre Device Shadows, consulte AWS IoT Servicio Device Shadow.

Enrutamiento de los datos del dispositivo a los servicios

Obtén información sobre cómo AWS IoT Core envía el estado del dispositivo a otros AWS servicios.



La animación de este paso muestra cómo se AWS IoT Core envían los datos desde los dispositivos a otros AWS servicios mediante AWS IoT reglas. AWS IoT las reglas se suscriben a mensajes específicos de los dispositivos, interpretan los datos de esos mensajes y envían los datos interpretados a otros servicios. En este ejemplo, una AWS IoT regla interpreta los datos de un sensor de movimiento y envía comandos a un Device Shadow, que luego los envía a la bombilla inteligente. Como en el ejemplo anterior, Device Shadow almacena la información de estado del dispositivo de control.

Para obtener más información sobre AWS IoT las reglas, consulteReglas para AWS IoT.

Pruebe el tutorial de conexión AWS IoT Core rápida

En este tutorial, creará su primer objeto, conectará un dispositivo a él y verá cómo envía mensajes MQTT.

Calcule dedicar unos 15-20 minutos a este tutorial.

Este tutorial es el mejor para las personas que quieren empezar rápidamente AWS IoT a ver cómo funciona en un escenario limitado. Si está buscando un ejemplo que le ayude a explorar más características y servicios, pruebe Explora AWS IoT Core en tutoriales prácticos.

En este tutorial, descargará y ejecutará software en un dispositivo que se conecta a un recurso de cosa AWS IoT Core como parte de una solución de IoT muy pequeña. El dispositivo puede ser un

dispositivo IoT, como una Raspberry Pi, o también puede ser un ordenador que ejecute Linux, OS y OSX, o Windows. Si quieres conectar un dispositivo WAN (WAN) de largo alcance a una LoRa WAN AWS IoT, consulta el tutorial >Conexión de dispositivos y puertas de enlace a AWS IoT Core una WAN. LoRa

Si el dispositivo es compatible con un navegador que pueda ejecutar la consola de AWS IoT, le recomendamos que realice este tutorial en ese dispositivo.

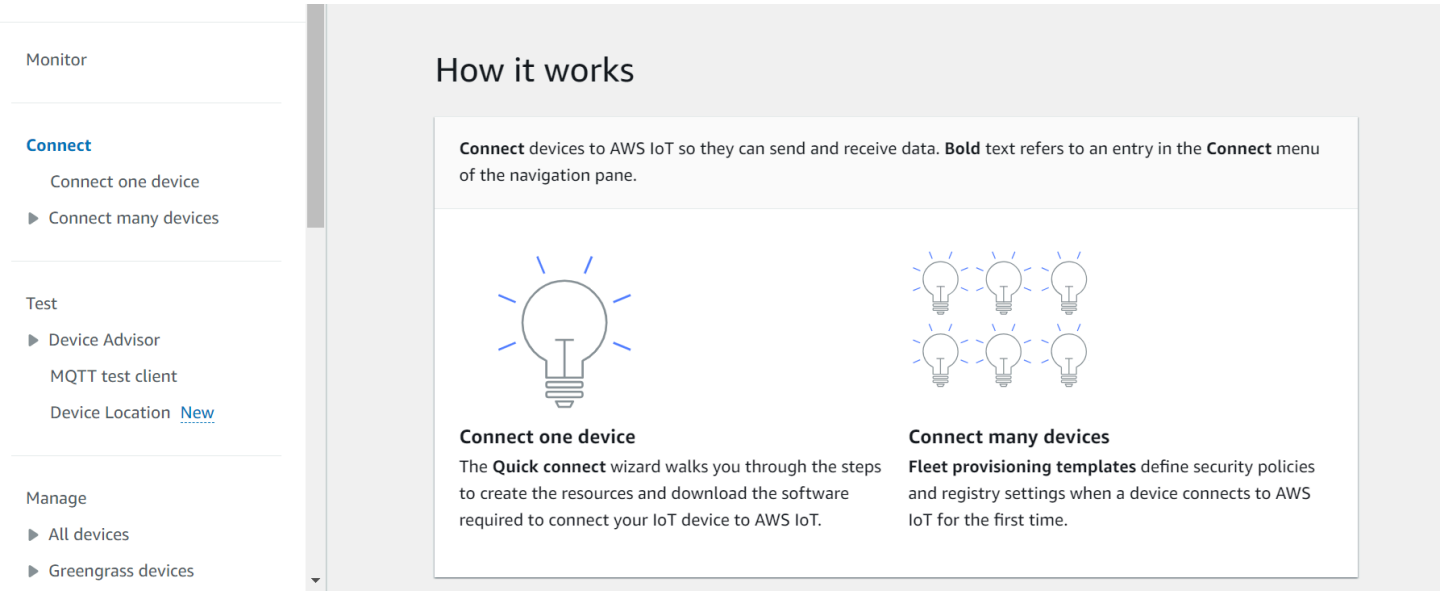
> **Note**
>
> Si su dispositivo no tiene un navegador compatible, realice este tutorial en un ordenador. Cuando el procedimiento le pida que descargue el archivo, descárguelo al ordenador y, a continuación, transfiera el archivo descargado al dispositivo mediante Secure Copy (SCP) o un proceso similar.

El tutorial requiere que su dispositivo IoT se comunique con el puerto 8443 del punto de conexión de datos de dispositivo de la Cuenta de AWS. Para comprobar si puede acceder a ese puerto, pruebe los procedimientos que se indican en Prueba de la conectividad con el punto de conexión de datos del dispositivo.

Paso 1. Comenzar el tutorial

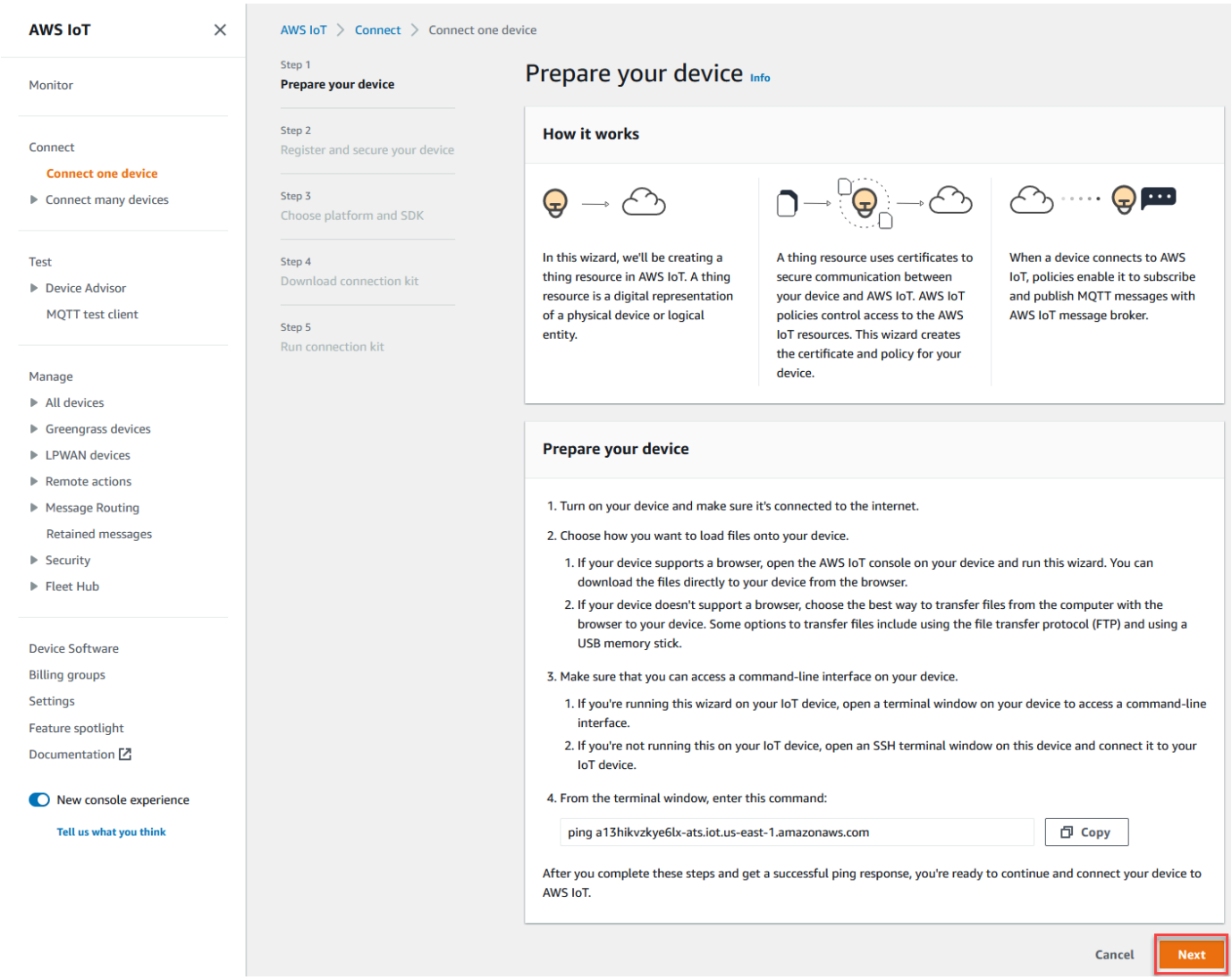
Si es posible, realice este procedimiento en el dispositivo; de lo contrario, prepárese para transferir un archivo al dispositivo más adelante durante el procedimiento.

Para comenzar el tutorial, inicie sesión en la consola de AWS IoT. En la página de inicio de la AWS IoT consola, a la izquierda, selecciona Conectar y, a continuación, selecciona Conectar un dispositivo.



Paso 2. Crear un objeto

1. En la sección Preparar su dispositivo, siga las instrucciones que aparecen en pantalla para preparar el dispositivo y conectarlo a AWS IoT.

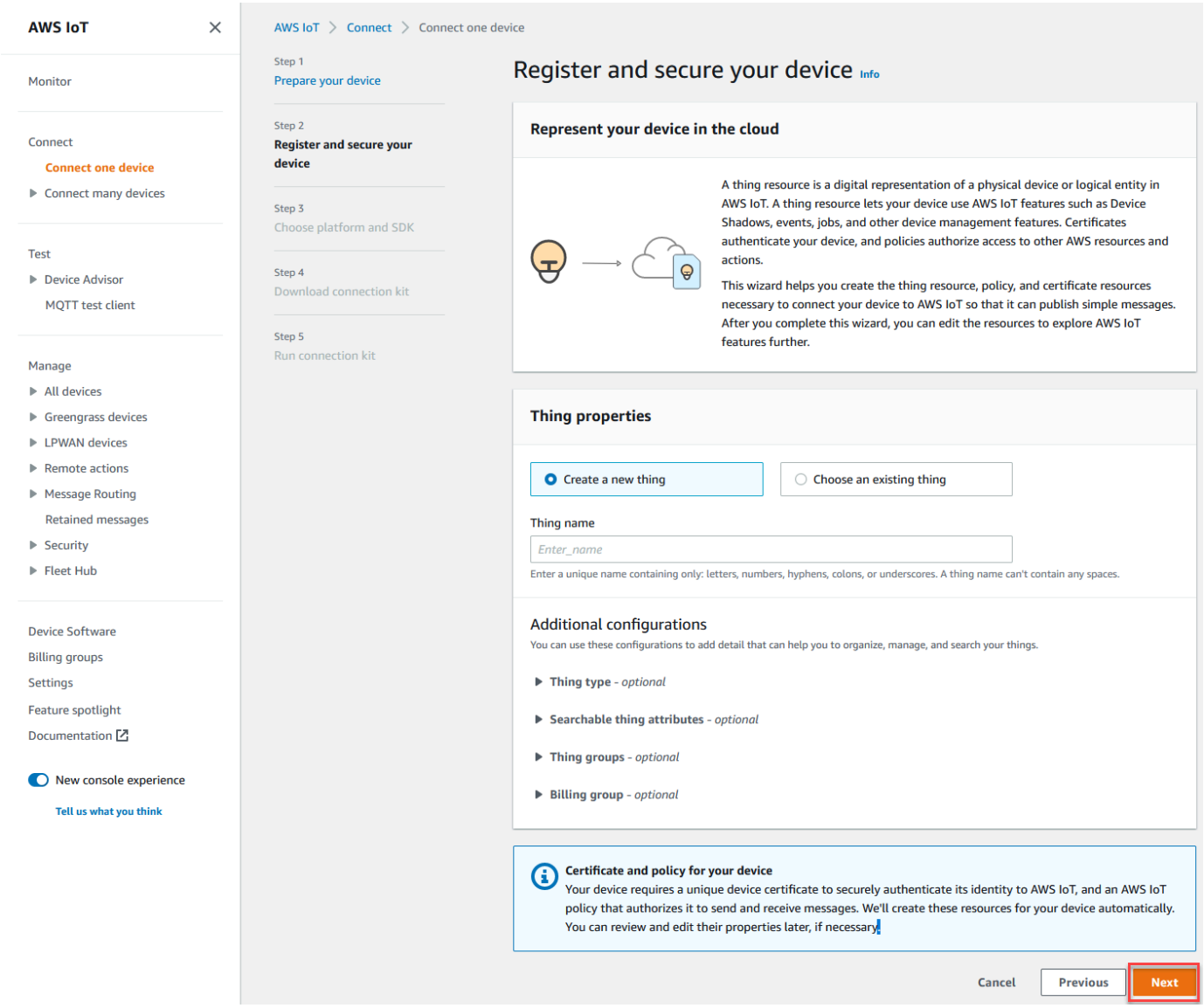


2. En la sección Registrar y proteger su dispositivo, seleccione Crear un nuevo objeto o Elegir un objeto existente. En el campo Nombre del objeto, introduzca el nombre del objeto. El nombre del objeto utilizado en este ejemplo es **`TutorialTestThing`**.

> **Important**
>
> Compruebe el nombre del objeto antes de continuar.
> El nombre de un objeto no se puede cambiar una vez creado el objeto. Si desea cambiar el nombre de un objeto, debe crear un objeto nuevo con el nombre correcto y eliminar después el objeto con el nombre incorrecto.

En la sección Configuraciones adicionales, personalice aún más su recurso de objeto utilizando las configuraciones opcionales que se muestran.

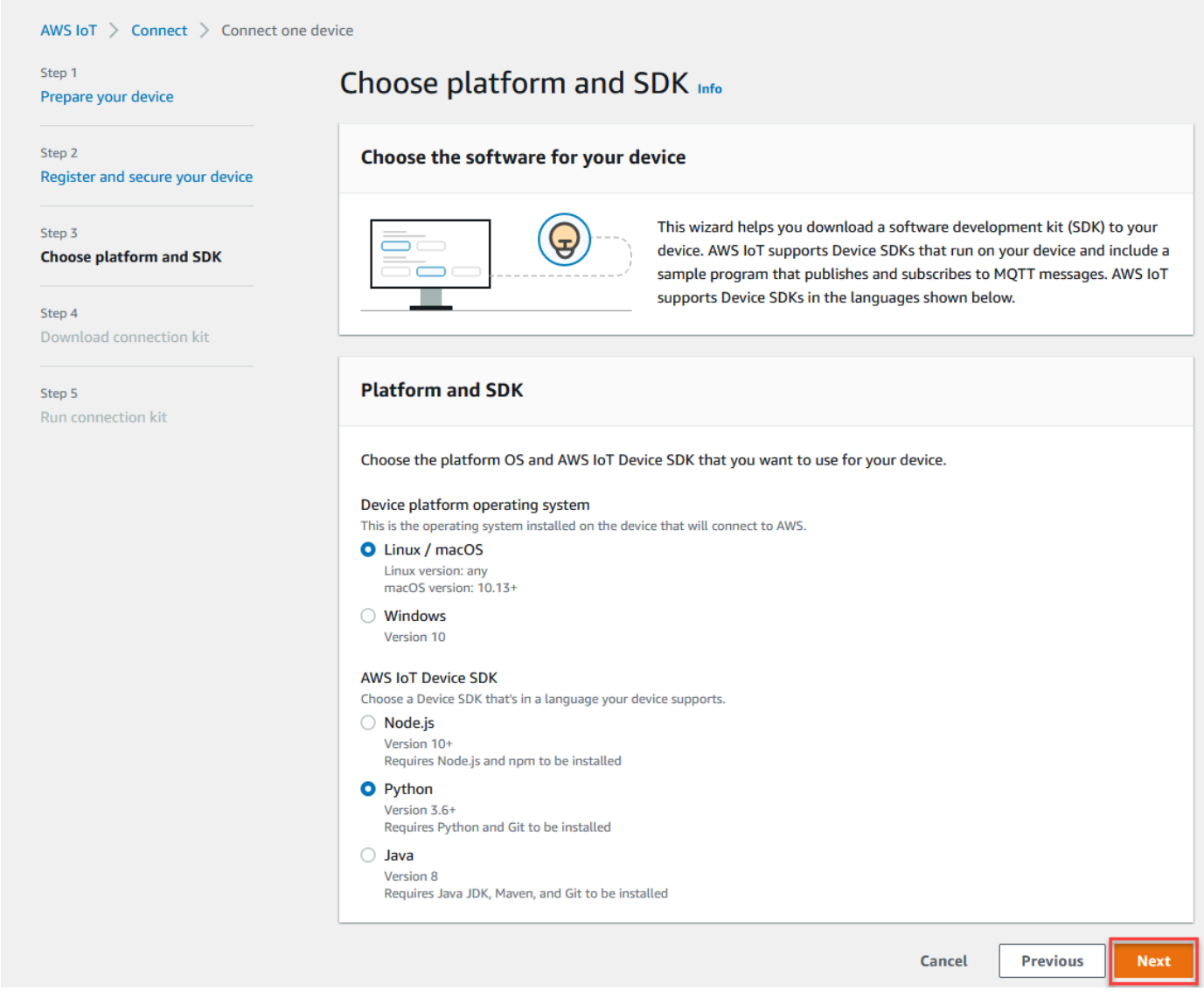
Tras asignar un nombre al objeto y seleccionar cualquier configuración adicional, elija Siguiente.



3. En la sección Elegir plataforma y SDK, elige la plataforma y el idioma del SDK del AWS IoT dispositivo que quieres usar. En este ejemplo, se utiliza la Linux/OSX plataforma y el SDK de Python. Asegúrese de tener python3 y pip3 instalados en el dispositivo de destino antes de continuar con el siguiente paso.

> **Note**
>
> Asegúrese de consultar la lista de requisitos de software necesarios para el SDK que haya elegido en la parte inferior de la página de la consola.
> Debe tener el software necesario instalado en el ordenador de destino para poder continuar con el siguiente paso.

Tras elegir el idioma del SDK de la plataforma y el dispositivo, seleccione Siguiente.

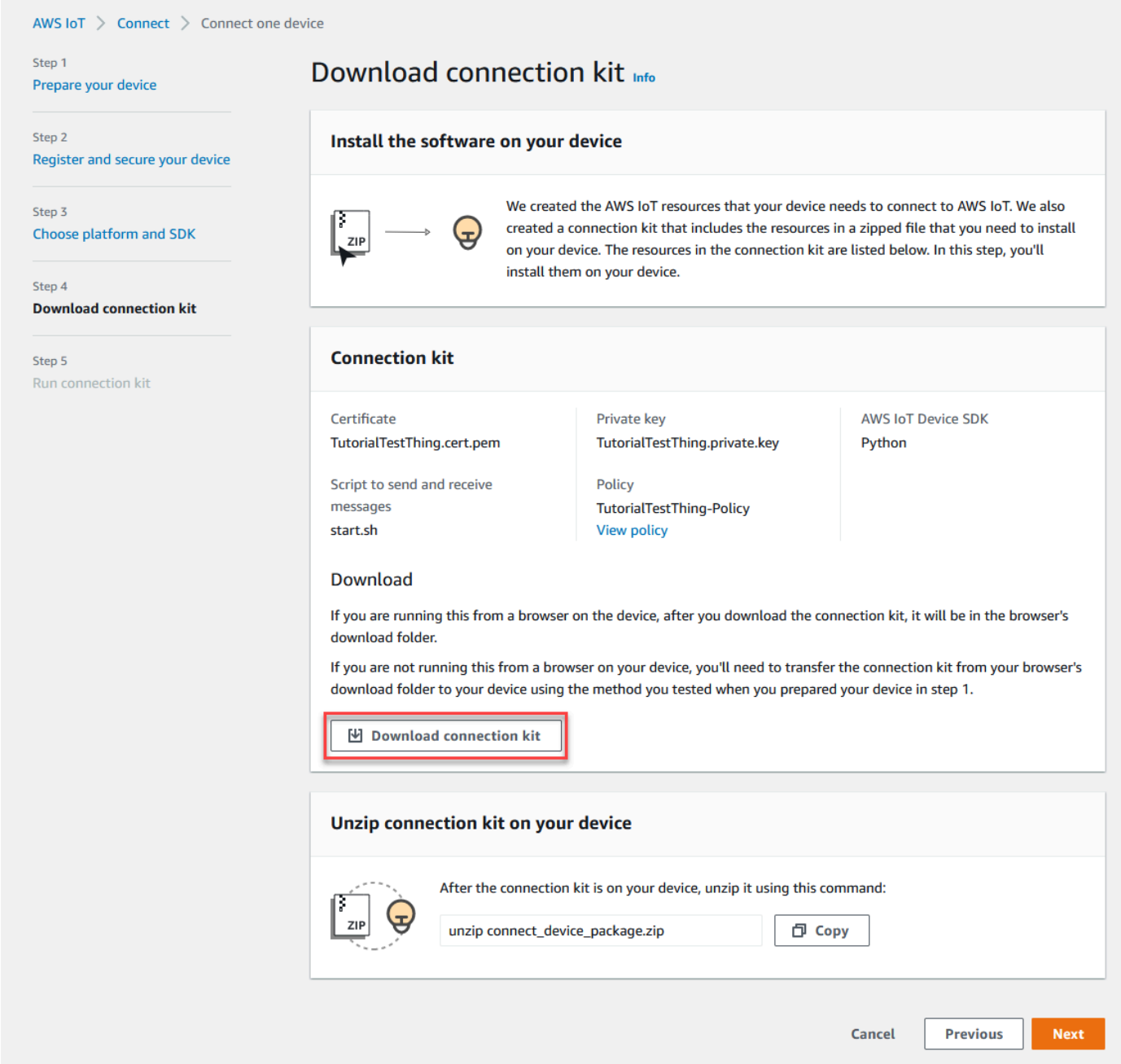


Paso 3. Descargar archivos en su dispositivo

Esta página aparece después de AWS IoT crear el kit de conexión, que incluye los siguientes archivos y recursos que requiere el dispositivo:

- Los archivos de certificado del objeto que se utilizan para autenticar el dispositivo.
- Un recurso de política para autorizar la interacción entre el objeto y AWS IoT.
- El script para descargar el SDK del AWS dispositivo y ejecutar el programa de muestra en el dispositivo

1. Cuando esté listo para continuar, elija el botón Descargar el kit de conexión para con el fin de descargar el kit de conexión para la plataforma que eligió anteriormente.



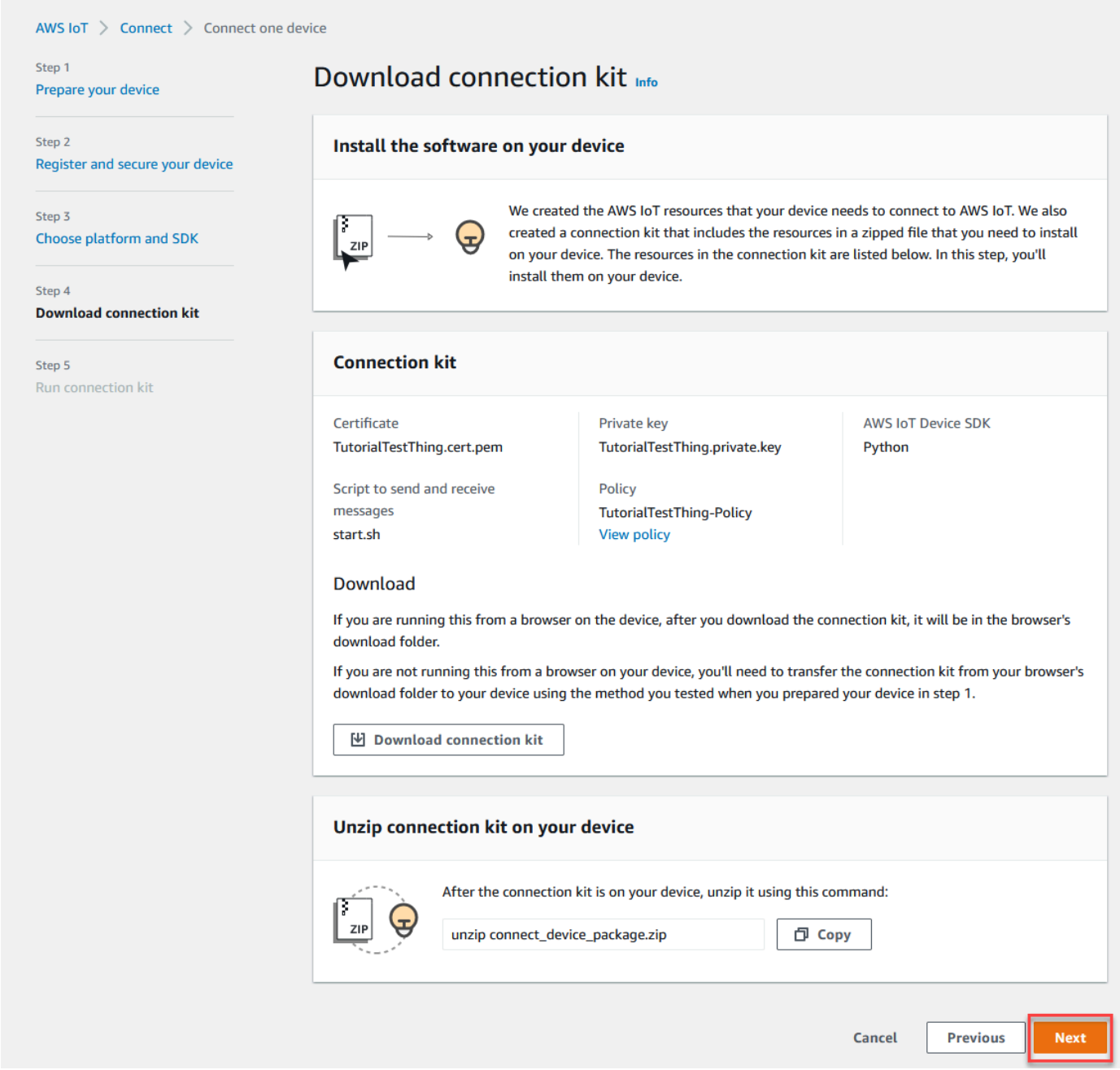
2. Si está ejecutando este procedimiento en su dispositivo, guarde el archivo del kit de conexión en un directorio desde el que pueda ejecutar los comandos de la línea de comandos.

   Si no va a ejecutar este procedimiento en el dispositivo, guarde el archivo del kit de conexión en un directorio local y, a continuación, transfiéralo al dispositivo.

3. En la sección Descomprima el kit de conexión en su dispositivo, introduzca unzip connect_device_package.zip en el directorio en el que se encuentran los archivos del kit de conexión.

Si utilizas una ventana de PowerShell comandos de Windows y el unzip comando no funciona, unzip expand-archive sustitúyelo por la línea de comandos y vuelve a intentarlo.

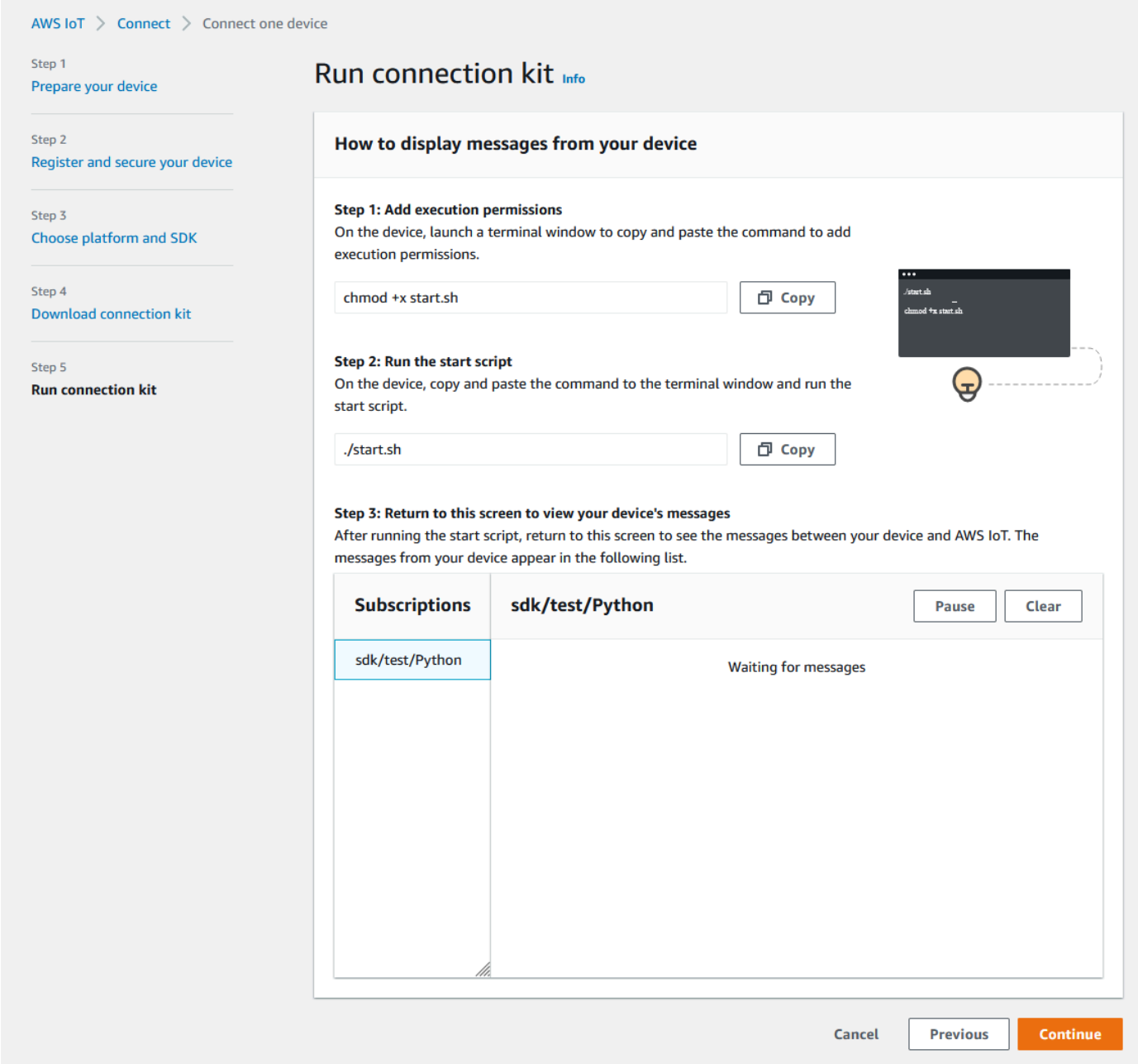
4. Una vez que tenga el archivo del kit de conexión en el dispositivo, continúe con el tutorial seleccionando Siguiente.



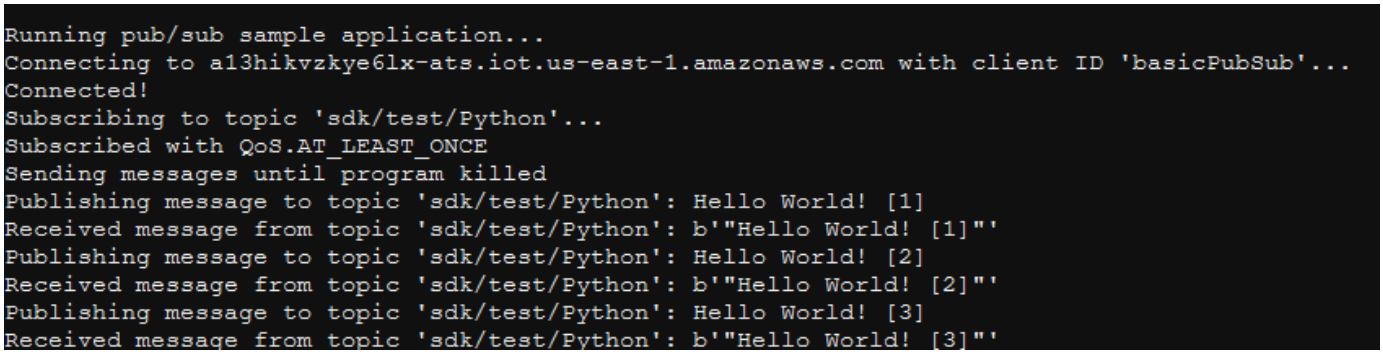
Paso 4. Ejecutar la muestra

Realice este procedimiento en una ventana de comandos o del terminal en el dispositivo mientras sigue las instrucciones que aparecen en la consola. Los comandos que ve en la consola son para el sistema operativo que ha elegido en the section called “Paso 2. Crear un objeto”. Los que se muestran aquí son para los sistemas Linux/OSX operativos.

1. En una terminal o ventana de comandos de su dispositivo, en el directorio que contiene el archivo del kit de conexión, lleve a cabo los pasos que se muestran en la AWS IoT consola.



2. Tras introducir el comando del paso 2 en la consola, debería aparecer un resultado similar al siguiente en la ventana de comandos o del terminal del dispositivo. Este resultado proviene de los mensajes que el programa envía y recibe en AWS IoT Core.



   Mientras se ejecuta el programa de ejemplo, también aparecerá el mensaje de prueba `Hello World!`. El mensaje de prueba aparece en la ventana de comandos o del terminal del dispositivo.

   > **Note**
   >
   > Para obtener más información sobre la suscripción y publicación de temas, consulte el código de ejemplo del SDK que haya elegido.

3. Para volver a ejecutar el programa de ejemplo, puede repetir los comandos del paso 2 en la consola de este procedimiento.

4. (Opcional) Si desea ver los mensajes de su cliente de IoT en la AWS IoT consola, abra el cliente de prueba de MQTT en la página de pruebas de la AWS IoT consola. Si ha elegido el SDK de Python, en el cliente de prueba MQTT, en Filtro de temas, introduzca el tema, por ejemplo **`sdk/test/python`**, para suscribirse a los mensajes de su dispositivo. Los filtros de temas distinguen entre mayúsculas y minúsculas y dependen del lenguaje de programación del SDK que haya elegido en el paso 1. Para obtener más información sobre la suscripción y publicación de temas, consulte el código de ejemplo del SDK que haya elegido.

5. Tras suscribirse al tema de prueba, ejecute ./start.sh en el dispositivo. Para obtener más información, consulte the section called “Vea los mensajes MQTT con el cliente AWS IoT MQTT”.

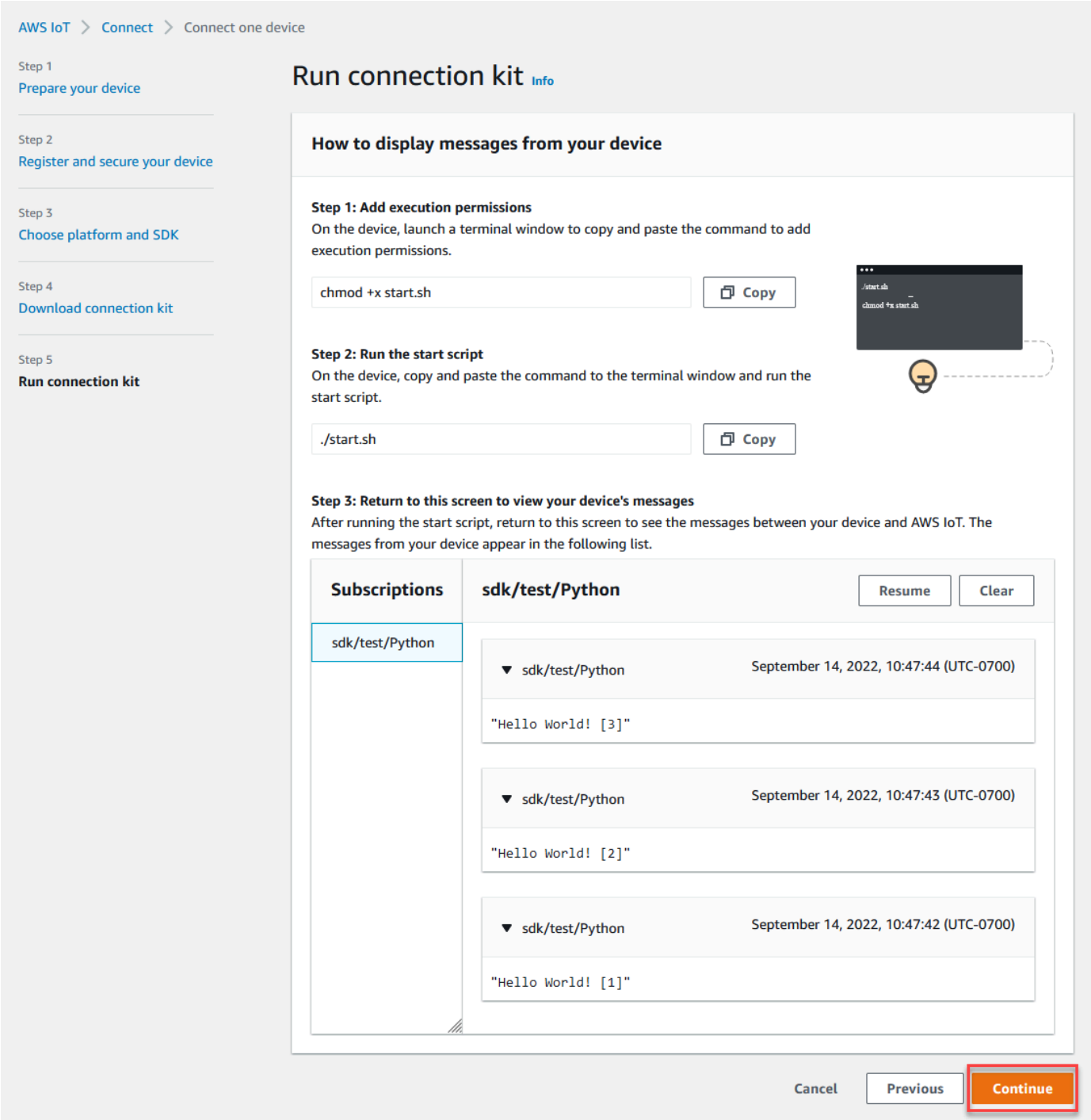
   Tras la ejecución de ./start.sh, aparecen mensajes en el cliente MQTT similares a los siguientes:

```
{
  "message": "Hello World!" [1]
```

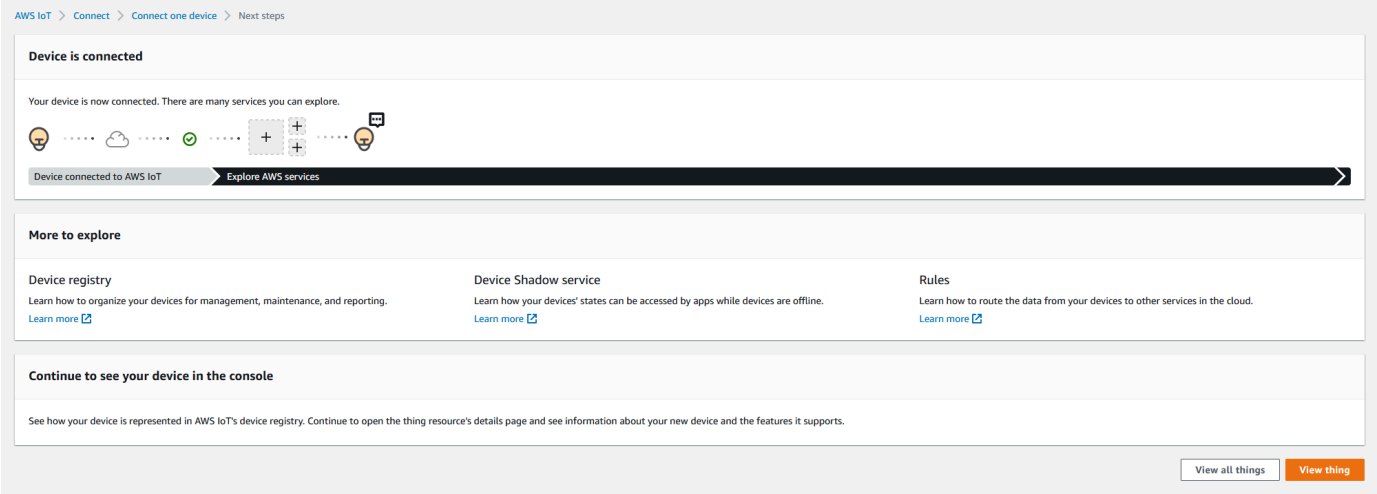
```
}
```

El número de `sequence` incluido en `[]` aumenta en uno cada vez que se recibe un mensaje `Hello World!` nuevo y se detiene al finalizar el programa.

6. Para terminar el tutorial y ver un resumen, en la AWS IoT consola, selecciona Continuar.



7. Ahora aparecerá un resumen del tutorial de conexión AWS IoT rápida.



Paso 5. Seguir explorando

Estas son algunas ideas para explorar AWS IoT más a fondo después de completar el inicio rápido.

- Consultar mensajes MQTT en el cliente de prueba MQTT

  En la consola de AWS IoT, puede abrir el cliente de MQTT en la página Prueba de la consola de AWS IoT . En el cliente de prueba de MQTT, suscríbase a **#**. A continuación, ejecute el programa ./start.sh en su dispositivo tal y como se describe en el paso anterior. Para obtener más información, consulte the section called “Vea los mensajes MQTT con el cliente AWS IoT MQTT”.

- Realizar pruebas en sus dispositivos con Device Advisor

  Utilice Device Advisor para comprobar si sus dispositivos pueden conectarse e interactuar con ellos de forma segura y fiable AWS IoT.

- the section called “Tutorial interactivo”

  Para iniciar el tutorial interactivo, en la página de aprendizaje de la AWS IoT consola, en el icono Vea cómo AWS IoT funciona, seleccione Iniciar el tutorial.

- Prepararse para explorar más tutoriales

  Este inicio rápido solo le ofrece una muestra de AWS IoT. Si desea explorar AWS IoT más a fondo y conocer las características que la convierten en una potente plataforma de soluciones de IoT, comience a preparar su plataforma de desarrollo de la siguiente maneraExplora AWS IoT Core en tutoriales prácticos.

Prueba de la conectividad con el punto de conexión de datos del dispositivo

En este tema se describe cómo probar la conexión de un dispositivo con el punto de conexión de datos de dispositivo de su cuenta, el punto de conexión que sus dispositivos de IoT utilizan para conectarse a AWS IoT.

Realice estos procedimientos en el dispositivo que quiera probar o mediante una sesión de terminal SSH conectada al dispositivo que desee.

Para probar la conectividad de un dispositivo con el punto de conexión de datos del dispositivo siga estos pasos:

- Encontrar el punto de conexión de datos de su dispositivo
- Probar la conexión rápidamente
- Obtener la aplicación para probar la conexión con el punto de conexión de datos del dispositivo y el puerto
- Pruebe la conexión con el punto de conexión de datos del dispositivo y el puerto

Encontrar el punto de conexión de datos de su dispositivo

Este procedimiento explica cómo encontrar el punto de conexión de datos del dispositivo en la consola de AWS IoT para probar la conexión con el dispositivo de IoT.

Para encontrar el punto de conexión de datos de su dispositivo

1. En la consola de AWS IoT, en la sección Conectar, vaya a Configuraciones de dominio.
2. En la página Configuraciones de dominio, vaya al contenedor Configuraciones de dominio y copie Nombre de dominio. El valor de su punto final es exclusivo para Cuenta de AWS usted y es similar al de este ejemplo:`a3qEXAMPLEsffp-ats.iot.eu-west-1.amazonaws.com`.
3. Guarde el punto de conexión de datos del dispositivo para utilizarlo en los procedimientos siguientes.

Probar la conexión rápidamente

Este procedimiento comprueba la conectividad general con el punto de conexión de datos del dispositivo, pero no comprueba el puerto específico que utilizarán los dispositivos. Esta prueba utiliza

un programa común y suele ser suficiente para averiguar si los dispositivos se pueden conectar a AWS IoT.

Si quiere probar la conectividad con el puerto específico que utilizarán sus dispositivos, omita este procedimiento y continúe con Obtener la aplicación para probar la conexión con el punto de conexión de datos del dispositivo y el puerto.

Para probar rápidamente el punto de conexión de datos del dispositivo

1. En una ventana del terminal o de la línea de comandos del dispositivo, sustituya el punto de conexión de datos del dispositivo de ejemplo (*a3qEXAMPLEsffp-ats.iot.eu-west-1.amazonaws.com*) por el punto de conexión de datos del dispositivo de su cuenta y, a continuación, introduzca este comando.

   Linux

   ```
   ping -c 5 a3qEXAMPLEsffp-ats.iot.eu-west-1.amazonaws.com
   ```

   Windows

   ```
   ping -n 5 a3qEXAMPLEsffp-ats.iot.eu-west-1.amazonaws.com
   ```

2. Si `ping` muestra una salida similar a la siguiente, significa que se ha conectado correctamente al punto de conexión de datos del dispositivo. Si bien no se comunicó AWS IoT directamente, encontró el servidor y es probable que AWS IoT esté disponible a través de este punto final.

   ```
   PING a3qEXAMPLEsffp-ats.iot.eu-west-1.amazonaws.com (xx.xx.xxx.xxx) 56(84) bytes of
    data.
   64 bytes from ec2-EXAMPLE-218.eu-west-1.compute.amazonaws.com (xx.xx.xxx.xxx):
    icmp_seq=1 ttl=231 time=127 ms
   64 bytes from ec2-EXAMPLE-218.eu-west-1.compute.amazonaws.com (xx.xx.xxx.xxx):
    icmp_seq=2 ttl=231 time=127 ms
   64 bytes from ec2-EXAMPLE-218.eu-west-1.compute.amazonaws.com (xx.xx.xxx.xxx):
    icmp_seq=3 ttl=231 time=127 ms
   64 bytes from ec2-EXAMPLE-218.eu-west-1.compute.amazonaws.com (xx.xx.xxx.xxx):
    icmp_seq=4 ttl=231 time=127 ms
   64 bytes from ec2-EXAMPLE-218.eu-west-1.compute.amazonaws.com (xx.xx.xxx.xxx):
    icmp_seq=5 ttl=231 time=127 ms
   ```

   Si le satisface este resultado, puede detener las pruebas aquí.

Si desea probar la conectividad con el puerto específico que utiliza AWS IoT, continúe con Obtener la aplicación para probar la conexión con el punto de conexión de datos del dispositivo y el puerto.

3. Si `ping` no da un resultado correcto, compruebe el valor del punto de conexión para asegurarse de es el correcto y compruebe la conexión del dispositivo a internet.

Obtener la aplicación para probar la conexión con el punto de conexión de datos del dispositivo y el puerto

Se puede realizar una prueba de conectividad más exhaustiva utilizando `nmap`. Este procedimiento comprueba si `nmap` está instalado en el dispositivo.

Para comprobar si **`nmap`** está en el dispositivo

1. En una ventana del terminal o de la línea de comandos del dispositivo que desee probar, introduzca este comando para comprobar si `nmap` está instalado.

   ```
   nmap --version
   ```

2. Si se incluye un resultado similar al siguiente, `nmap` está instalado y puede continuar con the section called “Pruebe la conexión con el punto de conexión de datos del dispositivo y el puerto”.

   ```
   Nmap version 6.40 ( http://nmap.org )
   Platform: x86_64-koji-linux-gnu
   Compiled with: nmap-liblua-5.2.2 openssl-1.0.2k libpcre-8.32 libpcap-1.5.3 nmap-
   libdnet-1.12 ipv6
   Compiled without:
   Available nsock engines: epoll poll select
   ```

3. Si no se incluye una respuesta similar a la del paso anterior, deberá instalar `nmap` en el dispositivo. Seleccione el procedimiento correspondiente al sistema operativo de su dispositivo.

Linux

Este procedimiento requiere tener permiso para instalar software en el equipo.

Para instalar nmap en su ordenador Linux

1. En una ventana del terminal o de la línea de comandos de su dispositivo, introduzca el comando que corresponda a la versión de Linux que esté ejecutando.

   a. Debian o Ubuntu:

   ```
   sudo apt install nmap
   ```

   b. CentOS o RHEL:

   ```
   sudo yum install nmap
   ```

2. Pruebe la instalación con este comando:

   ```
   nmap --version
   ```

3. Si se incluye un resultado similar al siguiente, nmap está instalado y puede continuar con the section called “Pruebe la conexión con el punto de conexión de datos del dispositivo y el puerto”.

   ```
   Nmap version 6.40 ( http://nmap.org )
   Platform: x86_64-koji-linux-gnu
   Compiled with: nmap-liblua-5.2.2 openssl-1.0.2k libpcre-8.32 libpcap-1.5.3 nmap-
   libdnet-1.12 ipv6
   Compiled without:
   Available nsock engines: epoll poll select
   ```

macOS

Este procedimiento requiere tener permiso para instalar software en el equipo.

Para instalar nmap en su ordenador macOS

1. En un navegador, abra https://nmap.org/download#macosx y descargue el instalador estable más reciente.

   Cuando se te pida, selecciona Abrir con DiskImageInstaller.

2. En la ventana de instalación, mueva el paquete a la carpeta Aplicaciones.

3. En el Finder, localice el paquete `nmap-xxxx-mpkg` en la carpeta Aplicaciones. Haga Ctrl-click en el paquete correspondiente y seleccione Abrir para abrirlo.
4. Revise el cuadro de diálogo de seguridad. Si está todo preparado para instalar nmap, seleccione Abrir para instalar nmap.
5. En Terminal, pruebe la instalación con este comando:

   ```
   nmap --version
   ```

6. Si se incluye un resultado similar al siguiente, `nmap` está instalado y puede continuar con the section called “Pruebe la conexión con el punto de conexión de datos del dispositivo y el puerto”.

   ```
   Nmap version 7.92 ( https://nmap.org )
   Platform: x86_64-apple-darwin17.7.0
   Compiled with: nmap-liblua-5.3.5 openssl-1.1.1k nmap-libssh2-1.9.0 libz-1.2.11
    nmap-libpcre-7.6 nmap-libpcap-1.9.1 nmap-libdnet-1.12 ipv6 Compiled without:
   Available nsock engines: kqueue poll select
   ```

Windows

Este procedimiento requiere tener permiso para instalar software en el equipo.

Para instalar nmap en su ordenador Windows

1. En un navegador, abra https://nmap.org/download#windows y descargue la última versión estable del programa de configuración.

   Si se le solicita, seleccione Guardar archivo. Una vez descargado el archivo, ábralo desde la carpeta de descargas.
2. Cuando el archivo de configuración termine de descargarse, abra el archivo nmap-xxxx-setup.exe descargado para instalar la aplicación.
3. Acepte la configuración predeterminada a medida que se instala el programa.

   No necesita la aplicación Npcap para esta prueba. Puede deseleccionar esa opción si no quiere instalarla.
4. En Command, pruebe la instalación con este comando:

```
nmap --version
```

5. Si se incluye un resultado similar al siguiente, nmap está instalado y puede continuar con the section called “Pruebe la conexión con el punto de conexión de datos del dispositivo y el puerto”.

```
Nmap version 7.92 ( https://nmap.org )
Platform: i686-pc-windows-windows
Compiled with: nmap-liblua-5.3.5 openssl-1.1.1k nmap-libssh2-1.9.0 nmap-
libz-1.2.11 nmap-libpcre-7.6 Npcap-1.50 nmap-libdnet-1.12 ipv6
Compiled without:
Available nsock engines: iocp poll select
```

Pruebe la conexión con el punto de conexión de datos del dispositivo y el puerto

Este procedimiento comprueba la conexión del dispositivo de IoT con el punto de conexión de datos del dispositivo con el puerto seleccionado.

Para probar la conexión con el punto de conexión de datos del dispositivo y el puerto

1. En una ventana del terminal o de la línea de comandos del dispositivo, sustituya el punto de conexión de datos del dispositivo de ejemplo (*a3qEXAMPLEsffp-ats.iot.eu-west-1.amazonaws.com*) por el punto de conexión de datos del dispositivo de su cuenta y, a continuación, introduzca este comando.

```
nmap -p 8443 a3qEXAMPLEsffp-ats.iot.eu-west-1.amazonaws.com
```

2. Si nmap muestra un resultado similar al siguiente, nmap ha podido conectarse correctamente al punto de conexión de datos de su dispositivo en el puerto seleccionado.

```
Starting Nmap 7.92 ( https://nmap.org ) at 2022-02-18 16:23 Pacific Standard Time
Nmap scan report for a3qEXAMPLEsffp-ats.iot.eu-west-1.amazonaws.com
 (xx.xxx.147.160)
Host is up (0.036s latency).
Other addresses for a3qEXAMPLEsffp-ats.iot.eu-west-1.amazonaws.com (not scanned):
 xx.xxx.134.144 xx.xxx.55.139 xx.xxx.110.235 xx.xxx.174.233 xx.xxx.74.65
 xx.xxx.122.179 xx.xxx.127.126
rDNS record for xx.xxx.147.160: ec2-EXAMPLE-160.eu-west-1.compute.amazonaws.com
```

```
PORT     STATE SERVICE
8443/tcp open  https-alt
MAC Address: 00:11:22:33:44:55 (Cimsys)

Nmap done: 1 IP address (1 host up) scanned in 0.91 seconds
```

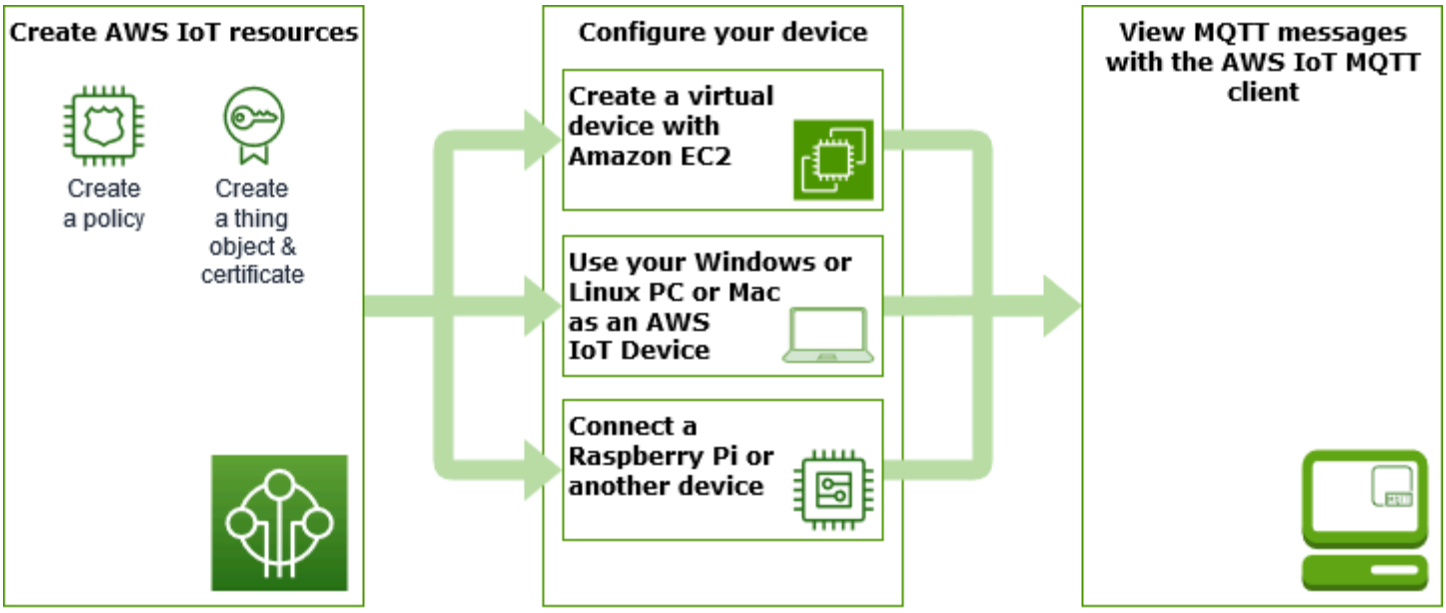
3. Si nmap no da un resultado correcto, compruebe el valor del punto de conexión para asegurarse de es el correcto y compruebe la conexión del dispositivo a internet.

Puede probar otros puertos del punto de conexión de datos del dispositivo, como el puerto 443 o el puerto HTTPS principal, sustituyendo el puerto utilizado en el paso 1, **8443**, por el puerto que deseas probar.

Explora AWS IoT Core en tutoriales prácticos

En este tutorial, instalará el software y creará los AWS IoT recursos necesarios para conectar un dispositivo al que AWS IoT Core pueda enviar y recibir mensajes MQTT. AWS IoT Core Verás los mensajes en el cliente MQTT de la AWS IoT consola.

Calcule dedicar unos 20-30 minutos a este tutorial. Si utiliza un dispositivo IoT o una Raspberry Pi, este tutorial puede tardar más si, por ejemplo, necesita instalar el sistema operativo y configurar el dispositivo.



Este tutorial es ideal para los desarrolladores que quieran empezar a usarlo para AWS IoT Core poder seguir explorando funciones más avanzadas, como el motor de reglas y las sombras. Este

tutorial lo prepara para seguir aprendiendo sobre otros AWS servicios AWS IoT Core y cómo interactúa con ellos, ya que explica los pasos con más detalle que en el tutorial de inicio rápido. Si lo que busca es una experiencia rápida de Hello World, pruebe el tutorial Pruebe el tutorial de conexión AWS IoT Core rápida.

Tras configurar tu AWS IoT consola Cuenta de AWS y consola, seguirás estos pasos para ver cómo conectar un dispositivo y hacer que envíe mensajes a AWS IoT Coreél.

Siguientes pasos

- Elegir qué dispositivo se adapta mejor a sus necesidades
- the section called “Crea AWS IoT recursos” si no va a crear un dispositivo virtual con Amazon EC2
- the section called “Configuración del dispositivo”
- the section called “Vea los mensajes MQTT con el cliente AWS IoT MQTT”

Para obtener más información AWS IoT Core, consulta ¿Qué es AWS IoT Core?

¿Qué dispositivo se adapta mejor a sus necesidades?

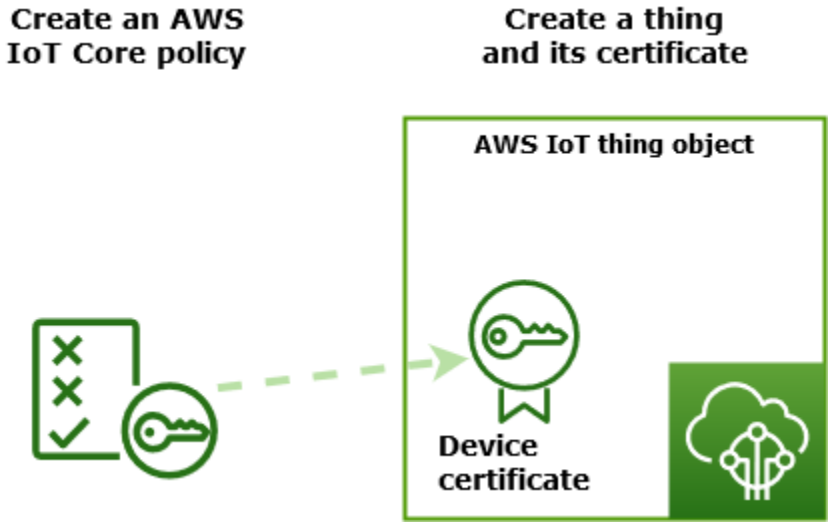
Si no sabe con certeza qué opción elegir, utilice la siguiente lista de ventajas y desventajas de cada opción para decidir cuál es la mejor para usted.

Opción	Esta podría ser una buena opción si:	Esta podría no ser una buena opción si:
the section called “Crear un dispositivo virtual con Amazon EC2”	• No tiene su propio dispositivo para probar. • No quiere instalar ningún software en su propio sistema. • Quiere realizar la prueba en un sistema operativo Linux.	• No suele usar la interfaz de línea de comandos. • No quiere incurrir en cargos adicionales de AWS . • No quiere realizar la prueba en un sistema operativo Linux.
the section called “Utilice su PC o Mac con Windows o Linux como dispositivo AWS IoT”	• No quiere incurrir en cargos adicionales de AWS .	• No quiere instalar ningún software en su ordenador.

Opción	Esta podría ser una buena opción si:	Esta podría no ser una buena opción si:
	• No quiere configurar ningún dispositivo adicional.	• Quiere una plataforma de pruebas más representativa.
the section called “Conexión de una Raspberry Pi u otro dispositivo”	• Desea realizar una prueba AWS IoT con un dispositivo real. • Ya tiene un dispositivo con el que realizar la prueba. • Tiene experiencia en la integración de hardware en los sistemas.	• No quiere comprar ni configurar un dispositivo solo para probarlo. • Por ahora, quieres hacer AWS IoT la prueba de la forma más sencilla posible.

Crea AWS IoT recursos

En este tutorial, crearás los AWS IoT recursos que un dispositivo necesita para conectarse AWS IoT Core e intercambiar mensajes.



1. Cree un documento AWS IoT de política que autorice a su dispositivo a interactuar con AWS IoT los servicios.
2. Cree un objeto con su certificado de dispositivo X.509 AWS IoT y, a continuación, adjunte el documento de política. El objeto es la representación virtual de su dispositivo en el AWS IoT

registro. El certificado autentica el dispositivo y AWS IoT Core el documento de política lo autoriza a interactuar con él. AWS IoT

> **Note**
>
> Si tiene previsto the section called “Crear un dispositivo virtual con Amazon EC2”, puede saltarse esta página y continuar en the section called “Configuración del dispositivo”. Creará estos recursos cuando cree su objeto virtual.

En este tutorial se utiliza la AWS IoT consola para crear los AWS IoT recursos. Si el dispositivo es compatible con un navegador web, puede ser más fácil ejecutar este procedimiento en el navegador web del dispositivo, ya que podrá descargar los archivos de certificado directamente al dispositivo. Si ejecuta este procedimiento en otro ordenador, tendrá que copiar los archivos de certificado en su dispositivo para que la aplicación de ejemplo pueda utilizarlos.

Cree una AWS IoT política

Los dispositivos utilizan un certificado X.509 para autenticarse. AWS IoT Core El certificado tiene AWS IoT políticas adjuntas. Estas políticas determinan qué operaciones de AWS IoT , como suscribirse o publicar en temas de MQTT, puede realizar el dispositivo. El dispositivo presenta su certificado cuando se conecta y envía mensajes a AWS IoT Core.

Siga los pasos de creación de una política que permita al dispositivo realizar las operaciones de AWS IoT necesarias para ejecutar el programa de ejemplo. Debe crear la política de AWS IoT antes de poder asociarla al certificado del dispositivo que creará más tarde.

Para crear una AWS IoT política

1. En la consola de AWS IoT, en el menú de la izquierda, seleccione Seguridad y, a continuación, Políticas.
2. En la página Aún no tiene ninguna política, elija Crear una política.

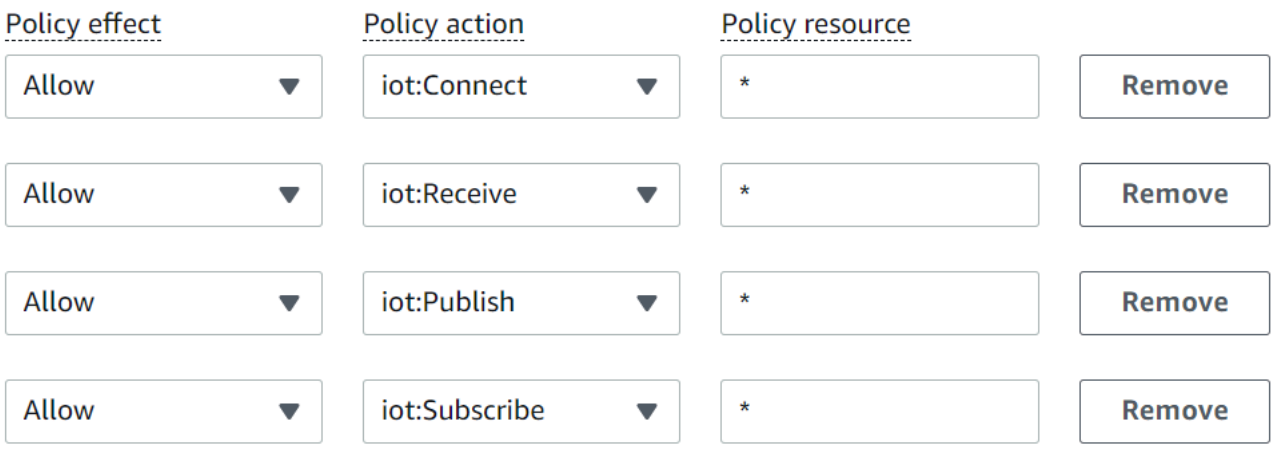
   Si su cuenta ya tiene políticas, seleccione Crear política.
3. En la página Crear una política:

   1. En la sección Propiedades de la política, en el campo Nombre de la política, introduzca un nombre para la política (por ejemplo, **`My_Iot_Policy`**). No utilice información personalmente identificable en sus nombres de política.

2. En la sección Documento de política, cree las declaraciones de política que conceden o deniegan el acceso de los recursos a las operaciones de AWS IoT Core . Para crear una declaración de política que permita a todos los clientes realizar **`iot:Connect`**, siga estos pasos:

   - En el campo Efecto de la política, seleccione Permitir. Esto permite a todos los clientes que tienen esta política asociada a su certificado realizar la acción que se indica en el campo Acción de la política.
   - En el campo Acción de la política, elija una acción de política como **`iot:Connect`**. Las acciones de política son las acciones para las que el dispositivo necesita permiso cuando ejecuta el programa de ejemplo del SDK del dispositivo.
   - En el campo Recurso de la política, introduzca un nombre de recurso de Amazon (ARN) de un recurso o un **`*`**. un **`*`** para seleccionar cualquier cliente (dispositivo).

   Para crear las declaraciones de política para **`iot:Receive`**, **`iot:Publish`** y **`iot:Subscribe`**, elija Agregar nueva declaración y repita los pasos.



> **Note**
>
> En este inicio rápido, el carácter comodín (*) se utiliza por motivos de simplicidad. Para aumentar la seguridad, debe restringir qué clientes (dispositivos) pueden conectarse y publicar mensajes especificando un ARN de cliente en lugar del carácter comodín como recurso. El cliente ARNs sigue este formato:`arn:aws:iot:`*`your-region`*`:`*`your-aws-account`*`:client/`*`my-client-id`*.

Sin embargo, primero debe crear el recurso (como un dispositivo cliente o una sombra de objeto) antes de poder asignar su ARN a una política. Para obtener más información, consulte Recursos de acción de AWS IoT Core.

4. Una vez que haya especificado la información de su política, elija Crear.

Para obtener más información, consulte ¿Cómo AWS IoT funciona con IAM.

Crear un objeto

Los dispositivos conectados AWS IoT Core están representados por objetos en el AWS IoT registro. Una cosa representa un dispositivo específico o 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 entidad física que no se conecta AWS IoT, pero que está relacionada con otros dispositivos que sí lo hacen (por ejemplo, un automóvil que tiene sensores de motor o un panel de control).

Para crear algo en la AWS IoT consola

1. En la consola de AWS IoT, en el menú de la izquierda, seleccione Todos los dispositivos y, a continuación, Cosas.
2. En la página Cosas seleccione Crear cosas.
3. En la página Crear cosas, elija Crear una sola cosa y luego seleccione Siguiente.
4. En la página Especificar propiedades de cosa en Nombre de la cosa, introduzca un nombre para la cosa, por ejemplo, **`MyIotThing`**.

   Elija los nombres de cosa con cuidado, ya que no podrá cambiarlos más adelante.

   Para cambiar el nombre de una cosa, debe crear otra nueva, asignarle el nuevo nombre y eliminar después la anterior.

   > **Note**
   >
   > No utilice información personal identificable en el nombre de la cosa. El nombre de la cosa puede aparecer en comunicaciones e informes no cifrados.

5. Mantenga el resto de los campos de esta página vacíos. Elija Siguiente.

6. En la página Configurar el certificado del dispositivo: opcional, seleccione Generar automáticamente un nuevo certificado (opción recomendada). Elija Siguiente.
7. En la página Asociar políticas al certificado (opcional), seleccione la política que creó en la sección anterior. En esa sección, la política se denominó **`My_Iot_Policy`**. Elija Crear objeto.
8. En la página Descargar certificados y claves:

   1. Descargue cada uno de los archivos de certificados y claves, y guárdelos para más adelante. Deberá instalar estos archivos en el dispositivo.

      Cuando guarde los archivos de certificado, asígneles los nombres de la siguiente tabla. Estos son los nombres de archivo que se utilizan en los ejemplos posteriores.

      Nombres de archivo de certificado

Archivos	Ruta de archivo
Clave privada	`private.pem.key`
Clave pública	(no se usa en estos ejemplos)
Certificado de dispositivo	`device.pem.crt`
Certificado de entidad de certificación raíz	`Amazon-root-CA-1.pem`

   2. Para descargar el archivo de entidad de certificación raíz de estos archivos, elija el enlace de descarga del archivo de certificado de entidad de certificación raíz que corresponda al tipo de punto de conexión de datos y conjunto de cifrado que esté utilizando. En este tutorial, seleccione Descargar a la derecha de RSA 2048 bit key: Amazon Root CA 1 y descargue el archivo de certificado RSA 2048 bit key: Amazon Root CA 1.

      > **Important**
      >
      > Debe guardar los archivos de certificado antes de salir de esta página. Tras salir de esta página en la consola, ya no tendrá acceso a los archivos de certificado.
      > Si olvidó descargar los archivos de certificado creados en este paso, debe salir de la pantalla de la consola, ir a la lista de objetos de la consola, eliminar el objeto creado y, a continuación, volver a realizar este procedimiento desde el principio.

   3. Seleccione Listo.

Tras completar este procedimiento, debería ver el objeto nuevo en la lista de objetos.

Configuración del dispositivo

En esta sección se describe cómo configurar el dispositivo para conectarse a AWS IoT Core. Si quiere empezar con un dispositivo AWS IoT Core pero aún no lo tiene, puede crear un dispositivo virtual mediante Amazon EC2 o puede utilizar su PC con Windows o Mac como dispositivo de IoT.

Seleccione la mejor opción de dispositivo para AWS IoT Core probarla. Por supuesto, puede probarlos todos, pero siempre de uno en uno. Si no sabe con certeza cuál es la mejor opción de dispositivo para usted, consulte cómo elegir la mejor opción de dispositivo y, después, vuelva a esta página.

Opciones de dispositivos

- Crear un dispositivo virtual con Amazon EC2
- Utilice su PC o Mac con Windows o Linux como dispositivo AWS IoT
- Conexión de una Raspberry Pi u otro dispositivo

Crear un dispositivo virtual con Amazon EC2

En este tutorial, creará una instancia de Amazon EC2 para que sirva como dispositivo virtual en la nube.

Para completar este tutorial, necesitas un Cuenta de AWS. Si no la tiene, complete los pasos que se describen en Configurar Cuenta de AWS antes de continuar.

En este tutorial, recorrerá los siguientes pasos:

- Configurar una instancia de Amazon EC2
- Instale Git, Node.js y configure AWS CLI
- Cree AWS IoT recursos para su dispositivo virtual
- Instale el SDK del AWS IoT dispositivo para JavaScript
- Ejecutar la aplicación de ejemplo
- Ve los mensajes de la aplicación de ejemplo en la AWS IoT consola

Configurar una instancia de Amazon EC2

Los siguientes pasos le muestran cómo crear una instancia de Amazon EC2 que actuará como su dispositivo virtual en lugar de un dispositivo físico.

Si es la primera vez que crea una instancia de Amazon EC2, puede que las instrucciones de Introducción a las instancias de Amazon EC2 Linux le resulten más útiles.

Lanzamiento de una instancia de

1. Abra la consola de Amazon EC2 en https://console.aws.amazon.com/ec2/.
2. En el menú de la consola de la izquierda, expanda la sección Instancias y elija Instancias. En el panel Instancias, seleccione Lanzar instancias a la derecha para ver una lista de configuraciones básicas.
3. En la sección Nombre y etiquetas, introduzca un nombre para la instancia y, si lo desea, agregue etiquetas.
4. En la sección Imágenes de aplicaciones y sistema operativo (Imagen de máquina de Amazon), elija una plantilla de AMI para la instancia, como la de Amazon Linux 2 AMI (HVM). Observe que estas AMI están marcadas como "Free tier eligible" (Apta para el nivel gratuito).
5. En la sección Tipo de instancia, puede seleccionar la configuración de hardware de su instancia. Seleccione el tipo `t2.micro`, que es la opción predeterminada. Observe que este tipo de instancia es apta para la capa gratuita.
6. En la sección Par de claves (inicio de sesión), elija un nombre de par de claves de la lista desplegable o elija Crear un nuevo par de claves para crear uno nuevo. Al crear un nuevo par de claves, asegúrese de descargar el archivo de clave privada y guardarlo en un lugar seguro, ya que es la única forma de descargarlo y guardarlo. Deberá proporcionar el nombre de su par de claves al lanzar una instancia, y la clave privada correspondiente cada vez que se conecte a dicha instancia.

   > **Warning**
   >
   > No seleccione la opción Continuar sin un par de claves. Si lanza la instancia sin un par de claves, no podrá conectarse a ella.

7. En la sección Configuración de redes y en la sección Configurar el almacenamiento, puede conservar la configuración predeterminada. Cuando todo esté preparado, elija Lanzar instancias.

8. Verá una página de confirmación que indicará que la instancia se está lanzando. Elija View instances para cerrar la página de confirmación y volver a la consola.
9. Puede ver el estado del lanzamiento en la pantalla Instancias. La instancia tarda poco tiempo en lanzarse. Al lanzar una instancia, su estado inicial es `pending`. Una vez iniciada la instancia, el estado cambia a `running` y recibe un nombre del DNS público. (Si la columna DNS público (IPv4) está oculta, elija Mostrar u ocultar columnas (el icono con forma de engranaje) en la esquina superior derecha de la página y, a continuación, seleccione DNS público ().) IPv4
10. Puede que transcurran unos minutos hasta que la instancia esté lista para conectarse a ella. Compruebe que la instancia haya superado las comprobaciones de estado; puede ver esta información en la columna Status Checks.

    Cuando la nueva instancia haya superado las comprobaciones de estado, continúe con el siguiente procedimiento y conéctese a ella.

Para conectarse a la instancia

Puede conectarse a una instancia mediante el cliente basado en navegador seleccionando la instancia en la consola de Amazon EC2 y eligiendo conectarse mediante Amazon EC2 Instance Connect. Instance Connect gestiona los permisos y proporciona una conexión exitosa.

1. Abra la consola de Amazon EC2 en https://console.aws.amazon.com/ec2/.
2. En el menú de la izquierda, seleccione Instancias.
3. Seleccione la instancia y elija Connect (Conectar).
4. Elija Amazon EC2 Instance Connect , Conectar.

Ahora debería ver una ventana Amazon EC2 Instance Connect conectada a su nueva instancia de Amazon EC2.

Instale Git, Node.js y configure AWS CLI

En esta sección, instalará Git y Node.js en su instancia de Linux.

Para instalar Git

1. En la ventana Amazon EC2 Instance Connect, actualice la instancia mediante el siguiente comando.

```
sudo yum update -y
```

2. En la ventana Amazon EC2 Instance Connect, instale Git mediante el siguiente comando.

```
sudo yum install git -y
```

3. Para comprobar si Git está instalado y cuál es su versión actual, ejecute el siguiente comando:

```
git --version
```

Para instalar Node.js

1. En la ventana Amazon EC2 Instance Connect, instale el administrador de versiones de nodos (nvm) mediante el siguiente comando.

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.34.0/install.sh | bash
```

   Utilizaremos nvm para instalar Node.js, ya que nvm puede instalar varias versiones de Node.js y permitirle alternar entre ellas.

2. En la ventana Amazon EC2 Instance Connect, active nvm mediante este comando.

```
. ~/.nvm/nvm.sh
```

3. En la ventana Amazon EC2 Instance Connect, utilice nvm para instalar la última versión de Node.js mediante este comando.

```
nvm install 16
```

> **Note**
>
> Esto instala la versión de LTS más reciente de Node.js.

   Si instala Node.js también instalará el administrador de paquetes de nodos (npm) para poder instalar módulos adicionales según sea necesario.

4. En la ventana Amazon EC2 Instance Connect, compruebe que Node.js está instalado y se ejecuta correctamente mediante este comando.

   ```
   node -e "console.log('Running Node.js ' + process.version)"
   ```

   Este tutorial requiere la versión Node v10.0 o posterior. Para obtener más información, consulte Tutorial: Configuración de Node.js en una instancia de Amazon EC2.

Para configurar AWS CLI

La instancia de Amazon EC2 viene precargada con la AWS CLI. Sin embargo, debe completar su AWS CLI perfil. Para obtener más información acerca de cómo configurar la CLI, consulte Configuración de la AWS CLI.

1. En el ejemplo siguiente se muestran los valores de ejemplo. Reemplácelos con valores propios. Puede encontrar estos valores en la consola de AWS , en la información de su cuenta, en la sección Credenciales de seguridad.

   En la ventana Amazon EC2 Instance Connect, escriba este comando:

   ```
   aws configure
   ```

   A continuación, introduzca los valores de su cuenta en las instrucciones que se muestran.

   ```
   AWS Access Key ID [None]: AKIAIOSFODNN7EXAMPLE
   AWS Secret Access Key [None]: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
   Default region name [None]: us-west-2
   Default output format [None]: json
   ```

2. Puede probar la AWS CLI configuración con este comando:

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

   Si AWS CLI está configurada correctamente, el comando debería devolver una dirección de punto final suya Cuenta de AWS.

Cree AWS IoT recursos para su dispositivo virtual

En esta sección se describe cómo usarlo AWS CLI para crear el objeto objeto y sus archivos de certificado directamente en el dispositivo virtual. Esto se hace directamente en el dispositivo para evitar las posibles complicaciones que podrían surgir al copiarlos al dispositivo desde otro ordenador. En esta sección, creará los siguientes recursos para su dispositivo virtual:

- Un objeto objeto en el que representar su dispositivo virtual AWS IoT.
- Un certificado para autenticar su dispositivo virtual.
- Un documento de política para autorizar al dispositivo virtual a conectarse a AWS IoT, y para publicar, recibir y suscribirse a mensajes.

Para crear un objeto « AWS IoT cosa» en tu instancia de Linux

Los dispositivos a los que AWS IoT se conecta se representan mediante objetos tipo cosa en el AWS IoT registro. Una cosa representa un dispositivo específico o una entidad lógica. En este caso, el objeto representará su dispositivo virtual, esta instancia de Amazon EC2.

1. En la ventana Amazon EC2 Instance Connect, ejecute el comando que se incluye a continuación para crear su objeto.

   ```
   aws iot create-thing --thing-name "MyIotThing"
   ```

2. El resultado JSON debe tener el siguiente aspecto:

   ```
   {
       "thingArn": "arn:aws:iot:your-region:your-aws-account:thing/MyIotThing",
       "thingName": "MyIotThing",
       "thingId": "6cf922a8-d8ea-4136-f3401EXAMPLE"
   }
   ```

Para crear y adjuntar AWS IoT claves y certificados en su instancia de Linux

El comando create-keys-and-certificate proporciona certificados de cliente firmado por la entidad emisora de certificados de Amazon Root. Este certificado se utiliza para autenticar la identidad del dispositivo virtual.

1. En la ventana Amazon EC2 Instance Connect, cree un directorio para almacenar los archivos de certificados y claves.

```
mkdir ~/certs
```

2. En la ventana Amazon EC2 Instance Connect, descargue una copia del certificado de la autoridad de certificación (CA) de Amazon mediante este comando.

```
curl -o ~/certs/Amazon-root-CA-1.pem \
    https://www.amazontrust.com/repository/AmazonRootCA1.pem
```

3. En la ventana Amazon EC2 Instance Connect, ejecute el siguiente comando para crear los archivos de clave privada, clave pública y certificado X.509. Este comando también registra y activa el certificado con AWS IoT.

```
aws iot create-keys-and-certificate \
    --set-as-active \
    --certificate-pem-outfile "~/certs/device.pem.crt" \
    --public-key-outfile "~/certs/public.pem.key" \
    --private-key-outfile "~/certs/private.pem.key"
```

La respuesta tiene este aspecto: Guarde `certificateArn` para poder utilizarlo en los comandos posteriores. Lo necesitará para asociar el certificado a su objeto y para asociar la política al certificado más adelante.

```
{
    "certificateArn": "arn:aws:iot:us-
west-2:123456789012:cert/9894ba17925e663f1d29c23af4582b8e3b7619c31f3fbd93adcb51ae54b83dc2",
    "certificateId":
 "9894ba17925e663f1d29c23af4582b8e3b7619c31f3fbd93adcb51ae54b83dc2",
    "certificatePem": "
-----BEGIN CERTIFICATE-----
MIICiTCCEXAMPLE6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC
VVMxCzAJBgNVBAgEXAMPLEAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6
b24xFDASBgNVBAsTC0lBTSEXAMPLE2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd
BgkqhkiG9w0BCQEWEG5vb25lQGFtYEXAMPLEb20wHhcNMTEwNDI1MjA0NTIxWhcN
MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCEXAMPLEJBgNVBAgTAldBMRAwDgYD
VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDAEXAMPLEsTC0lBTSBDb25z
b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEXAMPLE25lQGFt
YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+aEXAMPLE
EXAMPLEfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9T
rDHudUZEXAMPLELG5M43q7Wgc/MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE
Ibb3OhjZnzcvQAEXAMPLEWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
```

```
nUhVVxYUntneD9+h8Mg9qEXAMPLEyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb
FFBjvSfpJIlJ00zbhNYS5f6GuoEDEXAMPLEBHjJnyp378OD8uTs7fLvjx79LjSTb
NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
-----END CERTIFICATE-----\n",
    "keyPair": {
        "PublicKey": "-----BEGIN PUBLIC
 KEY-----\nMIIBIjANBgkqhkEXAMPLEQEFAAOCAQ8AMIIBCgKCAQEAEXAMPLE1nnyJwKSMHw4h
\nMMEXAMPLEuuN/dMAS3fyce8DW/4+EXAMPLEyjmoF/YVF/
gHr99VEEXAMPLE5VF13\n59VK7cEXAMPLE67GK+y+jikqXOgHh/xJTwo
+sGpWEXAMPLEDz18xOd2ka4tCzuWEXAMPLEahJbYkCPUBSU8opVkR7qkEXAMPLE1DR6sx2HocliOOLtu6Fkw91swQWE
\GB3ZPrNh0PzQYvjUStZeccyNCx2EXAMPLEvp9mQOUXP6plfgxwKRX2fEXAMPLEDa
\nhJLXkX3rHU2xbxJSq7D+XEXAMPLEcw+LyFhI5mgFRl88eGdsAEXAMPLElnI9EesG\nFQIDAQAB\n-----
END PUBLIC KEY-----\n",
        "PrivateKey": "-----BEGIN RSA PRIVATE KEY-----\nkey omitted for security
 reasons\n-----END RSA PRIVATE KEY-----\n"
    }
}
```

4. En la ventana Amazon EC2 Instance Connect, adjunte su objeto al certificado que acaba de crear mediante el siguiente comando y *certificateArn* en la respuesta del comando anterior.

```
aws iot attach-thing-principal \
    --thing-name "MyIotThing" \
    --principal "certificateArn"
```

Este comando no devuelve ningún resultado si se realiza correctamente.

Para crear y asociar una política a un usuario

1. En la ventana Amazon EC2 Instance Connect, cree el archivo de política copiando y pegando este documento de política en un archivo denominado **~/policy.json**.

Si no tiene un editor favorito de Linux, puede abrir nano con este comando.

```
nano ~/policy.json
```

Pegue el documento de política de `policy.json` en él. Pulse Ctrl-X para salir del editor de nano y guardar el archivo.

Contenido del documento de política de `policy.json`.

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

2. En la ventana Amazon EC2 Instance Connect, cree su política utilizando el siguiente comando.

```
aws iot create-policy \
    --policy-name "MyIotThingPolicy" \
    --policy-document "file://~/policy.json"
```

Salida:

```
{
    "policyName": "MyIotThingPolicy",
    "policyArn": "arn:aws:iot:us-east-1:your-aws-account:policy/
MyIotThingPolicy",
    "policyDocument": "{
        \"Version\": \"2012-10-17\",
        \"Statement\": [
            {
                \"Effect\": \"Allow\",
                \"Action\": [
                    \"iot:Publish\",
                    \"iot:Receive\",
                    \"iot:Subscribe\",
                    \"iot:Connect\"
```

```
                ],
                \"Resource\": [
                    \"*\"
                ]
            }
            ]
    }",
    "policyVersionId": "1"
}
```

3. En la ventana Amazon EC2 Instance Connect, asocie la política al certificado de su dispositivo virtual mediante el siguiente comando.

```
aws iot attach-policy \
    --policy-name "MyIotThingPolicy" \
    --target "certificateArn"
```

Este comando no devuelve ningún resultado si se realiza correctamente.

Instale el SDK del AWS IoT dispositivo para JavaScript

En esta sección, instalará el SDK para AWS IoT dispositivos JavaScript, que contiene el código que las aplicaciones pueden usar para comunicarse, así AWS IoT como los programas de muestra. Para obtener más información, consulta el JavaScript GitHub repositorio del SDK de AWS IoT dispositivos.

Para instalar el AWS IoT Device SDK for JavaScript en tu instancia de Linux

1. En la ventana Amazon EC2 Instance Connect, clone el AWS IoT Device SDK para el JavaScript repositorio en el `aws-iot-device-sdk-js-v2` directorio de su directorio principal mediante este comando.

```
cd ~
git clone https://github.com/aws/aws-iot-device-sdk-js-v2.git
```

2. Vaya al directorio `aws-iot-device-sdk-js-v2` que creó en el paso anterior.

```
cd aws-iot-device-sdk-js-v2
```

3. Utilice npm para instalar el SDK.

```
npm install
```

Ejecutar la aplicación de ejemplo

Los comandos de las siguientes secciones suponen que los archivos de clave y certificado se almacenan en el dispositivo virtual, tal y como se muestra en esta tabla.

Nombres de archivo de certificado

Archivos	Ruta de archivo
Clave privada	`~/certs/private.pem.key`
Certificado de dispositivo	`~/certs/device.pem.crt`
Certificado de entidad de certificación raíz	`~/certs/Amazon-root-CA-1.pem`

En esta sección, instalará y ejecutará la aplicación de `pub-sub.js` muestra que se encuentra en el `aws-iot-device-sdk-js-v2/samples/node` directorio del AWS IoT Device SDK for JavaScript. Esta aplicación muestra cómo un dispositivo, su instancia de Amazon EC2, utiliza la biblioteca de MQTT para publicar mensajes MQTT y suscribirse a ellos. La aplicación de `pub-sub.js` de ejemplo se suscribe a un tema,`topic_1`, publica 10 mensajes sobre ese tema y muestra los mensajes tal como los recibe del agente de mensajes.

Para instalar y ejecutar la aplicación de ejemplo

1. En la ventana Amazon EC2 Instance Connect, navegue hasta el directorio `aws-iot-device-sdk-js-v2/samples/node/pub_sub` que creó el SDK e instale la aplicación de ejemplo con estos comandos.

   ```
   cd ~/aws-iot-device-sdk-js-v2/samples/node/pub_sub
   npm install
   ```

2. En la ventana Amazon EC2 Instance Connect, vaya *`your-iot-endpoint`* desde AWS IoT mediante este comando.

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

3. En la ventana Amazon EC2 Instance Connect, insértelo *your-iot-endpoint* como se indica y ejecute este comando.

```
node dist/index.js --topic topic_1 --ca_file ~/certs/Amazon-root-CA-1.pem --cert ~/
certs/device.pem.crt --key ~/certs/private.pem.key --endpoint your-iot-endpoint
```

La aplicación de ejemplo:

1. Se conecta a AWS IoT Core para su cuenta.
2. Se suscribe al tema de mensaje, topic_1, y muestra los mensajes que recibe sobre ese tema.
3. Publica 10 mensajes sobre el tema, topic_1.
4. Muestra una salida similar a la siguiente:

```
Publish received. topic:"topic_1" dup:false qos:1 retain:false
{"message":"Hello world!","sequence":1}
Publish received. topic:"topic_1" dup:false qos:1 retain:false
{"message":"Hello world!","sequence":2}
Publish received. topic:"topic_1" dup:false qos:1 retain:false
{"message":"Hello world!","sequence":3}
Publish received. topic:"topic_1" dup:false qos:1 retain:false
{"message":"Hello world!","sequence":4}
Publish received. topic:"topic_1" dup:false qos:1 retain:false
{"message":"Hello world!","sequence":5}
Publish received. topic:"topic_1" dup:false qos:1 retain:false
{"message":"Hello world!","sequence":6}
Publish received. topic:"topic_1" dup:false qos:1 retain:false
{"message":"Hello world!","sequence":7}
Publish received. topic:"topic_1" dup:false qos:1 retain:false
{"message":"Hello world!","sequence":8}
Publish received. topic:"topic_1" dup:false qos:1 retain:false
{"message":"Hello world!","sequence":9}
Publish received. topic:"topic_1" dup:false qos:1 retain:false
{"message":"Hello world!","sequence":10}
```

Si tiene problemas para ejecutar la aplicación de ejemplo, revise the section called “Solución de problemas con la aplicación de ejemplo”.

También puede agregar el parámetro `--verbosity debug` a la línea de comandos para que la aplicación de ejemplo muestre mensajes detallados sobre lo que está haciendo. Esa información podría proporcionarle la ayuda que necesita para corregir el problema.

Ve los mensajes de la aplicación de ejemplo en la AWS IoT consola

Puede ver los mensajes de la aplicación de ejemplo a medida que pasan por el agente de mensajes mediante el cliente de prueba de MQTT de la consola de AWS IoT .

Para ver los mensajes MQTT publicados por la aplicación de ejemplo

1. Consulte Vea los mensajes MQTT con el cliente AWS IoT MQTT. Esto le ayudará a aprender a utilizar el cliente de prueba de MQTT de la consola de AWS IoT para ver los mensajes MQTT a medida que pasan por el agente de mensajes.
2. Abra la opción Cliente de prueba de MQTT en la consola de AWS IoT .
3. En Suscribirse a un tema, suscríbase al tema topic_1.
4. En la ventana Amazon EC2 Instance Connect, vuelva a ejecutar la aplicación de ejemplo y observe los mensajes del cliente de prueba de MQTT de la consola de AWS IoT .

```
cd ~/aws-iot-device-sdk-js-v2/samples/node/pub_sub
node dist/index.js --topic topic_1 --ca_file ~/certs/Amazon-root-CA-1.pem --cert ~/
certs/device.pem.crt --key ~/certs/private.pem.key --endpoint your-iot-endpoint
```

Para obtener más información sobre MQTT y cómo AWS IoT Core es compatible con el protocolo, consulte MQTT.

Utilice su PC o Mac con Windows o Linux como dispositivo AWS IoT

En este tutorial, configurará un ordenador personal para usarlo con AWS IoT. Estas instrucciones son compatibles con Windows, Linux PCs y Mac. Para lograr esto, necesita instalar cierto software en el ordenador. Si no quiere instalar software en su ordenador, puede consultar la sección Crear un dispositivo virtual con Amazon EC2 para saber cómo instalar todo el software en una máquina virtual.

En este tutorial, recorrerá los siguientes pasos:

- Configurar su ordenador personal
- Instalar Git, Python y el SDK de AWS IoT dispositivos para Python
- Configurar la política y ejecutar la aplicación de ejemplo

- Ve los mensajes de la aplicación de ejemplo en la AWS IoT consola
- Ejecutar el ejemplo de suscripción compartida en Python

Configurar su ordenador personal

Para completar este tutorial, necesita un PC con Windows o Linux o un Mac con conexión a internet.

Antes de continuar con el siguiente paso, confirme que puede abrir una ventana de la línea de comandos en el ordenador. Utilice cmd.exe en un PC con Windows. En un PC con Linux o un Mac, utilice Terminal.

Instalar Git, Python y el SDK de AWS IoT dispositivos para Python

En esta sección, instalará Python y el SDK de AWS IoT dispositivo para Python en su ordenador.

Instalar la versión más reciente de Git y Python

Este procedimiento explica cómo instalar la versión más reciente de Git y Python en su ordenador personal.

Para descargar e instalar Git y Python en su ordenador

1. Compruebe si tiene Git instalado en el equipo. Introduzca este comando en la línea de comandos.

   ```
   git --version
   ```

   Si el comando muestra la versión de Git, significa que Git está instalado y puede continuar con el paso siguiente.

   Si el comando muestra un error, abra https://git-scm.com/download e instale Git en el ordenador.
2. Compruebe si Python ya está instalado. Introduzca el comando en la línea de comandos.

   ```
   python -V
   ```

   > **Note**
   >
   > Si este comando produce un error, `Python was not found`, puede deberse a que el sistema operativo llama al ejecutable de Python v3.x como `Python3`. En ese caso,

sustituya todas las instancias de `python` por `python3` y continúe con el resto de este tutorial.

Si el comando muestra la versión de Python, significa que Python ya está instalado. Este tutorial requiere Python v3.7 o versiones posteriores.

3. Si Python está instalado, puede omitir el resto de los pasos de esta sección. Si no es así, continúe.
4. Abre https://www.python.org/downloads/ y descarga el instalador para tu ordenador.
5. Si la descarga no se inició automáticamente, ejecute el programa descargado para instalar Python.
6. Compruebe la instalación de Python.

```
python -V
```

Confirme que el comando muestra la versión de Python. Si no se muestra la versión de Python, pruebe a descargar e instalar Python de nuevo.

Instalar el SDK AWS IoT del dispositivo para Python

Para instalar el SDK de AWS IoT dispositivo para Python en su ordenador

1. Instale la versión 2 del SDK de AWS IoT dispositivos para Python.

```
python3 -m pip install awsiotsdk
```

2. Clona el repositorio AWS IoT Device SDK para Python en el directorio aws-iot-device-sdk - python-v2 de tu directorio principal. Este procedimiento hace referencia al directorio base de los archivos en los que estás instalando. *home*

   La ubicación real del *home* directorio depende del sistema operativo.

   Linux/macOS

   En macOS y Linux, el *home* directorio es~.

```
cd ~
```

```
git clone https://github.com/aws/aws-iot-device-sdk-python-v2.git
```

Windows

En Windows, puede encontrar la ruta del *home* directorio ejecutando este comando en la `cmd` ventana.

```
echo %USERPROFILE%
cd %USERPROFILE%
git clone https://github.com/aws/aws-iot-device-sdk-python-v2.git
```

> **Note**
>
> Si utilizas Windows PowerShell en lugar de hacerlocmd.exe, utiliza el siguiente comando.
>
> ```
> echo $home
> ```

Para obtener más información, consulte el GitHub repositorio AWS IoT Device SDK para Python.

Preparación antes de ejecutar las aplicaciones de ejemplo

Para prepararse antes de ejecutar las aplicaciones de ejemplo

- Cree el directorio `certs`. En el directorio `certs`, copie los archivos de clave privada, certificado de dispositivo y certificado de entidad de certificación raíz que guardó al crear y registrar el objeto en the section called “Crea AWS IoT recursos”. Los nombres de cada archivo del directorio de destino deben coincidir con los de la tabla.

  Los comandos de la siguiente sección suponen que los archivos de clave y certificado se almacenan en el dispositivo como se muestra en esta tabla.

  Linux/macOS

  Ejecute este comando para crear el subdirectorio `certs` que utilizará al ejecutar las aplicaciones de ejemplo.

```
mkdir ~/certs
```

En el nuevo subdirectorio, copie los archivos a las rutas de archivo de destino que se muestran en la siguiente tabla.

Nombres de archivo de certificado

Archivos	Ruta de archivo
Clave privada	`~/certs/private.pem.key`
Certificado de dispositivo	`~/certs/device.pem.crt`
Certificado de entidad de certificación raíz	`~/certs/Amazon-root-CA-1.pem`

Ejecute este comando para enumerar los archivos del directorio `certs` y compararlos con los que aparecen en la tabla.

```
ls -l ~/certs
```

Windows

Ejecute este comando para crear el subdirectorio `certs` que utilizará al ejecutar las aplicaciones de ejemplo.

```
mkdir %USERPROFILE%\certs
```

En el nuevo subdirectorio, copie los archivos a las rutas de archivo de destino que se muestran en la siguiente tabla.

Nombres de archivo de certificado

Archivos	Ruta de archivo
Clave privada	`%USERPROFILE%\certs\private.pem.key`

Archivos	Ruta de archivo
Certificado de dispositivo	`%USERPROFILE%\certs\device.pem.crt`
Certificado de entidad de certificación raíz	`%USERPROFILE%\certs\Amazon-root-CA-1.pem`

Ejecute este comando para enumerar los archivos del directorio `certs` y compararlos con los que aparecen en la tabla.

```
dir %USERPROFILE%\certs
```

Configurar la política y ejecutar la aplicación de ejemplo

En esta sección, configurará su política y ejecutará el script de ejemplo de `pubsub.py` que se encuentra en el directorio `aws-iot-device-sdk-python-v2/samples` del AWS IoT Device SDK para Python. Este script muestra cómo su dispositivo utiliza la biblioteca de MQTT para publicar y suscribirse a mensajes MQTT.

La aplicación de `pubsub.py` de ejemplo se suscribe a un tema,`test/topic`, publica 10 mensajes sobre ese tema y muestra los mensajes tal como los recibe del agente de mensajes.

Para ejecutar el script de ejemplo de `pubsub.py`, necesita la siguiente información:

Valores de los parámetros de la aplicación

Parámetro	Dónde encontrar el valor
`your-iot-endpoint`	1. En la consola de AWS IoT, en el menú de la izquierda, seleccione Configuración. 2. En la página Configuración, su punto de conexión aparece en la sección Punto de enlace de datos de dispositivo.

El *your-iot-endpoint* valor tiene el formato:*endpoint_id*-ats.iot.*region*.amazonaws.com, por ejemplo,a3qj468EXAMPLE-ats.iot.us-west-2.amazonaws.com.

Antes de ejecutar el script, asegúrese de que la política del objeto proporciona permisos para que el script de ejemplo pueda conectarse, suscribirse, publicar y recibir.

Para buscar y revisar el documento de política de un recurso de objeto

1. En la consola de AWS IoT, en la lista Objetos, busque el recurso de objeto que represente a su dispositivo.
2. Seleccione el enlace Nombre del recurso de objeto que representa a su dispositivo para abrir la página Detalles del objeto.
3. En la página Detalles del objeto, en la pestaña Certificados, elija el certificado asociado al recurso de objeto. Solo debe haber un certificado en la lista. Si hay más de uno, elija el certificado cuyos archivos estén instalados en el dispositivo y con el que se vaya a conectar a AWS IoT Core.

   En la página Detalles del certificado, en la pestaña Políticas, elija la política asociada al certificado. Solo debe haber una. Si hay más de una, repita el paso siguiente para cada una de ellas con el fin de garantizar que al menos una política conceda el acceso requerido.
4. En la página de descripción general de la política, busque el editor de JSON y seleccione Editar el documento de política para revisar y editar el documento de política según sea necesario.
5. El JSON de la política se muestra en el siguiente ejemplo. En el `"Resource"` elemento, *region:account* sustitúyalo por el suyo Región de AWS y Cuenta de AWS en cada uno de los `Resource` valores.

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

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

Linux/macOS

Para ejecutar el script de ejemplo en Linux/macOS

1. En la ventana de la línea de comandos, navegue hasta el directorio `~/aws-iot-device-sdk-python-v2/samples/node/pub_sub` que el SDK creó con estos comandos.

```
cd ~/aws-iot-device-sdk-python-v2/samples
```

2. En la ventana de la línea de comandos, reemplace *`your-iot-endpoint`* como se indica y ejecute este comando.

```
python3 pubsub.py --endpoint your-iot-endpoint --ca_file ~/certs/Amazon-root-CA-1.pem --cert ~/certs/device.pem.crt --key ~/certs/private.pem.key
```

Windows

Para ejecutar la aplicación de ejemplo en un PC con Windows

1. En la ventana de la línea de comandos, navegue hasta el directorio `%USERPROFILE%\aws-iot-device-sdk-python-v2\samples` que el SDK creó e instale la aplicación de ejemplo con estos comandos.

```
cd %USERPROFILE%\aws-iot-device-sdk-python-v2\samples
```

2. En la ventana de la línea de comandos, *`your-iot-endpoint`* sustituya como se indica y ejecute este comando.

```
python3 pubsub.py --endpoint your-iot-endpoint --ca_file %USERPROFILE%
\certs\Amazon-root-CA-1.pem --cert %USERPROFILE%\certs\device.pem.crt --key
 %USERPROFILE%\certs\private.pem.key
```

El script de ejemplo:

1. Se conecta a la AWS IoT Core de su cuenta.
2. Se suscribe al tema de mensaje, test/topic, y muestra los mensajes que recibe sobre ese tema.
3. Publica 10 mensajes sobre el tema test/topic.
4. Muestra una salida similar a la siguiente:

```
Connected!
Subscribing to topic 'test/topic'...
Subscribed with QoS.AT_LEAST_ONCE
Sending 10 message(s)
Publishing message to topic 'test/topic': Hello World! [1]
Received message from topic 'test/topic': b'"Hello World! [1]"'
Publishing message to topic 'test/topic': Hello World! [2]
Received message from topic 'test/topic': b'"Hello World! [2]"'
Publishing message to topic 'test/topic': Hello World! [3]
Received message from topic 'test/topic': b'"Hello World! [3]"'
Publishing message to topic 'test/topic': Hello World! [4]
Received message from topic 'test/topic': b'"Hello World! [4]"'
Publishing message to topic 'test/topic': Hello World! [5]
Received message from topic 'test/topic': b'"Hello World! [5]"'
Publishing message to topic 'test/topic': Hello World! [6]
```

```
Received message from topic 'test/topic': b'"Hello World! [6]"'
Publishing message to topic 'test/topic': Hello World! [7]
Received message from topic 'test/topic': b'"Hello World! [7]"'
Publishing message to topic 'test/topic': Hello World! [8]
Received message from topic 'test/topic': b'"Hello World! [8]"'
Publishing message to topic 'test/topic': Hello World! [9]
Received message from topic 'test/topic': b'"Hello World! [9]"'
Publishing message to topic 'test/topic': Hello World! [10]
Received message from topic 'test/topic': b'"Hello World! [10]"'
10 message(s) received.
Disconnecting...
Disconnected!
```

Si tiene problemas para ejecutar la aplicación de ejemplo, revise the section called “Solución de problemas con la aplicación de ejemplo”.

También puede agregar el parámetro `--verbosity Debug` a la línea de comandos para que la aplicación de ejemplo muestre mensajes detallados sobre lo que está haciendo. Esa información puede ayudarle a corregir el problema.

Ve los mensajes de la aplicación de ejemplo en la AWS IoT consola

Puede ver los mensajes de la aplicación de ejemplo a medida que pasan por el agente de mensajes mediante el cliente de prueba de MQTT de la consola de AWS IoT .

Para ver los mensajes MQTT publicados por la aplicación de ejemplo

1. Consulte Vea los mensajes MQTT con el cliente AWS IoT MQTT. Esto le ayudará a aprender a utilizar el cliente de prueba de MQTT de la consola de AWS IoT para ver los mensajes MQTT a medida que pasan por el agente de mensajes.
2. Abra la opción Cliente de prueba de MQTT en la consola de AWS IoT .
3. En Suscribirse a un tema, suscríbase al tema test/topic.
4. En su ventana de línea de comandos, ejecute de nuevo la aplicación de ejemplo y observe los mensajes del cliente MQTT en la consola de AWS IoT .

   Linux/macOS

   ```
   cd ~/aws-iot-device-sdk-python-v2/samples
   ```

```
python3 pubsub.py --topic test/topic --ca_file ~/certs/Amazon-root-CA-1.pem --
cert ~/certs/device.pem.crt --key ~/certs/private.pem.key --endpoint your-iot-
endpoint
```

Windows

```
cd %USERPROFILE%\aws-iot-device-sdk-python-v2\samples
python3 pubsub.py --topic test/topic --ca_file %USERPROFILE%\certs\Amazon-root-
CA-1.pem --cert %USERPROFILE%\certs\device.pem.crt --key %USERPROFILE%\certs
\private.pem.key --endpoint your-iot-endpoint
```

Para obtener más información sobre MQTT y cómo AWS IoT Core es compatible con el protocolo, consulte MQTT.

Ejecutar el ejemplo de suscripción compartida en Python

AWS IoT Core admite suscripciones compartidas para MQTT 3 y MQTT 5. Las suscripciones compartidas permiten que varios clientes compartan una suscripción a un tema y solo un cliente recibirá los mensajes publicados sobre ese tema mediante una distribución aleatoria. Para usar las suscripciones compartidas, los clientes se suscriben al filtro de temas de una suscripción compartida: `$share/{ShareName}/{TopicFilter}`.

Para configurar la política y ejecutar el ejemplo de suscripción compartida

1. Para ejecutar el ejemplo de suscripción compartida, debe configurar la política de su empresa tal y como se describe en la Suscripción compartida de MQTT 5.
2. Para ejecutar el ejemplo de suscripción compartida, ejecute los siguientes comandos.

   Linux/macOS

   Para ejecutar el script de ejemplo en Linux/macOS

   1. En la ventana de la línea de comandos, navegue hasta el directorio `~/aws-iot-device-sdk-python-v2/samples` que el SDK creó con estos comandos.

      ```
      cd ~/aws-iot-device-sdk-python-v2/samples
      ```

   2. En la ventana de la línea de comandos, *your-iot-endpoint* sustituya como se indica y ejecute este comando.

```
python3 mqtt5_shared_subscription.py --endpoint your-iot-endpoint --ca_file
 ~/certs/Amazon-root-CA-1.pem --cert ~/certs/device.pem.crt --key ~/certs/
private.pem.key --group_identifier consumer
```

Windows

Para ejecutar la aplicación de ejemplo en un PC con Windows

1. En la ventana de la línea de comandos, navegue hasta el directorio `%USERPROFILE%\aws-iot-device-sdk-python-v2\samples` que el SDK creó e instale la aplicación de ejemplo con estos comandos.

```
cd %USERPROFILE%\aws-iot-device-sdk-python-v2\samples
```

2. En la ventana de la línea de comandos, *`your-iot-endpoint`* sustituya como se indica y ejecute este comando.

```
python3 mqtt5_shared_subscription.py --endpoint your-iot-endpoint --ca_file
 %USERPROFILE%\certs\Amazon-root-CA-1.pem --cert %USERPROFILE%\certs
\device.pem.crt --key %USERPROFILE%\certs\private.pem.key --group_identifier
 consumer
```

> **Note**
>
> Si lo desea, puede especificar un identificador de grupo en función de sus necesidades al ejecutar la muestra (por ejemplo, `--group_identifier consumer`). Si no especifica uno, `python-sample` es el identificador de grupo predeterminado.

3. El resultado de la línea de comandos puede ser similar al siguiente:

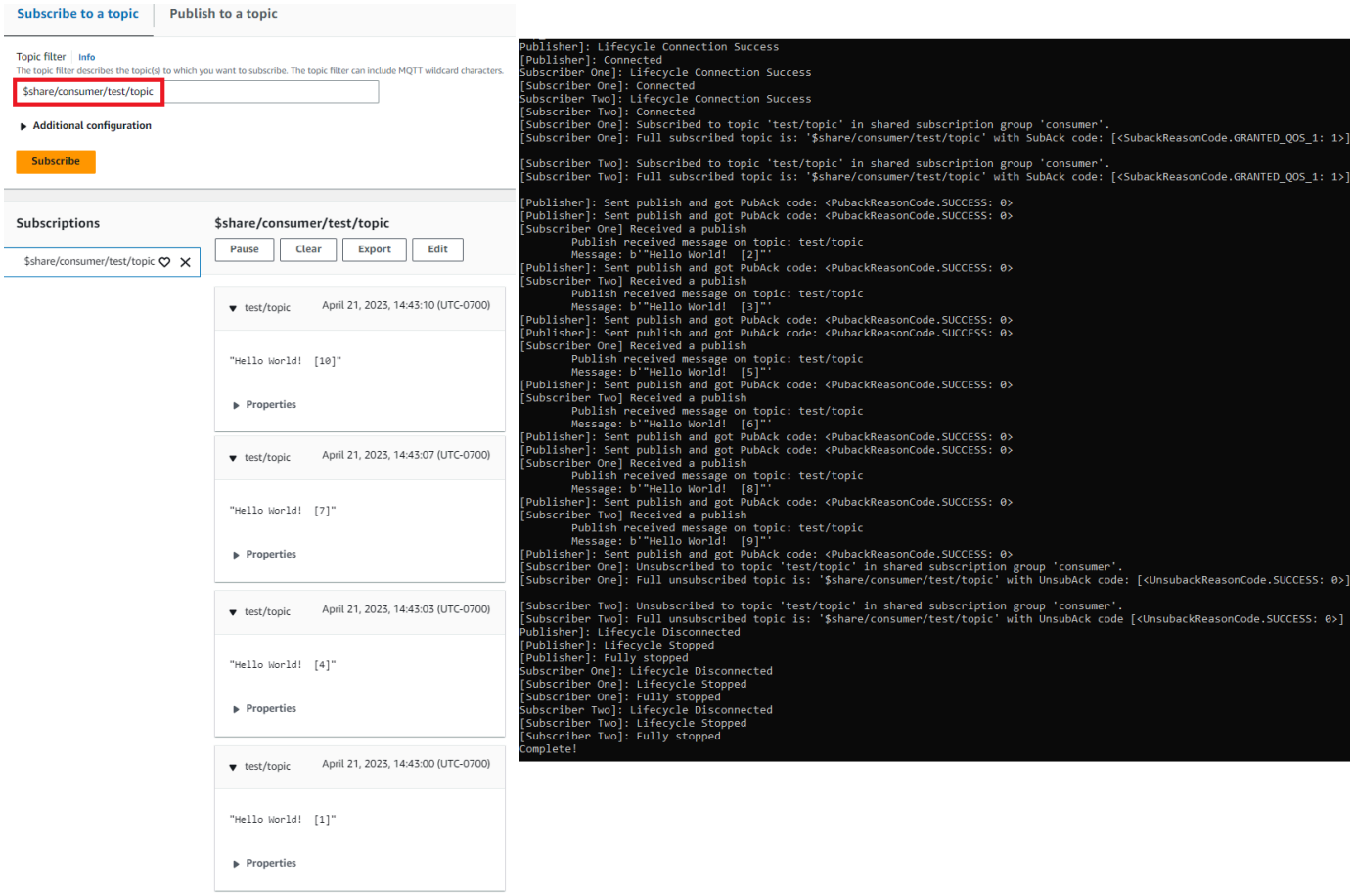
```
Publisher]: Lifecycle Connection Success
[Publisher]: Connected
Subscriber One]: Lifecycle Connection Success
[Subscriber One]: Connected
Subscriber Two]: Lifecycle Connection Success
[Subscriber Two]: Connected
[Subscriber One]: Subscribed to topic 'test/topic' in shared subscription group
 'consumer'.
```

```
[Subscriber One]: Full subscribed topic is: '$share/consumer/test/topic' with
 SubAck code: [<SubackReasonCode.GRANTED_QOS_1: 1>]
[Subscriber Two]: Subscribed to topic 'test/topic' in shared subscription group
 'consumer'.
[Subscriber Two]: Full subscribed topic is: '$share/consumer/test/topic' with
 SubAck code: [<SubackReasonCode.GRANTED_QOS_1: 1>]
[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
[Subscriber Two] Received a publish
        Publish received message on topic: test/topic
        Message: b'"Hello World!  [1]"'
[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
[Subscriber One] Received a publish
        Publish received message on topic: test/topic
        Message: b'"Hello World!  [2]"'
[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
[Subscriber Two] Received a publish
        Publish received message on topic: test/topic
        Message: b'"Hello World!  [3]"'
[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
[Subscriber One] Received a publish
        Publish received message on topic: test/topic
        Message: b'"Hello World!  [4]"'
[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
[Subscriber Two] Received a publish
        Publish received message on topic: test/topic
        Message: b'"Hello World!  [5]"'
[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
[Subscriber One] Received a publish
        Publish received message on topic: test/topic
        Message: b'"Hello World!  [6]"'
[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
[Subscriber Two] Received a publish
        Publish received message on topic: test/topic
        Message: b'"Hello World!  [7]"'
[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
[Subscriber One] Received a publish
        Publish received message on topic: test/topic
        Message: b'"Hello World!  [8]"'
[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
[Subscriber Two] Received a publish
        Publish received message on topic: test/topic
        Message: b'"Hello World!  [9]"'
[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
[Subscriber One] Received a publish
```

```
        Publish received message on topic: test/topic
        Message: b'"Hello World!  [10]"'
[Subscriber One]: Unsubscribed to topic 'test/topic' in shared subscription group
 'consumer'.
[Subscriber One]: Full unsubscribed topic is: '$share/consumer/test/topic' with
 UnsubAck code: [<UnsubackReasonCode.SUCCESS: 0>]
[Subscriber Two]: Unsubscribed to topic 'test/topic' in shared subscription group
 'consumer'.
[Subscriber Two]: Full unsubscribed topic is: '$share/consumer/test/topic' with
 UnsubAck code [<UnsubackReasonCode.SUCCESS: 0>]
Publisher]: Lifecycle Disconnected
[Publisher]: Lifecycle Stopped
[Publisher]: Fully stopped
Subscriber One]: Lifecycle Disconnected
[Subscriber One]: Lifecycle Stopped
[Subscriber One]: Fully stopped
Subscriber Two]: Lifecycle Disconnected
[Subscriber Two]: Lifecycle Stopped
[Subscriber Two]: Fully stopped
Complete!
```

4. Abra el cliente de prueba de MQTT en la consola de AWS IoT . En Suscribirse a un tema, suscríbase al tema de la suscripción compartida: `$share/consumer/test/topic`. Puede especificar un identificador de grupo en función de sus necesidades al ejecutar la muestra (por ejemplo, `--group_identifier consumer`). Si no especifica uno, el valor predeterminado es `python-sample`. Para obtener más información, consulte el Ejemplo de Python de suscripción compartida de MQTT 5 y Suscripciones compartidas en la Guía de AWS IoT Core para desarrolladores.

   En la ventana de la línea de comandos, ejecute de nuevo la aplicación de ejemplo y observe la distribución de mensajes en el cliente de prueba de MQTT de la consola de AWS IoT y de la línea de comandos.



Conexión de una Raspberry Pi u otro dispositivo

En esta sección, configuraremos una Raspberry Pi para usarla con AWS IoT. Si tiene otro dispositivo que gustaría conectar, las instrucciones de la Raspberry Pi incluyen referencias que pueden ayudarlo a adaptar estas instrucciones a otros dispositivos.

Normalmente se tarda unos 20 minutos, pero puede tardar más si tiene que instalar muchas actualizaciones del software del sistema.

En este tutorial, recorrerá los siguientes pasos:

- Configuración del dispositivo
- Instale las herramientas y bibliotecas necesarias para el SDK del AWS IoT dispositivo
- Instala AWS IoT el SDK del dispositivo
- Instalar y ejecutar la aplicación de ejemplo
- Vea los mensajes de la aplicación de ejemplo en la AWS IoT consola

> **Important**
>
> Adaptar estas instrucciones a otros dispositivos y sistemas operativos puede resultar difícil. Deberás entender el dispositivo lo suficientemente bien como para poder interpretar estas instrucciones y aplicarlas al dispositivo.
> Si tienes dificultades al configurar tu dispositivo AWS IoT, puedes probar una de las otras opciones del dispositivo como alternativa, como Crear un dispositivo virtual con Amazon EC2 oUtilice su PC o Mac con Windows o Linux como dispositivo AWS IoT.

Configuración del dispositivo

El objetivo de este paso es recopilar todo lo necesario para configurar su dispositivo de forma que pueda iniciar el sistema operativo (SO), conectarse a internet y permitirle interactuar con ella en una interfaz de línea de comandos.

Necesitará lo siguiente para completar este tutorial:

- Y Cuenta de AWS. Si no la tiene, complete los pasos que se describen en Configurar Cuenta de AWS antes de continuar.
- Una Raspberry Pi 3 Modelo B o un modelo más reciente. Esto podría funcionar en versiones anteriores de Raspberry Pi, pero no se han probado.
- Sistema operativo Raspberry Pi (32 bits) o posterior. Recomendamos utilizar la última versión del sistema operativo Raspberry Pi. Esto podría funcionar en versiones anteriores de Raspberry Pi, pero no se han probado.

  Para ejecutar este ejemplo, no necesita instalar el escritorio con la interfaz gráfica de usuario (GUI); sin embargo, si es la primera vez que utiliza Raspberry Pi y su hardware de Raspberry Pi lo admite, utilizar el escritorio con la GUI puede resultar más fácil.
- Una WiFi conexión Ethernet o.
- Teclado, ratón, monitor, cables, fuentes de alimentación y demás hardware que necesite el dispositivo.

> **Important**
>
> Antes de continuar con el siguiente paso, el sistema operativo del dispositivo debe estar instalado, configurado y en funcionamiento. El dispositivo debe estar conectado a internet y

deberá poder acceder al dispositivo mediante su interfaz de línea de comandos. El acceso a través de la línea de comandos se puede realizar mediante un teclado, un ratón y un monitor conectados directamente, o mediante una interfaz remota de terminal SSH.

Si utiliza un sistema operativo en la Raspberry Pi que tiene una interfaz gráfica de usuario (GUI), abra una ventana del terminal en el dispositivo y siga estas instrucciones en la ventana. De lo contrario, si se conecta al dispositivo mediante un terminal remoto, como PuTTY, abra un terminal remoto en el dispositivo y úselo.

Instale las herramientas y bibliotecas necesarias para el SDK del AWS IoT dispositivo

Antes de instalar el SDK del AWS IoT dispositivo y el código de muestra, asegúrese de que el sistema esté actualizado y de que disponga de las herramientas y bibliotecas necesarias para instalarlo SDKs.

1. Actualizar el sistema operativo e instalar las bibliotecas necesarias

   Antes de instalar un SDK para AWS IoT dispositivos, ejecuta estos comandos en una ventana de terminal del dispositivo para actualizar el sistema operativo e instalar las bibliotecas necesarias.

   ```
   sudo apt-get update
   ```

   ```
   sudo apt-get upgrade
   ```

   ```
   sudo apt-get install cmake
   ```

   ```
   sudo apt-get install libssl-dev
   ```

2. Instalar Git

   Si el sistema operativo de tu dispositivo no incluye Git instalado, debes instalarlo para instalar el SDK del AWS IoT dispositivo JavaScript.

   a. Ejecute este comando para comprobar si Git ya está instalado.

      ```
      git --version
      ```

b. Si el comando anterior devuelve la versión de Git, significa que Git ya está instalado y puede ir directamente al paso 3.

c. Si aparece un error al ejecutar el comando git, instale Git ejecutando este comando.

```
sudo apt-get install git
```

d. Ejecute este comando para volver a comprobar si Git está instalado.

```
git --version
```

e. Si Git está instalado, vaya a la siguiente sección. Si no es así, solucione el problema y corrija el error antes de continuar. Necesitas Git para instalar el SDK del AWS IoT dispositivo JavaScript.

Instala AWS IoT el SDK del dispositivo

Instale el SDK AWS IoT del dispositivo.

Python

En esta sección, instalará Python, sus herramientas de desarrollo y el SDK de AWS IoT dispositivos para Python en su dispositivo. Estas instrucciones son para una Raspberry Pi que ejecute la versión más reciente del sistema operativo Raspberry Pi. Si tiene otro dispositivo o utiliza otro sistema operativo, es posible que tenga que adaptar estas instrucciones a su dispositivo.

1. Instalar Python y sus herramientas de desarrollo

   El SDK de AWS IoT dispositivo para Python requiere que Python v3.5 o posterior esté instalado en tu Raspberry Pi.

   En una ventana del terminal de su dispositivo, ejecute estos comandos.

   1. Ejecute este comando para determinar la versión de Python instalada en el dispositivo.

      ```
      python3 --version
      ```

      Si Python está instalado, mostrará su versión.

   2. Si la versión que se muestra es `Python 3.5` o superior, puede ir directamente al paso 2.

3. Si la versión que se muestra es inferior a `Python 3.5`, puede instalar la versión correcta ejecutando este comando.

```
sudo apt install python3
```

4. Ejecute este comando para confirmar que ahora está instalada la versión correcta de Python.

```
python3 --version
```

2. Prueba de pip3

   En una ventana del terminal de su dispositivo, ejecute estos comandos.

   1. Ejecute este comando para comprobar si pip3 está instalado.

```
pip3 --version
```

   2. Si el comando devuelve un número de versión, significa que pip3 está instalado y puede ir al paso 3.
   3. Si el comando anterior devuelve un error, ejecute este comando para instalar pip3.

```
sudo apt install python3-pip
```

   4. Ejecute este comando para comprobar si pip3 está instalado.

```
pip3 --version
```

3. Instalar el SDK de AWS IoT dispositivo actual para Python

   Instala el SDK de AWS IoT dispositivos para Python y descarga las aplicaciones de muestra en tu dispositivo.

   En el dispositivo, ejecute estos comandos.

```
cd ~
python3 -m pip install awsiotsdk
```

```
git clone https://github.com/aws/aws-iot-device-sdk-python-v2.git
```

JavaScript

En esta sección, instalarás Node.js, el administrador de paquetes npm y el SDK para AWS IoT dispositivos JavaScript en tu dispositivo. Estas instrucciones son para una Raspberry Pi que ejecute el sistema operativo Raspberry Pi. Si tiene otro dispositivo o utiliza otro sistema operativo, es posible que tenga que adaptar estas instrucciones a su dispositivo.

1. Instale la versión más reciente de Node.js.

   El SDK del AWS IoT dispositivo JavaScript requiere que Node.js y el administrador de paquetes npm estén instalados en tu Raspberry Pi.

   a. Descargue la última versión del repositorio de Node mediante el siguiente comando.

   ```
   cd ~
   curl -sL https://deb.nodesource.com/setup_12.x | sudo -E bash -
   ```

   b. Instale Node y npm.

   ```
   sudo apt-get install -y nodejs
   ```

   c. Verifique la instalación de Node.

   ```
   node -v
   ```

   Confirme que el comando muestra la versión de Node. Este tutorial requiere la versión Node v10.0 o posterior. Si no se muestra la versión de Node, pruebe a descargar de nuevo el repositorio de Node.

   d. Verifique la instalación de npm.

   ```
   npm -v
   ```

   Confirme que el comando muestra la versión de npm. Si no se muestra la versión de npm, pruebe a instalar Node y npm de nuevo.

   e. Reinicie el dispositivo.

   ```
   sudo shutdown -r 0
   ```

   Continúe después de que se reinicie el dispositivo.

2. Instale el SDK del AWS IoT dispositivo para JavaScript

   Instala el SDK del AWS IoT dispositivo JavaScript en tu Raspberry Pi.

   a. Clona el SDK del AWS IoT dispositivo para el JavaScript repositorio en el `aws-iot-device-sdk-js-v2` directorio de tu *home* directorio. En la Raspberry Pi`~/`, el *home* directorio es el que se utiliza como *home* directorio en los siguientes comandos. Si su dispositivo usa una ruta diferente para el *home* directorio, debe `~/` reemplazarla por la ruta correcta para su dispositivo en los siguientes comandos.

      Estos comandos crean el directorio `~/aws-iot-device-sdk-js-v2` y copian el código del SDK en él.

      ```
      cd ~
      git clone https://github.com/aws/aws-iot-device-sdk-js-v2.git
      ```

   b. Cambie al directorio `aws-iot-device-sdk-js-v2` que creó en el paso anterior y ejecute `npm install` para instalar el SDK. El comando `npm install` invocará la creación de la biblioteca `aws-crt`, que puede tardar algunos minutos en completarse.

      ```
      cd ~/aws-iot-device-sdk-js-v2
      npm install
      ```

Instalar y ejecutar la aplicación de ejemplo

En esta sección, instalarás y ejecutarás la aplicación de `pubsub` muestra que se encuentra en el SDK del AWS IoT dispositivo. Esta aplicación muestra cómo su dispositivo utiliza la biblioteca de MQTT para publicar y suscribirse a mensajes MQTT. La aplicación de ejemplo se suscribe a un tema, `topic_1`, publica 10 mensajes sobre ese tema y muestra los mensajes tal como los recibe del agente de mensajes.

Instalar los archivos de certificado

La aplicación de ejemplo requiere que los archivos de certificado que autentican el dispositivo estén instalados en el dispositivo.

Para instalar los archivos de certificado del dispositivo para la aplicación de ejemplo

1. Crea un `certs` subdirectorio en tu *home* directorio ejecutando estos comandos.

```
cd ~
mkdir certs
```

2. En el directorio `~/certs`, copie la clave privada, el certificado de entidad de certificación del dispositivo y el certificado de entidad de certificación raíz que creó anteriormente en the section called “Crea AWS IoT recursos”.

   La forma de copiar los archivos de certificado en el dispositivo depende del dispositivo y del sistema operativo, y no se describe aquí. Sin embargo, si el dispositivo admite una interfaz gráfica de usuario (GUI) y tiene un navegador web, puede realizar el procedimiento descrito en the section called “Crea AWS IoT recursos” en el navegador web del dispositivo para descargar los archivos resultantes directamente al dispositivo.

   Los comandos de la siguiente sección suponen que los archivos de clave y certificado se almacenan en el dispositivo como se muestra en esta tabla.

   Nombres de archivo de certificado

Archivos	Ruta de archivo
Certificado de entidad de certificación raíz	`~/certs/Amazon-root-CA-1.pem`
Certificado de dispositivo	`~/certs/device.pem.crt`
Clave privada	`~/certs/private.pem.key`

Para ejecutar la aplicación de ejemplo, necesita la siguiente información:

Valores de los parámetros de la aplicación

Parámetro	Dónde encontrar el valor
`your-iot-endpoint`	En la consola de AWS IoT, seleccione Todos los dispositivos y, a continuación, Objetos. En la página Configuración del menú de AWS IoT . Su punto de conexión aparece en la sección Punto de enlace de datos de dispositi vo.

El *your-iot-endpoint* valor tiene el formato:*endpoint_id*-ats.iot.*region*.amazonaws.com, por ejemplo,a3qj468EXAMPLE-ats.iot.us-west-2.amazonaws.com.

Python

Para instalar y ejecutar la aplicación de ejemplo

1. Navegue hasta el directorio de la aplicación de ejemplo.

```
cd ~/aws-iot-device-sdk-python-v2/samples
```

2. En la ventana de la línea de comandos, *your-iot-endpoint* sustituya como se indica y ejecute este comando.

```
python3 pubsub.py --topic topic_1 --ca_file ~/certs/Amazon-root-CA-1.pem --
cert ~/certs/device.pem.crt --key ~/certs/private.pem.key --endpoint your-iot-
endpoint
```

3. Observe que la aplicación de ejemplo:

   1. Se conecta al AWS IoT servicio de su cuenta.
   2. Se suscribe al tema de mensaje, topic_1, y muestra los mensajes que recibe sobre ese tema.
   3. Publica 10 mensajes sobre el tema, topic_1.
   4. Muestra una salida similar a la siguiente:

```
Connecting to a3qEXAMPLEffp-ats.iot.us-west-2.amazonaws.com with client ID
 'test-0c8ae2ff-cc87-49d2-a82a-ae7ba1d0ca5a'...
Connected!
Subscribing to topic 'topic_1'...
Subscribed with QoS.AT_LEAST_ONCE
Sending 10 message(s)
Publishing message to topic 'topic_1': Hello World! [1]
Received message from topic 'topic_1': b'Hello World! [1]'
Publishing message to topic 'topic_1': Hello World! [2]
Received message from topic 'topic_1': b'Hello World! [2]'
Publishing message to topic 'topic_1': Hello World! [3]
Received message from topic 'topic_1': b'Hello World! [3]'
Publishing message to topic 'topic_1': Hello World! [4]
```

```
Received message from topic 'topic_1': b'Hello World! [4]'
Publishing message to topic 'topic_1': Hello World! [5]
Received message from topic 'topic_1': b'Hello World! [5]'
Publishing message to topic 'topic_1': Hello World! [6]
Received message from topic 'topic_1': b'Hello World! [6]'
Publishing message to topic 'topic_1': Hello World! [7]
Received message from topic 'topic_1': b'Hello World! [7]'
Publishing message to topic 'topic_1': Hello World! [8]
Received message from topic 'topic_1': b'Hello World! [8]'
Publishing message to topic 'topic_1': Hello World! [9]
Received message from topic 'topic_1': b'Hello World! [9]'
Publishing message to topic 'topic_1': Hello World! [10]
Received message from topic 'topic_1': b'Hello World! [10]'
10 message(s) received.
Disconnecting...
Disconnected!
```

Si tiene problemas para ejecutar la aplicación de ejemplo, revise the section called “Solución de problemas con la aplicación de ejemplo”.

También puede agregar el parámetro `--verbosity Debug` a la línea de comandos para que la aplicación de ejemplo muestre mensajes detallados sobre lo que está haciendo. Esa información podría proporcionarle la ayuda que necesita para corregir el problema.

JavaScript

Para instalar y ejecutar la aplicación de ejemplo

1. En la ventana de la línea de comandos, navegue hasta el directorio `~/aws-iot-device-sdk-js-v2/samples/node/pub_sub` que el SDK creó e instale la aplicación de ejemplo con estos comandos. El comando `npm install` invocará la creación de la biblioteca `aws-crt`, que puede tardar algunos minutos en completarse.

   ```
   cd ~/aws-iot-device-sdk-js-v2/samples/node/pub_sub
   npm install
   ```

2. En la ventana de la línea de comandos, reemplace *`your-iot-endpoint`* como se indica y ejecute este comando.

```
node dist/index.js --topic topic_1 --ca_file ~/certs/Amazon-root-CA-1.pem --
cert ~/certs/device.pem.crt --key ~/certs/private.pem.key --endpoint your-iot-
endpoint
```

3. Observe que la aplicación de ejemplo:

   1. Se conecta al AWS IoT servicio de su cuenta.
   2. Se suscribe al tema de mensaje, topic_1, y muestra los mensajes que recibe sobre ese tema.
   3. Publica 10 mensajes sobre el tema, topic_1.
   4. Muestra una salida similar a la siguiente:

```
Publish received on topic topic_1
{"message":"Hello world!","sequence":1}
Publish received on topic topic_1
{"message":"Hello world!","sequence":2}
Publish received on topic topic_1
{"message":"Hello world!","sequence":3}
Publish received on topic topic_1
{"message":"Hello world!","sequence":4}
Publish received on topic topic_1
{"message":"Hello world!","sequence":5}
Publish received on topic topic_1
{"message":"Hello world!","sequence":6}
Publish received on topic topic_1
{"message":"Hello world!","sequence":7}
Publish received on topic topic_1
{"message":"Hello world!","sequence":8}
Publish received on topic topic_1
{"message":"Hello world!","sequence":9}
Publish received on topic topic_1
{"message":"Hello world!","sequence":10}
```

   Si tiene problemas para ejecutar la aplicación de ejemplo, revise the section called “Solución de problemas con la aplicación de ejemplo”.

   También puede agregar el parámetro `--verbosity Debug` a la línea de comandos para que la aplicación de ejemplo muestre mensajes detallados sobre lo que está haciendo. Esa información podría proporcionarle la ayuda que necesita para corregir el problema.

Vea los mensajes de la aplicación de ejemplo en la AWS IoT consola

Puede ver los mensajes de la aplicación de ejemplo a medida que pasan por el agente de mensajes mediante el cliente de prueba de MQTT de la consola de AWS IoT .

Para ver los mensajes MQTT publicados por la aplicación de ejemplo

1. Consulte Vea los mensajes MQTT con el cliente AWS IoT MQTT. Esto le ayudará a aprender a utilizar el cliente de prueba de MQTT de la consola de AWS IoT para ver los mensajes MQTT a medida que pasan por el agente de mensajes.
2. Abra la opción Cliente de prueba de MQTT en la consola de AWS IoT .
3. Para suscribirse al tema topic_1.
4. En su ventana de línea de comandos, ejecute de nuevo la aplicación de ejemplo y observe los mensajes del cliente MQTT en la consola de AWS IoT .

   Python

   ```
   cd ~/aws-iot-device-sdk-python-v2/samples
   python3 pubsub.py --topic topic_1 --ca_file ~/certs/Amazon-root-CA-1.pem --
   cert ~/certs/device.pem.crt --key ~/certs/private.pem.key --endpoint your-iot-
   endpoint
   ```

   JavaScript

   ```
   cd ~/aws-iot-device-sdk-js-v2/samples/node/pub_sub
   node dist/index.js --topic topic_1 --ca_file ~/certs/Amazon-root-CA-1.pem --
   cert ~/certs/device.pem.crt --key ~/certs/private.pem.key --endpoint your-iot-
   endpoint
   ```

Solución de problemas con la aplicación de ejemplo

Si se produce un error al intentar ejecutar la aplicación de ejemplo, compruebe lo siguiente.

Compruebe el certificado

Si el certificado no está activo, AWS IoT no aceptará ningún intento de conexión que lo utilice para la autorización. Al crear el certificado, es fácil pasar por alto el botón Activar. Afortunadamente, es posible activar el certificado desde la consola de AWS IoT.

Para comprobar la activación del certificado

1. En la consola de AWS IoT, en el menú de la izquierda, seleccione Seguridad y, a continuación, Certificados.
2. En la lista de certificados, busque el certificado que creó para el ejercicio y compruebe su estado en la columna Estado.

   Si no recuerda el nombre del certificado, compruebe si hay alguno que esté inactivo para ver si es el que está utilizando.

   Seleccione el certificado de la lista para abrir su página de detalles. En la página de detalles, puede ver su fecha de creación para ayudarle a identificar el certificado.
3. Para activar un certificado inactivo, en la página de detalles del certificado, elija Acciones y, a continuación, seleccione Activar.

Si ha encontrado el certificado correcto y está activo, pero sigue teniendo problemas para ejecutar la aplicación de ejemplo, consulte su política, tal y como se describe en el siguiente paso.

También puede intentar crear un objeto y un certificado nuevo siguiendo los pasos que se indican en the section called “Crear un objeto”. Si crea un nuevo objeto, tendrá que asignarle un nombre nuevo y descargar los nuevos archivos de certificado en su dispositivo.

Comprobación de la política asociada al certificado

Las políticas autorizan las acciones en AWS IoT. Si el certificado utilizado para conectarse a AWS IoT no tiene una política o no tiene una política que le permita conectarse, se rechazará la conexión, incluso si el certificado está activo.

Para comprobar las políticas asociadas al certificado

1. Busque el certificado tal y como se describe en el punto anterior y abra su página de detalles.
2. En el menú de la izquierda de la página de detalles del certificado, seleccione Políticas para ver las políticas asociadas al certificado.
3. Si no hay políticas asociadas al certificado, agregue una seleccionando el menú Acciones y, a continuación, Asociar política.

   Elija la política que creó anteriormente en the section called “Crea AWS IoT recursos”.
4. Si hay una política asociada, elija el mosaico de la política para abrir su página de detalles.

En la página de detalles, revise el documento de política para asegurarse de que contiene la misma información que la que creó en the section called “Cree una AWS IoT política”.

Comprobación de la línea de comandos

Asegúrese de haber utilizado la línea de comandos correcta para su sistema. Los comandos que se utilizan en los sistemas Linux y macOS suelen ser diferentes de los que se utilizan en los sistemas Windows.

Comprobación de la dirección del punto de conexión

Revise el comando introducido y compruebe que la dirección del punto de conexión del comando coincide con la de la consola de AWS IoT.

Comprobación de los nombres de los archivos de certificado

Compare los nombres de los archivos del comando que introdujo con los nombres de los archivos de certificado del directorio `certs`.

Es posible que algunos sistemas requieran que los nombres de los archivos estén entre comillas para que funcionen correctamente.

Comprobación de la instalación del SDK

Asegúrese de que la instalación del SDK haya finalizado correctamente.

En caso de duda, vuelva a instalar el SDK en el dispositivo. En la mayoría de los casos, basta con encontrar la sección del tutorial titulada Instalar el SDK del AWS IoT dispositivo ***SDK language*** y volver a seguir el procedimiento.

Si utilizas el SDK para AWS IoT dispositivos JavaScript, recuerda instalar las aplicaciones de muestra antes de intentar ejecutarlas. Al instalar el SDK no se instalan automáticamente las aplicaciones de ejemplo. Las aplicaciones de ejemplo se deben instalar manualmente después de instalar el SDK.

Vea los mensajes MQTT con el cliente AWS IoT MQTT

En esta sección se describe cómo utilizar el cliente de prueba de AWS IoT MQTT de la AWS IoT consola para ver los mensajes MQTT enviados y recibidos por. AWS IoT El ejemplo utilizado en esta

sección se refiere a los ejemplos utilizados enCómo empezar con AWS IoT Core los tutoriales; sin embargo, puede reemplazar el *topicName* utilizado en los ejemplos por cualquier nombre de tema o filtro de tema que utilice su solución de IoT.

Los dispositivos publican mensajes MQTT que se identifican por temas para comunicar su estado y AWS IoT publican mensajes MQTT para informar a los dispositivos y aplicaciones de los cambios y eventos. AWS IoT Puede utilizar el cliente MQTT para suscribirse a estos temas y ver los mensajes a medida que se producen. También puede usar el cliente de prueba de MQTT para publicar mensajes MQTT en los dispositivos y servicios suscritos en su dispositivo. Cuenta de AWS

Contenido

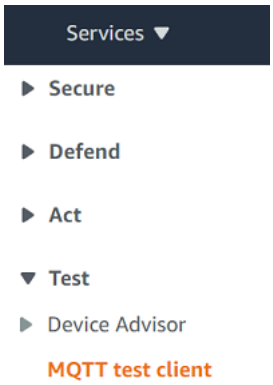
- Visualización de mensajes MQTT en el cliente MQTT
- Publicación de mensajes MQTT desde el cliente MQTT
- Probar las suscripciones compartidas en el cliente MQTT

Visualización de mensajes MQTT en el cliente MQTT

El siguiente procedimiento explica cómo suscribirse a un tema de MQTT específico en el que su dispositivo publique mensajes y cómo ver esos mensajes en la consola de AWS IoT.

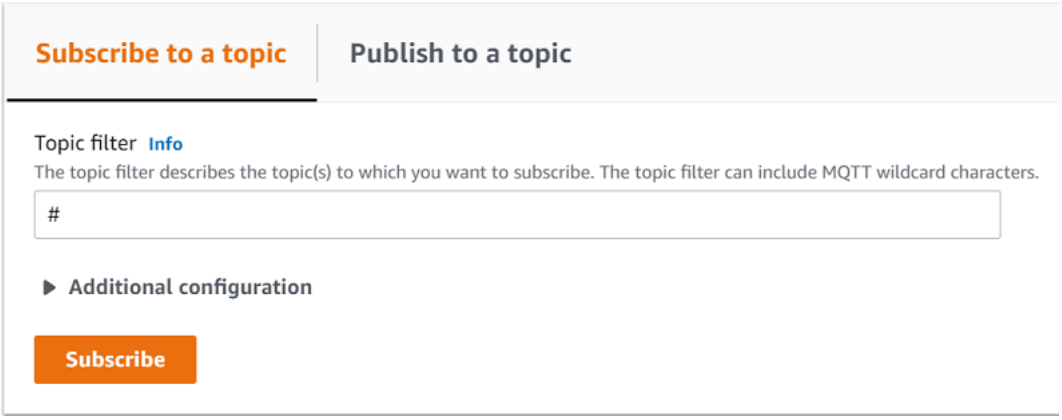
Para consultar los mensajes MQTT en el cliente MQTT

1. En la consola de AWS IoT, en el menú de la izquierda, elija Prueba y, a continuación, Cliente de prueba de MQTT.

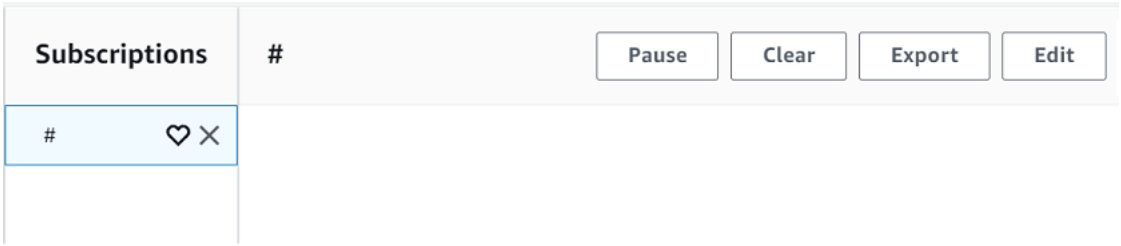


2. En la pestaña Suscribirse a un tema, introduzca la *topicName* para suscribirse al tema en el que publica su dispositivo. Para la aplicación de ejemplo de introducción, suscríbase a **#**, que se suscribe a todos los temas de los mensajes.

Continuando con el ejemplo de introducción, en la pestaña Suscribirse a un tema, en el campo Filtro de temas, introduzca **#** y, a continuación, seleccione Suscribirse.



Se abre la página de registro de mensajes del tema, #, y **#** aparece en la lista de suscripciones. Si el dispositivo que configuraste the section called “Configuración del dispositivo” ejecuta el programa de ejemplo, deberías ver los mensajes a los que se envía AWS IoT en el registro de mensajes #. Las entradas del registro de mensajes aparecerán debajo de la sección Publicar cuando se reciban mensajes con el tema suscrito. AWS IoT



3. En la página de registro de mensajes #, también puede publicar mensajes en un tema, pero tendrá que especificar el nombre del tema. No se puede publicar en el tema #.

   Los mensajes publicados en los temas suscritos aparecen en el registro de mensajes a medida que se reciben, con el mensaje más reciente primero.

Solución de problemas con los mensajes MQTT

Utilizar el filtro de temas comodín

Si sus mensajes no aparecen como esperaba en el registro de mensajes, pruebe a suscribirse a un tema comodín, tal y como se describe en Filtros de nombres de temas. El filtro de temas comodín de varios niveles de MQTT es el signo de almohadilla (#) y se puede utilizar como filtro de tema en el campo del tema de suscripción.

Al suscribirse al filtro de temas de #, se suscriben a todos los temas recibidos por el agente de mensajes. Puede restringir el filtro sustituyendo los elementos de la ruta del filtro de temas por un carácter comodín # de varios niveles o el carácter comodín de un solo nivel con el signo «+».

Cuando se utilizan caracteres comodín en un filtro de temas

- El carácter comodín de varios niveles debe ser el último carácter del filtro de temas.
- La ruta del filtro de temas solo puede tener un carácter comodín de un solo nivel por cada nivel de tema.

Por ejemplo:

Filtro de temas	Muestra mensajes con
#	Cualquier nombre de tema
`topic_1/#`	Un nombre de tema que comience por `topic_1/`
`topic_1/level_2/#`	Un nombre de tema que comience por `topic_1/level_2/`
`topic_1/+/level_3`	Un nombre de tema que comience por `topic_1/`, termine por `/level_3` y tenga un elemento de cualquier valor intermedio.

Para obtener más información acerca de los filtros de temas, consulte Filtros de nombres de temas.

Comprobación de errores en los nombres de temas

Los nombres de temas y los filtros de temas de MQTT distinguen entre mayúsculas y minúsculas. Si, por ejemplo, su dispositivo está publicando mensajes en `Topic_1` (con T mayúscula) en lugar de `topic_1`, el tema al que se suscribió, sus mensajes no aparecerían en el cliente de prueba de MQTT. Sin embargo, si se suscribe al filtro de temas comodín, se mostrará que el dispositivo está publicando mensajes y podría ver que está usando un nombre de tema que no es el que se esperaba.

Publicación de mensajes MQTT desde el cliente MQTT

Para publicar un mensaje en un tema de MQTT

1. En la página del cliente de prueba de MQTT, en la pestaña Publicar en un tema, en el campo Nombre del tema, introduzca su *topicName* mensaje. En este ejemplo, use **my/topic**.

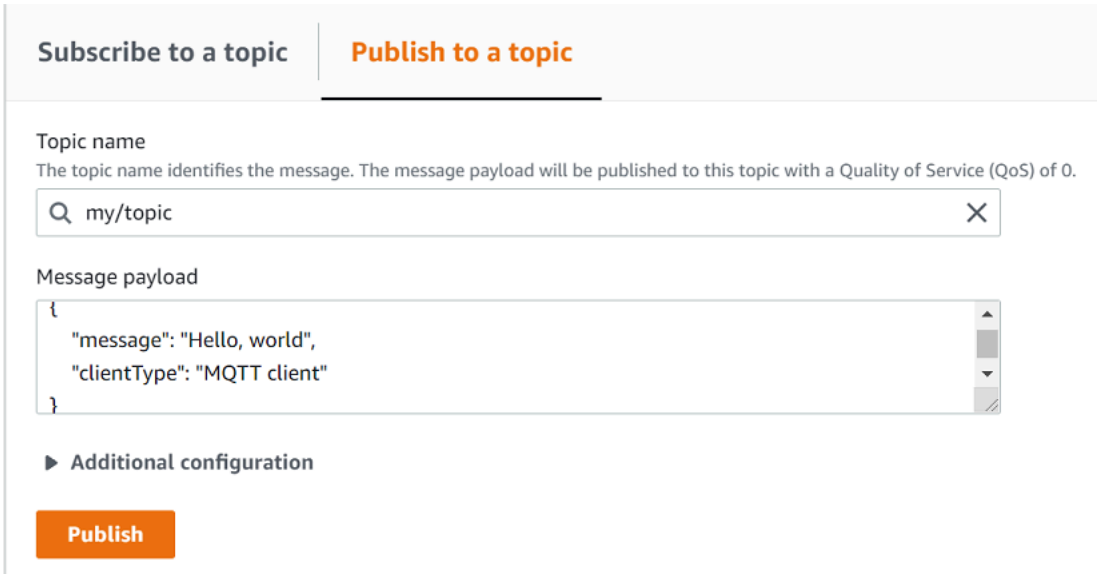
   > **Note**
   >
   > No utilice información de identificación personal en nombres de temas si los utiliza en el cliente MQTT o en la implementación de su sistema. Los nombres de temas pueden aparecer en comunicaciones e informes no cifrados.

2. En la ventana de carga del mensaje, introduzca el siguiente JSON:

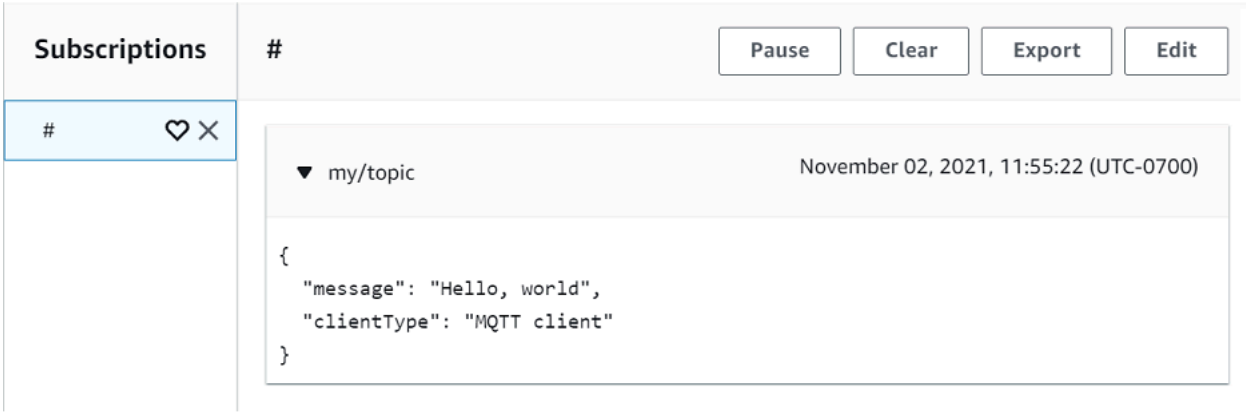
   ```
   {
       "message": "Hello, world",
       "clientType": "MQTT test client"
   }
   ```

3. Elija Publicar para publicar el mensaje en AWS IoT.

   > **Note**
   >
   > Asegúrese de estar suscrito al tema my/topic antes de publicar su mensaje.



4. En la columna Suscripciones, elija my/topic para ver el mensaje. Debería ver el mensaje en el cliente de prueba de MQTT, debajo de la ventana de carga para publicar el mensaje.



Puede publicar mensajes de MQTT *`topicName`* en otros temas cambiando el campo Nombre del tema y pulsando el botón Publicar.

> **Important**
>
> Al crear varias suscripciones con temas superpuestos (por ejemplo, probe1/temperature y probe1/#), existe la posibilidad de que un solo mensaje publicado sobre un tema que coincida con ambas suscripciones se entregue varias veces, una por cada suscripción superpuesta.

Probar las suscripciones compartidas en el cliente MQTT

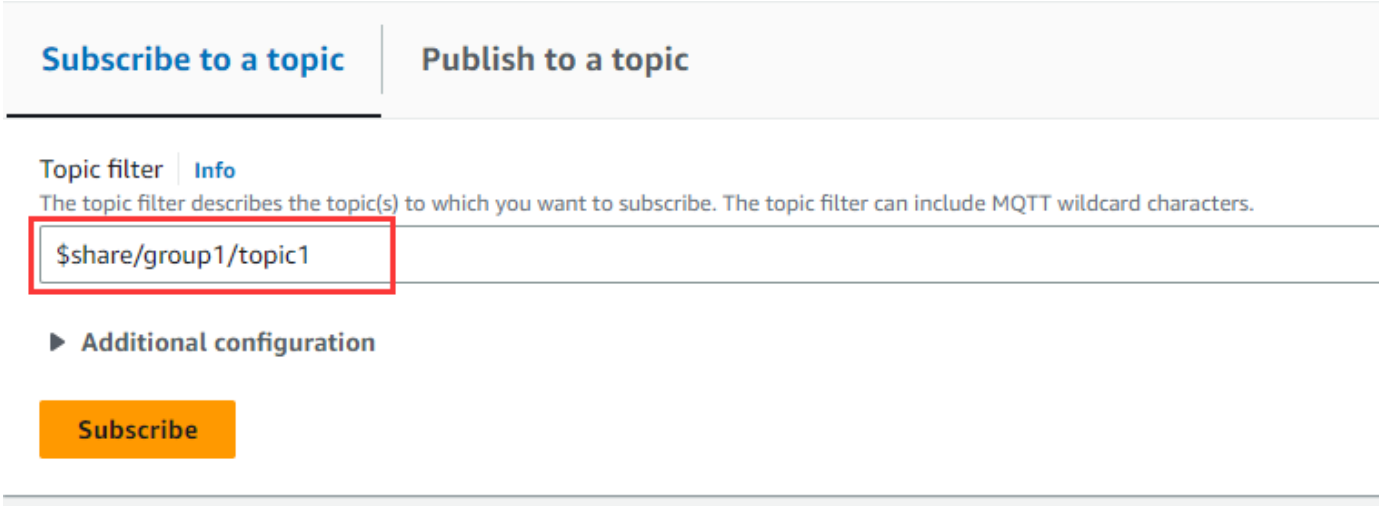
En esta sección se describe cómo utilizar el cliente AWS IoT MQTT de la AWS IoT consola para ver los mensajes MQTT enviados y recibidos AWS IoT mediante suscripciones compartidas. ???permiten que varios clientes compartan una suscripción a un tema y que solo un cliente reciba los mensajes publicados sobre ese tema mediante una distribución aleatoria. Para simular varios clientes MQTT (en este ejemplo, dos clientes MQTT) que comparten la misma suscripción, abra el cliente AWS IoT MQTT en la AWS IoT consola desde varios navegadores web. El ejemplo utilizado en esta sección no está relacionado con los ejemplos utilizados en Cómo empezar con AWS IoT Core los tutoriales. Para obtener más información, consulte Suscripciones compartidas.

Para compartir una suscripción a un tema de MQTT

1. En el panel de navegación de la consola de AWS IoT, seleccione Prueba y, a continuación, Cliente de prueba de MQTT.
2. En la pestaña Suscribirse a un tema, introduzca la *topicName* para suscribirse al tema en el que publica su dispositivo. Para usar las suscripciones compartidas, subscríbase al filtro de temas de una suscripción compartida de la siguiente manera:

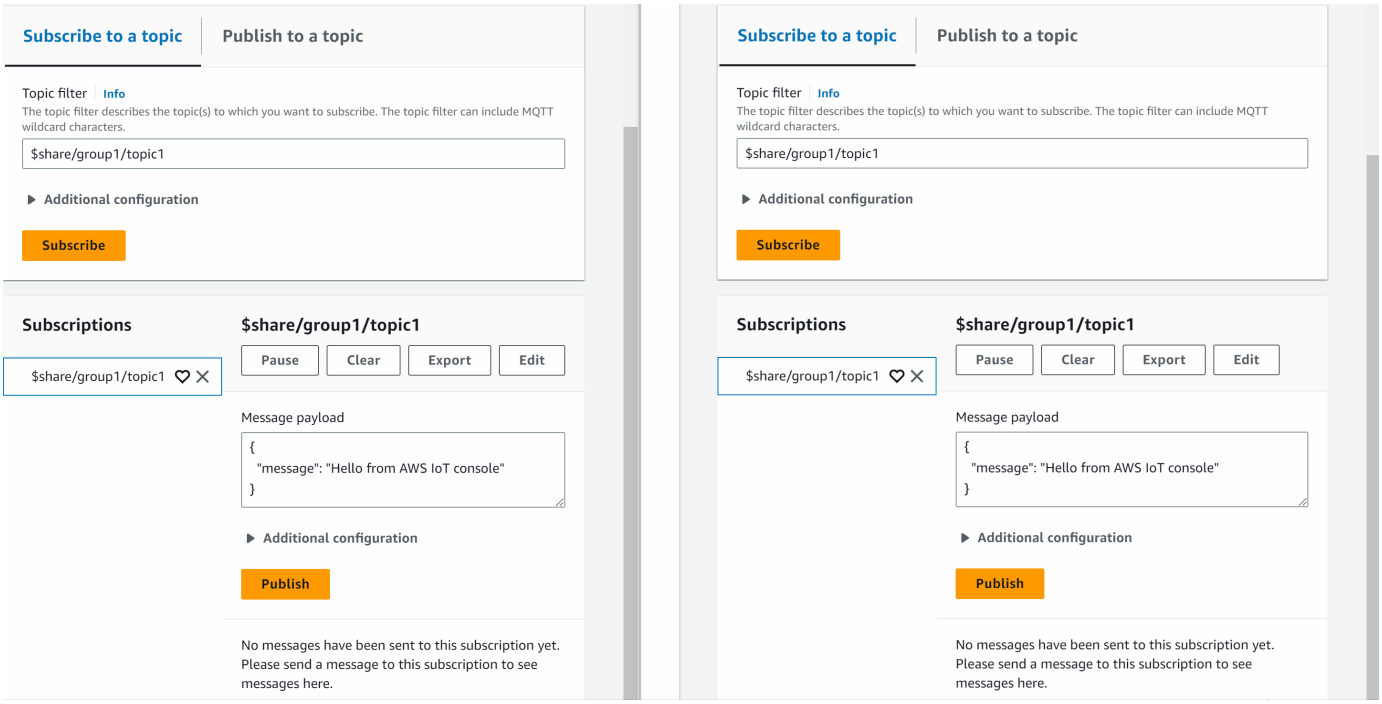
   ```
   $share/{ShareName}/{TopicFilter}
   ```

   Un ejemplo de filtro de temas puede ser **$share/group1/topic1**, que se suscribe al tema **topic1** del mensaje.

3. Abra otro navegador web y repita los pasos 1 y 2. De esta forma, está simulando dos clientes MQTT diferentes que comparten la misma suscripción **$share/group1/topic1**.

4. Elija un cliente MQTT, en la pestaña Publicar en un tema, en el campo Nombre del tema, introduzca el mensaje que *topicName* desee enviar. En este ejemplo, use **topic1**. Intente publicar el mensaje varias veces. En la lista Suscripciones de ambos clientes MQTT, debería poder ver que los clientes reciben el mensaje mediante una distribución aleatoria. En este ejemplo, publicamos el mismo mensaje “Hola desde la consola de AWS IoT ” tres veces. El cliente MQTT de la izquierda recibió el mensaje dos veces y el cliente MQTT de la derecha recibió el mensaje una vez.



AWS IoT tutoriales

Los AWS IoT tutoriales se dividen en dos rutas de aprendizaje para apoyar dos objetivos diferentes. Elija la mejor ruta de aprendizaje para su objetivo.

- ¿Desea crear una idea de AWS IoT solución proof-of-concept para probarla o demostrarla

  Para demostrar las tareas y aplicaciones comunes de IoT mediante el AWS IoT Device Client en sus dispositivos, siga la ruta de the section called “Creación de demostraciones con el cliente de AWS IoT dispositivos” aprendizaje. El AWS IoT Device Client proporciona un software de dispositivo con el que puede aplicar sus propios recursos en la nube para demostrar una end-to-end solución con un desarrollo mínimo.

  Para obtener información sobre el cliente del AWS IoT dispositivo, consulte el cliente del AWS IoT dispositivo.
- Desea aprender a crear software de producción para implementar una solución

  Para crear su propia solución de software que cumpla con sus requisitos específicos mediante un SDK para AWS IoT dispositivos, siga la ruta de the section called “Creación de soluciones con el dispositivo AWS IoT SDKs” aprendizaje.

  Para obtener información sobre el AWS IoT dispositivo disponible SDKs, consulte???. Para obtener información sobre el AWS SDKs, consulte Herramientas sobre las que construir AWS.

AWS IoT tutorial: opciones de ruta de aprendizaje

- Creación de demostraciones con el cliente de AWS IoT dispositivos
- Creación de soluciones con el dispositivo AWS IoT SDKs

Creación de demostraciones con el cliente de AWS IoT dispositivos

Los tutoriales de esta ruta de aprendizaje explican los pasos necesarios para desarrollar un software de demostración mediante el AWS IoT Device Client. El AWS IoT Device Client proporciona un software que se ejecuta en su dispositivo de IoT para probar y demostrar aspectos de una solución de IoT basada en ella AWS IoT.

El objetivo de estos tutoriales es facilitar la exploración y la experimentación para que pueda estar seguro de que es AWS IoT compatible con su solución antes de desarrollar el software de su dispositivo.

Lo que aprenderá en estos tutoriales:

- Cómo preparar una Raspberry Pi para usarla como dispositivo IoT con AWS IoT
- ¿Cómo demostrar AWS IoT las funciones mediante el AWS IoT Device Client de su dispositivo

En esta ruta de aprendizaje, instalará el AWS IoT Device Client en su propia Raspberry Pi y creará los AWS IoT recursos en la nube para demostrar ideas de soluciones de IoT. Si bien los tutoriales de esta ruta de aprendizaje muestran las características que se utilizan con una Raspberry Pi, en ellos se explican los objetivos y los procedimientos para ayudarlo a adaptarlas a otros dispositivos.

Requisitos previos para crear demostraciones con el AWS IoT Device Client

En esta sección se describe lo que necesitará antes de comenzar los tutoriales de esta ruta de aprendizaje.

Para completar los tutoriales de esta ruta de aprendizaje, necesitará lo siguiente:

- ¿Un Cuenta de AWS

  Puedes usar los existentes Cuenta de AWS, si los tienes, pero es posible que necesites agregar roles o permisos adicionales para usar las AWS IoT funciones que utilizan estos tutoriales.

  Si necesita crear uno nuevo Cuenta de AWS, consultethe section called “Configurar Cuenta de AWS”.
- Una Raspberry Pi o un dispositivo de IoT compatible

  Los tutoriales usan una Raspberry Pi porque viene en diferentes formatos y es un dispositivo de demostración de uso común y relativamente económico. Los tutoriales se han probado en la Raspberry Pi 3 Modelo B+, la Raspberry Pi 4 Modelo B y en una instancia de Amazon EC2 que ejecuta Ubuntu Server 20.04 LTS (HVM). Para utilizar los comandos AWS CLI y ejecutarlos, le recomendamos que utilice la última versión del sistema operativo Raspberry Pi (sistema operativo Raspberry Pi (64 bits) o OS Lite). Este tutorial puede funcionar en versiones anteriores del sistema operativo, pero no lo hemos probado.

> **Note**
>
> Los tutoriales explican los objetivos de cada paso para ayudarlo a adaptarlos al hardware de IoT en el que no los hemos probado; sin embargo, no describen específicamente cómo adaptarlos a otros dispositivos.

- Familiaridad con el sistema operativo del dispositivo de IoT

  En los pasos de estos tutoriales se supone que está familiarizado con el uso de los comandos y operaciones básicos de Linux desde la interfaz de línea de comandos compatible con una Raspberry Pi. Si no está familiarizado con estas operaciones, quizás desee darse más tiempo para completar los tutoriales.

  Para completar estos tutoriales, ya debe saber cómo:

  - Realizar de forma segura las operaciones básicas del dispositivo, como ensamblar y conectar los componentes, conectar el dispositivo a las fuentes de alimentación necesarias e instalar y extraer las tarjetas de memoria.
  - Cargar y descargar el software y los archivos del sistema en el dispositivo. Si el dispositivo no utiliza un dispositivo de almacenamiento extraíble, como una tarjeta microSD, tendrá que saber cómo conectarse al dispositivo y cómo cargar y descargar el software del sistema y los archivos en el dispositivo.
  - Conectar el dispositivo a las redes en las que piensa usarlo.
  - Conectarse al dispositivo desde otro equipo mediante un terminal SSH o un programa similar.
  - Usar una interfaz de línea de comandos para crear, copiar, mover, cambiar el nombre y configurar los permisos de los archivos y directorios del dispositivo.
  - Instalar nuevos programas en el dispositivo.
  - Transferir archivos desde y hacia el dispositivo mediante herramientas como FTP o SCP.
- Un entorno de desarrollo y pruebas para la solución de IoT

  En los tutoriales se describe el software y el hardware necesarios; sin embargo, se supone que podrá realizar operaciones que tal vez no estén descritas de forma explícita. Algunos ejemplos de este tipo de hardware y operaciones son:

  - Un equipo host local para descargar y almacenar archivos

Para la Raspberry Pi, suele ser una computadora personal o portátil que puede leer y escribir en tarjetas de memoria microSD. La equipo host local debe:

- Estar conectado a Internet.
- Tener instalada y configurada la AWS CLI.
- Tenga un navegador web compatible con la AWS consola.

- Una forma de conectar el equipo host local al dispositivo para comunicarse con él, introducir comandos y transferir archivos

  En la Raspberry Pi, esto suele hacerse mediante SSH y SCP desde el equipo host local.

- Un monitor y teclado para conectarse al dispositivo de IoT

  Estos pueden ser útiles, pero no son obligatorios para completar los tutoriales.

- Una forma de que su equipo host local y sus dispositivos de IoT se conecten a Internet

  Puede ser una conexión de red cableada o inalámbrica a un router o puerta de enlace que estén conectados a Internet. El host local también debe poder conectarse a la Raspberry Pi. Esto puede requerir que estén en la misma red de área local. Los tutoriales no muestran cómo configurar esto para el dispositivo o configuración de dispositivo en particular, pero sí cómo puede probar esta conectividad.

- Acceder al router de la red de área local para ver los dispositivos conectados

  Para completar los tutoriales de esta ruta de aprendizaje, tendrá que poder encontrar la dirección IP del dispositivo de IoT.

  En una red de área local, esto se puede hacer accediendo a la interfaz de administración del router de red al que se conectan los dispositivos. Si puede asignar una dirección IP fija al dispositivo en el router, puede simplificar la reconexión cada vez que se reinicie el dispositivo.

  Si tiene un teclado y un monitor conectados al dispositivo, ifconfig puede mostrar la dirección IP del dispositivo.

  Si ninguna de estas opciones es posible, tendrá que encontrar una forma de identificar la dirección IP del dispositivo cada vez que se reinicie.

Una vez que tenga todos los materiales, pase a the section called “Preparación para usar IoT Device Client”.

Tutoriales en esta ruta de aprendizaje

- Tutorial: Preparación de los dispositivos para el cliente de AWS IoT dispositivos
- Tutorial: Instalación y configuración del cliente de AWS IoT dispositivos
- Tutorial: Demuestre la comunicación por mensaje MQTT con el cliente del AWS IoT dispositivo
- Tutorial: Demuestre acciones remotas (trabajos) con el cliente del AWS IoT dispositivo
- Tutorial: Limpiar después de ejecutar los tutoriales de AWS IoT Device Client

Tutorial: Preparación de los dispositivos para el cliente de AWS IoT dispositivos

En este tutorial se explica cómo inicializar la Raspberry Pi para prepararla para los siguientes tutoriales de esta ruta de aprendizaje.

El objetivo de este tutorial es instalar la versión actual del sistema operativo del dispositivo y asegurarse de que puede comunicarse con él en el contexto de su entorno de desarrollo.

Requisitos previos

Antes de empezar este tutorial, asegúrese de que dispone de los elementos enumerados en the section called “Requisitos previos para crear demostraciones con el AWS IoT Device Client” y que se puedan usar.

Para completar este tutorial se necesitan aproximadamente 90 minutos.

En este tutorial, podrá:

- Instale y actualice el sistema operativo del dispositivo.
- Instale y verifique cualquier software adicional necesario para ejecutar los tutoriales.
- Pruebe la conectividad del dispositivo e instale los certificados necesarios.

Tras completar este tutorial, el siguiente prepara el dispositivo para las demostraciones que utilizan el AWS IoT Device Client.

Procedimientos de este tutorial

- Instalación y actualización del sistema operativo del dispositivo
- Instalación y verificación del software necesario en el dispositivo
- Prueba del dispositivo y almacenamiento del certificado de CA de Amazon

Instalación y actualización del sistema operativo del dispositivo

En los procedimientos de esta sección, se describe cómo inicializar la tarjeta microSD que la Raspberry Pi utiliza como unidad de sistema. La tarjeta microSD de la Raspberry Pi contiene el software de su sistema operativo (SO), así como espacio para el almacenamiento de los archivos de la aplicación. Si no utiliza una Raspberry Pi, siga las instrucciones del dispositivo para instalar y actualizar el software del sistema operativo del dispositivo.

Después de completar esta sección, debería poder iniciar el dispositivo IoT y conectarse a él desde el programa del terminal de su ordenador host local.

Equipo necesario:

- Su entorno local de desarrollo y pruebas
- Una Raspberry Pi o su dispositivo IoT (que se pueda conectar a Internet)
- Una tarjeta de memoria microSD con una capacidad mínima de 8 GB o suficiente espacio de almacenamiento para el sistema operativo y el software necesario.

> **Note**
>
> Al seleccionar una tarjeta microSD para estos ejercicios, elija una que sea tan grande como sea necesario pero lo más pequeña posible.
> Será más rápido realizar copias de seguridad y actualizar una tarjeta SD pequeña. En la Raspberry Pi, no necesitará más que una tarjeta microSD de 8 GB para estos tutoriales. Si necesita más espacio para su aplicación específica, los archivos de imagen más pequeños que guarde en estos tutoriales pueden cambiar el tamaño del sistema de archivos de una tarjeta más grande para ocupar todo el espacio compatible de la tarjeta que elija.

Equipamiento opcional:

- Un teclado USB conectado a la Raspberry Pi
- Un monitor HDMI y un cable para conectar el monitor a la Raspberry Pi

Procedimientos de esta sección:

- Cargue el sistema operativo del dispositivo en la tarjeta microSD
- Iniciar el dispositivo IoT con el nuevo sistema operativo

- Conectar el ordenador host local al dispositivo

Cargue el sistema operativo del dispositivo en la tarjeta microSD

Este procedimiento utiliza el ordenador host local para cargar el sistema operativo del dispositivo en una tarjeta microSD.

> **Note**
>
> Si el dispositivo no utiliza un medio de almacenamiento extraíble para su sistema operativo, instale el sistema operativo siguiendo el procedimiento de ese dispositivo y continúe con the section called “Inicio del dispositivo de IoT”.

Para instalar el sistema operativo en su Raspberry Pi

1. En el ordenador host local, descargue y descomprima la imagen del sistema operativo Raspberry Pi que desee usar. Las versiones más recientes están disponibles en https://www.raspberrypi.com/software/operating-systems/

   Elección de una versión del sistema operativo Raspberry Pi

   Este tutorial usa la versión Raspberry Pi OS Lite porque es la versión más pequeña que admite los tutoriales de esta ruta de aprendizaje. Esta versión del sistema operativo Raspberry Pi solo tiene una interfaz de línea de comandos y no tiene una interfaz gráfica de usuario. Con estos tutoriales también funcionará una versión del sistema operativo Raspberry Pi más reciente con una interfaz gráfica de usuario; sin embargo, los procedimientos descritos en esta ruta de aprendizaje utilizan únicamente la interfaz de línea de comandos para realizar operaciones en la Raspberry Pi.
2. Inserte su tarjeta microSD en el ordenador host local.
3. Con una herramienta de creación de imágenes de tarjetas SD, escriba el archivo de imagen del sistema operativo descomprimido en la tarjeta microSD.
4. Después de escribir la imagen del sistema operativo Raspberry Pi en la tarjeta microSD:

   a. Abra la partición BOOT de la tarjeta microSD en una ventana de línea de comandos o en una ventana del explorador de archivos.

   b. En la partición BOOT de la tarjeta microSD, en el directorio raíz, cree un archivo vacío denominado `ssh` sin extensión de archivo ni contenido. Esto le indica a la Raspberry Pi que habilite las comunicaciones SSH la primera vez que se inicie.

5. Extraiga la tarjeta microSD y extráigala de forma segura del ordenador host local.

La tarjeta microSD está lista para the section called “Inicio del dispositivo de IoT”.

Iniciar el dispositivo IoT con el nuevo sistema operativo

Este procedimiento instala la tarjeta microSD e inicia la Raspberry Pi por primera vez con el sistema operativo descargado.

Para poner en marcha su dispositivo IoT con el nuevo sistema operativo

1. Con la alimentación desconectada del dispositivo, insert la tarjeta microSD del paso anterior, the section called “Carga del sistema operativo”, en la Raspberry Pi.
2. Conecte el dispositivo a una red cableada.
3. Estos tutoriales interactuarán con su Raspberry Pi desde el ordenador host local mediante un terminal SSH.

   Si también quiere interactuar directamente con el dispositivo, puede:

   a. Conectar un monitor HDMI para ver los mensajes de la consola de la Raspberry Pi antes de poder conectar la ventana del terminal del ordenador host local a la Raspberry Pi.
   b. Conectar un teclado USB si quiere interactuar directamente con la Raspberry Pi.

4. Conectar la Raspberry Pi a la alimentación y esperar aproximadamente un minuto a que se inicialice.

   Si tiene un monitor conectado a la Raspberry Pi, puede usarlo para ver el proceso de arranque.

5. Encuentre la dirección IP de su dispositivo:

   - Si ha conectado un monitor HDMI a la Raspberry Pi, la dirección IP aparece en los mensajes del monitor
   - Si tiene acceso al router al que está conectada la Raspberry Pi, puede ver su dirección en la interfaz de administración del router.

Una vez que tenga la dirección IP de la Raspberry Pi, estará listo para the section called “Conexión del equipo host”.

Conectar el ordenador host local al dispositivo

Este procedimiento utiliza el programa del terminal del ordenador host local para conectarse a su Raspberry Pi y cambiar la contraseña predeterminada.

Para conectar el ordenador host local al dispositivo

1. En el ordenador host local, abra el programa del terminal SSH:

   - Windows: `PuTTY`
   - Linux/macOS: `Terminal`

   > **Note**
   >
   > PuTTY no se instala automáticamente en Windows. Si no está en su ordenador, es posible que tenga que descargarlo e instalarlo.

2. Conecte el programa del terminal a la dirección IP de la Raspberry Pi e inicie sesión con sus credenciales predeterminadas.

   ```
   username: pi
   password: raspberry
   ```

3. Después de iniciar sesión en la Raspberry Pi, cambie la contraseña del usuario `pi`.

   ```
   passwd
   ```

   Siga las instrucciones para cambiar la contraseña.

   ```
   Changing password for pi.
   Current password: raspberry
   New password: YourNewPassword
   Retype new password: YourNewPassword
   passwd: password updated successfully
   ```

Una vez que aparezca la línea de comandos de la Raspberry Pi en la ventana del terminal y hayas cambiado la contraseña, estará listo para continuar con the section called “Instalación y verificación del software necesario”.

Instalación y verificación del software necesario en el dispositivo

Los procedimientos de esta sección son la continuación de los de la sección anterior para actualizar el sistema operativo de su Raspberry Pi e instalar el software en la Raspberry Pi que se utilizará en la siguiente sección para crear e instalar el Device Client. AWS IoT

Después de completar esta sección, tu Raspberry Pi tendrá un sistema up-to-date operativo, el software requerido en los tutoriales de esta ruta de aprendizaje, y se configurará para tu ubicación.

Equipo necesario:

- Su entorno local de desarrollo y pruebas de la sección anterior
- La Raspberry Pi que utilizó en la sección anterior
- La tarjeta de memoria microSD de la sección anterior

> **Note**
>
> La Raspberry Pi Model 3+ y la Raspberry Pi Model 4 pueden ejecutar todos los comandos descritos en esta ruta de aprendizaje. Si tu dispositivo de IoT no puede compilar software ni ejecutarlo AWS Command Line Interface, es posible que tengas que instalar los compiladores necesarios en tu ordenador host local para crear el software y, a continuación, transferirlo a tu dispositivo de IoT. Para obtener más información acerca de cómo instalar y crear software para su dispositivo, consulte la documentación del software de su dispositivo.

Procedimientos de esta sección:

- Actualización del software del sistema operativo
- Instalación de las aplicaciones y las bibliotecas necesarias
- (Opcional) Guardar la imagen de la tarjeta microSD

Actualización del software del sistema operativo

Este procedimiento actualiza el software del sistema operativo.

Para actualizar el software del sistema operativo de la Raspberry Pi

Realice estos pasos en la ventana del terminal de su ordenador host local.

1. Escriba estos comandos para actualizar el software del sistema de la Raspberry Pi.

```
sudo apt-get -y update
sudo apt-get -y upgrade
sudo apt-get -y autoremove
```

2. Actualice la configuración regional y de zona horaria de la Raspberry Pi (opcional).

   Escriba este comando para actualizar la configuración regional y de zona horaria del dispositivo.

```
sudo raspi-config
```

   a. Para establecer la configuración regional del dispositivo:

      i. En la pantalla Raspberry Pi Software Configuration Tool (raspi-config), elija la opción 5.

         5 Localisation Options Configure language and regional settings

         Utilice la tecla Tab para desplazarse a <Select> y, a continuación, presione la space bar.

      ii. En el menú de opciones de localización, elija la opción L1.

         L1 Locale Configure language and regional settings

         Utilice la tecla Tab para desplazarse a <Select> y, a continuación, presione la space bar.

      iii. En la lista de opciones de configuración regional, elija las configuraciones regionales que desee instalar en la Raspberry Pi utilizando las teclas de flecha para desplazarse y space bar para marcar las que quiera.

         En Estados Unidos, **en_US.UTF-8** es una buena opción para elegir.

      iv. Tras seleccionar las configuraciones regionales del dispositivo, utilice la tecla Tab para ir a <OK> y, a continuación, presione la space bar para que aparezca la página de confirmación de configuración regional.

   b. Para configurar la zona horaria del dispositivo:

i. En la pantalla raspi-config, seleccione la opción 5.

`5 Localisation Options Configure language and regional settings`

Utilice la tecla Tab para desplazarse a <Select> y, a continuación, presione la space bar.

ii. En el menú de opciones de localización, utilice la tecla de flecha para elegir la opción L2:

`L2 time zone Configure time zone`

Utilice la tecla Tab para desplazarse a <Select> y, a continuación, presione la space bar.

iii. En el menú Configuración de tzdata, seleccione su área geográfica de la lista.

Utilice la tecla Tab para desplazarse a <OK> y, a continuación, presione la space bar.

iv. En la lista de ciudades, use las teclas de flecha para elegir una ciudad de su zona horaria.

Para configurar la zona horaria, utilice la tecla Tab para ir a <OK> y, a continuación, presione la space bar.

c. Cuando termine de actualizar la configuración, usa la tecla Tab para ir a <Finish> y, a continuación, presione la space bar para cerrar la aplicación raspi-config.

3. Escriba este comando para reiniciar la Raspberry Pi.

```
sudo shutdown -r 0
```

4. Espere a que la Raspberry Pi se reinicie.
5. Una vez reiniciada la Raspberry Pi, vuelva a conectar la ventana del terminal del ordenador host local a la Raspberry Pi.

El software del sistema de la Raspberry Pi ya está configurado y está listo para continuar con the section called “Instalación de las aplicaciones y las bibliotecas”.

Instalación de las aplicaciones y las bibliotecas necesarias

Este procedimiento instala el software de la aplicación y las bibliotecas que se utilizan en los tutoriales siguientes.

Si utiliza una Raspberry Pi o si puede compilar el software necesario en el dispositivo IoT, realice estos pasos en la ventana del terminal del ordenador host local. Si debe compilar software para su dispositivo IoT en el ordenador host local, consulte la documentación del software del dispositivo IoT para obtener información sobre cómo realizar estos pasos en el dispositivo.

Instalación de la aplicación, el software y las bibliotecas en la Raspberry Pi

1. Escriba este comando para instalar el software y las bibliotecas de la aplicación.

```
sudo apt-get -y install build-essential libssl-dev cmake unzip git python3-pip
```

2. Escriba estos comandos para confirmar que se instaló la versión correcta del software.

```
gcc --version
cmake --version
openssl version
git --version
```

3. Confirme que estén instaladas las siguientes versiones del software de la aplicación:

   - `gcc`: 9.3.0 o versiones posteriores
   - `cmake`: 3.10.x o versiones posteriores
   - `OpenSSL`: 1.1.1 o versiones posteriores
   - `git`: 2.20.1 o versiones posteriores

Si la Raspberry Pi tiene versiones aceptables del software de aplicación requerido, está listo para continuar con the section called “(Opcional) Almacenamiento de la imagen de microSD”.

(Opcional) Guardar la imagen de la tarjeta microSD

A lo largo de los tutoriales de esta ruta de aprendizaje, encontrará estos procedimientos para guardar una copia de la imagen de la tarjeta microSD de la Raspberry Pi en un archivo de su ordenador host local. Si bien se recomiendan, no son tareas obligatorias. Al guardar la imagen de la tarjeta microSD donde se sugiere, puede saltarte los procedimientos que preceden al punto de guardado en esta ruta de aprendizaje, lo que puede ahorrar tiempo si necesita volver a intentar algo. La consecuencia de no guardar la imagen de la tarjeta microSD periódicamente es que puede que tenga que reiniciar los tutoriales de la ruta de aprendizaje desde el principio si la tarjeta microSD está dañada si accidentalmente configura una aplicación o sus ajustes de forma incorrecta.

En este punto, la tarjeta microSD de la Raspberry Pi tiene un sistema operativo actualizado y el software de aplicación básico cargado. Puede ahorrar el tiempo que le llevó completar los pasos anteriores guardando ahora el contenido de la tarjeta microSD en un archivo. Tener la imagen actual de la imagen de la tarjeta microSD de su dispositivo le permite empezar desde este punto para continuar o volver a intentar un tutorial o procedimiento sin necesidad de instalar y actualizar el software desde cero.

Para guardar la imagen de la tarjeta microSD en un archivo

1. Escriba este comando para apagar la Raspberry Pi.

   ```
   sudo shutdown -h 0
   ```

2. Una vez apagada la Raspberry Pi por completo, desconecte su alimentación.
3. Extraiga la tarjeta microSD de la Raspberry Pi.
4. En el ordenador host local:

   a. Inserte la tarjeta microSD.

   b. Con la herramienta de creación de imágenes de tarjetas SD, guarde la imagen de la tarjeta microSD en un archivo.

   c. Una vez guardada la imagen de la tarjeta microSD, extraiga la tarjeta del ordenador host local.

5. Con la alimentación desconectada de la Raspberry Pi, inserte la tarjeta microSD en la Raspberry Pi.
6. Alimente la Raspberry Pi.
7. Tras esperar aproximadamente un minuto, en el ordenador host local, vuelva a conectar la ventana del terminal del computadora host local que estaba conectada a la Raspberry Pi, y luego inicie sesión en la Raspberry Pi.

Prueba del dispositivo y almacenamiento del certificado de CA de Amazon

Los procedimientos de esta sección son los mismos que los de la sección anterior para instalar el certificado de la autoridad de certificación utilizado para autenticar las conexiones AWS Command Line Interface y el certificado con el que se autentican las conexiones. AWS IoT Core

Después de completar esta sección, sabrás que tu Raspberry Pi tiene el software de sistema necesario para instalar el cliente del AWS IoT dispositivo y que tiene una conexión a Internet que funciona.

Equipo necesario:

- Su entorno local de desarrollo y pruebas de la sección anterior
- La Raspberry Pi que utilizó en la sección anterior
- La tarjeta de memoria microSD de la sección anterior

Procedimientos de esta sección:

- Instala el AWS Command Line Interface
- Configura tus Cuenta de AWS credenciales
- Descargar el certificado de entidad de certificación raíz de Amazon
- (Opcional) Guardar la imagen de la tarjeta microSD

Instala el AWS Command Line Interface

Este procedimiento lo instala AWS CLI en su Raspberry Pi.

Si utiliza una Raspberry Pi o si puede compilar el software en el dispositivo IoT, realice estos pasos en la ventana del terminal del ordenador host local. Si debe compilar software para su dispositivo IoT en el ordenador host local, consulte la documentación del software del dispositivo IoT para obtener información sobre las bibliotecas necesarias.

Para instalarlo AWS CLI en tu Raspberry Pi

1. Ejecute estos comandos para descargar e instalar la AWS CLI.

```
export PATH=$PATH:~/.local/bin # configures the path to include the directory with
 the AWS CLI
git clone https://github.com/aws/aws-cli.git # download the AWS CLI code from
 GitHub
cd aws-cli && git checkout v2 # go to the directory with the repo and checkout
 version 2
pip3 install -r requirements.txt # install the prerequisite software
```

2. Ejecute este comando para instalar el AWS CLI. Este comando puede tardar hasta 15 minutos en finalizar.

```
pip3 install . # install the AWS CLI
```

3. Ejecute este comando para confirmar que se AWS CLI ha instalado la versión correcta de.

```
aws --version
```

   La versión de AWS CLI debe ser 2.2 o posterior.

Si AWS CLI se muestra la versión actual, estás listo para continuar con ellathe section called "Configuración de las credenciales de la cuenta".

Configura tus Cuenta de AWS credenciales

En este procedimiento, obtendrá Cuenta de AWS las credenciales y las agregará para usarlas en su Raspberry Pi.

Para añadir tus Cuenta de AWS credenciales a tu dispositivo

1. Obtenga un identificador de clave de acceso y una clave de acceso secreta Cuenta de AWS para autenticarlos AWS CLI en su dispositivo.

   Si es la primera vez que utiliza AWS IAM, https://aws.amazon.com/premiumsupport/knowledge-center/create-access-key/describe el proceso que debe ejecutarse en la AWS consola para crear las credenciales de AWS IAM y utilizarlas en su dispositivo.
2. En la ventana del terminal del ordenador host local que esté conectado a la Raspberry Pi y con las credenciales de clave de acceso y clave de acceso secreta del dispositivo:

   a. Ejecute la aplicación de AWS configuración con este comando:

```
aws configure
```

   b. Introduzca las credenciales y la información de configuración cuando se le pida:

```
AWS Access Key ID: your Access Key ID
AWS Secret Access Key: your Secret Access Key
Default region name: your Región de AWS code
```

```
Default output format: json
```

3. Ejecute este comando para probar el acceso del dispositivo a su AWS IoT Core terminal Cuenta de AWS y al terminal.

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

Debería devolver tu punto final Cuenta de AWS de AWS IoT datos específico, como en este ejemplo:

```
{
    "endpointAddress": "a3EXAMPLEffp-ats.iot.us-west-2.amazonaws.com"
}
```

Si ves tu punto final Cuenta de AWS de AWS IoT datos específico, tu Raspberry Pi tiene la conectividad y los permisos necesarios para seguir accediendo a él. the section called “Descargar el certificado de entidad de certificación raíz de Amazon”

> **Important**
>
> Sus Cuenta de AWS credenciales ahora están almacenadas en la tarjeta microSD de su Raspberry Pi. Si bien esto facilita tus futuras interacciones con el software que crearás en estos tutoriales, también se guardarán y duplicarán en cualquier imagen de tarjeta microSD que hagas después de este paso de AWS forma predeterminada.
> Para proteger la seguridad de sus Cuenta de AWS credenciales, antes de guardar más imágenes de tarjetas microSD, considere la posibilidad de borrar las credenciales ejecutándolas `aws configure` nuevamente e ingresando caracteres aleatorios para el ID de la clave de acceso y la clave de acceso secreta para evitar que sus Cuenta de AWS credenciales se vean comprometidas.
> Si descubre que ha guardado sus Cuenta de AWS credenciales sin darse cuenta, puede desactivarlas en la consola de IAM. AWS

Descargar el certificado de entidad de certificación raíz de Amazon

Este procedimiento descarga y guarda una copia de un certificado de la entidad de certificación (CA) raíz de Amazon. Al descargar este certificado, se guarda para usarlo en los siguientes tutoriales y, además, se comprueba la conectividad del dispositivo con los servicios de AWS .

Para descargar y guardar el certificado de entidad de certificación raíz de Amazon

1. Ejecute este comando para crear un directorio para el certificado.

   ```
   mkdir ~/certs
   ```

2. Ejecute este comando para descargar el certificado de entidad de certificación raíz de Amazon.

   ```
   curl -o ~/certs/AmazonRootCA1.pem https://www.amazontrust.com/repository/
   AmazonRootCA1.pem
   ```

3. Ejecute estos comandos para configurar el acceso al directorio del certificado y a su archivo.

   ```
   chmod 745 ~
   chmod 700 ~/certs
   chmod 644 ~/certs/AmazonRootCA1.pem
   ```

4. Ejecute este comando para ver el archivo de certificado de entidad de certificación en el nuevo directorio.

   ```
   ls -l ~/certs
   ```

   Debería ver una entrada como esta. La fecha y la hora serán diferentes; sin embargo, el tamaño del archivo y el resto de la información deben ser los mismos que se muestran aquí.

   ```
   -rw-r--r-- 1 pi pi 1188 Oct 28 13:02 AmazonRootCA1.pem
   ```

   Si el tamaño del archivo no es 1188, compruebe los parámetros del comando curl. Puede que haya descargado un archivo incorrecto.

(Opcional) Guardar la imagen de la tarjeta microSD

En este punto, la tarjeta microSD de la Raspberry Pi tiene un sistema operativo actualizado y el software de aplicación básico cargado.

Para guardar la imagen de la tarjeta microSD en un archivo

1. En la ventana del terminal del ordenador host local, borre sus credenciales de AWS .

   a. Ejecute la aplicación de AWS configuración con este comando:

```
aws configure
```

b. Reemplace las credenciales cuando se le solicite. Puede dejar el nombre de región predeterminado y el formato de salida predeterminado tal como están presionando Intro.

```
AWS Access Key ID [****************YT2H]: XYXYXYXYX
AWS Secret Access Key [****************9plH]: XYXYXYXYX
Default region name [us-west-2]:
Default output format [json]:
```

2. Escriba este comando para apagar la Raspberry Pi.

```
sudo shutdown -h 0
```

3. Una vez apagada la Raspberry Pi por completo, desconecte su alimentación.
4. Extraiga la tarjeta microSD del dispositivo.
5. En el ordenador host local:

   a. Inserte la tarjeta microSD.

   b. Con la herramienta de creación de imágenes de tarjetas SD, guarde la imagen de la tarjeta microSD en un archivo.

   c. Una vez guardada la imagen de la tarjeta microSD, extraiga la tarjeta del ordenador host local.

6. Con la alimentación desconectada de la Raspberry Pi, inserte la tarjeta microSD en la Raspberry Pi.
7. Alimente el dispositivo.
8. Después de aproximadamente un minuto, en el ordenador host local, reinicie la sesión de la ventana del terminal e inicie sesión en el dispositivo.

   No vuelvas a introducir tus Cuenta de AWS credenciales todavía.

Una vez que haya reiniciado la Raspberry Pi e iniciado sesión de nuevo, estará listo para continuar con the section called “Instalación y configuración del cliente de dispositivo de IoT”.

Tutorial: Instalación y configuración del cliente de AWS IoT dispositivos

Este tutorial explica cómo instalar y configurar el AWS IoT Device Client y cómo crear los AWS IoT recursos que utilizará en esta y otras demostraciones.

Para comenzar este tutorial:

- Tenga preparados su ordenador host local y la Raspberry Pi del tutorial anterior.

Para completar este tutorial se necesitan aproximadamente 90 minutos.

Cuando haya terminado con este tema:

- Su dispositivo IoT estará listo para usarse en otras demostraciones de AWS IoT Device Client.
- Habrás aprovisionado tu dispositivo IoT. AWS IoT Core
- Habrás descargado e instalado el AWS IoT Device Client en tu dispositivo.
- Habrá guardado una imagen de la tarjeta microSD de su dispositivo que podrá usar en tutoriales posteriores.

Equipo necesario:

- Su entorno local de desarrollo y pruebas de la sección anterior
- La Raspberry Pi que utilizó en la sección anterior
- La tarjeta de memoria microSD de la Raspberry Pi que utilizó en la sección anterior

Procedimientos de este tutorial

- Descarga y guarda el cliente AWS IoT del dispositivo
- Aprovisione su Raspberry Pi AWS IoT
- Configura el cliente AWS IoT del dispositivo para probar la conectividad

Descarga y guarda el cliente AWS IoT del dispositivo

Los procedimientos de esta sección descargan el AWS IoT Device Client, lo compilan y lo instalan en su Raspberry Pi. Después de probar la instalación, puede guardar la imagen de la tarjeta microSD de la Raspberry Pi para usarla más tarde cuando quiera volver a probar los tutoriales.

Procedimientos de esta sección:

- Descarga y compila el cliente de AWS IoT dispositivo
- Creación de los directorios utilizados en los tutoriales
- (Opcional) Guardar la imagen de la tarjeta microSD

Descarga y compila el cliente de AWS IoT dispositivo

Este procedimiento instala el cliente de AWS IoT dispositivo en su Raspberry Pi.

Ejecute estos comandos en la ventana del terminal de su ordenador host local que esté conectado a su Raspberry Pi.

Para instalar el cliente de AWS IoT dispositivo en su Raspberry Pi

1. Introduzca estos comandos para descargar y construir el cliente de AWS IoT dispositivo en su Raspberry Pi.

   ```
   cd ~
   git clone https://github.com/awslabs/aws-iot-device-client aws-iot-device-client
   mkdir ~/aws-iot-device-client/build && cd ~/aws-iot-device-client/build
   cmake ../
   ```

2. Ejecute este comando para crear el AWS IoT Device Client. Este comando puede tardar hasta 15 minutos en finalizar.

   ```
   cmake --build . --target aws-iot-device-client
   ```

   Se pueden ignorar los mensajes de advertencia que aparecen cuando el AWS IoT Device Client compila.

   Estos tutoriales se han probado con el AWS IoT Device Client integradogcc, la versión 10.2.1 20210110 (Raspbian 10.2.1-6+rpi1), la versión 8.3.0 del 30 de octubre de 2021 del sistema operativo Raspberry Pi (bullseye) y la versión 8.3.0 del 7 de mayo gcc de 2021 del sistema operativo Raspberry Pi (buster).

3. Cuando el cliente del dispositivo termine de compilarse, pruébelo ejecutando este comando. AWS IoT

   ```
   ./aws-iot-device-client --help
   ```

Si ve la ayuda de la línea de comandos del AWS IoT Device Client, significa que el AWS IoT Device Client se ha creado correctamente y está listo para su uso.

Creación de los directorios utilizados en los tutoriales

Este procedimiento crea los directorios en la Raspberry Pi que se utilizarán para almacenar los archivos utilizados en los tutoriales de esta ruta de aprendizaje.

Para crear los directorios utilizados en los tutoriales de esta ruta de aprendizaje:

1. Ejecute estos comandos para crear los directorios necesarios.

```
mkdir ~/dc-configs
mkdir ~/policies
mkdir ~/messages
mkdir ~/certs/testconn
mkdir ~/certs/pubsub
mkdir ~/certs/jobs
```

2. Ejecute estos comandos para establecer los permisos en los nuevos directorios.

```
chmod 745 ~
chmod 700 ~/certs/testconn
chmod 700 ~/certs/pubsub
chmod 700 ~/certs/jobs
```

Tras crear estos directorios y establecer sus permisos, continúe con the section called “(Opcional) Guardar la imagen de la tarjeta microSD”.

(Opcional) Guardar la imagen de la tarjeta microSD

En este punto, la tarjeta microSD de tu Raspberry Pi tiene un sistema operativo actualizado, el software de aplicación básico y el cliente del AWS IoT dispositivo.

Si quiere volver a probar estos ejercicios y tutoriales, puede omitir los procedimientos anteriores y escribir la imagen de la tarjeta microSD que guarda con este procedimiento en una nueva tarjeta microSD desde la que continuar con los tutoriales de the section called “Aprovisionamiento de Raspberry Pi”.

Para guardar la imagen de la tarjeta microSD en un archivo:

En la ventana del terminal de su ordenador host local que esté conectado a la Raspberry Pi:

1. Confirma que tus Cuenta de AWS credenciales no estén almacenadas.

   a. Ejecute la aplicación de AWS configuración con este comando:

   ```
   aws configure
   ```

   b. Si sus credenciales están almacenadas (si aparecen en el mensaje), introduzca la cadena **XYXYXYXYX** cuando se le pida, tal y como se muestra aquí. Deje en blanco el nombre de la región y el formato de salida predeterminado.

   ```
   AWS Access Key ID [****************YXYX]: XYXYXYXYX
   AWS Secret Access Key [****************YXYX]: XYXYXYXYX
   Default region name:
   Default output format:
   ```

2. Escriba este comando para apagar la Raspberry Pi.

   ```
   sudo shutdown -h 0
   ```

3. Una vez apagada la Raspberry Pi por completo, desconecte su alimentación.
4. Extraiga la tarjeta microSD del dispositivo.
5. En el ordenador host local:

   a. Inserte la tarjeta microSD.

   b. Con la herramienta de creación de imágenes de tarjetas SD, guarde la imagen de la tarjeta microSD en un archivo.

   c. Una vez guardada la imagen de la tarjeta microSD, extraiga la tarjeta del ordenador host local.

Puede continuar con esta tarjeta microSD en the section called “Aprovisionamiento de Raspberry Pi”.

Aprovisione su Raspberry Pi AWS IoT

Los procedimientos de esta sección comienzan con la imagen microSD guardada que tiene instalado el cliente AWS CLI and AWS IoT Device y crean los AWS IoT recursos y los certificados de dispositivo que aprovisionan tu Raspberry Pi. AWS IoT

Instalación de la tarjeta microSD en la Raspberry Pi

Este procedimiento instala la tarjeta microSD con el software necesario cargado y configurado en la Raspberry Pi y la configura Cuenta de AWS para que pueda continuar con los tutoriales de esta ruta de aprendizaje.

Utilice una tarjeta microSD de the section called “(Opcional) Guardar la imagen de la tarjeta microSD” que tenga el software necesario para los ejercicios y tutoriales de esta ruta de aprendizaje.

Para instalar la tarjeta microSD en la Raspberry Pi

1. Con la alimentación desconectada de la Raspberry Pi, inserte la tarjeta microSD en la Raspberry Pi.
2. Alimente la Raspberry Pi.
3. Después de aproximadamente un minuto, en el ordenador host local, reinicie la sesión de la ventana del terminal e inicie sesión en la Raspberry Pi.
4. En su ordenador host local, en la ventana del terminal y con las credenciales ID de clave de acceso y Clave de acceso secreta de su Raspberry Pi:

   a. Ejecute la aplicación de AWS configuración con este comando:

   ```
   aws configure
   ```

   b. Introduzca sus Cuenta de AWS credenciales e información de configuración cuando se le solicite:

   ```
   AWS Access Key ID [****************YXYX]: your Access Key ID
   AWS Secret Access Key [****************YXYX]: your Secret Access Key
   Default region name [us-west-2]: your Región de AWS code
   Default output format [json]: json
   ```

Una vez que haya restaurado sus Cuenta de AWS credenciales, estará listo para continuar haciéndolothe section called “Aprovisione su dispositivo en AWS IoT Core”.

Aprovisione su dispositivo en AWS IoT Core

Los procedimientos de esta sección crean los AWS IoT recursos que aprovisionan tu Raspberry Pi AWS IoT. A medida que cree estos recursos, se le pedirá que registre varios datos. La configuración del cliente del AWS IoT dispositivo utilizará esta información en el siguiente procedimiento.

Para que su Raspberry Pi funcione AWS IoT, debe estar aprovisionada. El aprovisionamiento es el proceso de crear y configurar los AWS IoT recursos necesarios para respaldar su Raspberry Pi como dispositivo IoT.

Con la Raspberry Pi encendida y reiniciada, conecte la ventana del terminal del ordenador host local a la Raspberry Pi y siga estos procedimientos.

Procedimientos de esta sección:

- Crear y descargar archivos de certificado del dispositivo
- Crea recursos AWS IoT

Crear y descargar archivos de certificado del dispositivo

Este procedimiento crea los archivos de certificado del dispositivo para esta demostración.

Para crear y descargar los archivos de certificado del dispositivo para la Raspberry Pi

1. En la ventana del terminal de su ordenador host local, introduzca estos comandos para crear los archivos de certificado para su dispositivo.

   ```
   mkdir ~/certs/testconn
   aws iot create-keys-and-certificate \
   --set-as-active \
   --certificate-pem-outfile "~/certs/testconn/device.pem.crt" \
   --public-key-outfile "~/certs/testconn/public.pem.key" \
   --private-key-outfile "~/certs/testconn/private.pem.key"
   ```

   El comando devuelve una respuesta similar a la siguiente: Anote el valor *certificateArn*: lo necesitará más adelante.

   ```
   {
       "certificateArn": "arn:aws:iot:us-
   west-2:57EXAMPLE833:cert/76e7e4edb3e52f52334be2f387a06145b2aa4c7fcd810f3aea2d92abc227d269",
       "certificateId":
    "76e7e4edb3e52f5233EXAMPLE7a06145b2aa4c7fcd810f3aea2d92abc227d269",
       "certificatePem": "-----BEGIN CERTIFICATE-----
   \nMIIDWTCCAkGgAwIBAgI_SHORTENED_FOR_EXAMPLE_Lgn4jfgtS\n-----END CERTIFICATE-----
   \n",
       "keyPair": {
   ```

```
        "PublicKey": "-----BEGIN PUBLIC KEY-----
\nMIIBIjANBgkqhkiG9w0BA_SHORTENED_FOR_EXAMPLE_ImwIDAQAB\n-----END PUBLIC KEY-----
\n",
        "PrivateKey": "-----BEGIN RSA PRIVATE KEY-----
\nMIIEowIBAAKCAQE_SHORTENED_FOR_EXAMPLE_T9RoDiukY\n-----END RSA PRIVATE KEY-----\n"
    }
}
```

2. Introduzca los siguientes comandos para configurar los permisos en el directorio de certificados y sus archivos.

```
chmod 745 ~
chmod 700 ~/certs/testconn
chmod 644 ~/certs/testconn/*
chmod 600 ~/certs/testconn/private.pem.key
```

3. Ejecute este comando para revisar los permisos de sus directorios y archivos de certificados.

```
ls -l ~/certs/testconn
```

   El resultado del comando debe ser el mismo que el que se muestra aquí, excepto que las fechas y horas de los archivos serán diferentes.

```
-rw-r--r-- 1 pi pi 1220 Oct 28 13:02 device.pem.crt
-rw------- 1 pi pi 1675 Oct 28 13:02 private.pem.key
-rw-r--r-- 1 pi pi  451 Oct 28 13:02 public.pem.key
```

En este punto, ya tiene los archivos de certificado del dispositivo instalados en su Raspberry Pi y puede continuar con the section called “Crea recursos AWS IoT”.

Crea recursos AWS IoT

Este procedimiento aprovisiona el dispositivo AWS IoT mediante la creación de los recursos que el dispositivo necesita para acceder a las AWS IoT funciones y los servicios.

Para aprovisionar su dispositivo en AWS IoT

1. En la ventana del terminal de su ordenador host local, introduzca el siguiente comando para obtener la dirección del punto de conexión de datos de dispositivo correspondiente a su Cuenta de AWS.

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

El comando de los pasos anteriores devuelve una respuesta similar a la siguiente. Anote el valor *endpointAddress*: lo necesitará más adelante.

```
{
    "endpointAddress": "a3qjEXAMPLEffp-ats.iot.us-west-2.amazonaws.com"
}
```

2. Ingresa este comando para crear un recurso AWS IoT para tu Raspberry Pi.

```
aws iot create-thing --thing-name "DevCliTestThing"
```

Si se AWS IoT creó tu recurso, el comando devuelve una respuesta como esta.

```
{
    "thingName": "DevCliTestThing",
    "thingArn": "arn:aws:iot:us-west-2:57EXAMPLE833:thing/DevCliTestThing",
    "thingId": "8ea78707-32c3-4f8a-9232-14bEXAMPLEfd"
}
```

3. En la ventana del terminal:

   a. Abra un editor de texto, como `nano`.

   b. Copie este documento de política de JSON y péguelo en el editor de texto abierto.

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

```
            ]
        }
    ]
}
```

> **Note**
>
> Este documento de política otorga generosamente a todos los permisos de recursos para conectarse, recibir, publicar y suscribirse. Normalmente, las políticas solo otorgan permiso a recursos específicos para realizar acciones específicas. Sin embargo, para la prueba inicial de conectividad del dispositivo, se utiliza esta política demasiado general y permisiva para minimizar la posibilidad de que se produzca un problema de acceso durante la prueba. En los siguientes tutoriales, se utilizarán documentos de políticas con un alcance más limitado para demostrar las prácticas recomendadas de diseño de políticas.

c. Guarde el archivo en el editor de texto como **`~/policies/dev_cli_test_thing_policy.json`**.

4. Ejecute este comando para usar el documento de política de los pasos anteriores para crear una AWS IoT política.

```
aws iot create-policy \
--policy-name "DevCliTestThingPolicy" \
--policy-document "file://~/policies/dev_cli_test_thing_policy.json"
```

Si se crea la política, el comando devuelve una respuesta como esta.

```
{
    "policyName": "DevCliTestThingPolicy",
    "policyArn": "arn:aws:iot:us-west-2:57EXAMPLE833:policy/DevCliTestThingPolicy",
    "policyDocument": "{\n    \"Version\": \"2012-10-17\",       \n    \"Statement
\": [\n        {\n            \"Effect\": \"Allow\",\n            \"Action\": [\n
                \"iot:Publish\",\n                \"iot:Subscribe\",\n
    \"iot:Receive\",\n                \"iot:Connect\"\n            ],\n
 \"Resource\": [\n                \"*\"\n            ]\n        }\n    ]\n}\n",
    "policyVersionId": "1"
}
```

5. Ejecute este comando para asociar la política al certificado del dispositivo. Reemplace *certificateArn* por el valor `certificateArn` que guardó anteriormente.

```
aws iot attach-policy \
--policy-name "DevCliTestThingPolicy" \
--target "certificateArn"
```

   Si se ejecuta correctamente, este comando no devuelve nada.

6. Ejecute este comando para adjuntar el certificado del dispositivo al recurso AWS IoT del objeto. Reemplace *certificateArn* por el valor `certificateArn` que guardó anteriormente.

```
aws iot attach-thing-principal \
--thing-name "DevCliTestThing" \
--principal "certificateArn"
```

   Si se ejecuta correctamente, este comando no devuelve nada.

Una vez que hayas aprovisionado correctamente el dispositivo AWS IoT, estarás listo para continuar the section called “Configuración de Device Client para probar la conectividad” haciéndolo.

Configura el cliente AWS IoT del dispositivo para probar la conectividad

Los procedimientos de esta sección configuran el cliente del AWS IoT dispositivo para publicar un mensaje MQTT desde su Raspberry Pi.

Procedimientos de esta sección:

- Crear el archivo de configuración
- Abrir el cliente de prueba de MQTT
- Ejecute Device Client AWS IoT

Crear el archivo de configuración

Este procedimiento crea el archivo de configuración para probar el cliente del AWS IoT dispositivo.

Para crear el archivo de configuración para probar el AWS IoT Device Client

- En la ventana del terminal de su ordenador host local que esté conectado a la Raspberry Pi:

a. Introduzca estos comandos para crear un directorio para los archivos de configuración y establecer el permiso en el directorio:

```
mkdir ~/dc-configs
chmod 745 ~/dc-configs
```

b. Abra un editor de texto, como `nano`.

c. Copie este documento JSON y péguelo en el editor de texto abierto.

```
{
  "endpoint": "a3qEXAMPLEaffp-ats.iot.us-west-2.amazonaws.com",
  "cert": "~/certs/testconn/device.pem.crt",
  "key": "~/certs/testconn/private.pem.key",
  "root-ca": "~/certs/AmazonRootCA1.pem",
  "thing-name": "DevCliTestThing",
  "logging": {
    "enable-sdk-logging": true,
    "level": "DEBUG",
    "type": "STDOUT",
    "file": ""
  },
  "jobs": {
    "enabled": false,
    "handler-directory": ""
  },
  "tunneling": {
    "enabled": false
  },
  "device-defender": {
    "enabled": false,
    "interval": 300
  },
  "fleet-provisioning": {
    "enabled": false,
    "template-name": "",
    "template-parameters": "",
    "csr-file": "",
    "device-key": ""
  },
  "samples": {
    "pub-sub": {
      "enabled": true,
```

```
      "publish-topic": "test/dc/pubtopic",
      "publish-file": "",
      "subscribe-topic": "test/dc/subtopic",
      "subscribe-file": ""
    }
  },
  "config-shadow": {
    "enabled": false
  },
  "sample-shadow": {
    "enabled": false,
    "shadow-name": "",
    "shadow-input-file": "",
    "shadow-output-file": ""
  }
}
```

d. Sustituya el *endpoint* valor por el punto final de datos del dispositivo Cuenta de AWS que encontró enthe section called “Aprovisione su dispositivo en AWS IoT Core”.

e. Guarde el archivo en el editor de texto como **~/dc-configs/dc-testconn-config.json**.

f. Ejecute este comando para establecer los permisos en el nuevo archivo de configuración.

```
chmod 644 ~/dc-configs/dc-testconn-config.json
```

Después de guardar el archivo, estará listo para continuar con the section called “Abrir el cliente de prueba de MQTT”.

Abrir el cliente de prueba de MQTT

Este procedimiento prepara al cliente de prueba de MQTT de la AWS IoT consola para suscribirse al mensaje de MQTT que el cliente del AWS IoT dispositivo publica cuando se ejecuta.

Para preparar el cliente de prueba de MQTT para que se suscriba a todos los mensajes de MQTT

1. En el ordenador host local, en la consola de AWS IoT, elija Cliente de prueba de MQTT.
2. En la pestaña Suscribirse a un tema, en Filtro de temas, introduzca **#** (un solo signo de almohadilla) y elija Suscribirse para suscribirse a todos los temas de MQTT.
3. Debajo de la etiqueta Suscripciones, confirme que aparece **#** (un solo signo de almohadilla).

Deje abierta la ventana con el cliente de prueba de MQTT mientras continúa con the section called "Ejecute Device Client AWS IoT".

Ejecute Device Client AWS IoT

Este procedimiento ejecuta el cliente de AWS IoT dispositivo de forma que publique un único mensaje MQTT que reciba y muestre el cliente de prueba de MQTT.

Para enviar un mensaje MQTT desde el cliente del dispositivo AWS IoT

1. Asegúrese de que tanto la ventana del terminal que está conectada a su Raspberry Pi como la ventana con el cliente de prueba de MQTT estén visibles mientras realiza este procedimiento.
2. En la ventana del terminal, introduzca estos comandos para ejecutar el AWS IoT Device Client mediante el archivo de configuración creado enthe section called "Crear el archivo de configuración".

   ```
   cd ~/aws-iot-device-client/build
   ./aws-iot-device-client --config-file ~/dc-configs/dc-testconn-config.json
   ```

   En la ventana del terminal, el cliente del AWS IoT dispositivo muestra los mensajes de información y cualquier error que se produzca durante su ejecución.

   Si no se muestra ningún error en la ventana del terminal, revise el cliente de prueba de MQTT.
3. En el cliente de prueba de MQTT, en la ventana Suscripciones, consulte el mensaje Hello World! enviado al tema del mensaje `test/dc/pubtopic`.
4. Si el cliente del AWS IoT dispositivo no muestra ningún error y aparece Hello World! enviado al `test/dc/pubtopic` mensaje en el cliente de prueba de MQTT, ha demostrado que la conexión se ha realizado correctamente.
5. En la ventana del terminal, introduzca **^C** (Ctrl-C) para detener el cliente del AWS IoT dispositivo.

Una vez que hayas demostrado que el cliente del AWS IoT dispositivo funciona correctamente en tu Raspberry Pi y con el que se puede comunicar AWS IoT, puedes continuar con. the section called "Comunicación con Device Client mediante MQTT"

Tutorial: Demuestre la comunicación por mensaje MQTT con el cliente del AWS IoT dispositivo

En este tutorial, se muestra cómo el cliente del AWS IoT dispositivo puede suscribirse y publicar mensajes MQTT, que se utilizan habitualmente en las soluciones de IoT.

Para comenzar este tutorial:

- Configure su ordenador host local y una Raspberry Pi, tal como se utilizó en la sección anterior.

  Si guardaste la imagen de la tarjeta microSD después de instalar el AWS IoT Device Client, puedes usar una tarjeta microSD con esa imagen con tu Raspberry Pi.
- Si ya has realizado esta demostración anteriormente, procura ??? eliminar todos los AWS IoT recursos que creaste en ejecuciones anteriores para evitar errores de recursos duplicados.

Para completar este tutorial se necesitan aproximadamente 45 minutos.

Cuando haya terminado con este tema:

- Habrás demostrado diferentes maneras en las que tu dispositivo IoT puede suscribirse a los mensajes MQTT AWS IoT y publicarlos en ellos. AWS IoT

Equipo necesario:

- Su entorno local de desarrollo y pruebas de la sección anterior
- La Raspberry Pi que utilizó en la sección anterior
- La tarjeta de memoria microSD de la Raspberry Pi que utilizó en la sección anterior

Procedimientos de este tutorial

- Preparación de Raspberry Pi para demostrar la comunicación de mensajes de MQTT
- Demuestre la publicación de mensajes con el cliente AWS IoT del dispositivo
- Demuestre la suscripción a los mensajes con el cliente del dispositivo AWS IoT

Preparación de Raspberry Pi para demostrar la comunicación de mensajes de MQTT

Este procedimiento crea los recursos en AWS IoT y dentro de la Raspberry Pi para demostrar la comunicación de los mensajes MQTT mediante el AWS IoT Device Client.

Procedimientos de esta sección:

- Crear los archivos de certificado para demostrar la comunicación MQTT
- Aprovisionar su dispositivo para demostrar la comunicación MQTT
- Configure el archivo de configuración del cliente del AWS IoT dispositivo y el cliente de prueba de MQTT para demostrar la comunicación con MQTT

Crear los archivos de certificado para demostrar la comunicación MQTT

Este procedimiento crea los archivos de certificado del dispositivo para esta demostración.

Para crear y descargar los archivos de certificado del dispositivo para la Raspberry Pi

1. En la ventana del terminal de su ordenador host local, introduzca el siguiente comando para crear los archivos de certificado para su dispositivo.

   ```
   mkdir ~/certs/pubsub
   aws iot create-keys-and-certificate \
   --set-as-active \
   --certificate-pem-outfile "~/certs/pubsub/device.pem.crt" \
   --public-key-outfile "~/certs/pubsub/public.pem.key" \
   --private-key-outfile "~/certs/pubsub/private.pem.key"
   ```

   El comando devuelve una respuesta similar a la siguiente. Guarde el valor de *certificateArn* para utilizarlo más tarde.

   ```
   {
   "certificateArn": "arn:aws:iot:us-
   west-2:57EXAMPLE833:cert/76e7e4edb3e52f52334be2f387a06145b2aa4c7fcd810f3aea2d92abc227d269",
   "certificateId":
    "76e7e4edb3e52f5233EXAMPLE7a06145b2aa4c7fcd810f3aea2d92abc227d269",
   "certificatePem": "-----BEGIN CERTIFICATE-----
   \nMIIDWTCCAkGgAwIBAgI_SHORTENED_FOR_EXAMPLE_Lgn4jfgtS\n-----END CERTIFICATE-----
   \n",
   "keyPair": {
   ```

```
    "PublicKey": "-----BEGIN PUBLIC KEY-----
\nMIIBIjANBgkqhkiG9w0BA_SHORTENED_FOR_EXAMPLE_ImwIDAQAB\n-----END PUBLIC KEY-----
\n",
    "PrivateKey": "-----BEGIN RSA PRIVATE KEY-----
\nMIIEowIBAAKCAQE_SHORTENED_FOR_EXAMPLE_T9RoDiukY\n-----END RSA PRIVATE KEY-----\n"
}
}
```

2. Introduzca los siguientes comandos para configurar los permisos en el directorio de certificados y sus archivos.

```
chmod 700 ~/certs/pubsub
chmod 644 ~/certs/pubsub/*
chmod 600 ~/certs/pubsub/private.pem.key
```

3. Ejecute este comando para revisar los permisos de sus directorios y archivos de certificados.

```
ls -l ~/certs/pubsub
```

   El resultado del comando debe ser el mismo que el que se muestra aquí, excepto que las fechas y horas de los archivos serán diferentes.

```
-rw-r--r-- 1 pi pi 1220 Oct 28 13:02 device.pem.crt
-rw------- 1 pi pi 1675 Oct 28 13:02 private.pem.key
-rw-r--r-- 1 pi pi  451 Oct 28 13:02 public.pem.key
```

4. Introduzca estos comandos para crear los directorios de los archivos de registro.

```
mkdir ~/.aws-iot-device-client
mkdir ~/.aws-iot-device-client/log
chmod 745 ~/.aws-iot-device-client/log
echo " " > ~/.aws-iot-device-client/log/aws-iot-device-client.log
echo " " > ~/.aws-iot-device-client/log/pubsub_rx_msgs.log
chmod 600 ~/.aws-iot-device-client/log/*
```

Aprovisionar su dispositivo para demostrar la comunicación MQTT

Esta sección crea los AWS IoT recursos que aprovisionan tu Raspberry Pi. AWS IoT

Para aprovisionar el dispositivo en AWS IoT:

1. En la ventana del terminal de su ordenador host local, introduzca el siguiente comando para obtener la dirección del punto de conexión de datos de dispositivo correspondiente a su Cuenta de AWS.

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

   El valor del punto de conexión no ha cambiado desde el momento en que ejecutó este comando para el tutorial anterior. Al volver a ejecutar el comando aquí, será más fácil buscar y pegar el valor del punto de conexión de datos en el archivo de configuración utilizado en este tutorial.

   El comando de los pasos anteriores devuelve una respuesta similar a la siguiente. Anote el valor *endpointAddress*: lo necesitará más adelante.

   ```
   {
   "endpointAddress": "a3qjEXAMPLEffp-ats.iot.us-west-2.amazonaws.com"
   }
   ```

2. Ingresa este comando para crear un nuevo recurso AWS IoT para tu Raspberry Pi.

   ```
   aws iot create-thing --thing-name "PubSubTestThing"
   ```

   Como un AWS IoT recurso es una representación virtual de su dispositivo en la nube, podemos crear varios recursos AWS IoT para usarlos con diferentes propósitos. Todos pueden ser utilizados por el mismo dispositivo físico IoT para representar diferentes aspectos del dispositivo.

   Estos tutoriales solo utilizarán un recurso a la vez para representar la Raspberry Pi. De esta forma, en estos tutoriales, se representan las diferentes demostraciones para que, después de crear los AWS IoT recursos para una demostración, pueda volver atrás y repetir la demostración utilizando los recursos que ha creado específicamente para cada una de ellas.

   Si AWS IoT se ha creado el recurso deseado, el comando devuelve una respuesta como esta.

   ```
   {
   "thingName": "PubSubTestThing",
   "thingArn": "arn:aws:iot:us-west-2:57EXAMPLE833:thing/PubSubTestThing",
   "thingId": "8ea78707-32c3-4f8a-9232-14bEXAMPLEfd"
   }
   ```

3. En la ventana del terminal:

   a. Abra un editor de texto, como `nano`.

   b. Copie este documento JSON y péguelo en el editor de texto abierto.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-west-2:123456789012:client/PubSubTestThing"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-west-2:123456789012:topic/test/dc/pubtopic"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-west-2:123456789012:topicfilter/test/dc/
subtopic"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
```

```
                "arn:aws:iot:us-west-2:123456789012:topic/test/dc/subtopic"
            ]
        }
    ]
}
```

c. En el editor, en cada `Resource` sección del documento de política, *us-west-2:57EXAMPLE833* sustitúyalo por el tuyo Región de AWS, dos puntos (:)) y tu Cuenta de AWS número de 12 dígitos.

d. Guarde el archivo en el editor de texto como **~/policies/pubsub_test_thing_policy.json**.

4. Ejecute este comando para usar el documento de política de los pasos anteriores para crear una AWS IoT política.

```
aws iot create-policy \
--policy-name "PubSubTestThingPolicy" \
--policy-document "file://~/policies/pubsub_test_thing_policy.json"
```

Si se crea la política, el comando devuelve una respuesta como esta.

```
{
                                    "policyName": "PubSubTestThingPolicy",
                                    "policyArn": "arn:aws:iot:us-
west-2:57EXAMPLE833:policy/PubSubTestThingPolicy",
                                    "policyDocument": "{\n\"Version\":
 \"2012-10-17\",       \n\"Statement\": [\n{\n\"Effect\": \"Allow\",\n
\"Action\": [\n\"iot:Connect\"\n],\n\"Resource\": [\n\"arn:aws:iot:us-
west-2:57EXAMPLE833:client/PubSubTestThing\"\n]\n},\n{\n\"Effect\": \"Allow
\",\n\"Action\": [\n\"iot:Publish\"\n],\n\"Resource\": [\n\"arn:aws:iot:us-
west-2:57EXAMPLE833:topic/test/dc/pubtopic\"\n]\n},\n{\n\"Effect\": \"Allow
\",\n\"Action\": [\n\"iot:Subscribe\"\n],\n\"Resource\": [\n\"arn:aws:iot:us-
west-2:57EXAMPLE833:topicfilter/test/dc/subtopic\"\n]\n},\n{\n\"Effect\": \"Allow
\",\n\"Action\": [\n\"iot:Receive\"\n],\n\"Resource\": [\n\"arn:aws:iot:us-
west-2:57EXAMPLE833:topic/test/dc/*\"\n]\n}\n]\n}\n",
                                    "policyVersionId": "1"
                                    }
```

5. Ejecute este comando para asociar la política al certificado del dispositivo. Reemplace *certificateArn* por el valor `certificateArn` que guardó anteriormente en esta sección.

```
aws iot attach-policy \
```

```
--policy-name "PubSubTestThingPolicy" \
--target "certificateArn"
```

Si se ejecuta correctamente, este comando no devuelve nada.

6. Ejecute este comando para asociar el certificado del dispositivo al recurso de objeto de AWS IoT . Reemplace *certificateArn* por el valor `certificateArn` que guardó anteriormente en esta sección.

```
aws iot attach-thing-principal \
--thing-name "PubSubTestThing" \
--principal "certificateArn"
```

Si se ejecuta correctamente, este comando no devuelve nada.

Una vez que hayas aprovisionado correctamente el dispositivo AWS IoT, estarás listo para continuar con élthe section called “Configuración del archivo de configuración de Device Client y el cliente MQTT”.

Configure el archivo de configuración del cliente del AWS IoT dispositivo y el cliente de prueba de MQTT para demostrar la comunicación con MQTT

Este procedimiento crea un archivo de configuración para probar el AWS IoT Device Client.

Para crear el archivo de configuración para probar el AWS IoT Device Client

1. En la ventana del terminal de su ordenador host local que esté conectado a la Raspberry Pi:
   a. Abra un editor de texto, como `nano`.
   b. Copie este documento JSON y péguelo en el editor de texto abierto.

```
{
  "endpoint": "a3qEXAMPLEaffp-ats.iot.us-west-2.amazonaws.com",
  "cert": "~/certs/pubsub/device.pem.crt",
  "key": "~/certs/pubsub/private.pem.key",
  "root-ca": "~/certs/AmazonRootCA1.pem",
  "thing-name": "PubSubTestThing",
  "logging": {
    "enable-sdk-logging": true,
    "level": "DEBUG",
    "type": "STDOUT",
```

```
    "file": ""
  },
  "jobs": {
    "enabled": false,
    "handler-directory": ""
  },
  "tunneling": {
    "enabled": false
  },
  "device-defender": {
    "enabled": false,
    "interval": 300
  },
  "fleet-provisioning": {
    "enabled": false,
    "template-name": "",
    "template-parameters": "",
    "csr-file": "",
    "device-key": ""
  },
  "samples": {
    "pub-sub": {
      "enabled": true,
      "publish-topic": "test/dc/pubtopic",
      "publish-file": "",
      "subscribe-topic": "test/dc/subtopic",
      "subscribe-file": "~/.aws-iot-device-client/log/pubsub_rx_msgs.log"
    }
  },
  "config-shadow": {
    "enabled": false
  },
  "sample-shadow": {
    "enabled": false,
    "shadow-name": "",
    "shadow-input-file": "",
    "shadow-output-file": ""
  }
}
```

c. Sustituya el *`endpoint`* valor por el punto final de datos del dispositivo Cuenta de AWS que encontró enthe section called “Aprovisione su dispositivo en AWS IoT Core”.

d. Guarde el archivo en el editor de texto como **`~/dc-configs/dc-pubsub-config.json`**.

e. Ejecute este comando para establecer los permisos en el nuevo archivo de configuración.

```
chmod 644 ~/dc-configs/dc-pubsub-config.json
```

2. Para preparar el cliente de prueba de MQTT para que se suscriba a todos los mensajes de MQTT:

   a. En el ordenador host local, en la consola de AWS IoT, elija Cliente de prueba de MQTT.
   b. En la pestaña Suscribirse a un tema, en Filtro de temas, introduzca **#** (un solo signo de almohadilla) y elija Suscribirse.
   c. Debajo de la etiqueta Suscripciones, confirme que aparece **#** (un solo signo de almohadilla).

   Deje abierta la ventana con el cliente de prueba de MQTT mientras continúa con este tutorial.

Después de guardar el archivo y configurar el cliente de prueba de MQTT, estará listo para continuar con the section called “Publicación de mensajes con IoT Device Client”.

Demuestre la publicación de mensajes con el cliente AWS IoT del dispositivo

Los procedimientos de esta sección muestran cómo el cliente del AWS IoT dispositivo puede enviar mensajes MQTT predeterminados y personalizados.

Estas declaraciones en la política que ha creado en el paso anterior para estos ejercicios dan permiso a la Raspberry Pi para realizar estas acciones:

- **`iot:Connect`**

  Proporciona el nombre del clientePubSubTestThing, su Raspberry Pi que ejecuta el AWS IoT Device Client, para conectarse.

```
{
  "Effect": "Allow",
  "Action": [
    "iot:Connect"
  ],
  "Resource": [
    "arn:aws:iot:us-west-2:57EXAMPLE833:client/PubSubTestThing"
  ]
}
```

- **`iot:Publish`**

  Dar permiso a la Raspberry Pi para publicar mensajes con un tema de MQTT de `test/dc/pubtopic`.

  ```
  {
    "Effect": "Allow",
    "Action": [
      "iot:Publish"
    ],
    "Resource": [
      "arn:aws:iot:us-west-2:57EXAMPLE833:topic/test/dc/pubtopic"
    ]
  }
  ```

  La acción `iot:Publish` da permiso para publicar en los temas de MQTT que figuran en la matriz de recursos. El contenido de esos mensajes no está controlado por la declaración de política.

Publica el mensaje predeterminado mediante el AWS IoT Device Client

Este procedimiento ejecuta el cliente del AWS IoT dispositivo para publicar un único mensaje MQTT predeterminado que el cliente de prueba MQTT recibe y muestra.

Para enviar el mensaje MQTT predeterminado desde el cliente del dispositivo AWS IoT

1. Asegúrese de que tanto la ventana del terminal de su ordenador host local conectada a su Raspberry Pi como la ventana con el cliente de prueba de MQTT estén visibles mientras realiza este procedimiento.
2. En la ventana del terminal, introduzca estos comandos para ejecutar el AWS IoT Device Client mediante el archivo de configuración creado enthe section called “Crear el archivo de configuración”.

   ```
   cd ~/aws-iot-device-client/build
   ./aws-iot-device-client --config-file ~/dc-configs/dc-pubsub-config.json
   ```

   En la ventana del terminal, el cliente del AWS IoT dispositivo muestra los mensajes de información y cualquier error que se produzca durante su ejecución.

   Si no se muestra ningún error en la ventana del terminal, revise el cliente de prueba de MQTT.

3. En la opción Cliente de prueba de MQTT, en la ventana Suscripciones, consulte el mensaje Hello World! enviado al tema del mensaje `test/dc/pubtopic`.
4. Si el cliente del AWS IoT dispositivo no muestra ningún error y aparece Hello World! enviado al `test/dc/pubtopic` mensaje en el cliente de prueba de MQTT, ha demostrado que la conexión se ha realizado correctamente.
5. En la ventana del terminal, introduzca **`^C`** (Ctrl-C) para detener el cliente del AWS IoT dispositivo.

Una vez que haya demostrado que el cliente del AWS IoT dispositivo publicó el mensaje MQTT predeterminado, puede continuar con el. the section called “Publicación de un mensaje MQTT personalizado”

Publique un mensaje personalizado mediante el AWS IoT Device Client

Los procedimientos de esta sección crean un mensaje de MQTT personalizado y, a continuación, ejecutan AWS IoT Device Client para publicar el mensaje de MQTT personalizado una vez para que el cliente de prueba de MQTT lo reciba y lo muestre.

Cree un mensaje MQTT personalizado para el cliente del AWS IoT dispositivo

Realice estos pasos en la ventana del terminal del ordenador host local que está conectado a su Raspberry Pi.

Para crear un mensaje personalizado para que el cliente del AWS IoT dispositivo lo publique

1. En la ventana del terminal, abra un editor de texto, por ejemplo, `nano`.
2. En el editor de texto, copie y pegue el siguiente documento JSON. Esta será la carga útil de mensajes MQTT que publique el cliente del AWS IoT dispositivo.

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

3. Guarde el contenido del editor de texto como **`~/messages/sample-ws-message.json`**.

4. Introduzca el siguiente comando para configurar los permisos del archivo de mensaje que acaba de crear.

```
chmod 600 ~/messages/*
```

Crear un archivo de configuración para que el cliente del AWS IoT dispositivo lo utilice para enviar el mensaje personalizado

1. En la ventana del terminal, en un editor de texto como`nano`, por ejemplo, abra el archivo de configuración de AWS IoT Device Client existente:**`~/dc-configs/dc-pubsub-config.json`**.
2. Edita el objeto de `samples` para que tenga este aspecto. No es necesario cambiar ninguna otra parte de este archivo.

```
  "samples": {
    "pub-sub": {
      "enabled": true,
      "publish-topic": "test/dc/pubtopic",
      "publish-file": "~/messages/sample-ws-message.json",
      "subscribe-topic": "test/dc/subtopic",
      "subscribe-file": "~/.aws-iot-device-client/log/pubsub_rx_msgs.log"
```

3. Guarde el contenido del editor de texto como **`~/dc-configs/dc-pubsub-custom-config.json`**.
4. Ejecute este comando para establecer los permisos en el nuevo archivo de configuración.

```
chmod 644 ~/dc-configs/dc-pubsub-custom-config.json
```

Publique el mensaje MQTT personalizado mediante el AWS IoT Device Client

Este cambio afecta únicamente al contenido de la carga del mensaje de MQTT, por lo que la política actual seguirá funcionando. Sin embargo, si se cambiara el tema de MQTT (tal como lo define el valor `publish-topic` en `~/dc-configs/dc-pubsub-custom-config.json`), también habría que modificar la declaración de política de `iot::Publish` para permitir que la Raspberry Pi publique en el nuevo tema de MQTT.

Para enviar el mensaje MQTT desde el cliente del dispositivo AWS IoT

1. Asegúrese de que tanto la ventana del terminal como la ventana con el cliente de prueba de MQTT estén visibles mientras realiza este procedimiento. Además, verifique que su cliente de prueba de MQTT siga suscrito al filtro de temas #. Si no lo está, vuelva a suscribirse al filtro de temas #.
2. En la ventana del terminal, introduzca estos comandos para ejecutar AWS IoT Device Client mediante el archivo de configuración creado en the section called “Crear el archivo de configuración”.

   ```
   cd ~/aws-iot-device-client/build
   ./aws-iot-device-client --config-file ~/dc-configs/dc-pubsub-custom-config.json
   ```

   En la ventana del terminal, el cliente del AWS IoT dispositivo muestra los mensajes de información y cualquier error que se produzca durante su ejecución.

   Si no se muestra ningún error en la ventana del terminal, revise el cliente de prueba de MQTT.
3. En el cliente de prueba de MQTT, en la ventana Suscripciones, observe la carga del mensaje personalizado enviado al tema del mensaje `test/dc/pubtopic`.
4. Si el cliente del AWS IoT dispositivo no muestra ningún error y usted ve la carga útil del mensaje personalizado que publicó en el `test/dc/pubtopic` mensaje en el cliente de prueba de MQTT, el mensaje personalizado se ha publicado correctamente.
5. En la ventana del terminal, introduzca **^C** (Ctrl-C) para detener el cliente del dispositivo. AWS IoT

Una vez que hayas demostrado que el cliente del AWS IoT dispositivo publicó una carga de mensajes personalizada, puedes continuar haciéndolo. the section called “Suscripción a mensajes con IoT Device Client”

Demuestre la suscripción a los mensajes con el cliente del dispositivo AWS IoT

En esta sección, mostrará dos tipos de suscripciones de mensajes:

- Suscripción a un solo tema
- Suscripción a un tema comodín

Estas declaraciones en la política creadas para estos ejercicios dan permiso a la Raspberry Pi para realizar estas acciones:

- **`iot:Receive`**

  Otorga al cliente del AWS IoT dispositivo permiso para recibir temas de MQTT que coincidan con los nombrados en el `Resource` objeto.

  ```
  {
    "Effect": "Allow",
    "Action": [
      "iot:Receive"
    ],
    "Resource": [
      "arn:aws:iot:us-west-2:57EXAMPLE833:topic/test/dc/subtopic"
    ]
  }
  ```

- **`iot:Subscribe`**

  Otorga al cliente del AWS IoT dispositivo permiso para suscribirse a los filtros de temas de MQTT que coincidan con los nombres del `Resource` objeto.

  ```
  {
    "Effect": "Allow",
    "Action": [
      "iot:Subscribe"
    ],
    "Resource": [
      "arn:aws:iot:us-west-2:57EXAMPLE833:topicfilter/test/dc/subtopic"
    ]
  }
  ```

Suscripción a un solo tema de mensaje de MQTT

Este procedimiento demuestra cómo el cliente del AWS IoT dispositivo puede suscribirse a los mensajes MQTT y registrarlos.

En la ventana del terminal de un ordenador host local que esté conectado a su Raspberry Pi, enumere el contenido de **`~/dc-configs/dc-pubsub-custom-config.json`** o abra el archivo en un editor de texto para revisarlo. Localice el objeto `samples`. Debería tener este aspecto.

```
  "samples": {
    "pub-sub": {
      "enabled": true,
      "publish-topic": "test/dc/pubtopic",
      "publish-file": "~/messages/sample-ws-message.json",
      "subscribe-topic": "test/dc/subtopic",
      "subscribe-file": "~/.aws-iot-device-client/log/pubsub_rx_msgs.log"
```

Observe que el valor `subscribe-topic` es el tema de MQTT al que AWS IoT Device Client se suscribirá cuando se ejecute. El cliente del AWS IoT dispositivo escribe las cargas útiles de mensajes que recibe de esta suscripción en el archivo mencionado en el `subscribe-file` valor.

Para suscribirse a un tema de mensajes MQTT desde el AWS IoT Device Client

1. Asegúrese de que tanto la ventana del terminal como la ventana con el cliente de prueba de MQTT estén visibles mientras realiza este procedimiento. Además, verifique que su cliente de prueba de MQTT siga suscrito al filtro de temas #. Si no lo está, vuelva a suscribirse al filtro de temas #.
2. En la ventana del terminal, introduzca estos comandos para ejecutar el AWS IoT Device Client mediante el archivo de configuración creado enthe section called “Crear el archivo de configuración”.

   ```
   cd ~/aws-iot-device-client/build
   ./aws-iot-device-client --config-file ~/dc-configs/dc-pubsub-custom-config.json
   ```

   En la ventana del terminal, el cliente del AWS IoT dispositivo muestra los mensajes de información y cualquier error que se produzca durante su ejecución.

   Si no se muestra ningún error en la ventana del terminal, continúe en la consola de AWS IoT .
3. En la AWS IoT consola, en el cliente de pruebas de MQTT, seleccione la pestaña Publicar en un tema.
4. En Nombre del tema, introduzca **`test/dc/subtopic`**
5. En Carga útil del mensaje, revise el contenido del mensaje.
6. Elija Publicar para publicar el mensaje de MQTT.
7. En la ventana del terminal, busque la entrada de mensaje recibida del cliente del AWS IoT dispositivo que tiene este aspecto.

```
2021-11-10T16:02:20.890Z [DEBUG] {samples/PubSubFeature.cpp}: Message received on
 subscribe topic, size: 45 bytes
```

8. Cuando veas la entrada del mensaje recibido que indica que se ha recibido el mensaje, ingresa **^C** (Ctrl-C) para detener el cliente del AWS IoT dispositivo.
9. Introduzca este comando para ver el final del archivo de registro de mensajes y ver el mensaje que publicó desde el cliente de prueba de MQTT.

```
tail ~/.aws-iot-device-client/log/pubsub_rx_msgs.log
```

Al ver el mensaje en el archivo de registro, ha demostrado que AWS IoT Device Client recibió el mensaje que usted publicó desde el cliente de prueba de MQTT.

Suscribirse a varios temas de mensajes de MQTT utilizando caracteres comodín

Estos procedimientos demuestran cómo el cliente del AWS IoT dispositivo puede suscribirse a los mensajes MQTT y registrarlos mediante caracteres comodín. Para ello, deberá:

1. Actualice el filtro de temas que el cliente del AWS IoT dispositivo utiliza para suscribirse a los temas de MQTT.
2. Actualizar la política utilizada por el dispositivo para permitir las nuevas suscripciones.
3. Ejecute el cliente del AWS IoT dispositivo y publique los mensajes desde la consola de pruebas de MQTT.

Para crear un archivo de configuración y suscribirse a varios temas de mensajes de MQTT mediante un filtro de temas de MQTT con caracteres comodín

1. En la ventana del terminal de su ordenador host local conectado a la Raspberry Pi, abra para **~/dc-configs/dc-pubsub-custom-config.json** para editar y localice el objeto `samples`.
2. En el editor de texto, localice el objeto `samples` y actualice el valor `subscribe-topic` para que tenga este aspecto.

```
  "samples": {
    "pub-sub": {
      "enabled": true,
      "publish-topic": "test/dc/pubtopic",
      "publish-file": "~/messages/sample-ws-message.json",
```

```
    "subscribe-topic": "test/dc/#",
    "subscribe-file": "~/.aws-iot-device-client/log/pubsub_rx_msgs.log"
```

El nuevo valor `subscribe-topic` es un filtro de temas de MQTT con un carácter comodín de MQTT al final. Esto describe una suscripción a todos los temas de MQTT que comiencen con `test/dc/`. El cliente del AWS IoT dispositivo escribe las cargas útiles de mensajes que recibe de esta suscripción en el archivo mencionado. `subscribe-file`

3. Guarde el archivo de configuración modificado como **`~/dc-configs/dc-pubsub-wild-config.json`** y salga del editor de texto.

Para modificar la política utilizada por su Raspberry Pi y permitir la suscripción y la recepción de varios temas de mensajes de MQTT

1. En la ventana del terminal de su ordenador host local que esté conectado a la Raspberry Pi, en el editor de texto que prefiera, abra **`~/policies/pubsub_test_thing_policy.json`** para su edición y luego busque las declaraciones de política de `iot::Subscribe` y `iot::Receive` en el archivo.
2. En la declaración de política `iot::Subscribe`, actualice la cadena del objeto Resource para reemplazar `subtopic` por `*`, de forma que tenga el siguiente aspecto.

```
{
  "Effect": "Allow",
  "Action": [
    "iot:Subscribe"
  ],
  "Resource": [
    "arn:aws:iot:us-west-2:57EXAMPLE833:topicfilter/test/dc/*"
  ]
}
```

> **Note**
>
> Los caracteres comodín del filtro de temas de MQTT son + (signo más) y # (signo de almohadilla). Una solicitud de suscripción con un símbolo # al final suscribe todos los temas que comiencen por la cadena que precede al carácter # (por ejemplo, `test/dc/` en este caso).
> Sin embargo, el valor del recurso de la declaración de política que autoriza esta suscripción debe usar * (un asterisco) en lugar de # (signo de almohadilla) en el ARN

del filtro de temas. Esto se debe a que el procesador de políticas utiliza un carácter comodín diferente al que utiliza MQTT.
Para obtener más información sobre el uso de caracteres comodín en los temas y los filtros de temas en las políticas, consulte Uso de caracteres comodín en MQTT y en las políticas AWS IoT Core.

3. En la declaración de política `iot::Receive`, actualice la cadena del objeto Resource para reemplazar `subtopic` por `*`, de forma que tenga el siguiente aspecto.

```
    {
      "Effect": "Allow",
      "Action": [
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:us-west-2:57EXAMPLE833:topic/test/dc/*"
      ]
    }
```

4. Guarde el documento de política actualizado como **`~/policies/pubsub_wild_test_thing_policy.json`** y salga del editor.

5. Introduzca este comando para actualizar la política de este tutorial con el fin de utilizar las nuevas definiciones de recursos.

```
aws iot create-policy-version \
--set-as-default \
--policy-name "PubSubTestThingPolicy" \
--policy-document "file://~/policies/pubsub_wild_test_thing_policy.json"
```

Si el comando se ejecuta correctamente, devuelve una respuesta como esta. Observe que `policyVersionId` ahora es 2, lo que indica que esta es la segunda versión de esta política.

Si ha actualizado correctamente la política, puede continuar con el siguiente procedimiento.

```
{
    "policyArn": "arn:aws:iot:us-west-2:57EXAMPLE833:policy/PubSubTestThingPolicy",
    "policyDocument": "{\n  \"Version\": \"2012-10-17\",       \n  \"Statement
\": [\n    {\n      \"Effect\": \"Allow\",\n      \"Action\": [\n
 \"iot:Connect\"\n      ],\n      \"Resource\": [\n        \"arn:aws:iot:us-
west-2:57EXAMPLE833:client/PubSubTestThing\"\n      ]\n    },\n    {\n
```

```
 \"Effect\": \"Allow\",\n      \"Action\": [\n        \"iot:Publish\"\n      ],
\n      \"Resource\": [\n        \"arn:aws:iot:us-west-2:57EXAMPLE833:topic/
test/dc/pubtopic\"\n      ]\n    },\n    {\n      \"Effect\": \"Allow\",\n
 \"Action\": [\n        \"iot:Subscribe\"\n      ],\n      \"Resource\": [\n
  \"arn:aws:iot:us-west-2:57EXAMPLE833:topicfilter/test/dc/*\"\n      ]\n    },\n
   {\n      \"Effect\": \"Allow\",\n      \"Action\": [\n        \"iot:Receive\"\n
     ],\n      \"Resource\": [\n        \"arn:aws:iot:us-west-2:57EXAMPLE833:topic/
test/dc/*\"\n      ]\n    }\n  ]\n}\n",
    "policyVersionId": "2",
    "isDefaultVersion": true
}
```

Si aparece un error que indica que hay demasiadas versiones de la política como para guardar una nueva, introduzca este comando para ver una lista de las versiones actuales de la política. Revise la lista que devuelve este comando para buscar una versión de la política que pueda eliminar.

```
aws iot list-policy-versions --policy-name "PubSubTestThingPolicy"
```

Introduzca este comando para eliminar una versión que ya no necesite. Tenga en cuenta que no puede eliminar la versión de política predeterminada. La versión de política predeterminada es la que tiene un valor `isDefaultVersion` de `true`.

```
aws iot delete-policy-version \
--policy-name "PubSubTestThingPolicy" \
--policy-version-id policyId
```

Tras eliminar una versión de la política, vuelva a intentar este paso.

Con el archivo de configuración y la política actualizados, estará listo para hacer una demostración de las suscripciones estándar con el AWS IoT Device Client.

Para demostrar cómo el cliente del AWS IoT dispositivo se suscribe y recibe varios temas de mensajes de MQTT

1. En el cliente de prueba de MQTT, compruebe las suscripciones. Si el cliente de pruebas de MQTT está suscrito al filtro incluido en el tema **#**, continúe con el siguiente paso. De lo contrario, en la opción Cliente de prueba de MQTT, en la pestaña Suscribirse a un tema, en Filtro de

temas, introduzca **#** (un carácter de almohadilla) y, a continuación, seleccione Suscribirse para suscribirse.

2. En la ventana del terminal de su ordenador host local conectado a su Raspberry Pi, introduzca estos comandos para iniciar AWS IoT Device Client.

```
cd ~/aws-iot-device-client/build
./aws-iot-device-client --config-file ~/dc-configs/dc-pubsub-wild-config.json
```

3. Mientras observa el resultado del AWS IoT Device Client en la ventana del terminal del ordenador host local, vuelva al cliente de prueba de MQTT. En la pestaña Publicar en un tema, en Nombre del tema, introduzca **`test/dc/subtopic`** y, a continuación, seleccione Publicar.

4. En la ventana del terminal, confirme que el mensaje se ha recibido buscando un mensaje como:

```
2021-11-10T16:34:20.101Z [DEBUG] {samples/PubSubFeature.cpp}: Message received on
 subscribe topic, size: 76 bytes
```

5. Mientras observa el resultado del AWS IoT Device Client en la ventana del terminal del ordenador host local, vuelva al cliente de prueba de MQTT. En la pestaña Publicar en un tema, en Nombre del tema, introduzca **`test/dc/subtopic2`** y, a continuación, seleccione Publicar.

6. En la ventana del terminal, confirme que el mensaje se ha recibido buscando un mensaje como:

```
2021-11-10T16:34:32.078Z [DEBUG] {samples/PubSubFeature.cpp}: Message received on
 subscribe topic, size: 77 bytes
```

7. Cuando vea los mensajes que confirman la recepción de ambos mensajes, introduzca **`^C`** (Ctrl-C) para detener el AWS IoT Device Client.

8. Introduzca este comando para ver el final del archivo de registro de mensajes y ver el mensaje que publicó desde el cliente de prueba de MQTT.

```
tail -n 20 ~/.aws-iot-device-client/log/pubsub_rx_msgs.log
```

> **Note**
>
> El archivo de registro contiene solo cargas de mensajes. Los temas de los mensajes no se registran en el archivo de registro de mensajes recibidos.
> Es posible que también vea el mensaje publicado por el cliente del AWS IoT dispositivo en el registro recibido. Esto se debe a que el filtro de temas comodín incluye ese tema

del mensaje y, en ocasiones, el agente de mensajes puede procesar la solicitud de suscripción antes de enviar el mensaje publicado a los suscriptores.

Las entradas del archivo de registro demuestran que se recibieron los mensajes. Puede repetir este procedimiento utilizando otros nombres de temas. Se deben recibir y registrar todos los mensajes que tengan un nombre de tema que comience por `test/dc/`. Se ignoran los mensajes con nombres de temas que comiencen por cualquier otro texto.

Tras demostrar cómo el cliente del AWS IoT dispositivo puede publicar y suscribirse a los mensajes MQTT, continúe Tutorial: Demuestre acciones remotas (trabajos) con el cliente del AWS IoT dispositivo haciéndolo.

Tutorial: Demuestre acciones remotas (trabajos) con el cliente del AWS IoT dispositivo

En estos tutoriales, configurará e implementará trabajos en su Raspberry Pi para demostrar cómo puede enviar operaciones remotas a sus dispositivos IoT.

Para comenzar este tutorial:

- Configure su ordenador host local, una Raspberry Pi, tal como se utilizó en la sección anterior.
- Si no has completado el tutorial de la sección anterior, puedes probar este tutorial utilizando la Raspberry Pi con una tarjeta microSD que contenga la imagen que guardaste después de instalar el AWS IoT Device Client. (Opcional) Guardar la imagen de la tarjeta microSD
- Si ya ha realizado esta demostración anteriormente, procure ??? eliminar todos los AWS IoT recursos que haya creado en ejecuciones anteriores para evitar errores de recursos duplicados.

Para completar este tutorial se necesitan aproximadamente 45 minutos.

Cuando haya terminado con este tema:

- Habrás demostrado las diferentes formas en que tu dispositivo de IoT puede AWS IoT Core utilizarlos para ejecutar operaciones remotas gestionadas por AWS IoT .

Equipo necesario:

- Su entorno local de desarrollo y pruebas que probó en una sección anterior

- La Raspberry Pi que probó en una sección anterior
- La tarjeta de memoria microSD de la Raspberry Pi que probó en una sección anterior

Procedimientos de este tutorial

- Preparación de Raspberry Pi para ejecutar trabajos
- Crea y ejecuta el trabajo AWS IoT con AWS IoT Device Client

Preparación de Raspberry Pi para ejecutar trabajos

Los procedimientos de esta sección describen cómo preparar la Raspberry Pi para ejecutar tareas mediante el AWS IoT Device Client.

> **Note**
>
> Estos procedimientos son específicos del dispositivo. Si desea realizar los procedimientos de esta sección con más de un dispositivo al mismo tiempo, cada dispositivo necesitará su propia política y un certificado y un nombre únicos y específicos del dispositivo. Para asignar a cada dispositivo sus recursos exclusivos, realice este procedimiento una vez para cada dispositivo y, al mismo tiempo, cambie los elementos específicos del dispositivo, tal como se describe en los procedimientos.

Procedimientos de este tutorial

- Aprovisionar la Raspberry Pi para demostrar trabajos
- Configure el cliente del AWS IoT dispositivo para ejecutar el agente de trabajos

Aprovisionar la Raspberry Pi para demostrar trabajos

Los procedimientos de esta sección aprovisionan tu Raspberry Pi AWS IoT mediante la creación de AWS IoT recursos y certificados de dispositivo para ella.

Temas

- Cree y descargue archivos de certificados de dispositivos para demostrar los AWS IoT trabajos
- Cree AWS IoT recursos para demostrar los AWS IoT trabajos

Cree y descargue archivos de certificados de dispositivos para demostrar los AWS IoT trabajos

Este procedimiento crea los archivos de certificado del dispositivo para esta demostración.

Si está preparando más de un dispositivo, este procedimiento debe realizarse en cada uno de ellos.

Para crear y descargar los archivos de certificado del dispositivo para la Raspberry Pi:

En la ventana del terminal de su ordenador host local que esté conectado a la Raspberry Pi, introduzca estos comandos:

1. Escriba el siguiente comando para crear los archivos de certificado del dispositivo.

   ```
   aws iot create-keys-and-certificate \
   --set-as-active \
   --certificate-pem-outfile "~/certs/jobs/device.pem.crt" \
   --public-key-outfile "~/certs/jobs/public.pem.key" \
   --private-key-outfile "~/certs/jobs/private.pem.key"
   ```

   El comando devuelve una respuesta similar a la siguiente: Guarde el valor de *certificateArn* para utilizarlo más tarde.

   ```
   {
   "certificateArn": "arn:aws:iot:us-
   west-2:57EXAMPLE833:cert/76e7e4edb3e52f52334be2f387a06145b2aa4c7fcd810f3aea2d92abc227d269",
   "certificateId":
    "76e7e4edb3e52f5233EXAMPLE7a06145b2aa4c7fcd810f3aea2d92abc227d269",
   "certificatePem": "-----BEGIN CERTIFICATE-----
   \nMIIDWTCCAkGgAwIBAgI_SHORTENED_FOR_EXAMPLE_Lgn4jfgtS\n-----END CERTIFICATE-----
   \n",
   "keyPair": {
       "PublicKey": "-----BEGIN PUBLIC KEY-----
   \nMIIBIjANBgkqhkiG9w0BA_SHORTENED_FOR_EXAMPLE_ImwIDAQAB\n-----END PUBLIC KEY-----
   \n",
       "PrivateKey": "-----BEGIN RSA PRIVATE KEY-----
   \nMIIEowIBAAKCAQE_SHORTENED_FOR_EXAMPLE_T9RoDiukY\n-----END RSA PRIVATE KEY-----\n"
   }
   }
   ```

2. Introduzca los siguientes comandos para configurar los permisos en el directorio de certificados y sus archivos.

```
chmod 700 ~/certs/jobs
chmod 644 ~/certs/jobs/*
chmod 600 ~/certs/jobs/private.pem.key
```

3. Ejecute este comando para revisar los permisos de sus directorios y archivos de certificados.

   ```
   ls -l ~/certs/jobs
   ```

   El resultado del comando debe ser el mismo que el que se muestra aquí, excepto que las fechas y horas de los archivos serán diferentes.

   ```
   -rw-r--r-- 1 pi pi 1220 Oct 28 13:02 device.pem.crt
   -rw------- 1 pi pi 1675 Oct 28 13:02 private.pem.key
   -rw-r--r-- 1 pi pi  451 Oct 28 13:02 public.pem.key
   ```

Una vez descargados los archivos de certificado del dispositivo en la Raspberry Pi, estará listo para continuar con the section called “Aprovisionamiento de Raspberry Pi para los trabajos”.

Cree AWS IoT recursos para demostrar los AWS IoT trabajos

Cree los AWS IoT recursos para este dispositivo.

Si está preparando más de un dispositivo, este procedimiento debe realizarse en cada uno de ellos.

Para aprovisionar el dispositivo en AWS IoT:

En la ventana del terminal de su ordenador host local que esté conectado a la Raspberry Pi:

1. Introduzca el siguiente comando para obtener la dirección del punto de conexión de datos del dispositivo de su Cuenta de AWS.

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

   El valor del punto de conexión no ha cambiado desde la última vez que ejecutó este comando. Al volver a ejecutar el comando aquí, será más fácil buscar y pegar el valor del punto de conexión de datos en el archivo de configuración utilizado en este tutorial.

El comando describe-endpoint devuelve una respuesta similar a la siguiente. Anote el valor *endpointAddress*: lo necesitará más adelante.

```
{
"endpointAddress": "a3qjEXAMPLEffp-ats.iot.us-west-2.amazonaws.com"
}
```

2. *uniqueThingName*Sustitúyalos por un nombre único para el dispositivo. Si desea realizar este tutorial con varios dispositivos, asigne a cada dispositivo su propio nombre. Por ejemplo, **TestDevice01**, **TestDevice02**, etc.

   Ingresa este comando para crear un nuevo recurso AWS IoT para tu Raspberry Pi.

   ```
   aws iot create-thing --thing-name "uniqueThingName"
   ```

   Como un AWS IoT recurso es una representación virtual de su dispositivo en la nube, podemos crear varios recursos AWS IoT para usarlos con diferentes propósitos. Todos pueden ser utilizados por el mismo dispositivo físico IoT para representar diferentes aspectos del dispositivo.

   > **Note**
   >
   > Cuando desee proteger la política para varios dispositivos, puede utilizar ${iot:Thing.*ThingName*} en lugar del nombre del objeto estático, *uniqueThingName*.

   Estos tutoriales solo utilizarán un recurso a la vez por dispositivo. De esta forma, en estos tutoriales, se representan las diferentes demostraciones para que, después de crear los AWS IoT recursos para una demostración, pueda volver atrás y repetir las demostraciones utilizando los recursos que ha creado específicamente para cada una de ellas.

   Si AWS IoT se ha creado el recurso deseado, el comando devuelve una respuesta como esta. Registre el valor *thingArn* para usarlo más adelante cuando cree el trabajo para ejecutarlo en este dispositivo.

   ```
   {
   "thingName": "uniqueThingName",
   "thingArn": "arn:aws:iot:us-west-2:57EXAMPLE833:thing/uniqueThingName",
   ```

```
"thingId": "8ea78707-32c3-4f8a-9232-14bEXAMPLEfd"
}
```

3. En la ventana del terminal:

   a. Abra un editor de texto, como `nano`.

   b. Copie este documento JSON y péguelo en el editor de texto abierto.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-west-2:123456789012:client/uniqueThingName"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-west-2:123456789012:topic/test/dc/pubtopic",
                "arn:aws:iot:us-west-2:123456789012:topic/$aws/events/job/
*",
                "arn:aws:iot:us-west-2:123456789012:topic/$aws/events/
jobExecution/*",
                "arn:aws:iot:us-west-2:123456789012:topic/$aws/
things/uniqueThingName/jobs/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
```

```
                "arn:aws:iot:us-west-2:123456789012:topicfilter/test/dc/
subtopic",
                "arn:aws:iot:us-west-2:123456789012:topic/$aws/events/
jobExecution/*",
                "arn:aws:iot:us-west-2:123456789012:topicfilter/$aws/
things/uniqueThingName/jobs/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:us-west-2:123456789012:topic/test/dc/subtopic",
                "arn:aws:iot:us-west-2:123456789012:topic/$aws/
things/uniqueThingName/jobs/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:DescribeJobExecution",
                "iot:GetPendingJobExecutions",
                "iot:StartNextPendingJobExecution",
                "iot:UpdateJobExecution"
            ],
            "Resource": [
                "arn:aws:iot:us-west-2:123456789012:topic/$aws/
things/uniqueThingName"
            ]
        }
    ]
}
```

c. En el editor, en la `Resource` sección de cada declaración de política, *us-west-2:57EXAMPLE833* sustitúyalo por el tuyo Región de AWS, dos puntos (:)) y tu Cuenta de AWS número de 12 dígitos.

d. En el editor, en cada declaración de política, *uniqueThingName* sustitúyala por el nombre que le diste al recurso.

e. Guarde el archivo en el editor de texto como **~/policies/jobs_test_thing_policy.json**.

Si ejecuta este procedimiento para varios dispositivos, guarde el archivo con este nombre de archivo en cada dispositivo.

4. *uniqueThingName*Sustitúyalo por el nombre del dispositivo y, a continuación, ejecuta este comando para crear una AWS IoT política que se adapte a ese dispositivo.

```
aws iot create-policy \
--policy-name "JobTestPolicyFor uniqueThingName" \
--policy-document "file://~/policies/jobs_test_thing_policy.json"
```

Si se crea la política, el comando devuelve una respuesta como esta.

```
{
    "policyName": "JobTestPolicyFor uniqueThingName",
    "policyArn": "arn:aws:iot:us-west-2:57EXAMPLE833:policy/
JobTestPolicyFor uniqueThingName",
    "policyDocument": "{\n\"Version\": \"2012-10-17\",\n\"Statement\": [\n{\n
\"Effect\": \"Allow\",\n\"Action\": [\n\"iot:Connect\"\n],\n\"Resource\":
 [\n\"arn:aws:iot:us-west-2:57EXAMPLE833:client/PubSubTestThing\"\n]\n},\n{\n
\"Effect\": \"Allow\",\n\"Action\": [\n\"iot:Publish\"\n],\n\"Resource\":
 [\n\"arn:aws:iot:us-west-2:57EXAMPLE833:topic/test/dc/pubtopic\"\n]\n},\n{\n
\"Effect\": \"Allow\",\n\"Action\": [\n\"iot:Subscribe\"\n],\n\"Resource\": [\n
\"arn:aws:iot:us-west-2:57EXAMPLE833:topicfilter/test/dc/subtopic\"\n]\n},\n{\n
\"Effect\": \"Allow\",\n\"Action\": [\n\"iot:Receive\"\n],\n\"Resource\": [\n
\"arn:aws:iot:us-west-2:57EXAMPLE833:topic/test/dc/*\"\n]\n}\n]\n}\n",
    "policyVersionId": "1"
}
```

5. *uniqueThingName*Sustitúyalo por el nombre del dispositivo y *certificateArn* por el `certificateArn` valor que guardó anteriormente en esta sección para este dispositivo y, a continuación, ejecute este comando para adjuntar la política al certificado del dispositivo.

```
aws iot attach-policy \
--policy-name "JobTestPolicyFor uniqueThingName" \
--target "certificateArn"
```

Si se ejecuta correctamente, este comando no devuelve nada.

6. *uniqueThingName*Sustitúyalo por el nombre del dispositivo, *certificateArn* sustitúyalo por el `certificateArn` valor que guardó anteriormente en esta sección y, a continuación, ejecute este comando para adjuntar el certificado del dispositivo al AWS IoT recurso del elemento.

   ```
   aws iot attach-thing-principal \
   --thing-name "uniqueThingName" \
   --principal "certificateArn"
   ```

   Si se ejecuta correctamente, este comando no devuelve nada.

Una vez aprovisionada correctamente la Raspberry Pi, podrá repetir esta sección con otra Raspberry Pi de la prueba o, si ya se han aprovisionado todos los dispositivos, podrá continuar con the section called “Configuración de Device Client para ejecutar trabajos”.

Configure el cliente del AWS IoT dispositivo para ejecutar el agente de trabajos

Este procedimiento crea un archivo de configuración para que el cliente del AWS IoT dispositivo ejecute el agente de trabajos:.

Nota: Si está preparando más de un dispositivo, este procedimiento debe realizarse en cada uno de ellos.

Para crear el archivo de configuración para probar el cliente del AWS IoT dispositivo:

1. En la ventana del terminal de su ordenador host local que esté conectado a la Raspberry Pi:

   a. Abra un editor de texto, como `nano`.

   b. Copie este documento JSON y péguelo en el editor de texto abierto.

   ```
   {
     "endpoint": "a3qEXAMPLEaffp-ats.iot.us-west-2.amazonaws.com",
     "cert": "~/certs/jobs/device.pem.crt",
     "key": "~/certs/jobs/private.pem.key",
     "root-ca": "~/certs/AmazonRootCA1.pem",
     "thing-name": "uniqueThingName",
     "logging": {
       "enable-sdk-logging": true,
       "level": "DEBUG",
       "type": "STDOUT",
       "file": ""
     },
   ```

```
  "jobs": {
    "enabled": true,
    "handler-directory": ""
  },
  "tunneling": {
    "enabled": false
  },
  "device-defender": {
    "enabled": false,
    "interval": 300
  },
  "fleet-provisioning": {
    "enabled": false,
    "template-name": "",
    "template-parameters": "",
    "csr-file": "",
    "device-key": ""
  },
  "samples": {
    "pub-sub": {
      "enabled": false,
      "publish-topic": "",
      "publish-file": "",
      "subscribe-topic": "",
      "subscribe-file": ""
    }
  },
  "config-shadow": {
    "enabled": false
  },
  "sample-shadow": {
    "enabled": false,
    "shadow-name": "",
    "shadow-input-file": "",
    "shadow-output-file": ""
  }
}
```

c. Sustituya el *endpoint* valor por el valor del punto final de los datos del dispositivo Cuenta de AWS que encontró enthe section called “Aprovisione su dispositivo en AWS IoT Core”.

d. *uniqueThingName*Sustitúyalo por el nombre del objeto que utilizaste para este dispositivo.

e. Guarde el archivo en el editor de texto como **~/dc-configs/dc-jobs-config.json**.

2. Ejecute este comando para establecer los permisos de archivo del nuevo archivo de configuración.

```
chmod 644 ~/dc-configs/dc-jobs-config.json
```

No utilizará el cliente de prueba de MQTT para esta prueba. Si bien el dispositivo intercambiará mensajes MQTT relacionados con las tareas AWS IoT, los mensajes de progreso de las tareas solo se intercambiarán con el dispositivo que las esté ejecutando. Como los mensajes de progreso del trabajo solo se intercambian con el dispositivo que ejecuta el trabajo, no puedes suscribirte a ellos desde otro dispositivo, como la consola. AWS IoT

Después de guardar el archivo de configuración, estará listo para continuar con the section called “Creación y ejecución de un trabajo de IoT”.

Crea y ejecuta el trabajo AWS IoT con AWS IoT Device Client

Los procedimientos de esta sección crean un documento de trabajo y un recurso de AWS IoT trabajo. Tras crear el recurso de trabajo, AWS IoT envía el documento de trabajo a los destinos de trabajo especificados, a los que un agente de trabajos aplica el documento de trabajo al dispositivo o cliente.

Procedimientos de esta sección

- Creación y almacenamiento del documento de trabajo de IoT
- Ejecute un trabajo AWS IoT para un dispositivo de IoT

Creación y almacenamiento del documento de trabajo de IoT

Este procedimiento crea un documento de trabajo sencillo para incluirlo en un recurso de AWS IoT trabajo. Este documento de trabajo muestra el mensaje «¡Hola mundo!» en el destino del trabajo.

Para crear y almacenar un documento de trabajo:

1. Seleccione el bucket de Amazon S3 en el que guardará el documento de trabajo. Si no dispone de un bucket de Amazon S3 existente, deberá crear uno. Para obtener información sobre cómo crear buckets de Amazon S3, consulte los temas de Introducción a Amazon S3.
2. Crear y guardar el documento de trabajo para este trabajo

    a. En el ordenador host local, abra un editor de texto.

b. Copie y pegue este texto en el editor.

```
{
    "operation": "echo",
    "args": ["Hello world!"]
}
```

c. En el equipo host local, guarde el contenido del editor en un archivo denominado **`hello-world-job.json`**.

d. Confirme que el archivo se haya guardado correctamente. Algunos editores de texto agregan `.txt` automáticamente el nombre del archivo al guardar un archivo de texto. Si el editor lo ha agregado `.txt` al nombre del archivo, corrija el nombre del archivo antes de continuar.

3. Sustituya la *`path_to_file`* ruta por la ruta a**`hello-world-job.json`**, si no está en su directorio actual, *`s3_bucket_name`* sustitúyala por la ruta del bucket de Amazon S3 al bucket que ha seleccionado y, a continuación, ejecute este comando para colocar el documento de trabajo en el bucket de Amazon S3.

```
aws s3api put-object \
--key hello-world-job.json \
--body path_to_file/hello-world-job.json --bucket s3_bucket_name
```

La URL del documento de trabajo que identifica el documento de trabajo que ha almacenado en Amazon S3 se determina sustituyendo *`s3_bucket_name`* y *`AWS_region`* en la siguiente URL. Registre la URL resultante para utilizarla más adelante como *`job_document_path`*

```
https://s3_bucket_name.s3.AWS_Region.amazonaws.com/hello-world-job.json
```

> **Note**
>
> AWS la seguridad impide que puedas abrir esta URL fuera de la tuya Cuenta de AWS, por ejemplo, mediante un navegador. El motor de AWS IoT trabajos, que tiene acceso al archivo, utiliza la URL de forma predeterminada. En un entorno de producción, tendrá que asegurarse de que sus servicios de AWS IoT tengan permiso para acceder a los documentos de trabajo almacenados en Amazon S3.

Una vez que haya guardado la URL del documento de trabajo, continúe con the section called "Ejecución del trabajo para un solo dispositivo".

Ejecute un trabajo AWS IoT para un dispositivo de IoT

Los procedimientos de esta sección inician el cliente del AWS IoT dispositivo en su Raspberry Pi para ejecutar el agente de tareas en el dispositivo y esperar a que se ejecuten las tareas. También crea un recurso de trabajo en AWS IoT, que enviará el trabajo a tu dispositivo IoT y se ejecutará en él.

> **Note**
>
> Este procedimiento ejecuta un trabajo en un solo dispositivo.

Para iniciar el agente de trabajos en su Raspberry Pi:

1. En la ventana de terminal de tu ordenador host local que está conectado a tu Raspberry Pi, ejecuta este comando para iniciar el cliente del AWS IoT dispositivo.

   ```
   cd ~/aws-iot-device-client/build
   ./aws-iot-device-client --config-file ~/dc-configs/dc-jobs-config.json
   ```

2. En la ventana del terminal, confirma que el AWS IoT dispositivo es cliente y muestra estos mensajes

   ```
   2021-11-15T18:45:56.708Z [INFO]  {Main.cpp}: Jobs is enabled
                         .
                         .
                         .
   2021-11-15T18:45:56.708Z [INFO]  {Main.cpp}: Client base has been notified that
    Jobs has started
   2021-11-15T18:45:56.708Z [INFO]  {JobsFeature.cpp}: Running Jobs!
   2021-11-15T18:45:56.708Z [DEBUG] {JobsFeature.cpp}: Attempting to subscribe to
    startNextPendingJobExecution accepted and rejected
   2021-11-15T18:45:56.708Z [DEBUG] {JobsFeature.cpp}: Attempting to subscribe to
    nextJobChanged events
   2021-11-15T18:45:56.708Z [DEBUG] {JobsFeature.cpp}: Attempting to subscribe to
    updateJobExecutionStatusAccepted for jobId +
   2021-11-15T18:45:56.738Z [DEBUG] {JobsFeature.cpp}: Ack received for
    SubscribeToUpdateJobExecutionAccepted with code {0}
   ```

```
2021-11-15T18:45:56.739Z [DEBUG] {JobsFeature.cpp}: Attempting to subscribe to
 updateJobExecutionStatusRejected for jobId +
2021-11-15T18:45:56.753Z [DEBUG] {JobsFeature.cpp}: Ack received for
 SubscribeToNextJobChanged with code {0}
2021-11-15T18:45:56.760Z [DEBUG] {JobsFeature.cpp}: Ack received for
 SubscribeToStartNextJobRejected with code {0}
2021-11-15T18:45:56.776Z [DEBUG] {JobsFeature.cpp}: Ack received for
 SubscribeToStartNextJobAccepted with code {0}
2021-11-15T18:45:56.776Z [DEBUG] {JobsFeature.cpp}: Ack received for
 SubscribeToUpdateJobExecutionRejected with code {0}
2021-11-15T18:45:56.777Z [DEBUG] {JobsFeature.cpp}: Publishing
 startNextPendingJobExecutionRequest
2021-11-15T18:45:56.785Z [DEBUG] {JobsFeature.cpp}: Ack received for
 StartNextPendingJobPub with code {0}
2021-11-15T18:45:56.785Z [INFO]  {JobsFeature.cpp}: No pending jobs are scheduled,
 waiting for the next incoming job
```

3. En la ventana del terminal, cuando vea este mensaje, continúe con el siguiente procedimiento y cree el recurso de trabajo. Tenga en cuenta que puede que no sea la última entrada de la lista.

```
2021-11-15T18:45:56.785Z [INFO]  {JobsFeature.cpp}: No pending jobs are scheduled,
 waiting for the next incoming job
```

Para crear un recurso de AWS IoT trabajo

1. En el ordenador host local:

   a. *job_document_url* Sustitúyalo por la URL del documento de trabajo dethe section called “Creación y almacenamiento de un documento de trabajo”.

   b. *thing_arn* Reemplácelo por el ARN del recurso que creó para su dispositivo y, a continuación, ejecute este comando.

   ```
   aws iot create-job \
   --job-id hello-world-job-1 \
   --document-source "job_document_url" \
   --targets "thing_arn" \
   --target-selection SNAPSHOT
   ```

   Si se ejecuta correctamente, el comando devuelve un resultado como este.

```
{
  "jobArn": "arn:aws:iot:us-west-2:57EXAMPLE833:job/hello-world-job-1",
  "jobId": "hello-world-job-1"
}
```

2. En la ventana del terminal, deberías ver un resultado del cliente del AWS IoT dispositivo como este.

```
2021-11-15T18:02:26.688Z [INFO]  {JobsFeature.cpp}: No pending jobs are scheduled,
 waiting for the next incoming job
2021-11-15T18:10:24.890Z [DEBUG] {JobsFeature.cpp}: Job ids differ
2021-11-15T18:10:24.890Z [INFO]  {JobsFeature.cpp}: Executing job: hello-world-
job-1
2021-11-15T18:10:24.890Z [DEBUG] {JobsFeature.cpp}: Attempting to update job
 execution status!
2021-11-15T18:10:24.890Z [DEBUG] {JobsFeature.cpp}: Not including stdout with the
 status details
2021-11-15T18:10:24.890Z [DEBUG] {JobsFeature.cpp}: Not including stderr with the
 status details
2021-11-15T18:10:24.890Z [DEBUG] {JobsFeature.cpp}: Assuming executable is in PATH
2021-11-15T18:10:24.890Z [INFO]  {JobsFeature.cpp}: About to execute: echo Hello
 world!
2021-11-15T18:10:24.890Z [DEBUG] {Retry.cpp}: Retryable function starting, it will
 retry until success
2021-11-15T18:10:24.890Z [DEBUG] {JobsFeature.cpp}: Created EphermalPromise for
 ClientToken 3TEWba9Xj6 in the updateJobExecution promises map
2021-11-15T18:10:24.890Z [DEBUG] {JobEngine.cpp}: Child process now running
2021-11-15T18:10:24.890Z [DEBUG] {JobEngine.cpp}: Child process about to call
 execvp
2021-11-15T18:10:24.890Z [DEBUG] {JobEngine.cpp}: Parent process now running, child
 PID is 16737
2021-11-15T18:10:24.891Z [DEBUG] {16737}: Hello world!
2021-11-15T18:10:24.891Z [DEBUG] {JobEngine.cpp}: JobEngine finished waiting for
 child process, returning 0
2021-11-15T18:10:24.891Z [INFO]  {JobsFeature.cpp}: Job exited with status: 0
2021-11-15T18:10:24.891Z [INFO]  {JobsFeature.cpp}: Job executed successfully!
2021-11-15T18:10:24.891Z [DEBUG] {JobsFeature.cpp}: Attempting to update job
 execution status!
2021-11-15T18:10:24.891Z [DEBUG] {JobsFeature.cpp}: Not including stdout with the
 status details
2021-11-15T18:10:24.891Z [DEBUG] {JobsFeature.cpp}: Not including stderr with the
 status details
```

```
2021-11-15T18:10:24.892Z [DEBUG] {Retry.cpp}: Retryable function starting, it will
 retry until success
2021-11-15T18:10:24.892Z [DEBUG] {JobsFeature.cpp}: Created EphermalPromise for
 ClientToken GmQ0HTzWGg in the updateJobExecution promises map
2021-11-15T18:10:24.905Z [DEBUG] {JobsFeature.cpp}: Ack received for
 PublishUpdateJobExecutionStatus with code {0}
2021-11-15T18:10:24.905Z [DEBUG] {JobsFeature.cpp}: Removing ClientToken 3TEWba9Xj6
 from the updateJobExecution promises map
2021-11-15T18:10:24.905Z [DEBUG] {JobsFeature.cpp}: Success response after
 UpdateJobExecution for job hello-world-job-1
2021-11-15T18:10:24.917Z [DEBUG] {JobsFeature.cpp}: Ack received for
 PublishUpdateJobExecutionStatus with code {0}
2021-11-15T18:10:24.918Z [DEBUG] {JobsFeature.cpp}: Removing ClientToken GmQ0HTzWGg
 from the updateJobExecution promises map
2021-11-15T18:10:24.918Z [DEBUG] {JobsFeature.cpp}: Success response after
 UpdateJobExecution for job hello-world-job-1
2021-11-15T18:10:25.861Z [INFO]  {JobsFeature.cpp}: No pending jobs are scheduled,
 waiting for the next incoming job
```

3. Mientras el cliente del AWS IoT dispositivo está en ejecución y esperando un trabajo, puede enviar otro trabajo cambiando el `job-id` valor y volviendo a ejecutar el create-job paso 1.

Cuando termines de ejecutar las tareas, en la ventana del terminal, ingresa ^C (Control-C) para detener el cliente del AWS IoT dispositivo.

Tutorial: Limpiar después de ejecutar los tutoriales de AWS IoT Device Client

Los procedimientos de este tutorial le explican cómo eliminar los archivos y recursos que ha creado mientras completa los tutoriales de esta ruta de aprendizaje.

Procedimientos de este tutorial

- Paso 1: Limpiar sus dispositivos después de crear demostraciones con el cliente de AWS IoT dispositivos
- Paso 2: Limpiar las demostraciones Cuenta de AWS después de crear con el cliente del AWS IoT dispositivo

Paso 1: Limpiar sus dispositivos después de crear demostraciones con el cliente de AWS IoT dispositivos

Este tutorial describe dos opciones para limpiar la tarjeta microSD después de crear las demostraciones de esta ruta de aprendizaje. Elija la opción que ofrezca el nivel de seguridad que necesita.

Ten en cuenta que al limpiar la tarjeta microSD del dispositivo no se elimina ningún AWS IoT recurso que hayas creado. Para limpiar los AWS IoT recursos después de limpiar la tarjeta microSD del dispositivo, consulte el tutorial en. the section called “Limpiar después de crear demostraciones con el AWS IoT Device Client”

Opción 1: Limpiar sobrescribiendo la tarjeta microSD

La forma más fácil y completa de limpiar la tarjeta microSD después de completar los tutoriales de esta ruta de aprendizaje es sobrescribir la tarjeta microSD con un archivo de imagen guardado que creó al preparar el dispositivo por primera vez.

Este procedimiento utiliza el ordenador host local para escribir una imagen de tarjeta microSD guardada en una tarjeta microSD.

> **Note**
>
> Si el dispositivo no utiliza un medio de almacenamiento extraíble para su sistema operativo, consulte el procedimiento correspondiente a ese dispositivo.

Para escribir una nueva imagen en la tarjeta microSD

1. En el ordenador host local, localice la imagen de la tarjeta microSD guardada que desee escribir en la tarjeta microSD.
2. Inserte su tarjeta microSD en el ordenador host local.
3. Utilizando una herramienta de creación de imágenes de tarjetas SD, escriba el archivo de imagen seleccionado en la tarjeta microSD.
4. Después de escribir la imagen de Raspberry Pi SO en la tarjeta microSD, extraiga la tarjeta microSD y extráigala de forma segura del ordenador host local.

La tarjeta microSD está lista para usar.

Opción 2: Limpiar eliminando los directorios de usuario

Para limpiar la tarjeta microSD después de completar los tutoriales sin volver sobrescribir la imagen de la tarjeta microSD, puede eliminar los directorios de usuario individualmente. Esto no es tan exhaustivo como volver a escribir la tarjeta microSD a partir de una imagen guardada porque no elimina ningún archivo del sistema que pueda haberse instalado.

Si la eliminación de los directorios de usuario es lo suficientemente exhaustiva para sus necesidades, puede seguir este procedimiento.

Para eliminar los directorios de usuario de esta ruta de aprendizaje del dispositivo

1. Para eliminar los directorios y subdirectorios de los usuarios y todos sus archivos que se crearon en esta ruta de aprendizaje, ejecute estos comandos en la ventana del terminal conectada a su dispositivo.

   > **Note**
   >
   > Tras eliminar estos directorios y archivos, no podrá ejecutar las demostraciones sin volver a completar los tutoriales.

   ```
   rm -Rf ~/dc-configs
   rm -Rf ~/policies
   rm -Rf ~/messages
   rm -Rf ~/certs
   rm -Rf ~/.aws-iot-device-client
   ```

2. Ejecute estos comandos para eliminar los directorios y archivos de origen de la aplicación en la ventana del terminal conectada a su dispositivo.

   > **Note**
   >
   > Estos comandos no desinstalan ningún programa. Solo eliminan los archivos de origen utilizados para crearlos e instalarlos. Después de eliminar estos archivos, es posible que el cliente del AWS IoT dispositivo AWS CLI y el dispositivo no funcionen.

   ```
   rm -Rf ~/aws-cli
   ```

```
rm -Rf ~/aws
rm -Rf ~/aws-iot-device-client
```

Paso 2: Limpiar las demostraciones Cuenta de AWS después de crear con el cliente del AWS IoT dispositivo

Estos procedimientos le ayudan a identificar y eliminar los AWS recursos que ha creado al completar los tutoriales de esta ruta de aprendizaje.

Limpie AWS IoT los recursos

Este procedimiento le ayuda a identificar y eliminar los AWS IoT recursos que creó al completar los tutoriales de esta ruta de aprendizaje.

AWS IoT recursos creados en esta ruta de aprendizaje

Tutorial	Recurso de objetos	Recurso de políticas
the section called “Instalación y configuración del cliente de dispositivo de IoT”	DevCliTestThing	DevCliTestThingPolicy
the section called “Comunicación con Device Client mediante MQTT”	PubSubTestThing	PubSubTestThingPolicy
the section called “Ejecución de tareas de IoT con Device Client”	definido por el usuario (puede haber más de uno)	definido por el usuario (puede haber más de uno)

Para eliminar los AWS IoT recursos, siga este procedimiento para cada recurso que haya creado

1. Reemplace *thing_name* por el nombre del recurso objeto que quiera eliminar y, a continuación, ejecute este comando para enumerar los certificados asociados al recurso objeto desde el equipo host local.

```
aws iot list-thing-principals --thing-name thing_name
```

Este comando devuelve una respuesta como esta, en la que se enumeran los certificados asociados a *thing_name*. En la mayoría de los casos, solo habrá un certificado en la lista.

```
{
    "principals": [
        "arn:aws:iot:us-
west-2:57EXAMPLE833:cert/23853eea3cf0edc7f8a69c74abeafa27b2b52823cab5b3e156295e94b26ae8ac"
    ]
}
```

2. Para cada certificado enumerado en el comando anterior:

   a. Reemplace *certificate_ID* por el ID de certificado del comando anterior. El identificador del certificado son los caracteres alfanuméricos que siguen a `cert/` en el ARN devuelto por el comando anterior. A continuación, ejecute este comando para inactivar el certificado.

   ```
   aws iot update-certificate --new-status INACTIVE --certificate-
   id certificate_ID
   ```

   Si se ejecuta correctamente, el comando no devolverá nada.

   b. Reemplace *certificate_ARN* por el ARN del certificado de la lista de certificados devuelta anteriormente y, a continuación, ejecute este comando para enumerar las políticas asociadas a este certificado.

   ```
   aws iot list-attached-policies --target certificate_ARN
   ```

   Este comando devuelve una respuesta como esta que enumera las políticas asociadas al certificado. En la mayoría de los casos, solo habrá una política en la lista.

   ```
   {
       "policies": [
           {
               "policyName": "DevCliTestThingPolicy",
               "policyArn": "arn:aws:iot:us-west-2:57EXAMPLE833:policy/
   DevCliTestThingPolicy"
           }
       ]
   }
   ```

c. Para cada política asociada al certificado:

i. Reemplace *policy_name* por el valor `policyName` del comando anterior, reemplace *certificate_ARN* por el ARN del certificado y, a continuación, ejecute este comando para desasociar la política del certificado.

```
aws iot detach-policy --policy-name policy_name --target certificate_ARN
```

Si se ejecuta correctamente, el comando no devolverá nada.

ii. Reemplace *policy_name* por el valor `policyName` y, a continuación, ejecute este comando para comprobar si la política está asociada a más certificados.

```
aws iot list-targets-for-policy --policy-name policy_name
```

Si el comando devuelve una lista vacía como esta, la política no se asocia a ningún certificado y se seguirán enumerando las versiones de la política. Si aún hay certificados asociados a la política, continúe con el paso detach-thing-principal.

```
{
    "targets": []
}
```

iii. Reemplace *policy_name* por el valor `policyName` y, a continuación, ejecute este comando para comprobar las versiones de la política. Para eliminar la política, solo debe tener una versión.

```
aws iot list-policy-versions --policy-name policy_name
```

Si la política solo tiene una versión, como en este ejemplo, puede saltar al paso delete-policy y eliminar la política ahora.

```
{
    "policyVersions": [
        {
            "versionId": "1",
            "isDefaultVersion": true,
            "createDate": "2021-11-18T01:02:46.778000+00:00"
        }
    ]
```

```
}
```

Si la política tiene más de una versión, como en este ejemplo, se deben eliminar las versiones de la política con un valor `isDefaultVersion` de `false` antes de poder eliminar la política.

```
{
    "policyVersions": [
        {
            "versionId": "2",
            "isDefaultVersion": true,
            "createDate": "2021-11-18T01:52:04.423000+00:00"
        },
        {
            "versionId": "1",
            "isDefaultVersion": false,
            "createDate": "2021-11-18T01:30:18.083000+00:00"
        }
    ]
}
```

Si necesita eliminar una versión de la política, reemplace *`policy_name`* por el valor `policyName`, reemplace *`version_ID`* por el valor `versionId` del comando anterior y, a continuación, ejecute este comando para eliminar una versión de la política.

```
aws iot delete-policy-version --policy-name policy_name --policy-version-
id version_ID
```

Si se ejecuta correctamente, el comando no devolverá nada.

Después de eliminar una versión de la política, repita este paso hasta que la política tenga solo una versión de política.

iv. Reemplace *`policy_name`* por el valor `policyName` y, a continuación, ejecute este comando para eliminar la política.

```
aws iot delete-policy --policy-name policy_name
```

d. Reemplace *thing_name* por el nombre del objeto, reemplace *certificate_ARN* por el ARN del certificado y, a continuación, ejecute este comando para desasociar el certificado del recurso de objeto.

```
aws iot detach-thing-principal --thing-name thing_name --
principal certificate_ARN
```

Si se ejecuta correctamente, el comando no devolverá nada.

e. Reemplace *certificate_ID* por el ID de certificado del comando anterior. El identificador del certificado son los caracteres alfanuméricos que siguen a `cert/` en el ARN devuelto por el comando anterior. A continuación, ejecute este comando para eliminar el recurso del certificado.

```
aws iot delete-certificate --certificate-id certificate_ID
```

Si se ejecuta correctamente, el comando no devolverá nada.

3. Reemplace *thing_name* por el nombre del objeto y, a continuación, ejecute este comando para eliminarlo.

```
aws iot delete-thing --thing-name thing_name
```

Si se ejecuta correctamente, el comando no devolverá nada.

Limpie AWS los recursos

Este procedimiento le ayuda a identificar y eliminar otros AWS recursos que haya creado al completar los tutoriales de esta ruta de aprendizaje.

Otros AWS recursos creados en esta ruta de aprendizaje

Tutorial	Tipo de recurso	Nombre o ID del recurso
the section called “Ejecución de tareas de IoT con Device Client”	Objeto de Amazon S3	hello-world-job.json

Tutorial	Tipo de recurso	Nombre o ID del recurso
the section called “Ejecución de tareas de IoT con Device Client”	AWS IoT recursos laborales	definido por el usuario

Para eliminar los AWS recursos creados en esta ruta de aprendizaje

1. Para eliminar otros trabajos creados en esta ruta de aprendizaje

   a. Ejecute este comando para enumerar los trabajos de su Cuenta de AWS.

   ```
   aws iot list-jobs
   ```

   El comando devuelve una lista de los AWS IoT trabajos que tiene Cuenta de AWS y Región de AWS que tiene el siguiente aspecto.

   ```
   {
       "jobs": [
           {
               "jobArn": "arn:aws:iot:us-west-2:57EXAMPLE833:job/hello-world-
   job-2",
               "jobId": "hello-world-job-2",
               "targetSelection": "SNAPSHOT",
               "status": "COMPLETED",
               "createdAt": "2021-11-16T23:40:36.825000+00:00",
               "lastUpdatedAt": "2021-11-16T23:40:41.375000+00:00",
               "completedAt": "2021-11-16T23:40:41.375000+00:00"
           },
           {
               "jobArn": "arn:aws:iot:us-west-2:57EXAMPLE833:job/hello-world-
   job-1",
               "jobId": "hello-world-job-1",
               "targetSelection": "SNAPSHOT",
               "status": "COMPLETED",
               "createdAt": "2021-11-16T23:35:26.381000+00:00",
               "lastUpdatedAt": "2021-11-16T23:35:29.239000+00:00",
               "completedAt": "2021-11-16T23:35:29.239000+00:00"
           }
       ]
   ```

```
}
```

b. Para cada trabajo que reconozca de la lista como un trabajo que creó en esta ruta de aprendizaje, *jobId* sustitúyalo por el `jobId` valor del trabajo que desea eliminar y, a continuación, ejecute este comando para eliminar un AWS IoT trabajo.

```
aws iot delete-job --job-id jobId
```

Si se ejecuta correctamente, el comando no devuelve nada.

2. Para eliminar los documentos de trabajo que guardó en un bucket de Amazon S3 en esta ruta de aprendizaje.

a. Reemplace *bucket* por el nombre del bucket que utilizó y, a continuación, ejecute este comando para enumerar los objetos del bucket de Amazon S3 que utilizó.

```
aws s3api list-objects --bucket bucket
```

El comando devuelve una lista de los objetos de Amazon S3 en un bucket con el siguiente aspecto.

```
{
    "Contents": [
        {
            "Key": "hello-world-job.json",
            "LastModified": "2021-11-18T03:02:12+00:00",
            "ETag": "\"868c8bc3f56b5787964764d4b18ed5ef\"",
            "Size": 54,
            "StorageClass": "STANDARD",
            "Owner": {
                "DisplayName": "EXAMPLE",
                "ID":
 "e9e3d6ec1EXAMPLEf5bfb5e6bd0a2b6ed03884d1ed392a82ad011c144736a4ee"
            }
        },
        {
            "Key": "iot_job_firmware_update.json",
            "LastModified": "2021-04-13T21:57:07+00:00",
            "ETag": "\"7c68c591949391791ecf625253658c61\"",
            "Size": 66,
            "StorageClass": "STANDARD",
            "Owner": {
```

```
                "DisplayName": "EXAMPLE",
                "ID":
 "e9e3d6ec1EXAMPLEf5bfb5e6bd0a2b6ed03884d1ed392a82ad011c144736a4ee"
            }
        },
        {
            "Key": "order66.json",
            "LastModified": "2021-04-13T21:57:07+00:00",
            "ETag": "\"bca60d5380b88e1a70cc27d321caba72\"",
            "Size": 29,
            "StorageClass": "STANDARD",
            "Owner": {
                "DisplayName": "EXAMPLE",
                "ID":
 "e9e3d6ec1EXAMPLEf5bfb5e6bd0a2b6ed03884d1ed392a82ad011c144736a4ee"
            }
        }
    ]
}
```

b. Para cada objeto que reconozca de la lista como un objeto creado en esta ruta de aprendizaje, reemplace *bucket* por el nombre del bucket y *key* por el valor clave del objeto que quiera eliminar. A continuación, ejecute este comando para eliminar un objeto de Amazon S3.

```
aws s3api delete-object --bucket bucket --key key
```

Si se ejecuta correctamente, el comando no devuelve nada.

Después de eliminar todos los AWS recursos y objetos que creó al completar esta ruta de aprendizaje, puede empezar de nuevo y repetir los tutoriales.

Creación de soluciones con el dispositivo AWS IoT SDKs

Los tutoriales de esta sección lo guían a través de los pasos para desarrollar una solución de IoT que se pueda implementar en un entorno de producción mediante AWS IoT.

Estos tutoriales pueden tardar más tiempo en completarse que los de la sección correspondiente, the section called “Creación de demostraciones con el cliente de AWS IoT dispositivos” ya que utilizan

el AWS IoT dispositivo SDKs y explican los conceptos que se están aplicando con más detalle para ayudarle a crear soluciones seguras y fiables.

Comience a crear soluciones con el AWS IoT dispositivo SDKs

Estos tutoriales lo guían a través de diferentes AWS IoT escenarios. Cuando proceda, los tutoriales utilizan el AWS IoT dispositivo SDKs.

Temas

- Tutorial: Conexión de un dispositivo AWS IoT Core mediante el SDK del AWS IoT dispositivo
- Crear AWS IoT reglas para enrutar los datos del dispositivo a otros servicios
- Retener el estado del dispositivo mientras el dispositivo está fuera de línea con Device Shadows
- Tutorial: Crear un autorizador personalizado para AWS IoT Core
- Tutorial: Monitorización de la humedad del suelo con una AWS IoT Raspberry Pi

Tutorial: Conexión de un dispositivo AWS IoT Core mediante el SDK del AWS IoT dispositivo

En este tutorial se muestra cómo conectar un dispositivo para que AWS IoT Core pueda enviar y recibir datos desde y hacia AWS IoT. Después de completar este tutorial, el dispositivo se configurará para conectarse AWS IoT Core y comprenderá cómo se comunican los dispositivos con ellos AWS IoT.

Temas

- Requisitos previos
- Prepara tu dispositivo para AWS IoT
- Revise el protocolo MQTT
- Consulte la aplicación de ejemplo del SDK de dispositivo pubsub.py
- Conecta tu dispositivo y comunícate con AWS IoT Core
- Revisión de los resultados.
- Tutorial: Uso de la AWS IoT Device SDK para Embedded C

Requisitos previos

Antes de empezar este tutorial, asegúrese de que ha:

- Finalizado Cómo empezar con AWS IoT Core los tutoriales

  En la sección de ese tutorial donde debe the section called “Configuración del dispositivo”, seleccione la opción the section called “Conexión de una Raspberry Pi u otro dispositivo” para su dispositivo y utilice las opciones del lenguaje Python para configurar su dispositivo.

  > **Note**
  >
  > Mantenga abierta la ventana de terminal que utilizó en ese tutorial porque también la utilizará en este.

- Un dispositivo que puede ejecutar el AWS IoT Device SDK v2 para Python.

  Este tutorial muestra cómo conectar un dispositivo AWS IoT Core mediante ejemplos de código de Python, que requieren un dispositivo relativamente potente. Si trabaja con dispositivos con recursos limitados, es posible que estos ejemplos de código no funcionen en ellos. En dicho caso, es posible que tenga más éxito con el tutorial de the section called “Uso de AWS IoT Device SDK para Embedded C”.

- Obtención de la información necesaria para conectarse al dispositivo

  Para conectar el dispositivo AWS IoT, debe tener información sobre el nombre del dispositivo, el nombre del host y el número de puerto.

  > **Note**
  >
  > También puede usar la autenticación personalizada para conectar dispositivos a AWS IoT Core. Los datos de conexión que se transfieren a la función de Lambda de su autorizador dependen del protocolo que utilice.

  - Nombre de la cosa: el nombre de la AWS IoT cosa a la que quieres conectarte. Debes haberte registrado como dispositivo como una AWS IoT cosa. Para obtener más información, consulte Administrar dispositivos con AWS IoT.
  - Nombre de host: el nombre de host del punto de conexión de IoT específico de la cuenta.
  - Número de puerto: el número de puerto al que desea conectarse.

  Puede usar el `configureEndpoint` método del SDK de AWS IoT Python para configurar el nombre de host y el número de puerto.

```
myAWSIoTMQTTClient.configureEndpoint("random.iot.region.amazonaws.com", 8883)
```

Prepara tu dispositivo para AWS IoT

En Cómo empezar con AWS IoT Core los tutoriales, preparó su dispositivo y su cuenta AWS para que pudieran comunicarse. En esta sección se analizan los aspectos de esa preparación que se aplican a cualquier dispositivo que se conecte a AWS IoT Core.

Para conectar un dispositivo a AWS IoT Core:

1. Debe tener una Cuenta de AWS.

   En el procedimiento Configurar Cuenta de AWS se describe cómo crear una Cuenta de AWS si aún no tiene una.

2. En esa cuenta, debe tener definidos los siguientes AWS IoT recursos para el dispositivo de su Cuenta de AWS región.

   El procedimiento en Crea AWS IoT recursos se describe en este documento describe cómo crear estos recursos para el dispositivo en su Cuenta de AWS y su región.

   - Un certificado de dispositivo registrado con AWS IoT y activado para autenticar el dispositivo.

     El certificado suele crearse con un objeto y adjuntarse a AWS IoT él. Si bien no se requiere un objeto para conectarse a un dispositivo AWS IoT, este pone a su disposición AWS IoT funciones adicionales.

   - Una política adjunta al certificado del dispositivo que lo autoriza a conectarse AWS IoT Core y realizar todas las acciones que desee.

3. Una conexión a Internet que pueda acceder a los puntos de conexión del dispositivo Cuenta de AWS.

   Los puntos finales del dispositivo se describen en la página de configuración de la consola AWS IoT datos del dispositivo y puntos finales de servicio y se pueden ver en ella. AWS IoT

4. Software de comunicación, como el que SDKs proporciona el AWS IoT dispositivo. En este tutorial se utiliza el SDK de AWS IoT dispositivo v2 para Python.

Revise el protocolo MQTT

Antes de hablar de la aplicación de ejemplo, es útil entender el protocolo MQTT. El protocolo MQTT ofrece algunas ventajas sobre otros protocolos de comunicación de red, como HTTP, que lo convierten en una opción popular para los dispositivos IoT. En esta sección se analizan los aspectos clave del MQTT aplicables a este tutorial. Para obtener información sobre la comparación entre MQTT y HTTP, consulte Elección de un protocolo de aplicación para la comunicación entre dispositivos.

MQTT utiliza un modelo de publish/subscribe comunicación

El protocolo MQTT utiliza un modelo de publish/subscribe comunicación con su anfitrión. Este modelo es diferente del request/response modelo que utiliza HTTP. Con MQTT, los dispositivos establecen una sesión con el host que se identifica mediante un ID de cliente único. Para enviar datos, los dispositivos publican los mensajes identificados por temas en un agente de mensajes del host. Para recibir mensajes del agente de mensajes, los dispositivos se suscriben a los temas enviando filtros de temas en las solicitudes de suscripción al agente de mensajes.

MQTT admite sesiones persistentes

El agente de mensajes recibe los mensajes de los dispositivos y los publica en los dispositivos que se han suscrito a ellos. Con las sesiones persistentes (sesiones que permanecen activas incluso cuando el dispositivo de inicio está desconectado), los dispositivos pueden recuperar los mensajes que se publicaron mientras estaban desconectados. En cuanto al dispositivo, MQTT admite niveles de calidad de servicio (QoS) que garantizan que el host reciba los mensajes enviados por el dispositivo.

Consulte la aplicación de ejemplo del SDK de dispositivo pubsub.py

En esta sección se revisa la aplicación de ejemplo `pubsub.py` del SDK de dispositivos AWS IoT v2 para Python utilizada en este tutorial. Aquí veremos cómo se conecta para publicar y suscribirse AWS IoT Core a los mensajes MQTT. En la siguiente sección se presentan algunos ejercicios que le ayudarán a explorar cómo se conecta y se comunica un dispositivo con AWS IoT Coreél.

La aplicación pubsub.py de ejemplo muestra estos aspectos de una conexión MQTT con AWS IoT Core:

- Protocolos de comunicación
- Sesiones persistentes
- Calidad de servicio

- Publicación de mensajes
- Suscripción a mensajes
- Desconexión y reconexión del dispositivo

Protocolos de comunicación

En el ejemplo `pubsub.py` se muestra una conexión MQTT mediante los protocolos MQTT y MQTT sobre WSS. La biblioteca AWS Common Runtime (AWS CRT) proporciona compatibilidad con el protocolo de comunicación de bajo nivel y se incluye en el AWS IoT Device SDK v2 para Python.

MQTT

El `pubsub.py` ejemplo de llamadas `mtls_from_path` (que se muestra aquí) es `mqtt_connection_builder`para establecer una conexión AWS IoT Core mediante el protocolo MQTT. `mtls_from_path`utiliza los certificados X.509 y TLS v1.2 para autenticar el dispositivo. La biblioteca AWS CRT gestiona los detalles de nivel inferior de esa conexión.

```
mqtt_connection = mqtt_connection_builder.mtls_from_path(
    endpoint=args.endpoint,
    cert_filepath=args.cert,
    pri_key_filepath=args.key,
    ca_filepath=args.ca_file,
    client_bootstrap=client_bootstrap,
    on_connection_interrupted=on_connection_interrupted,
    on_connection_resumed=on_connection_resumed,
    client_id=args.client_id,
    clean_session=False,
    keep_alive_secs=6
)
```

`endpoint`

El punto final Cuenta de AWS de su dispositivo IoT

En la aplicación de ejemplo, este valor se transfiere desde la línea de comandos.

`cert_filepath`

La ruta al archivo del certificado del dispositivo

En la aplicación de ejemplo, este valor se transfiere desde la línea de comandos.

`pri_key_filepath`

La ruta al archivo de clave privada del dispositivo que se creó con su archivo de certificado

En la aplicación de ejemplo, este valor se transfiere desde la línea de comandos.

`ca_filepath`

La ruta al archivo de la CA Raíz. Solo es obligatorio si el servidor MQTT usa un certificado que aún no esté en tu almacén de confianza.

En la aplicación de ejemplo, este valor se transfiere desde la línea de comandos.

`client_bootstrap`

El objeto de tiempo de ejecución común que gestiona las actividades de comunicación de los sockets

En la aplicación de ejemplo, se crea una instancia de este objeto antes de la llamada a `mqtt_connection_builder.mtls_from_path`.

`on_connection_interrupted`, `on_connection_resumed`

y son funciones de devolución de llamada para llamar cuando la conexión del dispositivo se interrumpe y se reanuda.

`client_id`

El identificador que identifica de forma exclusiva a este dispositivo en la Región de AWS

En la aplicación de ejemplo, este valor se transfiere desde la línea de comandos.

`clean_session`

Si se debe iniciar una nueva sesión persistente o, si hay una, volver a conectarse a una existente

`keep_alive_secs`

El valor keep alive, en segundos, para enviar la solicitud `CONNECT`. Se enviará automáticamente un ping en este intervalo. Si el servidor no recibe un ping después de 1,5 veces este valor, asume que la conexión se ha perdido.

MQTT a través de WSS

Las llamadas de ejemplo `pubsub.py` `websockets_with_default_aws_signing` (mostradas aquí) en el `mqtt_connection_builder` para establecer una conexión con AWS IoT Core

mediante el protocolo MQTT sobre WSS. `websockets_with_default_aws_signing` crea una conexión de MQTT sobre WSS mediante Signature V4 para autenticar el dispositivo.

```
mqtt_connection = mqtt_connection_builder.websockets_with_default_aws_signing(
    endpoint=args.endpoint,
    client_bootstrap=client_bootstrap,
    region=args.signing_region,
    credentials_provider=credentials_provider,
    websocket_proxy_options=proxy_options,
    ca_filepath=args.ca_file,
    on_connection_interrupted=on_connection_interrupted,
    on_connection_resumed=on_connection_resumed,
    client_id=args.client_id,
    clean_session=False,
    keep_alive_secs=6
)
```

`endpoint`

El punto final Cuenta de AWS de su dispositivo IoT

En la aplicación de ejemplo, este valor se transfiere desde la línea de comandos.

`client_bootstrap`

El objeto de tiempo de ejecución común que gestiona las actividades de comunicación de los sockets

En la aplicación de ejemplo, se crea una instancia de este objeto antes de la llamada a `mqtt_connection_builder.websockets_with_default_aws_signing`.

`region`

La región de AWS firma utilizada por la autenticación Signature V4. En `pubsub.py`, pasa el parámetro introducido en la línea de comandos.

En la aplicación de ejemplo, este valor se transfiere desde la línea de comandos.

`credentials_provider`

Las AWS credenciales proporcionadas para usarlas en la autenticación

En la aplicación de ejemplo, se crea una instancia de este objeto antes de la llamada a `mqtt_connection_builder.websockets_with_default_aws_signing`.

`websocket_proxy_options`

Opciones del proxy HTTP, si utiliza un host proxy

En la aplicación de ejemplo, este valor se inicializa antes de la llamada a `mqtt_connection_builder.websockets_with_default_aws_signing`.

`ca_filepath`

La ruta al archivo de la CA Raíz. Solo es obligatorio si el servidor MQTT usa un certificado que aún no esté en tu almacén de confianza.

En la aplicación de ejemplo, este valor se transfiere desde la línea de comandos.

`on_connection_interrupted`, `on_connection_resumed`

La función de devolución de llamada para llamar cuando la conexión del dispositivo se interrumpe y se reanuda.

`client_id`

El identificador que identifica de forma exclusiva a este dispositivo en la Región de AWS.

En la aplicación de ejemplo, este valor se transfiere desde la línea de comandos.

`clean_session`

Si se debe iniciar una nueva sesión persistente o, si hay una, volver a conectarse a una existente

`keep_alive_secs`

El valor keep alive, en segundos, para enviar la solicitud `CONNECT`. Se enviará automáticamente un ping en este intervalo. Si el servidor no recibe un ping después de 1,5 veces este valor, se supone que se ha perdido la conexión.

HTTPS

¿Qué pasa con HTTPS? AWS IoT Core admite dispositivos que publican solicitudes HTTPS. Desde el punto de vista de la programación, los dispositivos envían solicitudes HTTPS a AWS IoT Core igual que cualquier otra aplicación. Para ver un ejemplo de un programa de Python que envía un mensaje HTTP desde un dispositivo, consulta el ejemplo de código HTTPS con la biblioteca `requests` de Python. En este ejemplo, se envía un mensaje AWS IoT Core mediante HTTPS de forma que se AWS IoT Core interpreta como un mensaje MQTT.

Si bien AWS IoT Core es compatible con las solicitudes HTTPS de los dispositivos, asegúrese de revisar la información al respecto Elección de un protocolo de aplicación para la comunicación entre dispositivos para poder tomar una decisión informada sobre qué protocolo utilizar para las comunicaciones de su dispositivo.

Sesiones persistentes

En la aplicación de ejemplo, si se establece el parámetro `clean_session` en `False` indica que la conexión debe ser persistente. En la práctica, esto significa que la conexión abierta por esta llamada se vuelve a conectar a una sesión persistente existente, si existe alguna. De lo contrario, crea una nueva sesión persistente y se conecta a ella.

Con una sesión persistente, el agente de mensajes almacena los mensajes que se envían al dispositivo mientras el dispositivo no está conectado. Cuando un dispositivo se vuelve a conectar a una sesión persistente, el agente de mensajes envía al dispositivo todos los mensajes almacenados a los que se haya suscrito.

Sin una sesión persistente, el dispositivo no recibirá los mensajes que se envíen mientras no esté conectado. La opción que utilices dependerá de la aplicación que utilices y de si es necesario comunicar los mensajes que se produzcan mientras el dispositivo no está conectado. Para obtener más información, consulte Sesiones persistentes de MQTT.

Calidad de servicio

Cuando el dispositivo publica mensajes y se suscribe a ellos, se puede establecer la calidad de servicio (QoS) preferida. AWS IoT admite los niveles de QoS 0 y 1 para las operaciones de publicación y suscripción. Para obtener más información sobre los niveles de QoS en AWS IoT, consulte. Opciones de calidad de servicio (QoS) de MQTT

El tiempo de ejecución AWS CRT para Python define estas constantes para los niveles de QoS que admite:

Niveles de calidad de servicio de Python

Nivel de QoS de MQTT	Valor simbólico de Python utilizado por el SDK	Description (Descripción)
QoS nivel 0	`mqtt.QoS.AT_MOST_ONCE`	Solo se intentará enviar el mensaje una vez, tanto si se recibe como si no. Es posible

Nivel de QoS de MQTT	Valor simbólico de Python utilizado por el SDK	Description (Descripción)
		que el mensaje no se envíe en absoluto, por ejemplo, si el dispositivo no está conectado o si hay un error de red.
QoS nivel 1	`mqtt.QoS.AT_LEAST_ONCE`	El mensaje se envía repetidamente hasta que se recibe un acuse de recibo PUBACK.

En la aplicación de ejemplo, las solicitudes de publicación y suscripción se realizan con un nivel de QoS de 1 (`mqtt.QoS.AT_LEAST_ONCE`).

- QoS al publicar

  Cuando un dispositivo publica un mensaje con el nivel 1 de QoS, envía el mensaje repetidamente hasta que recibe una respuesta PUBACK del agente de mensajes. Si el dispositivo no está conectado, el mensaje queda en cola para enviarse una vez que se vuelva a conectar.

- QoS al suscribirse

  Cuando un dispositivo se suscribe a un mensaje con QoS de nivel 1, el agente de mensajes guarda los mensajes a los que está suscrito el dispositivo hasta que se puedan enviar al dispositivo. El agente de mensajes vuelve a enviar los mensajes hasta que recibe una respuesta PUBACK del dispositivo.

Publicación de mensajes

Tras establecer correctamente una conexión AWS IoT Core, los dispositivos pueden publicar mensajes. La muestra `pubsub.py` lo hace llamando a la operación `publish` de la cosa `mqtt_connection`.

```
mqtt_connection.publish(
    topic=args.topic,
    payload=message,
    qos=mqtt.QoS.AT_LEAST_ONCE
```

```
)
```

`topic`

El nombre del tema del mensaje que lo identifica

En la aplicación de ejemplo, este se transfiere desde la línea de comandos.

`payload`

La carga útil del mensaje está formateada como una cadena (por ejemplo, un documento JSON)

En la aplicación de ejemplo, este se transfiere desde la línea de comandos.

Un documento JSON es un formato de carga útil común y otros AWS IoT servicios lo reconocen; sin embargo, el formato de datos de la carga útil del mensaje puede ser cualquier formato que acuerden los editores y los suscriptores. Sin embargo, otros servicios AWS IoT solo reconocen JSON y, en algunos casos, CBOR para la mayoría de las operaciones.

`qos`

El nivel de QoS de este mensaje

Suscripción a mensajes

Para recibir mensajes de otros servicios AWS IoT y dispositivos, los dispositivos se suscriben a esos mensajes por el nombre del tema. Los dispositivos pueden suscribirse a mensajes individuales especificando un nombre de tema y a un grupo de mensajes especificando un filtro de tema, que puede incluir caracteres comodín. En el ejemplo `pubsub.py`, se utiliza el código que se muestra aquí para suscribirse a los mensajes y registrar las funciones de devolución de llamada para procesar el mensaje una vez recibido.

```
subscribe_future, packet_id = mqtt_connection.subscribe(
    topic=args.topic,
    qos=mqtt.QoS.AT_LEAST_ONCE,
    callback=on_message_received
)
subscribe_result = subscribe_future.result()
```

`topic`

El tema al que suscribirse. Puede ser el nombre de un tema o un filtro de tema.

En la aplicación de ejemplo, este se transfiere desde la línea de comandos.

`qos`

Si el agente de mensajes debe almacenar estos mensajes mientras el dispositivo está desconectado.

Un valor de `mqtt.QoS.AT_LEAST_ONCE` (nivel de QoS 1) requiere que se especifique una sesión persistente (`clean_session=False`) al crear la conexión.

`callback`

La función a la que se debe llamar para procesar el mensaje suscrito.

La función `mqtt_connection.subscribe` devuelve un futuro y un ID de paquete. Si la solicitud de suscripción se inició correctamente, el identificador del paquete devuelto es superior a 0. Para asegurarse de que el agente de mensajes recibió y registró la suscripción, debe esperar a que se devuelva el resultado de la operación asíncrona, como se muestra en el ejemplo de código.

Función de devolución de llamada

La devolución de llamada del ejemplo `pubsub.py` procesa los mensajes suscritos a medida que el dispositivo los recibe.

```
def on_message_received(topic, payload, **kwargs):
    print("Received message from topic '{}': {}".format(topic, payload))
    global received_count
    received_count += 1
    if received_count == args.count:
        received_all_event.set()
```

`topic`

El tema del mensaje

Es el nombre del tema específico del mensaje recibido, incluso si te has suscrito a un filtro de temas.

`payload`

La carga útil del mensaje

El formato es específico de la aplicación.

`kwargs`

Posibles argumentos adicionales, tal como se describe en `mqtt.Connection.subscribe`.

En el ejemplo `pubsub.py`, `on_message_received` solo muestra el tema y su carga útil. También cuenta los mensajes recibidos para finalizar el programa una vez alcanzado el límite.

Su aplicación evaluaría el tema y la carga útil para determinar qué acciones debe realizar.

Desconexión y reconexión del dispositivo

El ejemplo `pubsub.py` incluye funciones de devolución de llamada que se invocan cuando el dispositivo se desconecta y cuando se restablece la conexión. Las acciones que realiza el dispositivo ante estos eventos son específicas de la aplicación.

Cuando un dispositivo se conecta por primera vez, debe suscribirse a los temas para poder recibirlos. Si la sesión de un dispositivo está presente cuando se vuelve a conectar, sus suscripciones se restauran y todos los mensajes almacenados en esas suscripciones se envían al dispositivo una vez que se vuelve a conectar.

Si la sesión de un dispositivo ya no existe cuando se vuelve a conectar, debe volver a suscribirse a sus suscripciones. Las sesiones persistentes tienen una vida útil limitada y pueden caducar si el dispositivo se desconecta durante demasiado tiempo.

Conecta tu dispositivo y comunícate con AWS IoT Core

En esta sección se presentan algunos ejercicios que le ayudarán a explorar diferentes aspectos de la conexión del dispositivo a AWS IoT Core. Para estos ejercicios, utilizará el cliente de pruebas MQTT de la consola de AWS IoT para ver lo que publica el dispositivo y publicar los mensajes en el dispositivo. Estos ejercicios utilizan el ejemplo `pubsub.py` del SDK para dispositivos con AWS IoT v2 para Python y se basan en su experiencia con los tutoriales de Tutoriales de introducción.

En esta sección, realizará las siguientes acciones:

- Suscríbase a los filtros de temas comodín
- Procesar suscripciones de filtro de temas
- Publique mensajes desde su dispositivo

Para estos ejercicios, empezará con el programa de ejemplo `pubsub.py`.

> **Note**
>
> En estos ejercicios se presupone que ha completado los tutoriales Tutoriales de introducción y ha utilizado la ventana de terminal del dispositivo que aparece en ese tutorial.

Suscríbase a los filtros de temas comodín

En este ejercicio, modificará la línea de comandos utilizada para llamar a `pubsub.py` para suscribirse a un filtro de temas comodín y procesará los mensajes recibidos en función del tema del mensaje.

Procedimiento del ejercicio

Para este ejercicio, imagine que su dispositivo contiene un control de temperatura y un control de luz. Utiliza estos nombres de temas para identificar los mensajes sobre ellos.

1. Antes de empezar el ejercicio, intente ejecutar este comando de los tutoriales Tutoriales de introducción del dispositivo para asegurarse de que todo está listo para el ejercicio.

   ```
   cd ~/aws-iot-device-sdk-python-v2/samples
   python3 pubsub.py --topic topic_1 --ca_file ~/certs/Amazon-root-CA-1.pem --cert ~/
   certs/device.pem.crt --key ~/certs/private.pem.key --endpoint your-iot-endpoint
   ```

   Debería ver el mismo resultado que en el tutorial de introducción.
2. Para este ejercicio, cambie estos parámetros de la línea de comandos.

Action	Parámetro de línea de comando	Efecto
agregar	--message ""	Configure `pubsub.py` para escuchar solo
adición	--count 2	Finalice el programa después de recibir dos mensajes
cambiar	--topic `device/+/details`	Defina el filtro de tema al que desea suscribirse

Al realizar estos cambios en la línea de comandos inicial, se obtiene esta línea de comandos. Introduzca este comando en la ventana del terminal de su dispositivo.

```
python3 pubsub.py --message "" --count 2 --topic device/+/details --ca_file
 ~/certs/Amazon-root-CA-1.pem --cert ~/certs/device.pem.crt --key ~/certs/
private.pem.key --endpoint your-iot-endpoint
```

El programa debe mostrar algo similar al siguiente:

```
Connecting to a3qexamplesffp-ats.iot.us-west-2.amazonaws.com with client ID
 'test-24d7cdcc-cc01-458c-8488-2d05849691e1'...
Connected!
Subscribing to topic 'device/+/details'...
Subscribed with QoS.AT_LEAST_ONCE
Waiting for all messages to be received...
```

Si ve algo como esto en su terminal, su dispositivo está listo y a la escucha de mensajes cuyos nombres de tema empiecen por `device` y terminen por `/detail`. Así que, vamos a probarlo.

3. Aquí hay un par de mensajes que su dispositivo podría recibir.

Nombre del tema	Carga útil de mensaje
`device/temp/details`	`{ "desiredTemp": 20, "currentTemp": 15 }`
`device/light/details`	`{ "desiredLight": 100, "currentLight": 50 }`

4. Con el cliente de prueba MQTT de la AWS IoT consola, envíe los mensajes descritos en el paso anterior a su dispositivo.

   a. Abra el cliente de prueba MQTT en la AWS IoT consola.

   b. En Suscribirse a un tema, en el campo Tema de suscripción escriba **`device/+/details`** y, a continuación, elija Suscribirse al tema.

   c. En la columna Suscripciones del cliente de prueba de MQTT, elija device/+/details.

d. Para cada uno de los temas de la tabla anterior, haga lo siguiente en el cliente de prueba de MQTT:

   1. En Publicar, introduzca el valor de la columna Nombre del tema de la tabla.
   2. En el campo de carga útil del mensaje situado debajo del nombre del tema, introduzca el valor de la columna de carga útil del mensaje de la tabla.
   3. Observe la ventana del terminal en la que `pubsub.py` se está ejecutando y, en el cliente de prueba de MQTT, elija Publicar en un tema.

   Debería ver que el mensaje ha sido recibido por `pubsub.py` en la ventana del terminal.

Resultado del ejercicio

Con esto, `pubsub.py`, se suscribió a los mensajes mediante un filtro de temas comodín, los recibió y los mostró en la ventana del terminal. Observa cómo te has suscrito a un único filtro de temas y que se ha llamado a la función de devolución de llamada para procesar los mensajes que tienen dos temas distintos.

Procesar suscripciones de filtro de temas

Basándose en el ejercicio anterior, modifique la aplicación de ejemplo `pubsub.py` para evaluar los temas de los mensajes y procesar los mensajes suscritos en función del tema.

Procedimiento del ejercicio

Para evaluar el tema del mensaje

1. Copie `pubsub.py` en `pubsub2.py`.
2. Abra `pubsub2.py` en el editor de texto o IDE.
3. En `pubsub2.py`, busque la función `on_message_received`.
4. En `on_message_received`, inserte el siguiente código después de la línea que empieza por `print("Received message` y antes de la línea que empieza por `global received_count`.

```
    topic_parsed = False
    if "/" in topic:
        parsed_topic = topic.split("/")
```

```
        if len(parsed_topic) == 3:
            # this topic has the correct format
            if (parsed_topic[0] == 'device') and (parsed_topic[2] == 'details'):
                # this is a topic we care about, so check the 2nd element
                if (parsed_topic[1] == 'temp'):
                    print("Received temperature request: {}".format(payload))
                    topic_parsed = True
                if (parsed_topic[1] == 'light'):
                    print("Received light request: {}".format(payload))
                    topic_parsed = True
    if not topic_parsed:
        print("Unrecognized message topic.")
```

5. Guarde los cambios y ejecute el programa modificado mediante esta línea de comandos.

```
python3 pubsub2.py --message "" --count 2 --topic device/+/details --ca_file
 ~/certs/Amazon-root-CA-1.pem --cert ~/certs/device.pem.crt --key ~/certs/
private.pem.key --endpoint your-iot-endpoint
```

6. En la AWS IoT consola, abra el cliente de pruebas MQTT.
7. En Suscribirse a un tema, en el campo Tema de suscripción escriba **`device/+/details`** y, a continuación, elija Suscribirse al tema.
8. En la columna Suscripciones del cliente de prueba de MQTT, elija device/+/details.
9. Para cada uno de los temas de la tabla, haga lo siguiente en el cliente de prueba de MQTT:

Nombre del tema	Carga útil de mensaje
`device/temp/details`	`{ "desiredTemp": 20, "currentTemp": 15 }`
`device/light/details`	`{ "desiredLight": 100, "currentLight": 50 }`

   1. En Publicar, introduzca el valor de la columna Nombre del tema de la tabla.
   2. En el campo de carga útil del mensaje situado debajo del nombre del tema, introduzca el valor de la columna de carga útil del mensaje de la tabla.
   3. Observe la ventana del terminal en la que `pubsub.py` se está ejecutando y, en el cliente de prueba de MQTT, elija Publicar en un tema.

Debería ver que el mensaje ha sido recibido por `pubsub.py` en la ventana del terminal.

Debería ver algo similar a esto en la ventana de su terminal.

```
Connecting to a3qexamplesffp-ats.iot.us-west-2.amazonaws.com with client ID 'test-
af794be0-7542-45a0-b0af-0b0ea7474517'...
Connected!
Subscribing to topic 'device/+/details'...
Subscribed with QoS.AT_LEAST_ONCE
Waiting for all messages to be received...
Received message from topic 'device/light/details': b'{ "desiredLight": 100,
 "currentLight": 50 }'
Received light request: b'{ "desiredLight": 100, "currentLight": 50 }'
Received message from topic 'device/temp/details': b'{ "desiredTemp": 20,
 "currentTemp": 15 }'
Received temperature request: b'{ "desiredTemp": 20, "currentTemp": 15 }'
2 message(s) received.
Disconnecting...
Disconnected!
```

Resultado del ejercicio

En este ejercicio, ha añadido código para que la aplicación de ejemplo reconozca y procese varios mensajes en la función de devolución de llamada. Con ello, su dispositivo podrá recibir mensajes y actuar en consecuencia.

Otra forma de que el dispositivo reciba y procese varios mensajes consiste en suscribirse a distintos mensajes por separado y asignar cada suscripción a su propia función de devolución de llamada.

Publique mensajes desde su dispositivo

Puede utilizar la aplicación de ejemplo pubsub.py para publicar mensajes desde su dispositivo. Aunque publicará los mensajes tal cual, éstos no podrán leerse como documentos JSON. En este ejercicio se modifica la aplicación de ejemplo para poder publicar documentos JSON en la carga útil del mensaje que puedan leerse. AWS IoT Core

Procedimiento del ejercicio

En este ejercicio, se enviará el siguiente mensaje con el tema `device/data`.

```
{
    "timestamp": 1601048303,
    "sensorId": 28,
    "sensorData": [
        {
        "sensorName": "Wind speed",
        "sensorValue": 34.2211224
        }
    ]
}
```

Para preparar su cliente de pruebas de MQTT para supervisar los mensajes de este ejercicio

1. En Suscribirse a un tema, en el campo Tema de suscripción escriba **`device/data`** y, a continuación, elija Suscribirse al tema.
2. En la columna Suscripciones del cliente de prueba MQTT, elija device/data.
3. Mantenga abierta la ventana del cliente de prueba de MQTT para esperar a que lleguen los mensajes de su dispositivo.

Para enviar documentos JSON con la aplicación de ejemplo pubsub.py

1. En su dispositivo, copie `pubsub.py` en`pubsub3.py`.
2. Edite `pubsub3.py` para cambiar el formato de los mensajes que publica.
    a. Abra `pubsub3.py` en un editor de texto.
    b. Localice esta línea de código:

    ```
    message = "{} [{}]".format(message_string, publish_count)
    ```

    c. Cambie a:

    ```
    message = "{}".format(message_string)
    ```

    d. Localice esta línea de código:

    ```
    message_json = json.dumps(message)
    ```

    e. Cambie a:

    ```
    message = "{}".json.dumps(json.loads(message))
    ```

    f. Guarde los cambios.

3. En su dispositivo, ejecute este comando para enviar el mensaje dos veces.

```
python3 pubsub3.py  --ca_file ~/certs/Amazon-root-CA-1.pem  --cert ~/certs/
device.pem.crt  --key ~/certs/private.pem.key  --topic device/data  --count 2 --
message '{"timestamp":1601048303,"sensorId":28,"sensorData":[{"sensorName":"Wind
 speed","sensorValue":34.2211224}]}'  --endpoint your-iot-endpoint
```

4. En el cliente de prueba de MQTT, compruebe que ha interpretado y formateado el documento JSON de la carga útil del mensaje, de la siguiente manera:



De forma predeterminada, `pubsub3.py` también se suscribe a los mensajes que envía. Debería ver que recibió los mensajes en la salida de la aplicación. La ventana de la terminal debe tener un aspecto similar al siguiente.

```
Connecting to a3qEXAMPLEsffp-ats.iot.us-west-2.amazonaws.com with client ID
 'test-5cff18ae-1e92-4c38-a9d4-7b9771afc52f'...
Connected!
Subscribing to topic 'device/data'...
Subscribed with QoS.AT_LEAST_ONCE
Sending 2 message(s)
Publishing message to topic 'device/data':
 {"timestamp":1601048303,"sensorId":28,"sensorData":[{"sensorName":"Wind
 speed","sensorValue":34.2211224}]}
Received message from topic 'device/data':
 b'{"timestamp":1601048303,"sensorId":28,"sensorData":[{"sensorName":"Wind
 speed","sensorValue":34.2211224}]}'
Publishing message to topic 'device/data':
 {"timestamp":1601048303,"sensorId":28,"sensorData":[{"sensorName":"Wind
 speed","sensorValue":34.2211224}]}
```

```
Received message from topic 'device/data':
 b'{"timestamp":1601048303,"sensorId":28,"sensorData":[{"sensorName":"Wind
 speed","sensorValue":34.2211224}]}'
2 message(s) received.
Disconnecting...
Disconnected!
```

Resultado del ejercicio

De este modo, el dispositivo puede generar mensajes para enviarlos a fin de probar AWS IoT Core la conectividad básica y proporcionar mensajes del dispositivo AWS IoT Core para su procesamiento. Por ejemplo, puedes usar esta aplicación para enviar datos de prueba desde tu dispositivo para probar las acciones de las AWS IoT reglas.

Revisión de los resultados.

Los ejemplos de este tutorial le proporcionaron una experiencia práctica sobre los conceptos básicos de cómo los dispositivos pueden comunicarse con ellos AWS IoT Core, una parte fundamental de su AWS IoT solución. Cuando sus dispositivos pueden comunicarse AWS IoT Core, pueden enviar mensajes a los AWS servicios y otros dispositivos desde los que pueden actuar. Del mismo modo, AWS los servicios y otros dispositivos pueden procesar información que hace que los mensajes se envíen a tus dispositivos.

Cuando esté listo para AWS IoT Core seguir explorando, pruebe estos tutoriales:

- the section called “Tutorial: Envío de una notificación de Amazon SNS”
- the section called “Almacenamiento de datos de dispositivos en una tabla de DynamoDB”
- the section called “Formatear una notificación mediante una AWS Lambda función”

Tutorial: Uso de la AWS IoT Device SDK para Embedded C

En esta sección, se explica cómo se ejecuta el AWS IoT Device SDK para Embedded C.

Procedimientos de esta sección

- Paso 1: Instalar la AWS IoT Device SDK para Embedded C
- Paso 2: Configurar la aplicación de ejemplo
- Paso 3: Compilar y ejecutar la aplicación de ejemplo

Paso 1: Instalar la AWS IoT Device SDK para Embedded C

AWS IoT Device SDK para Embedded C generalmente se dirige a dispositivos con limitaciones de recursos que requieren un tiempo de ejecución optimizado del lenguaje C. Puede usar el SDK en cualquier sistema operativo y alojarlo en cualquier tipo de procesador (por ejemplo, MCU y MPU). Si dispone de más memoria y recursos de procesamiento, le recomendamos que utilice uno de los SDK de AWS IoT para dispositivos y móviles superiores (por ejemplo, C++, Java, JavaScript y Python).

En general, AWS IoT Device SDK para Embedded C está dirigido a sistemas que utilizan MCU o MPU de bajo rendimiento que ejecutan sistemas operativos integrados. Para el ejemplo de programación de esta sección, asumimos que su dispositivo utiliza Linux.

Example

1. Descarga el AWS IoT Device SDK para Embedded C en tu dispositivo desde GitHub.

   ```
   git clone https://github.com/aws/aws-iot-device-sdk-embedded-c.git --recurse-submodules
   ```

   Esto crea un directorio denominado `aws-iot-device-sdk-embedded-c` en el directorio actual.
2. Vaya hasta ese directorio y consulte la última versión. Observe github.com/aws/aws-iot-device-sdk-embedded-C/tags para ver la etiqueta de la última versión.

   ```
   cd aws-iot-device-sdk-embedded-c
   git checkout latest-release-tag
   ```
3. Instale OpenSSL 1.1.0 o una versión posterior. Las bibliotecas de desarrollo de OpenSSL suelen denominarse «libssl-dev» u «openssl-devel» cuando se instalan mediante un administrador de paquetes.

   ```
   sudo apt-get install libssl-dev
   ```

Paso 2: Configurar la aplicación de ejemplo

El AWS IoT Device SDK para Embedded C contiene aplicaciones de ejemplo que puede probar. Para simplificar, este tutorial utiliza la aplicación `mqtt_demo_mutual_auth`, que ilustra cómo puede conectarse al agente de mensajes de AWS IoT Core, así como suscribirse a temas de MQTT y publicar en ellos.

1. Copie el certificado y la clave privada que ha creado en Cómo empezar con AWS IoT Core los tutoriales en el directorio `build/bin/certificates`.

   > **Note**
   >
   > Los certificados de dispositivo y de entidad de certificación raíz están sujetos a vencimiento o revocación. Si estos certificados caducan o se revocan, debe copiar un nuevo certificado de entidad de certificación o un certificado de clave privada y dispositivo en el dispositivo.

2. Debe configurar el ejemplo con su punto de conexión de AWS IoT Core, su clave privada, su certificado y su certificado de la entidad de certificación raíz personales. Vaya al directorio `aws-iot-device-sdk-embedded-c/demos/mqtt/mqtt_demo_mutual_auth`.

   Si tiene instalada la AWS CLI, puede utilizar este comando para buscar la URL del punto de conexión de su cuenta.

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

   Si no tiene instalada la AWS CLI, abra la consola de AWS IoT. Desde el panel de navegación, elija Manage (Administrar) y, a continuación, Things (Objetos). Elija el objeto IoT para su Raspberry Pi y, a continuación, elija Interactuar. Su punto de conexión se muestra en la sección HTTPS de la página de detalles del objeto.

3. Abra el archivo `demo_config.h` y actualice los valores de los elementos siguientes:

   AWS_IOT_ENDPOINT

   Su punto de conexión personal.

   CLIENT_CERT_PATH

   La ruta del archivo de su certificado, por ejemplo, `certificates/device.pem.crt"`.

   CLIENT_PRIVATE_KEY_PATH

   El nombre de su archivo de clave privada, por ejemplo `certificates/private.pem.key`.

   Por ejemplo:

   ```
   // Get from demo_config.h
   ```

```
// ================================================
#define AWS_IOT_ENDPOINT                "my-endpoint-ats.iot.us-
east-1.amazonaws.com"
#define AWS_MQTT_PORT                   8883
#define CLIENT_IDENTIFIER               "testclient"
#define ROOT_CA_CERT_PATH               "certificates/AmazonRootCA1.crt"
#define CLIENT_CERT_PATH                "certificates/my-device-cert.pem.crt"
#define CLIENT_PRIVATE_KEY_PATH         "certificates/my-device-private-key.pem.key"
// ================================================
```

4. Compruebe si tiene CMake instalado en su dispositivo mediante este comando.

   ```
   cmake --version
   ```

   Si ve la información de la versión del compilador, puede continuar con la siguiente sección.

   Si aparece un error o no ve ninguna información, deberá instalar el paquete cmake con este comando.

   ```
   sudo apt-get install cmake
   ```

   Vuelva a ejecutar el comando cmake --version y confirme que CMake se ha instalado y que está listo para continuar.

5. Compruebe si tiene las herramientas de desarrollo instaladas en su dispositivo mediante este comando.

   ```
   gcc --version
   ```

   Si ve la información de la versión del compilador, puede continuar con la siguiente sección.

   Si aparece un error o no ve la información del compilador, deberá instalar el paquete `build-essential` con este comando.

   ```
   sudo apt-get install build-essential
   ```

   Vuelva a ejecutar el comando gcc --version y confirme que las herramientas de compilación se han instalado y que está listo para continuar.

Paso 3: Compilar y ejecutar la aplicación de ejemplo

Este procedimiento explica cómo generar la aplicación `mqtt_demo_mutual_auth` en el dispositivo y conectarla a la consola de AWS IoT con el AWS IoT Device SDK para Embedded C.

Para ejecutar las aplicaciones de ejemplo de AWS IoT Device SDK para Embedded C

1. Vaya a `aws-iot-device-sdk-embedded-c` y cree un directorio de compilación.

   ```
   mkdir build && cd build
   ```

2. Introduzca el siguiente comando CMake para generar los Makefiles necesarios para la compilación.

   ```
   cmake ..
   ```

3. Escriba el siguiente comando para crear el archivo ejecutable de la aplicación.

   ```
   make
   ```

4. Ejecute la aplicación `mqtt_demo_mutual_auth` con este comando.

   ```
   cd bin
   ./mqtt_demo_mutual_auth
   ```

   Debería ver una salida similar a esta:

```
[INFO] [DEMO] [mqtt_demo_mutual_auth.c:584] Establishing a TLS session to a2zk5tjv9x07ct-ats.iot.us-west-2.amazonaws.com:8883.
[INFO] [DEMO] [mqtt_demo_mutual_auth.c:1264] Creating an MQTT connection to a2zk5tjv9x07ct-ats.iot.us-west-2.amazonaws.com.
[INFO] [MQTT] [core_mqtt.c:855] Packet received. ReceivedBytes=2.
[INFO] [MQTT] [core_mqtt_serializer.c:970] CONNACK session present bit not set.
[INFO] [MQTT] [core_mqtt_serializer.c:912] Connection accepted.
[INFO] [MQTT] [core_mqtt.c:1526] Received MQTT CONNACK successfully from broker.
[INFO] [MQTT] [core_mqtt.c:1792] MQTT connection established with the broker.
[INFO] [DEMO] [mqtt_demo_mutual_auth.c:1033] MQTT connection successfully established with broker.


[INFO] [DEMO] [mqtt_demo_mutual_auth.c:1296] A clean MQTT connection is established. Cleaning up all the stored outgoing publishes.


[INFO] [DEMO] [mqtt_demo_mutual_auth.c:1314] Subscribing to the MQTT topic testclient/example/topic.
[INFO] [DEMO] [mqtt_demo_mutual_auth.c:1097] SUBSCRIBE sent for topic testclient/example/topic to broker.


[INFO] [MQTT] [core_mqtt.c:855] Packet received. ReceivedBytes=3.
[INFO] [DEMO] [mqtt_demo_mutual_auth.c:921] Subscribed to the topic testclient/example/topic. with maximum QoS 1.


[INFO] [DEMO] [mqtt_demo_mutual_auth.c:1358] Sending Publish to the MQTT topic testclient/example/topic.
[INFO] [DEMO] [mqtt_demo_mutual_auth.c:1195] PUBLISH sent for topic testclient/example/topic to broker with packet ID 2.


[INFO] [MQTT] [core_mqtt.c:855] Packet received. ReceivedBytes=2.
[INFO] [MQTT] [core_mqtt.c:1126] Ack packet deserialized with result: MQTTSuccess.
[INFO] [MQTT] [core_mqtt.c:1139] State record updated. New state=MQTTPublishDone.
[INFO] [DEMO] [mqtt_demo_mutual_auth.c:946] PUBACK received for packet id 2.


[INFO] [DEMO] [mqtt_demo_mutual_auth.c:672] Cleaned up outgoing publish packet with packet id 2.


[INFO] [MQTT] [core_mqtt.c:855] Packet received. ReceivedBytes=40.
[INFO] [MQTT] [core_mqtt.c:1015] De-serialized incoming PUBLISH packet: DeserializerResult=MQTTSuccess.
```

El dispositivo está conectado a AWS IoT mediante la AWS IoT Device SDK para Embedded C.

También puede utilizar la consola de AWS IoT para ver los mensajes de MQTT que está publicando la aplicación de ejemplo. Para obtener información sobre cómo utilizar el cliente MQTT en la consola de AWS IoT, consulte the section called “Vea los mensajes MQTT con el cliente AWS IoT MQTT”.

Crear AWS IoT reglas para enrutar los datos del dispositivo a otros servicios

Estos tutoriales te muestran cómo crear y probar AWS IoT reglas utilizando algunas de las acciones de reglas más comunes.

AWS IoT las reglas envían datos desde tus dispositivos a otros AWS servicios. Escuchan mensajes MQTT específicos, formatean los datos de las cargas útiles de los mensajes y envían el resultado a otros servicios AWS .

Le recomendamos que las pruebe en el orden en que se muestran aquí, incluso si su objetivo es crear una regla que utilice una función de Lambda o algo más complejo. Los tutoriales se presentan en orden de básico a complejo. Presentan nuevos conceptos de forma gradual para ayudarlo a

aprender los conceptos que puede usar para crear las acciones de reglas que no tienen un tutorial específico.

> **Note**
>
> AWS IoT las reglas le ayudan a enviar los datos de sus dispositivos de IoT a otros AWS servicios. Sin embargo, para hacerlo correctamente, necesita un conocimiento práctico de los demás servicios a los que desea enviar datos. Si bien estos tutoriales proporcionan la información necesaria para completar las tareas, puede resultarle útil obtener más información sobre los servicios a los que desea enviar datos antes de utilizarlos en su solución. Una explicación detallada de los demás AWS servicios queda fuera del ámbito de estos tutoriales.

Información general del escenario del tutorial

El escenario de estos tutoriales es el de un dispositivo sensor meteorológico que publica sus datos periódicamente. Hay muchos dispositivos sensores de este tipo en este sistema imaginario. Los tutoriales de esta sección, sin embargo, se centran en un único dispositivo mientras muestran cómo se pueden acomodar múltiples sensores.

Los tutoriales de esta sección le muestran cómo usar AWS IoT las reglas para realizar las siguientes tareas con este sistema imaginario de dispositivos de sensores meteorológicos.

- Tutorial: Volver a publicar un mensaje MQTT

  En este tutorial se muestra cómo volver a publicar un mensaje MQTT recibido de los sensores meteorológicos como un mensaje que contiene únicamente el identificador del sensor y el valor de temperatura. Utiliza solo servicios AWS IoT Core y muestra una consulta SQL sencilla y cómo utilizar el cliente MQTT para probar la regla.
- Tutorial: Envío de una notificación de Amazon SNS

  Este tutorial muestra cómo enviar un mensaje SNS cuando un valor de un dispositivo sensor meteorológico supera un valor específico. Se basa en los conceptos presentados en el tutorial anterior y añade cómo trabajar con otro AWS servicio, el Amazon Simple Notification Service (Amazon SNS).

  Si es la primera vez que usa Amazon SNS, consulte sus ejercicios de introducción antes de comenzar este tutorial.

- Tutorial: Almacenamiento de datos de dispositivos en una tabla de DynamoDB

  En este tutorial se muestra cómo almacenar los datos de los dispositivos de sensores meteorológicos en una tabla de base de datos. Utiliza la declaración de consulta de reglas y las plantillas de sustitución para dar formato a los datos del mensaje para el servicio de destino, Amazon DynamoDB.

  Si es la primera vez que usa DynamoDB, consulte sus ejercicios de introducción antes de comenzar este tutorial.

- Tutorial: Formatear una notificación mediante una AWS Lambda función

  En este tutorial se muestra cómo llamar a una función de Lambda para volver a formatear los datos del dispositivo y, a continuación, enviarlos como mensaje de texto. Añade un script de Python y funciones de AWS SDK en una AWS Lambdafunción para formatear con el mensaje los datos de carga útil de los dispositivos con sensores meteorológicos y enviar un mensaje de texto.

  Si es la primera vez que usa Lambda, consulte sus ejercicios de introducción antes de comenzar este tutorial.

AWS IoT descripción general de las reglas

Todos estos tutoriales crean AWS IoT reglas.

Para que una AWS IoT regla envíe los datos de un dispositivo a otro AWS servicio, utiliza:

- Una declaración de consulta de reglas que consta de:
  - Una cláusula SELECT de SQL que selecciona y formatea los datos de la carga del mensaje
  - Un filtro de tema (el objeto FROM de la instrucción de consulta de la regla) que identifica los mensajes que se van a utilizar
  - Una sentencia condicional opcional (una cláusula SQL WHERE) que especifica condiciones concretas sobre las que actuar
- Al menos una acción de regla

Los dispositivos publican mensajes en temas MQTT. El filtro de temas de la instrucción SQL SELECT identifica los temas de MQTT a los que se va a aplicar la regla. Los campos especificados en la sentencia SQL SELECT dan formato a los datos de la carga útil del mensaje MQTT entrante

para que los utilicen las acciones de la regla. Para obtener una lista completa de las acciones de las reglas, consulte Acciones de las reglas de AWS IoT.

Tutoriales en esta sección

- Tutorial: Volver a publicar un mensaje MQTT
- Tutorial: Envío de una notificación de Amazon SNS
- Tutorial: Almacenamiento de datos de dispositivos en una tabla de DynamoDB
- Tutorial: Formatear una notificación mediante una AWS Lambda función

Tutorial: Volver a publicar un mensaje MQTT

En este tutorial se muestra cómo crear una AWS IoT regla que publique un mensaje MQTT cuando se reciba un mensaje MQTT específico. La regla puede modificar la carga útil de los mensajes entrantes antes de que se publique. Esto permite crear mensajes que se adapten a aplicaciones específicas sin necesidad de modificar el dispositivo o su firmware. También puede utilizar el aspecto de filtrado de una regla para publicar mensajes sólo cuando se cumpla una condición específica.

Los mensajes que se vuelven a publicar mediante una regla actúan como los mensajes enviados por cualquier otro AWS IoT dispositivo o cliente. Los dispositivos pueden suscribirse a los mensajes republicados del mismo modo que pueden suscribirse a cualquier otro tema de mensajes de MQTT.

Lo que aprenderá en este tutorial:

- Cómo utilizar consultas y funciones SQL sencillas en una declaración de consulta de reglas
- Cómo usar el cliente MQTT para probar una regla AWS IoT

Para completar este tutorial se necesitan aproximadamente 30 minutos.

En este tutorial, podrá:

- Revise los temas y AWS IoT reglas de MQTT
- Paso 1: Cree una AWS IoT regla para volver a publicar un mensaje MQTT
- Paso 2: Probar su nueva regla
- Paso 3: Revisar los resultados y los siguientes pasos

Antes de empezar este tutorial, asegúrese de que tiene:

- Configurar Cuenta de AWS

  Necesitará su AWS IoT consola Cuenta de AWS y su consola para completar este tutorial.
- Vea los mensajes MQTT con el cliente AWS IoT MQTT revisado

  Asegúrese de poder utilizar el cliente MQTT para suscribirse a un tema y publicar en él. En este procedimiento, utilizará el cliente MQTT para probar la nueva regla.

Revise los temas y AWS IoT reglas de MQTT

Antes de hablar de AWS IoT las reglas, es útil entender el protocolo MQTT. En las soluciones de IoT, el protocolo MQTT ofrece algunas ventajas sobre otros protocolos de comunicación de red, como HTTP, lo que lo convierte en una opción popular para su uso en dispositivos de IoT. En esta sección se analizan los aspectos clave del MQTT aplicables a este tutorial. Para obtener información sobre la comparación entre MQTT y HTTP, consulte Elección de un protocolo de aplicación para la comunicación entre dispositivos.

Protocolo MQTT

El protocolo MQTT utiliza un modelo de publish/subscribe comunicación con su anfitrión. Para enviar datos, los dispositivos publican los mensajes que se identifican por temas en el intermediario de AWS IoT mensajes. Para recibir mensajes del agente de mensajes, los dispositivos se suscriben a los temas que van a recibir enviando filtros de temas en las solicitudes de suscripción al agente de mensajes. El motor de reglas AWS IoT recibe los mensajes MQTT del agente de mensajes.

AWS IoT reglas

AWS IoT las reglas constan de una declaración de consulta de reglas y una o más acciones de regla. Cuando el motor de reglas AWS IoT recibe un mensaje MQTT, estos elementos actúan sobre el mensaje de la siguiente manera.

- Declaración de consulta de reglas

  La sentencia de consulta de la regla describe los temas de MQTT que se van a utilizar, interpreta los datos de la carga útil del mensaje y les da el formato que describe una sentencia SQL similar a las instrucciones utilizadas en las bases de datos SQL más conocidas. El resultado de la sentencia de consulta son los datos que se envían a las acciones de la regla.
- Acción de regla

Cada acción de una regla actúa sobre los datos que resultan de la declaración de consulta de la regla. AWS IoT admite muchas acciones de regla. Sin embargo, en este tutorial se concentrará en la acción de la regla Volver a publicar, que publica el resultado de la declaración de consulta como un mensaje MQTT con un tema específico.

Paso 1: Cree una AWS IoT regla para volver a publicar un mensaje MQTT

La AWS IoT regla que crearás en este tutorial se basa en los temas de `device/`*`device_id`*`/data` MQTT, donde *`device_id`* está el ID del dispositivo que envió el mensaje. Estos temas se describen mediante un filtro de temas como `device/+/data`: donde + es un carácter comodín que coincide con cualquier cadena entre los dos caracteres de barra diagonal.

Cuando la regla recibe un mensaje de un tema coincidente, vuelve a publicar los valores `device_id` y `temperature` como un nuevo mensaje MQTT con el tema `device/data/temp`.

Por ejemplo, la carga útil de un mensaje MQTT con el tema `device/22/data` tiene el siguiente aspecto:

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

La regla toma el valor `temperature` de la carga útil del mensaje, y el `device_id` del tema, y los vuelve a publicar como un mensaje MQTT con el tema `device/data/temp` y una carga útil del mensaje que se parece a esto:

```
{
  "device_id": "22",
  "temperature": 28
}
```

Con esta regla, los dispositivos que solo necesitan el identificador del dispositivo y los datos de temperatura se suscriben al tema `device/data/temp` para recibir únicamente esa información.

Para crear una regla que vuelva a publicar un mensaje MQTT

1. Abre el centro de reglas de la AWS IoT consola.
2. En Reglas, elija Crear y comience a crear su nueva regla.
3. En la parte superior de Crear una regla:

   a. En Nombre, introduzca el nombre de la regla. Para este tutorial, llámela **`republish_temp`**.

      Recuerde que el nombre de una regla debe ser único en su cuenta y región, y no puede tener espacios. Hemos utilizado un carácter de subrayado en este nombre para separar las dos palabras del nombre de la regla.

   b. En Descripción, describa la regla.

      Una descripción significativa le ayuda a recordar lo que hace esta regla y por qué la creó. La descripción puede ser tan larga como sea necesario, por lo que debe ser lo más detallada posible.
4. En Declaración de consulta de la regla de Crear una regla:

   a. En la versión Uso de SQL, seleccione **`2016-03-23`**.

   b. En el cuadro de edición de la declaración de consulta de reglas, introduzca la siguiente declaración:

      ```
      SELECT topic(2) as device_id, temperature FROM 'device/+/data'
      ```

      Esta declaración:

      - Escucha los mensajes de MQTT con un tema que coincida con el filtro de temas `device/+/data`.
      - Selecciona el segundo elemento de la cadena de temas y lo asigna al campo `device_id`.
      - Selecciona el campo `temperature` de valor de la carga útil del mensaje y lo asigna al campo `temperature`.
5. En Establecer una o más acciones:

   a. Para abrir la lista de acciones de regla para esta regla, seleccione Añadir acción.

   b. En Selecciona una acción, selecciona Volver a publicar un mensaje en un AWS IoT tema.

c. En la parte inferior de la lista de acciones, seleccione Configurar acción para abrir la página de configuración de la acción seleccionada.

6. En Acciones de configuración:

   a. En Tema, escriba **`device/data/temp`**. Este es el tema MQTT del mensaje que publicará esta regla.

   b. En Calidad de servicio, elija 0: el mensaje se entrega cero o más veces.

   c. En Elegir o crear un rol para conceder AWS IoT acceso para realizar esta acción:

      i. Seleccione Crear rol. Se abre el cuadro de diálogo Crear un rol de IAM.

      ii. Especifique un nombre que describa el recurso. Para este tutorial, use **`republish_role`**.

      Al crear un rol nuevo, se crean las políticas correctas para realizar la acción de la regla y se asocian al nuevo rol. Si cambia el tema de esta acción de regla o utiliza este rol en otra acción de regla, debe actualizar la política de ese rol para autorizar el nuevo tema o acción. Para actualizar un rol existente, seleccione Actualizar rol en esta sección.

      iii. Seleccione Crear rol para crear el rol y cerrar el cuadro de diálogo.

   d. Seleccione Añadir acción para añadir la acción a la regla y volver a la página Crear una regla.

7. La acción Volver a publicar un mensaje en un AWS IoT tema ahora aparece en Establecer una o más acciones.

   En el icono de la nueva acción, debajo de Volver a publicar un mensaje en un tema AWS IoT , puede ver el tema en el que se publicará la acción de volver a publicar.

   Esta es la única acción de regla que añadirá a esta regla.

8. En Crear una regla, desplácese hasta la parte inferior y seleccione Crear regla para crear la regla y completar este paso.

Paso 2: Probar su nueva regla

Para probar la nueva regla, utilizará el cliente MQTT para publicar y suscribirse a los mensajes MQTT utilizados por esta regla.

Abra el cliente MQTT de la consola AWS IoT en una ventana nueva. Esto le permitirá editar la regla sin perder la configuración de su cliente MQTT. El cliente MQTT no conserva las suscripciones ni los registros de mensajes si lo abandona para ir a otra página de la consola.

Para utilizar el cliente MQTT para probar su regla

1. En el cliente MQTT de la consola de AWS IoT, suscríbase a los temas de entrada, en este caso, `device/+/data`.

   a. En el cliente MQTT, en Suscripciones, seleccione Suscribirse a un tema.

   b. En el tema de suscripción, introduzca el tema del filtro de temas de entrada, **`device/+/data`**.

   c. No cambie el resto de los valores predeterminados de los demás ajustes.

   d. Elija Suscribirse al tema.

   En la columna Suscripciones, aparece **`device/+/data`** en Publicar en un tema.

2. Suscríbase al tema que publicará su regla: `device/data/temp`.

   a. En Suscripciones, elija Suscribirse nuevamente a un tema, y en Tema de suscripción, ingrese el tema del mensaje republicado, **`device/data/temp`**.

   b. No cambie el resto de los valores predeterminados de los demás ajustes.

   c. Elija Suscribirse al tema.

   En la columna Suscripciones, aparece **`device/data/temp`** en device/+/data.

3. Publique un mensaje en el tema de entrada con un identificador de dispositivo específico, **`device/22/data`** No puede publicar en MQTT temas que contengan caracteres comodín.

   a. En el cliente MQTT, en Suscripciones, seleccione Publicar en tema.

   b. En el campo Publicar, introduzca el nombre del tema de entrada,. **`device/22/data`**

   c. Copie los datos de ejemplo que se muestran aquí y, en el cuadro de edición situado debajo del nombre del tema, péguelos.

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
```

```
    "bearing": 255
  }
}
```

d. Para enviar su mensaje MQTT, seleccione Publicar en tema.

4. Revise los mensajes que se enviaron.

   a. En el cliente de MQTT, en Suscripciones, hay un punto verde junto a los dos temas a los que se ha suscrito anteriormente.

      Los puntos verdes indican que se han recibido uno o más mensajes nuevos desde la última vez que los consultó.

   b. En Suscripciones, seleccione device/+/data para comprobar que la carga útil de los mensajes coincide con la que acaba de publicar y tiene este aspecto:

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

   c. En Suscripciones, selecciona device/data/tempcomprobar que la carga útil de los mensajes republicados tenga el siguiente aspecto:

```
{
  "device_id": "22",
  "temperature": 28
}
```

      Observe que el valor `device_id` es una cadena entre comillas y el valor `temperature` es numérico. Esto se debe a que la función `topic()` extrajo la cadena del nombre del tema del mensaje de entrada, mientras que el valor `temperature` utiliza el valor numérico de la carga útil del mensaje de entrada.

      Si desea convertir el valor `device_id` en un valor numérico, sustituya `topic(2)` en la declaración de consulta de reglas por:

```
cast(topic(2) AS DECIMAL)
```

Tenga en cuenta que convertir el valor `topic(2)` en un valor numérico solo funcionará si esa parte del tema contiene solo caracteres numéricos.

5. Si ves que se publicó el mensaje correcto en el device/data/temptema, significa que la regla ha funcionado. Vea qué más puede aprender sobre la acción Regla de republicación en la siguiente sección.

   Si no ves que se ha publicado el mensaje correcto en el dispositivo/+/data o en los device/data/temptemas, consulta las sugerencias de solución de problemas.

Solución de problemas con la regla de volver a publicar mensajes

He aquí algunas cosas que debe comprobar en caso de que no esté viendo los resultados que espera.

- ¿Tiene un mensaje de error

  Si apareció un error al publicar el mensaje de entrada, corrija primero ese error. Los siguientes pasos pueden ayudarle a corregir ese error.
- No ve el mensaje de entrada en el cliente MQTT

  Cada vez que publique su mensaje de entrada en el tema `device/22/data`, dicho mensaje debería aparecer en el cliente MQTT si se ha suscrito al filtro de temas `device/+/data` tal y como se describe en el procedimiento.

  Cosas que debe comprobar

  - Compruebe el filtro de temas al que se ha suscrito

    Si se ha suscrito al tema del mensaje de entrada como se describe en el procedimiento, debería ver una copia del mensaje de entrada cada vez que lo publique.

    Si no ve el mensaje, compruebe el nombre del tema al que se suscribió y compárelo con el tema en el que lo publicó. Los nombres de los temas distinguen entre mayúsculas y minúsculas y el tema al que te suscribiste debe ser idéntico al tema en el que publicaste la carga útil del mensaje.
  - Compruebe la función de publicación de mensajes

En el cliente MQTT, en Suscripciones, elija device/+/data, marque el tema del mensaje de publicación y, a continuación, elija Publicar en tema. Debería ver la carga útil del mensaje en el cuadro de edición situado debajo del tema en la lista de mensajes.

- No ve su mensaje republicado en el cliente MQTT

  Para que su regla funcione, debe tener la política correcta que la autorice a recibir y volver a publicar un mensaje, y debe recibir el mensaje.

  Cosas que debe comprobar

  - Comprueba el Región de AWS de tu cliente MQTT y la regla que has creado

    La consola en la que ejecuta el cliente MQTT debe estar en la misma región AWS que la regla que creó.
  - Compruebe el tema del mensaje de entrada en la declaración de consulta de la regla

    Para que la regla funcione, debe recibir un mensaje con el nombre del tema que coincida con el filtro de tema de la cláusula FROM de la declaración de consulta de la regla.

    Compruebe la ortografía del filtro de temas de la declaración de consulta de reglas con la del tema en el cliente MQTT. Los nombres de los temas distinguen mayúsculas de minúsculas y el tema del mensaje debe coincidir con el filtro de tema de la declaración de consulta de reglas.
  - Compruebe el contenido de la carga útil del mensaje de entrada

    Para que la regla funcione, debe encontrar el campo de datos en la carga útil del mensaje que se declara en la sentencia SELECT.

    Compruebe la ortografía del campo `temperature` en la declaración de consulta de la regla con la de la carga útil del mensaje en el cliente MQTT. Los nombres de los campos distinguen mayúsculas de minúsculas y el campo `temperature` de la declaración de consulta de la regla debe ser idéntico al campo `temperature` de la carga útil del mensaje.

    Asegúrese de que el documento JSON de la carga útil del mensaje tenga el formato correcto. Si el JSON contiene algún error, como la falta de una coma, la regla no podrá leerlo.
  - Compruebe el tema del mensaje que se ha vuelto a publicar en la acción de la regla

    El tema en el que la acción de volver a publicar la regla publica el nuevo mensaje debe coincidir con el tema al que se suscribió en el cliente MQTT.

Abra la regla que creó en la consola y seleccione el tema en el que la acción de la regla volverá a publicar el mensaje.

- Compruebe el rol que utiliza la regla

  La acción de la regla debe tener permiso para recibir el tema original y publicar el tema nuevo.

  Las políticas que autorizan a la regla a recibir datos de los mensajes y volver a publicarlos son específicas de los temas utilizados. Si cambia el tema utilizado para volver a publicar los datos del mensaje, debe actualizar la función de la acción de regla para actualizar su política y que coincida con el tema actual.

  Si sospecha que este es el problema, edite la acción Volver a publicar la regla y cree un nuevo rol. Los nuevos roles creados por la acción de regla reciben las autorizaciones necesarias para realizar estas acciones.

Paso 3: Revisar los resultados y los siguientes pasos

En este tutorial:

- Ha utilizado una consulta SQL sencilla y un par de funciones en una sentencia de consulta de reglas para generar un nuevo mensaje MQTT.
- Creó una regla que volvió a publicar ese nuevo mensaje.
- Utilizó el cliente MQTT para probar la regla. AWS IoT

Siguientes pasos

Después de volver a publicar algunos mensajes con esta regla, intente experimentar con ella para ver cómo los cambios en algunos aspectos del tutorial afectan al mensaje que se ha vuelto a publicar. He aquí algunos ejemplos para empezar.

- Cambie *`device_id`* el tema del mensaje de entrada y observe el efecto en la carga útil del mensaje republicado.
- Cambie los campos seleccionados en la declaración de consulta de la regla y observe el efecto en la carga útil del mensaje republicado.
- Pruebe el siguiente tutorial de esta serie y aprenda cómo Tutorial: Envío de una notificación de Amazon SNS.

La acción Volver a publicar la regla utilizada en este tutorial también puede ayudarle a depurar las sentencias de consulta de reglas. Por ejemplo, puede añadir esta acción a una regla para ver cómo su sentencia de consulta de regla formatea los datos utilizados por sus acciones de regla.

Tutorial: Envío de una notificación de Amazon SNS

En este tutorial se muestra cómo crear una AWS IoT regla que envíe datos de mensajes MQTT a un tema de Amazon SNS para que puedan enviarse como un mensaje de texto SMS.

En este tutorial, creará una regla que envíe datos de mensajes de un sensor meteorológico a todos los suscriptores de un tema SNS de Amazon, siempre que la temperatura supere el valor establecido en la regla. La regla detecta si la temperatura notificada supera el valor establecido por la regla y, a continuación, crea una nueva carga de mensajes que incluye únicamente el identificador del dispositivo, la temperatura notificada y el límite de temperatura que se ha superado. La regla envía la carga útil del mensaje nuevo como un documento JSON a un tema de SNS, que notifica a todos los suscriptores del tema de SNS.

Lo que aprenderá en este tutorial:

- Cómo crear y probar una notificación de Amazon SNS
- Cómo llamar a una notificación de Amazon SNS desde una regla AWS IoT
- Cómo utilizar consultas y funciones SQL sencillas en una declaración de consulta de reglas
- ¿Cómo usar el cliente MQTT para probar una regla AWS IoT

Para completar este tutorial se necesitan aproximadamente 30 minutos.

En este tutorial, podrá:

- Paso 1: crear un tema de Amazon SNS que envía un mensaje de texto SMS
- Paso 2: Crea una AWS IoT regla para enviar el mensaje de texto
- Paso 3: Probar la AWS IoT regla y la notificación de Amazon SNS
- Paso 4: Revisar los resultados y los siguientes pasos

Antes de empezar este tutorial, asegúrese de que tiene:

- Configurar Cuenta de AWS

  Necesitará su AWS IoT consola Cuenta de AWS y su consola para completar este tutorial.

- Vea los mensajes MQTT con el cliente AWS IoT MQTT revisado

  Asegúrese de poder utilizar el cliente MQTT para suscribirse a un tema y publicar en él. En este procedimiento, utilizará el cliente MQTT para probar la nueva regla.
- Revisado el Amazon Simple Notification Service

  Si no ha utilizado Amazon SNS anteriormente, consulte Configuración del acceso a Amazon SNS. Si ya has completado otros AWS IoT tutoriales, ya Cuenta de AWS deberías estar configurado correctamente.

Paso 1: crear un tema de Amazon SNS que envía un mensaje de texto SMS

Este procedimiento explica cómo crear el tema de Amazon SNS al que su sensor meteorológico puede enviar datos de mensajes. A continuación, el tema de Amazon SNS notificará a todos sus suscriptores mediante un mensaje de texto SMS el límite de temperatura que se ha superado.

Paso 1: Crear un tema de Amazon SNS que envía un mensaje de texto SMS

1. Cree un tema de Amazon SNS.

   a. Inicie sesión en la consola de Amazon SNS.
   b. En el panel de navegación izquierdo, elija Topics (Temas).
   c. En la página Temas, elija Crear tema.
   d. En Detalles, elija el tipo Estándar. De forma predeterminada, la consola crea un tema FIFO.
   e. En Nombre, introduzca el nombre del tema de SNS. En este tutorial, escriba **`high_temp_notice`**.
   f. Desplácese hasta el final de la página y elija Crear tema.

      En la consola se abre la página Details (Detalles) del nuevo tema.
2. Crear un una suscripción de Amazon SNS.

   > **Note**
   >
   > Es posible que el número de teléfono que utilice en esta suscripción conlleve cargos por mensajería de texto debido a los mensajes que envíe en este tutorial.

   a. En la página de detalles del tema high_temp_notice, seleccione Crear suscripción.

b. En Crear suscripción, en la sección Detalles, en la lista de protocolos, selecciona SMS.
c. En punto de conexión, introduzca el número de teléfono que puede recibir mensajes de texto. Asegúrese de escribirlo de forma que comience con +, incluya el código de país y área y no incluya ningún otro carácter de puntuación.
d. Seleccione Crear subscripción.

3. Pruebe la notificación de Amazon SNS.

a. En la consola de Amazon SNS, en el panel de navegación izquierdo, seleccione Temas.
b. Para abrir la página de detalles del tema, en Temas, en la lista de temas, elija high_temp_notice.
c. Para abrir la página Publicar mensaje en un tema, en la página de detalles de high_temp_notice, seleccione Publicar mensaje.
d. En Publicar mensaje en un tema, en la sección Cuerpo del mensaje, en Cuerpo del mensaje para enviar al punto de conexión, introduzca un mensaje corto.
e. Desplácese a la parte inferior de la página y seleccione Publicar mensaje.
f. En el teléfono con el número que utilizó anteriormente al crear la suscripción, confirme que ha recibido el mensaje.

Si no ha recibido el mensaje de prueba, vuelva a comprobar el número de teléfono y la configuración de su teléfono.

Asegúrese de poder publicar los mensajes de prueba desde la consola de Amazon SNS antes de continuar con el tutorial.

Paso 2: Crea una AWS IoT regla para enviar el mensaje de texto

La AWS IoT regla que crearás en este tutorial se basa en los temas de `device/`*`device_id`*`/data` MQTT, donde *`device_id`* está el ID del dispositivo que envió el mensaje. Estos temas se describen mediante un filtro de temas como `device/+/data`, donde + es un carácter comodín que coincide con cualquier cadena entre los dos caracteres de barra diagonal. Esta regla también comprueba el valor del campo `temperature` en la carga útil del mensaje.

Cuando la regla recibe un mensaje de un tema coincidente, toma el *`device_id`* del nombre del tema, el valor `temperature` de la carga útil del mensaje y añade un valor constante para el límite que está probando, y envía estos valores como un documento JSON a un tema de notificación de Amazon SNS.

Por ejemplo, un mensaje MQTT del dispositivo sensor meteorológico número 32 utiliza el tema `device/32/data` y tiene una carga útil de mensajes similar a la siguiente:

```
{
  "temperature": 38,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

La declaración de consulta de reglas de la regla toma el valor `temperature` de la carga útil del mensaje, el *`device_id`* del nombre del tema, y añade el valor `max_temperature` constante para enviar una carga útil de mensajes similar a la siguiente al tema de Amazon SNS:

```
{
  "device_id": "32",
  "reported_temperature": 38,
  "max_temperature": 30
}
```

Para crear una AWS IoT regla que detecte un valor de temperatura superior al límite y crear los datos para enviarlos al tema Amazon SNS

1. Abra el centro de reglas de la AWS IoT consola.
2. Si esta es su primera regla, elija Crear o Crear una regla.
3. En Crear una regla:

   a. En Name (Nombre), escriba **`temp_limit_notify`**.

      Recuerda que el nombre de una regla debe ser único en tu región Cuenta de AWS y no puede tener espacios. Hemos utilizado un carácter de subrayado en este nombre para separar las dos palabras del nombre de la regla.

   b. En Descripción, describa la regla.

      Una descripción significativa hace que sea más fácil recordar lo que hace esta regla y por qué la creó. La descripción puede ser tan larga como sea necesario, por lo que debe ser lo más detallada posible.

4. En Declaración de consulta de la regla de Crear una regla:

   a. En la versión Uso de SQL, seleccione 2016-03-23.

   b. En el cuadro de edición de la declaración de consulta de reglas, introduzca la siguiente declaración:

   ```
   SELECT topic(2) as device_id,
       temperature as reported_temperature,
       30 as max_temperature
     FROM 'device/+/data'
     WHERE temperature > 30
   ```

   Esta declaración:

   - Escucha los mensajes MQTT con un tema que coincida con el filtro de temas `device/+/data` y que tengan un valor `temperature` superior a 30.
   - Selecciona el segundo elemento de la cadena de temas y lo asigna al campo `device_id`.
   - Selecciona el campo `temperature` de valor de la carga útil del mensaje y lo asigna al campo `reported_temperature`.
   - Crea un valor constante `30` para representar el valor límite y lo asigna al campo `max_temperature`.

5. Para abrir la lista de acciones de la regla para esta regla, en Establecer una o más acciones, seleccione Añadir acción.

6. En Seleccionar una acción, elija Enviar un mensaje como una notificación push SNS.

7. Para abrir la página de configuración de la acción seleccionada, en la parte inferior de la lista de acciones, seleccione Configurar acción.

8. En Acciones de configuración:

   a. En el objetivo SNS, elija Seleccionar, busque su tema SNS llamado high_temp_notice y elija Seleccionar.

   b. En Formato del mensaje, elija RAW.

   c. En Elegir o crear un rol para conceder AWS IoT acceso para realizar esta acción, selecciona Crear rol.

   d. En Crear un nuevo rol, en Nombre, escriba un nombre único para el nuevo rol. Para este tutorial, escriba **`sns_rule_role`**.

e. Elija Crear rol.

Si va a repetir este tutorial o a reutilizar un rol existente, elija Actualizar rol antes de continuar. Esto actualiza el documento de política del rol para que funcione con el objetivo de SNS.

9. Seleccione Añadir acción y vuelva a la página Crear una regla.

En el icono de la nueva acción, debajo de Enviar un mensaje como notificación push de SNS, puede ver el tema de SNS al que se referirá su regla.

Esta es la única acción de regla que añadirá a esta regla.

10. Para crear la regla y completar este paso, en Crear una regla, desplácese hasta la parte inferior y seleccione Crear regla.

Paso 3: Probar la AWS IoT regla y la notificación de Amazon SNS

Para probar la nueva regla, utilizará el cliente MQTT para publicar y suscribirse a los mensajes MQTT utilizados por esta regla.

Abra el cliente MQTT de la consola AWS IoT en una ventana nueva. Esto le permitirá editar la regla sin perder la configuración de su cliente MQTT. Si abandona el cliente MQTT para ir a otra página de la consola, no conservará ningún registro de suscripciones o mensajes.

Para utilizar el cliente MQTT para probar su regla

1. En el cliente MQTT de la consola de AWS IoT, suscríbase a los temas de entrada, en este caso, `device/+/data`.

   a. En el cliente MQTT, en Suscripciones, seleccione Suscribirse a un tema.
   b. En el tema de suscripción, introduzca el tema del filtro de temas de entrada, **`device/+/data`**.
   c. No cambie el resto de los valores predeterminados de los demás ajustes.
   d. Elija Suscribirse al tema.

   En la columna Suscripciones, aparece **`device/+/data`** en Publicar en un tema.

2. Publique un mensaje en el tema de entrada con un identificador de dispositivo específico, **`device/32/data`** No puede publicar en MQTT temas que contengan caracteres comodín.

   a. En el cliente MQTT, en Suscripciones, seleccione Publicar en tema.

b. En el campo Publicar, introduzca el nombre del tema de entrada,. **`device/32/data`**

c. Copie los datos de ejemplo que se muestran aquí y, en el cuadro de edición situado debajo del nombre del tema, péguelos.

```
{
  "temperature": 38,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

d. Elija Publicar en tema para publicar su mensaje MQTT.

3. Confirme que se envió el mensaje de texto.

a. En el cliente de MQTT, en Suscripciones, hay un punto verde junto al tema al que se ha suscrito anteriormente.

El punto verdes indica que se han recibido uno o más mensajes nuevos desde la última vez que los consultó.

b. En Suscripciones, seleccione device/+/data para comprobar que la carga útil de los mensajes coincide con la que acaba de publicar y tiene este aspecto:

```
{
  "temperature": 38,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

c. Compruebe el teléfono que utilizó para suscribirse al tema SNS y confirme que el contenido de la carga útil del mensaje tiene este aspecto:

```
{"device_id":"32","reported_temperature":38,"max_temperature":30}
```

Observe que el valor `device_id` es una cadena entre comillas y el valor `temperature` es numérico. Esto se debe a que la función `topic()` extrajo la cadena del nombre del tema del mensaje de entrada, mientras que el valor `temperature` utiliza el valor numérico de la carga útil del mensaje de entrada.

Si desea convertir el valor `device_id` en un valor numérico, sustituya `topic(2)` en la declaración de consulta de reglas por:

```
cast(topic(2) AS DECIMAL)
```

Tenga en cuenta que la conversión del valor `topic(2)` a un valor numérico `DECIMAL` solo funcionará si esa parte del tema sólo contiene caracteres numéricos.

4. Intente enviar un mensaje MQTT en el que la temperatura no supere el límite.

   a. En el cliente MQTT, en Suscripciones, seleccione Publicar en tema.

   b. En el campo Publicar, introduzca el nombre del tema de entrada,. **`device/33/data`**

   c. Copie los datos de ejemplo que se muestran aquí y, en el cuadro de edición situado debajo del nombre del tema, péguelos.

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

   d. Para enviar su mensaje MQTT, seleccione Publicar en tema.

Debería ver el mensaje que envió en la suscripción **`device/+/data`**. Sin embargo, como el valor de la temperatura está por debajo de la temperatura máxima en la sentencia de consulta de la regla, no debería recibir un mensaje de texto.

Si no observa el comportamiento correcto, consulte los consejos para la solución de problemas.

Solución de problemas con la regla de volver a publicar mensajes

He aquí algunas cosas que debe comprobar, en caso de que no esté viendo los resultados que espera.

- ¿Tiene un mensaje de error

  Si apareció un error al publicar el mensaje de entrada, corrija primero ese error. Los siguientes pasos pueden ayudarle a corregir ese error.
- No ve el mensaje de entrada en el cliente MQTT

  Cada vez que publique su mensaje de entrada en el tema `device/22/data`, dicho mensaje debería aparecer en el cliente MQTT si se ha suscrito al filtro de temas `device/+/data` tal y como se describe en el procedimiento.

  Cosas que debe comprobar

  - Compruebe el filtro de temas al que se ha suscrito

    Si se ha suscrito al tema del mensaje de entrada como se describe en el procedimiento, debería ver una copia del mensaje de entrada cada vez que lo publique.

    Si no ve el mensaje, compruebe el nombre del tema al que se suscribió y compárelo con el tema en el que lo publicó. Los nombres de los temas distinguen entre mayúsculas y minúsculas y el tema al que te suscribiste debe ser idéntico al tema en el que publicaste la carga útil del mensaje.
  - Compruebe la función de publicación de mensajes

    En el cliente MQTT, en Suscripciones, elija device/+/data, marque el tema del mensaje de publicación y, a continuación, elija Publicar en tema. Debería ver la carga útil del mensaje en el cuadro de edición situado debajo del tema en la lista de mensajes.
- Si no recibe el mensaje SMS:

  Para que su regla funcione, debe tener la política correcta que la autorice a recibir un mensaje y enviar una notificación SNS, y debe recibir el mensaje.

  Cosas que debe comprobar

  - Compruebe el Región de AWS de su cliente MQTT y la regla que creó

La consola en la que ejecuta el cliente MQTT debe estar en la misma región AWS que la regla que creó.

- Compruebe que el valor de temperatura de la carga útil del mensaje supera el umbral de prueba

  Si el valor de temperatura es inferior o igual a 30, tal como se define en la sentencia de consulta de la regla, la regla no realizará ninguna de sus acciones.

- Compruebe el tema del mensaje de entrada en la declaración de consulta de la regla

  Para que la regla funcione, debe recibir un mensaje con el nombre del tema que coincida con el filtro de tema de la cláusula FROM de la declaración de consulta de la regla.

  Compruebe la ortografía del filtro de temas de la declaración de consulta de reglas con la del tema en el cliente MQTT. Los nombres de los temas distinguen mayúsculas de minúsculas y el tema del mensaje debe coincidir con el filtro de tema de la declaración de consulta de reglas.

- Compruebe el contenido de la carga útil del mensaje de entrada

  Para que la regla funcione, debe encontrar el campo de datos en la carga útil del mensaje que se declara en la sentencia SELECT.

  Compruebe la ortografía del campo `temperature` en la declaración de consulta de la regla con la de la carga útil del mensaje en el cliente MQTT. Los nombres de los campos distinguen mayúsculas de minúsculas y el campo `temperature` de la declaración de consulta de la regla debe ser idéntico al campo `temperature` de la carga útil del mensaje.

  Asegúrese de que el documento JSON de la carga útil del mensaje tenga el formato correcto. Si el JSON contiene algún error, como la falta de una coma, la regla no podrá leerlo.

- Compruebe el tema del mensaje que se ha vuelto a publicar en la acción de la regla

  El tema en el que la acción de volver a publicar la regla publica el nuevo mensaje debe coincidir con el tema al que se suscribió en el cliente MQTT.

  Abra la regla que creó en la consola y seleccione el tema en el que la acción de la regla volverá a publicar el mensaje.

- Compruebe el rol que utiliza la regla

  La acción de la regla debe tener permiso para recibir el tema original y publicar el tema nuevo.

Las políticas que autorizan a la regla a recibir datos de los mensajes y volver a publicarlos son específicas de los temas utilizados. Si cambia el tema utilizado para volver a publicar los datos del mensaje, debe actualizar la función de la acción de regla para actualizar su política y que coincida con el tema actual.

Si sospecha que este es el problema, edite la acción Volver a publicar la regla y cree un nuevo rol. Los nuevos roles creados por la acción de regla reciben las autorizaciones necesarias para realizar estas acciones.

Paso 4: Revisar los resultados y los siguientes pasos

En este tutorial:

- Creó y probó una suscripción y un tema de notificación de Amazon SNS.
- Utilizó una consulta SQL sencilla y funciones en una declaración de consulta de reglas para crear un mensaje nuevo para su notificación.
- Creaste una AWS IoT regla para enviar una notificación de Amazon SNS que utilizaba tu carga de mensajes personalizada.
- Utilizó el cliente MQTT para probar la regla. AWS IoT

Siguientes pasos

Después de enviar unos cuantos mensajes de texto con esta regla, pruebe a experimentar con ella para ver cómo el cambio de algunos aspectos del tutorial afecta al mensaje y al momento en que se envía. He aquí algunos ejemplos para empezar.

- Cambie *`device_id`* el tema del mensaje de entrada y observe el efecto en el contenido del mensaje de texto.
- Cambie los campos seleccionados en la sentencia de consulta de la regla y observe el efecto en el contenido del mensaje de texto.
- Cambie la prueba en la declaración de consulta de reglas para probar una temperatura mínima en lugar de una temperatura máxima. ¡Recuerde cambiar el nombre de `max_temperature`!
- Añada una acción de regla de republicación para enviar un mensaje MQTT cuando se envíe una notificación SNS.
- Pruebe el siguiente tutorial de esta serie y aprenda cómo Tutorial: Almacenamiento de datos de dispositivos en una tabla de DynamoDB.

Tutorial: Almacenamiento de datos de dispositivos en una tabla de DynamoDB

En este tutorial se muestra cómo crear una AWS IoT regla que envíe datos de mensajes a una tabla de DynamoDB.

En este tutorial, va a crear una regla que envía datos de mensajes desde un dispositivo de sensor meteorológico imaginario a una tabla de DynamoDB. La regla formatea los datos de muchos sensores meteorológicos de forma que se puedan añadir a una sola tabla de base de datos.

Lo que aprenderá en este tutorial:

- Cómo crear una tabla de DynamoDB
- Cómo enviar datos de mensajes a una tabla de DynamoDB desde una regla AWS IoT
- ¿Cómo utilizar las plantillas de sustitución en una regla AWS IoT
- Cómo utilizar consultas y funciones SQL sencillas en una declaración de consulta de reglas
- ¿Cómo usar el cliente MQTT para probar una regla AWS IoT

Para completar este tutorial se necesitan aproximadamente 30 minutos.

En este tutorial, podrá:

- Paso 1: Crear la tabla DynamoDB para este tutorial
- Paso 2: Crear una AWS IoT regla para enviar datos a la tabla de DynamoDB
- Paso 3: Probar la AWS IoT regla y la tabla de DynamoDB
- Paso 4: Revisar los resultados y los siguientes pasos

Antes de empezar este tutorial, asegúrese de que tiene:

- Configurar Cuenta de AWS

  Necesitará su AWS IoT consola Cuenta de AWS y su consola para completar este tutorial.
- Vea los mensajes MQTT con el cliente AWS IoT MQTT revisado

  Asegúrese de poder utilizar el cliente MQTT para suscribirse a un tema y publicar en él. En este procedimiento, utilizará el cliente MQTT para probar la nueva regla.
- Revisó la descripción general de Amazon DynamoDB

Si no ha utilizado DynamoDB anteriormente, consulte Introducción a DynamoDB para familiarizarse con los conceptos básicos y las operaciones de DynamoDB.

Paso 1: Crear la tabla DynamoDB para este tutorial

En este tutorial, creará una tabla de DynamoDB con estos atributos para registrar los datos de los dispositivos sensores meteorológicos imaginarios:

- `sample_time` es una clave principal y describe el momento en que se grabó la muestra.
- `device_id` es una clave de clasificación y describe el dispositivo que proporcionó la muestra
- `device_data` son los datos recibidos del dispositivo y formateados según la declaración de consulta de la regla

Para crear la tabla DynamoDB para este tutorial

1. Abra la consola DynamoDB y elija Crear tabla.
2. En Crear tabla:
   a. En Nombre de tabla, introduzca el nombre de la tabla: **`wx_data`**.
   b. En la clave de partición, introduzca **`sample_time`** y, en la lista de opciones situada junto al campo, elija **`Number`**.
   c. En la clave de clasificación, introduzca **`device_id`** y, en la lista de opciones situada junto al campo, elija **`Number`**.
   d. En la parte inferior de la página, elija Crear.

Definirá `device_data` más adelante, cuando configure la acción de la regla de DynamoDB.

Paso 2: Crear una AWS IoT regla para enviar datos a la tabla de DynamoDB

En este paso, utilizará la sentencia de consulta de reglas para formatear los datos de los dispositivos sensores meteorológicos imaginarios y escribirlos en la tabla de la base de datos.

Un ejemplo de carga útil de un mensaje recibido desde un dispositivo sensor meteorológico tiene el siguiente aspecto:

```
{
  "temperature": 28,
```

```
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

Para la entrada de la base de datos, utilizará la sentencia de consulta de reglas para aplanar la estructura de la carga útil del mensaje de forma similar a la siguiente:

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind_velocity": 22,
  "wind_bearing": 255
}
```

En esta regla, también utilizarás un par de Plantillas de sustitución. Las plantillas de sustitución son expresiones que permiten insertar valores dinámicos a partir de funciones y datos de mensajes.

Para crear la AWS IoT regla para enviar datos a la tabla de DynamoDB

1. Abra el centro de reglas de la consola de AWS IoT. O bien, puede abrir la AWS IoT página de inicio dentro de la Consola de administración de AWS y navegar hasta Enrutamiento de mensajes > Reglas.
2. Para empezar a crear su nueva regla en Reglas, seleccione Crear regla.
3. En Propiedades de la regla:

   a. En Nombre de la regla, escriba **`wx_data_ddb`**.

      Recuerda que el nombre de una regla debe ser único en tu región Cuenta de AWS y no puede tener espacios. Hemos utilizado un carácter de subrayado en este nombre para separar las dos palabras del nombre de la regla.

   b. En Descripción de la regla, describa la regla.

      Una descripción significativa hace que sea más fácil recordar lo que hace esta regla y por qué la creó. La descripción puede ser tan larga como sea necesario, por lo que debe ser lo más detallada posible.

4. Elija Siguiente para continuar.
5. En declaración SQL:

   a. En la versión de SQL, seleccione **2016-03-23**.

   b. En el cuadro de edición de la Declaración de SQL, introduzca la siguiente declaración:

   ```
   SELECT temperature, humidity, barometer,
     wind.velocity as wind_velocity,
     wind.bearing as wind_bearing,
   FROM 'device/+/data'
   ```

   Esta declaración:

   - Escucha los mensajes de MQTT con un tema que coincida con el filtro de temas `device/+/data`.
   - Formatea los elementos del atributo `wind` como atributos individuales.
   - Pasa los atributos `temperature`, `humidity` y `barometer` sin cambios.
6. Elija Siguiente para continuar.
7. En Acciones de reglas:

   a. Para abrir la lista de acciones de la regla para esta regla, en Acción 1, seleccione **DynamoDB**.

   > **Note**
   >
   > Asegúrese de elegir DynamoDB y no DBv2 Dynamo como acción de regla.

   b. En Nombre de tabla, elija el nombre de la tabla de DynamoDB que creó en un paso anterior:. **wx_data**

   Los campos Tipo de clave de partición y Tipo de clave de clasificación se rellenan con los valores de la tabla de DynamoDB.

   c. En Partition key (Clave de partición), escriba **sample_time**.

   d. En Partition key value (Valor de clave de partición), escriba **${timestamp()}**.

   Esta es la primera de las Plantillas de sustitución que utilizará en esta regla. En lugar de usar un valor de la carga útil del mensaje, usará el valor devuelto por la función de marca de

tiempo. Para saber más, consulte timestamp en la Guía del desarrollador de AWS IoT Core .

e. En Clave de clasificación, escriba **`device_id`**.

f. En Sort key value (Valor de clave de clasificación), escriba **`${cast(topic(2) AS DECIMAL)}`**.

Esta es la segunda de las Plantillas de sustitución que utilizará en esta regla. Inserta el valor del segundo elemento en el nombre del tema, que es el identificador del dispositivo, y luego lo convierte en un valor DECIMAL para que coincida con el formato numérico de la clave. Para saber más sobre los temas, consulte el tema en la Guía del desarrollador de AWS IoT Core . O para saber más sobre el reparto, consulte reparto en la Guía del desarrollador de AWS IoT Core .

g. En Write message data to this column (Escribir datos del mensaje en esta columna) introduzca **`device_data`**.

Esto creará la columna `device_data` en la tabla de DynamoDB.

h. Deje Operation (Operación) en blanco.

i. En el rol de IAM, seleccione Crear nuevo rol.

j. En el cuadro de diálogo Crear rol, en Nombre del rol, escriba wx_ddb_role. Esta nueva función contendrá automáticamente una política con el prefijo aws-iot-rule "» que permitirá a la **`wx_data_ddb`** regla enviar datos a la tabla de **`wx_data`** DynamoDB que haya creado.

k. En Rol de IAM, seleccione **`wx_ddb_role`**.

l. En la parte inferior de la página, elija Siguiente.

8. En la parte inferior de la página de revisión y creación, seleccione Crear para crear la regla.

Paso 3: Probar la AWS IoT regla y la tabla de DynamoDB

Para probar la nueva regla, utilizará el cliente MQTT para publicar y suscribirse a los mensajes MQTT utilizados en esta prueba.

Abra el cliente MQTT de la consola AWS IoT en una ventana nueva. Esto le permitirá editar la regla sin perder la configuración de su cliente MQTT. El cliente MQTT no conserva las suscripciones ni los registros de mensajes si lo abandona para ir a otra página de la consola. También querrá abrir una ventana de consola independiente en el centro de tablas de DynamoDB de AWS IoT la consola para ver las nuevas entradas que envía la regla.

Para utilizar el cliente MQTT para probar su regla

1. En el cliente MQTT AWS IoT de la consola de `device/+/data`, suscríbase al tema de entrada, .

   a. En el Cliente de MQTT, elija Suscribirse a un tema.

   b. En el filtro de temas, introduzca el tema del filtro de temas de entrada, **`device/+/data`**.

   c. Seleccione Suscribirse.

2. Ahora, publique un mensaje en el tema de entrada con un identificador de dispositivo específico, **`device/22/data`** No puede publicar en MQTT temas que contengan caracteres comodín.

   a. En el cliente MQTT, elija Publicar en un tema.

   b. Para Nombre del tema, introduzca el nombre del tema de entrada, **`device/22/data`**.

   c. Para la carga útil de mensajes, introduzca los siguientes datos de ejemplo.

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

   d. Para publicar el mensaje MQTT, seleccione Publicar.

   e. Ahora, en el Cliente de MQTT, elija Suscribirse a un tema. En la columna Suscribirse, elija la suscripción **`device/+/data`**. Confirme que los datos de muestra del paso anterior aparecen allí.

3. Compruebe la fila de la tabla de DynamoDB que creó la regla.

   a. En el centro de tablas de DynamoDB de AWS IoT la consola, elija wx_data y, a continuación, elija la pestaña Elementos.

   Si ya está en la pestaña Elementos, puede que necesite actualizar la pantalla seleccionando el icono de actualización situado en la esquina superior derecha del encabezado de la tabla.

   b. Observa que los valores de sample_time de la tabla son enlaces y abre uno. Si acaba de enviar su primer mensaje, será el único de la lista.

Este enlace muestra todos los datos de esa fila de la tabla.

c. Amplíe la entrada device_data para ver los datos resultantes de la declaración de consulta de la regla.

d. Explore las diferentes representaciones de los datos que están disponibles en esta pantalla. También puede editar los datos en esta pantalla.

e. Cuando haya terminado de revisar esta fila de datos, para guardar los cambios que haya realizado, elija Guardar o, para salir sin guardar ningún cambio, elija Cancelar.

Si no observa el comportamiento correcto, consulte los consejos para la solución de problemas.

Solución de problemas de la regla de DynamoDB

He aquí algunas cosas que debe comprobar en caso de que no esté viendo los resultados que espera.

- ¿Tiene un mensaje de error

  Si apareció un error al publicar el mensaje de entrada, corrija primero ese error. Los siguientes pasos pueden ayudarle a corregir ese error.

- No ve el mensaje de entrada en el cliente MQTT

  Cada vez que publique su mensaje de entrada en el tema `device/22/data`, dicho mensaje debería aparecer en el cliente MQTT si se ha suscrito al filtro de temas `device/+/data` tal y como se describe en el procedimiento.

  Cosas que debe comprobar

  - Compruebe el filtro de temas al que se ha suscrito

    Si se ha suscrito al tema del mensaje de entrada como se describe en el procedimiento, debería ver una copia del mensaje de entrada cada vez que lo publique.

    Si no ve el mensaje, compruebe el nombre del tema al que se suscribió y compárelo con el tema en el que lo publicó. Los nombres de los temas distinguen entre mayúsculas y minúsculas y el tema al que te suscribiste debe ser idéntico al tema en el que publicaste la carga útil del mensaje.

  - Compruebe la función de publicación de mensajes

En el cliente MQTT, en Suscripciones, elija device/+/data, marque el tema del mensaje de publicación y, a continuación, elija Publicar en tema. Debería ver la carga útil del mensaje en el cuadro de edición situado debajo del tema en la lista de mensajes.

- No ve sus datos en la tabla de DynamoDB

La primera cosa que debe hacer es actualizar la pantalla seleccionando el icono de actualización situado en la esquina superior derecha del encabezado de la tabla. Si eso no muestra los datos que busca, compruebe lo siguiente.

Cosas que debe comprobar

- Compruebe el Región de AWS de su cliente MQTT y la regla que creó

  La consola en la que ejecuta el cliente MQTT debe estar en la misma región AWS que la regla que creó.
- Compruebe el tema del mensaje de entrada en la declaración de consulta de la regla

  Para que la regla funcione, debe recibir un mensaje con el nombre del tema que coincida con el filtro de tema de la cláusula FROM de la declaración de consulta de la regla.

  Compruebe la ortografía del filtro de temas de la declaración de consulta de reglas con la del tema en el cliente MQTT. Los nombres de los temas distinguen mayúsculas de minúsculas y el tema del mensaje debe coincidir con el filtro de tema de la declaración de consulta de reglas.
- Compruebe el contenido de la carga útil del mensaje de entrada

  Para que la regla funcione, debe encontrar el campo de datos en la carga útil del mensaje que se declara en la sentencia SELECT.

  Compruebe la ortografía del campo `temperature` en la declaración de consulta de la regla con la de la carga útil del mensaje en el cliente MQTT. Los nombres de los campos distinguen mayúsculas de minúsculas y el campo `temperature` de la declaración de consulta de la regla debe ser idéntico al campo `temperature` de la carga útil del mensaje.

  Asegúrese de que el documento JSON de la carga útil del mensaje tenga el formato correcto. Si el JSON contiene algún error, como la falta de una coma, la regla no podrá leerlo.
- Compruebe los nombres de las claves y los campos utilizados en la acción de la regla

  Los nombres de campo utilizados en la regla de tema deben coincidir con los que se encuentran en la carga útil del mensaje JSON del mensaje publicado.

Abra la regla que creó en la consola y compruebe los nombres de los campos de la configuración de acciones de la regla con los utilizados en el cliente MQTT.

- Compruebe el rol que utiliza la regla

  La acción de la regla debe tener permiso para recibir el tema original y publicar el tema nuevo.

  Las políticas que autorizan a la regla a recibir datos de mensajes y actualizar la tabla de DynamoDB son específicas de los temas utilizados. Si cambia el tema o el nombre de la tabla de DynamoDB que utiliza la regla, debe actualizar el rol de la acción de la regla para que su política coincida.

  Si sospecha que éste es el problema, edite la acción de la regla y cree un nuevo rol. Los nuevos roles creados por la acción de regla reciben las autorizaciones necesarias para realizar estas acciones.

Paso 4: Revisar los resultados y los siguientes pasos

Después de enviar algunos mensajes a la tabla de DynamoDB con esta regla, pruebe a experimentar con ella para ver cómo los cambios en algunos aspectos del tutorial afectan a los datos escritos en la tabla. He aquí algunos ejemplos para empezar.

- Cambie *`device_id`* el tema del mensaje de entrada y observe el efecto en los datos. Puede usarlo para simular la recepción de datos de varios sensores meteorológicos.
- Cambie los campos seleccionados en la sentencia de consulta de reglas y observe el efecto sobre los datos. Puede usar esto para filtrar los datos almacenados en la tabla.
- Agregue una acción de regla de republicación para enviar un mensaje MQTT por cada fila agregada a la tabla. Puede utilizarla para la depuración.

Una vez que haya completado este tutorial, consulte the section called “Formatear una notificación mediante una AWS Lambda función”.

Tutorial: Formatear una notificación mediante una AWS Lambda función

En este tutorial se muestra cómo enviar los datos de los mensajes MQTT a una AWS Lambda acción para formatearlos y enviarlos a otro AWS servicio. En este tutorial, la AWS Lambda acción utiliza el AWS SDK para enviar el mensaje formateado al tema de Amazon SNS que creó en el tutorial sobre cómo hacerlo. the section called “Tutorial: Envío de una notificación de Amazon SNS”

En el tutorial sobre cómo the section called “Tutorial: Envío de una notificación de Amazon SNS”, el documento JSON resultante de la declaración de consulta de la regla se envió como cuerpo del mensaje de texto. El resultado fue un mensaje de texto parecido al siguiente ejemplo:

```
{"device_id":"32","reported_temperature":38,"max_temperature":30}
```

En este tutorial, utilizarás una acción de AWS Lambda regla para llamar a una AWS Lambda función que formatea los datos de la declaración de consulta de la regla en un formato más sencillo, como en este ejemplo:

```
Device 32 reports a temperature of 38, which exceeds the limit of 30.
```

La AWS Lambda función que crearás en este tutorial formatea la cadena del mensaje con los datos de la declaración de consulta de la regla y llama a la función de publicación de SNS del AWS SDK para crear la notificación.

Lo que aprenderá en este tutorial:

- ¿Cómo crear y probar una función? AWS Lambda
- Cómo utilizar el AWS SDK en una AWS Lambda función para publicar una notificación de Amazon SNS
- Cómo utilizar consultas y funciones SQL sencillas en una declaración de consulta de reglas
- ¿Cómo usar el cliente MQTT para probar una regla AWS IoT

Para completar este tutorial se necesitan aproximadamente 45 minutos.

En este tutorial, podrá:

- Paso 1: Cree una AWS Lambda función que envíe un mensaje de texto
- Paso 2: Crea una AWS IoT regla con una AWS Lambda acción de regla
- Paso 3: Pruebe la AWS IoT regla y AWS Lambda la acción de la regla
- Paso 4: Revisar los resultados y los siguientes pasos

Antes de empezar este tutorial, asegúrese de que tiene:

- Configurar Cuenta de AWS

  Necesitará su AWS IoT consola Cuenta de AWS y su consola para completar este tutorial.

- Vea los mensajes MQTT con el cliente AWS IoT MQTT revisado

  Asegúrese de poder utilizar el cliente MQTT para suscribirse a un tema y publicar en él. En este procedimiento, utilizará el cliente MQTT para probar la nueva regla.

- Completó los demás tutoriales de reglas de esta sección

  Este tutorial requiere el tema de notificaciones de SNS que creó en el tutorial sobre cómo the section called “Tutorial: Envío de una notificación de Amazon SNS”. También asume que ha completado los otros tutoriales relacionados con las reglas de esta sección.

- Ha revisado la descripción general de AWS Lambda

  Si no lo ha utilizado AWS Lambda antes, consulte AWS LambdaIntroducción a Lambda para conocer sus términos y conceptos.

Paso 1: Cree una AWS Lambda función que envíe un mensaje de texto

La AWS Lambda función de este tutorial recibe el resultado de la declaración de consulta de reglas, inserta los elementos en una cadena de texto y envía la cadena resultante a Amazon SNS como mensaje en una notificación.

A diferencia del tutorial sobre cómo the section called “Tutorial: Envío de una notificación de Amazon SNS” hacerlo, que utilizaba una acción de AWS IoT regla para enviar la notificación, este tutorial envía la notificación desde la función Lambda mediante una función del AWS SDK. Sin embargo, el tema de notificación de Amazon SNS que se utiliza en este tutorial es el mismo que utilizó en el tutorial sobre cómo the section called “Tutorial: Envío de una notificación de Amazon SNS”.

Para crear una AWS Lambda función que envíe un mensaje de texto

1. Cree una AWS Lambda función nueva.

   a. En la consola de AWS Lambda, elija Create function (Crear función).

   b. En Crear función, seleccione Usar un esquema.

      Busque y seleccione el esquema **`hello-world-python`** y, a continuación, elija Configurar

   c. En Información básica:

      i. En Nombre de la función, escriba el nombre de esta función, **`format-high-temp-notification`**.

ii. En Función de ejecución, elija Crear una nueva función a partir de plantillas de AWS políticas.

iii. En Nombre del rol, introduzca el nombre del nuevo rol, **`format-high-temp-notification-role`**.

iv. En Plantillas de políticas (opcional), busque y seleccione la política de publicación de Amazon SNS.

v. Seleccione Creación de función.

2. Modifique el código del esquema para formatear y enviar una notificación de Amazon SNS.

a. Tras crear la función, debería ver la página de format-high-temp-notificationdetalles. Si no lo hace, ábrala desde la página de Funciones de Lambda.

b. En la página de format-high-temp-notificationdetalles, selecciona la pestaña Configuración y desplázate hasta el panel de códigos de función.

c. En la ventana Código de función, en el panel Entorno, elija el archivo Python, `lambda_function.py`.

d. En la ventana de códigos de funciones, elimine todo el código original del programa del esquema y sustitúyalo por este código.

```
import boto3
#
#   expects event parameter to contain:
#   {
#       "device_id": "32",
#       "reported_temperature": 38,
#       "max_temperature": 30,
#       "notify_topic_arn": "arn:aws:sns:us-
east-1:57EXAMPLE833:high_temp_notice"
#   }
#
#   sends a plain text string to be used in a text message
#
#      "Device {0} reports a temperature of {1}, which exceeds the limit of
 {2}."
#
#   where:
#       {0} is the device_id value
#       {1} is the reported_temperature value
#       {2} is the max_temperature value
#
```

```
def lambda_handler(event, context):

    # Create an SNS client to send notification
    sns = boto3.client('sns')

    # Format text message from data
    message_text = "Device {0} reports a temperature of {1}, which exceeds the
 limit of {2}.".format(
            str(event['device_id']),
            str(event['reported_temperature']),
            str(event['max_temperature'])
        )

    # Publish the formatted message
    response = sns.publish(
            TopicArn = event['notify_topic_arn'],
            Message = message_text
        )

    return response
```

e. Elija Implementar.

3. En una ventana nueva, busque el nombre de recurso de Amazon (ARN) del tema de Amazon SNS del tutorial sobre cómo the section called “Tutorial: Envío de una notificación de Amazon SNS”

   a. En una nueva ventana, abra la página Temas de la consola Amazon SNS.

   b. En la página de temas, busque el tema de notificación high_temp_notice en la lista de temas de Amazon SNS.

   c. Busque el ARN del tema de notificación high_temp_notice para usarlo en el siguiente paso.

4. Cree un caso de prueba para la función de Lambda.

   a. En la página Funciones Lambda de la consola, en la página de format-high-temp-notificationdetalles, elija Seleccionar un evento de prueba en la esquina superior derecha de la página (aunque parezca deshabilitado) y, a continuación, elija Configurar eventos de prueba.

   b. En Configurar evento de prueba, seleccione Crear nuevo evento de prueba.

   c. En Nombre del evento, escriba **`SampleRuleOutput`**.

d. En el editor de JSON, debajo del nombre del evento, pegue este ejemplo de documento JSON. Este es un ejemplo de lo que la AWS IoT regla enviará a la función Lambda.

```
{
  "device_id": "32",
  "reported_temperature": 38,
  "max_temperature": 30,
  "notify_topic_arn": "arn:aws:sns:us-east-1:57EXAMPLE833:high_temp_notice"
}
```

e. Consulte la ventana que contiene el ARN del tema de notificación high_temp_notice y copie el valor del ARN.

f. Sustituya el valor `notify_topic_arn` del editor JSON por el ARN del tema de notificaciones.

   Mantenga esta ventana abierta para poder volver a utilizar este valor de ARN al crear la regla AWS IoT .

g. Seleccione Crear.

5. Pruebe la función con datos de ejemplo.

   a. En la página de format-high-temp-notificationdetalles, en la esquina superior derecha de la página, confirme que SampleRuleOutputaparece junto al botón Probar. Si no es así, selecciónelo de la lista de eventos de prueba disponibles.

   b. Para enviar el ejemplo de mensaje de salida de la regla a su función, elija Probar.

Si tanto la función como la notificación han funcionado, recibirá un mensaje de texto en el teléfono suscrito a la notificación.

Si no recibió un mensaje de texto en el teléfono, verifique el resultado de la operación. En el panel de códigos de función, en la pestaña Resultado de la ejecución, revise la respuesta para encontrar cualquier error que se haya producido. No continúe con el siguiente paso hasta que la función pueda enviar la notificación a su teléfono.

Paso 2: Crea una AWS IoT regla con una AWS Lambda acción de regla

En este paso, utilizará la sentencia de consulta de reglas para formatear los datos del dispositivo sensor meteorológico imaginario y enviarlos a una función de Lambda, que formateará y enviará un mensaje de texto.

Una muestra de la carga útil de un mensaje recibido de los dispositivos meteorológicos tiene el siguiente aspecto:

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

En esta regla, utilizará la sentencia de consulta de reglas para crear una carga de mensajes para la función de Lambda con el siguiente aspecto:

```
{
  "device_id": "32",
  "reported_temperature": 38,
  "max_temperature": 30,
  "notify_topic_arn": "arn:aws:sns:us-east-1:57EXAMPLE833:high_temp_notice"
}
```

Contiene toda la información que la función de Lambda necesita para formatear y enviar el mensaje de texto correcto.

Para crear la AWS IoT regla para llamar a una función Lambda

1. Abra el centro de reglas de la AWS IoT consola.
2. Para empezar a crear su nueva regla en Reglas, seleccione Crear.
3. En la parte superior de Crear una regla:

    a. En Nombre, introduzca el nombre de la regla, **`wx_friendly_text`**

    Recuerda que el nombre de una regla debe ser único en tu región Cuenta de AWS y no puede tener espacios. Hemos utilizado un carácter de subrayado en este nombre para separar las dos palabras del nombre de la regla.

    b. En Descripción, describa la regla.

Una descripción significativa hace que sea más fácil recordar lo que hace esta regla y por qué la creó. La descripción puede ser tan larga como sea necesario, por lo que debe ser lo más detallada posible.

4. En Declaración de consulta de la regla de Crear una regla:

   a. En la versión Uso de SQL, seleccione **2016-03-23**.

   b. En el cuadro de edición de la declaración de consulta de reglas, introduzca la siguiente declaración:

```
SELECT
  cast(topic(2) AS DECIMAL) as device_id,
  temperature as reported_temperature,
  30 as max_temperature,
  'arn:aws:sns:us-east-1:57EXAMPLE833:high_temp_notice' as notify_topic_arn
FROM 'device/+/data' WHERE temperature > 30
```

   Esta declaración:

   - Escucha los mensajes MQTT con un tema que coincida con el filtro de temas `device/+/data` y que tengan un valor `temperature` superior a 30.
   - Selecciona el segundo elemento de la cadena de temas, lo convierte en un número decimal y lo asigna al campo `device_id`.
   - Selecciona el valor del campo `temperature`de la carga útil del mensaje y lo asigna al campo `reported_temperature`.
   - Crea un valor constante, `30`, para representar el valor límite y lo asigna al campo `max_temperature`.
   - Crea un valor constante para el campo `notify_topic_arn`.

   c. Consulte la ventana que contiene el ARN del tema de notificación high_temp_notice y copie el valor del ARN.

   d. Sustituya el valor del ARN (*arn:aws:sns:us-east-1:57EXAMPLE833:high_temp_notice*) en el editor de sentencias de consulta de reglas por el ARN del tema de notificación.

5. En Establecer una o más acciones:

   a. Para abrir la lista de acciones de regla para esta regla, seleccione Añadir acción.

   b. En Seleccionar una acción, elija Enviar un mensaje a una función de Lambda.

c. Para abrir la página de configuración de la acción seleccionada, en la parte inferior de la lista de acciones, seleccione Configurar acción.

6. En Acciones de configuración:

   a. En Nombre de la función, elija Seleccionar.
   b. Elija format-high-temp-notification.
   c. En la parte inferior de Configurar acción, seleccione Añadir acción.
   d. Para crear la regla, en la parte inferior de Crear una regla, seleccione Crear regla.

Paso 3: Pruebe la AWS IoT regla y AWS Lambda la acción de la regla

Para probar la nueva regla, utilizará el cliente MQTT para publicar y suscribirse a los mensajes MQTT utilizados por esta regla.

Abra el cliente MQTT de la consola AWS IoT en una ventana nueva. Ahora puede editar la regla sin perder la configuración de su cliente MQTT. Si abandona el cliente MQTT para ir a otra página de la consola, perderá sus suscripciones o los registros de mensajes.

Para utilizar el cliente MQTT para probar su regla

1. En el cliente MQTT de la consola de AWS IoT, suscríbase a los temas de entrada, en este caso, `device/+/data`.

   a. En el cliente MQTT, en Suscripciones, seleccione Suscribirse a un tema.
   b. En el tema de suscripción, introduzca el tema del filtro de temas de entrada, **`device/+/data`**.
   c. No cambie el resto de los valores predeterminados de los demás ajustes.
   d. Elija Suscribirse al tema.

   En la columna Suscripciones, aparece **`device/+/data`** en Publicar en un tema.

2. Publique un mensaje en el tema de entrada con un identificador de dispositivo específico, **`device/32/data`** No puede publicar en MQTT temas que contengan caracteres comodín.

   a. En el cliente MQTT, en Suscripciones, seleccione Publicar en tema.
   b. En el campo Publicar, introduzca el nombre del tema de entrada,. **`device/32/data`**
   c. Copie los datos de ejemplo que se muestran aquí y, en el cuadro de edición situado debajo del nombre del tema, péguelos.

```
{
  "temperature": 38,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

d. Para publicar su mensaje MQTT, elija Publicar en tema.

3. Confirme que se envió el mensaje de texto.

a. En el cliente de MQTT, en Suscripciones, hay un punto verde junto al tema al que se ha suscrito anteriormente.

El punto verdes indica que se han recibido uno o más mensajes nuevos desde la última vez que los consultó.

b. En Suscripciones, seleccione device/+/data para comprobar que la carga útil de los mensajes coincide con la que acaba de publicar y tiene este aspecto:

```
{
  "temperature": 38,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

c. Compruebe el teléfono que utilizó para suscribirse al tema SNS y confirme que el contenido de la carga útil del mensaje tiene este aspecto:

```
Device 32 reports a temperature of 38, which exceeds the limit of 30.
```

Si cambia el elemento ID del tema en el tema del mensaje, recuerde que la conversión del valor `topic(2)` a un valor numérico sólo funcionará si ese elemento en el tema del mensaje sólo contiene caracteres numéricos.

4. Intente enviar un mensaje MQTT en el que la temperatura no supere el límite.

   a. En el cliente MQTT, en Suscripciones, seleccione Publicar en tema.

   b. En el campo Publicar, introduzca el nombre del tema de entrada,. **`device/33/data`**

   c. Copie los datos de ejemplo que se muestran aquí y, en el cuadro de edición situado debajo del nombre del tema, péguelos.

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

   d. Para enviar su mensaje MQTT, seleccione Publicar en tema.

   Debería ver el mensaje que envió en la suscripción **`device/+/data`**; sin embargo, como el valor de la temperatura está por debajo de la temperatura máxima en la sentencia de consulta de la regla, no debería recibir un mensaje de texto.

   Si no observa el comportamiento correcto, consulte los consejos para la solución de problemas.

Solución de problemas con AWS Lambda la regla y la notificación

He aquí algunas cosas que debe comprobar, en caso de que no esté viendo los resultados que espera.

- ¿Tiene un mensaje de error

  Si apareció un error al publicar el mensaje de entrada, corrija primero ese error. Los siguientes pasos pueden ayudarle a corregir ese error.

- No ve el mensaje de entrada en el cliente MQTT

  Cada vez que publique su mensaje de entrada en el tema `device/32/data`, dicho mensaje debería aparecer en el cliente MQTT si se ha suscrito al filtro de temas `device/+/data` tal y como se describe en el procedimiento.

Cosas que debe comprobar

- Compruebe el filtro de temas al que se ha suscrito

  Si se ha suscrito al tema del mensaje de entrada como se describe en el procedimiento, debería ver una copia del mensaje de entrada cada vez que lo publique.

  Si no ve el mensaje, compruebe el nombre del tema al que se suscribió y compárelo con el tema en el que lo publicó. Los nombres de los temas distinguen entre mayúsculas y minúsculas y el tema al que te suscribiste debe ser idéntico al tema en el que publicaste la carga útil del mensaje.

- Compruebe la función de publicación de mensajes

  En el cliente MQTT, en Suscripciones, elija device/+/data, marque el tema del mensaje de publicación y, a continuación, elija Publicar en tema. Debería ver la carga útil del mensaje en el cuadro de edición situado debajo del tema en la lista de mensajes.

- Si no recibe el mensaje SMS:

  Para que su regla funcione, debe tener la política correcta que la autorice a recibir un mensaje y enviar una notificación SNS, y debe recibir el mensaje.

  Cosas que debe comprobar

  - Compruebe el Región de AWS de su cliente MQTT y la regla que creó

    La consola en la que ejecuta el cliente MQTT debe estar en la misma región AWS que la regla que creó.

  - Compruebe que el valor de temperatura de la carga útil del mensaje supera el umbral de prueba

    Si el valor de temperatura es inferior o igual a 30, tal como se define en la sentencia de consulta de la regla, la regla no realizará ninguna de sus acciones.

  - Compruebe el tema del mensaje de entrada en la declaración de consulta de la regla

    Para que la regla funcione, debe recibir un mensaje con el nombre del tema que coincida con el filtro de tema de la cláusula FROM de la declaración de consulta de la regla.

    Compruebe la ortografía del filtro de temas de la declaración de consulta de reglas con la del tema en el cliente MQTT. Los nombres de los temas distinguen mayúsculas de minúsculas y el tema del mensaje debe coincidir con el filtro de tema de la declaración de consulta de reglas.

- Compruebe el contenido de la carga útil del mensaje de entrada

  Para que la regla funcione, debe encontrar el campo de datos en la carga útil del mensaje que se declara en la sentencia SELECT.

  Compruebe la ortografía del campo `temperature` en la declaración de consulta de la regla con la de la carga útil del mensaje en el cliente MQTT. Los nombres de los campos distinguen mayúsculas de minúsculas y el campo `temperature` de la declaración de consulta de la regla debe ser idéntico al campo `temperature` de la carga útil del mensaje.

  Asegúrese de que el documento JSON de la carga útil del mensaje tenga el formato correcto. Si el JSON contiene algún error, como la falta de una coma, la regla no podrá leerlo.

- Compruebe la notificación de Amazon SNS

  En Paso 1: crear un tema de Amazon SNS que envía un mensaje de texto SMS, consulte el paso 3, que describe cómo probar la notificación de Amazon SNS y probar la notificación para asegurarse de que funciona.

- Comprobar la función de Lambda

  En Paso 1: Cree una AWS Lambda función que envíe un mensaje de texto, consulte el paso 5, que describe cómo probar la función de Lambda con datos de prueba y probar la función de Lambda.

- Compruebe el rol que utiliza la regla

  La acción de la regla debe tener permiso para recibir el tema original y publicar el tema nuevo.

  Las políticas que autorizan a la regla a recibir datos de los mensajes y volver a publicarlos son específicas de los temas utilizados. Si cambia el tema utilizado para volver a publicar los datos del mensaje, debe actualizar la función de la acción de regla para actualizar su política y que coincida con el tema actual.

  Si sospecha que este es el problema, edite la acción Volver a publicar la regla y cree un nuevo rol. Los nuevos roles creados por la acción de regla reciben las autorizaciones necesarias para realizar estas acciones.

Paso 4: Revisar los resultados y los siguientes pasos

En este tutorial:

- Creó una AWS IoT regla para llamar a una función de Lambda que enviaba una notificación de Amazon SNS que utilizaba su carga de mensajes personalizada.
- Utilizó una consulta y funciones SQL sencillas en una sentencia de consulta de reglas para crear una nueva carga útil de mensajes para la función de Lambda.
- Utilizó el cliente MQTT para probar la regla. AWS IoT

Siguientes pasos

Después de enviar unos cuantos mensajes de texto con esta regla, pruebe a experimentar con ella para ver cómo el cambio de algunos aspectos del tutorial afecta al mensaje y al momento en que se envía. He aquí algunos ejemplos para empezar.

- Cambie *`device_id`* el tema del mensaje de entrada y observe el efecto en el contenido del mensaje de texto.
- Cambie los campos seleccionados en la sentencia de consulta de reglas, actualice la función de Lambda para utilizarlos en un nuevo mensaje y observe el efecto en el contenido del mensaje de texto.
- Cambie la prueba en la declaración de consulta de reglas para probar una temperatura mínima en lugar de una temperatura máxima. Actualice la función de Lambda para formatear un mensaje nuevo y recuerde cambiar el nombre de `max_temperature`.
- Para obtener más información sobre cómo encontrar los errores que pueden producirse al desarrollar y utilizar AWS IoT reglas, consulteMonitorización AWS IoT.

Retener el estado del dispositivo mientras el dispositivo está fuera de línea con Device Shadows

Estos tutoriales muestran cómo utilizar el servicio AWS IoT Device Shadow para almacenar y actualizar la información de estado de un dispositivo. El documento Shadow, que es un documento JSON, muestra el cambio en el estado del dispositivo en función de los mensajes publicados por un dispositivo, una aplicación local o un servicio. En este tutorial, el documento de sombra muestra el cambio en el color de una bombilla. Estos tutoriales también muestran cómo la sombra almacena

esta información incluso cuando el dispositivo está desconectado de Internet y le devuelve la información de estado más reciente cuando vuelve a conectarse y solicita esta información.

Le recomendamos que pruebe estos tutoriales en el orden en que se muestran aquí, empezando por los recursos AWS IoT que necesita para crearlos y la configuración de hardware necesaria, lo que también le ayudará a aprender los conceptos de forma gradual. Estos tutoriales muestran cómo configurar y conectar un dispositivo Raspberry Pi para usarlo con él AWS IoT. Si no dispone del hardware necesario, puede seguir estos tutoriales adaptándolos al dispositivo que prefiera o creando un dispositivo virtual con Amazon EC2.

Información general del escenario del tutorial

El escenario de estos tutoriales es una aplicación o un servicio local que cambia el color de una bombilla y publica sus datos en temas ocultos reservados. Estos tutoriales son similares a la funcionalidad Device Shadow descrita en el tutorial interactivo de introducción y se implementan en un dispositivo Raspberry Pi. Los tutoriales de esta sección se centran en una única sombra clásica y, al mismo tiempo, muestran cómo se pueden acomodar sombras con nombres o varios dispositivos.

Los siguientes tutoriales le ayudarán a aprender a utilizar el servicio AWS IoT Device Shadow.

- Tutorial: Cómo preparar la Raspberry Pi para ejecutar la aplicación shadow

  En este tutorial se muestra cómo configurar un dispositivo Raspberry Pi para conectarse a él AWS IoT. También creará un documento de AWS IoT política y un recurso específico, descargará los certificados y, a continuación, adjuntará la política a ese recurso. Para completar este tutorial se necesitan aproximadamente 30 minutos.

- Tutorial: Instalación del SDK del dispositivo y ejecución de la aplicación de ejemplo para Device Shadows

  En este tutorial se muestra cómo instalar las herramientas, el software y el SDK de AWS IoT dispositivos necesarios para Python y, a continuación, ejecutar la aplicación oculta de ejemplo. Este tutorial se basa en los conceptos presentados en Conexión de una Raspberry Pi u otro dispositivo y tarda 20 minutos en completarse.

- Tutorial: Interactuar con Device Shadow mediante la aplicación de ejemplo y el cliente de pruebas MQTT

  En este tutorial se muestra cómo utilizar la aplicación y la AWS IoT consola de `shadow.py` ejemplo para observar la interacción entre AWS IoT Device Shadows y los cambios de estado de

la bombilla. El tutorial también muestra cómo enviar mensajes MQTT a los temas reservados de Device Shadow. Para completar este tutorial se necesitan aproximadamente 45 minutos.

AWS IoT Descripción general de Device Shadow

Un Device Shadow es una representación virtual y persistente de un dispositivo que se administra mediante un recurso creado en el AWS IoT registro. El documento Shadow es un JSON o un documento de JavaScript notación que se utiliza para almacenar y recuperar la información del estado actual de un dispositivo. Puede usar la sombra para obtener y establecer el estado de un dispositivo mediante temas de MQTT o HTTP REST APIs, independientemente de si el dispositivo está conectado a Internet.

Un documento Shadow contiene una propiedad `state` que describe estos aspectos del estado del dispositivo.

- `desired`: Las aplicaciones especifican los estados deseados de las propiedades del dispositivo actualizando el objeto `desired`.
- `reported`: Los dispositivos notifican su estado actual en el objeto `reported`.
- `delta`: AWS IoT informa de las diferencias entre el estado deseado y el estado reportado en el `delta` objeto.

He aquí un ejemplo de documento de estado en Shadow.

```
{
  "state": {
    "desired": {
      "color": "green"
      },
    "reported": {
      "color": "blue"
      },
    "delta": {
      "color": "green"
      }
   }
}
```

Para actualizar el documento Shadow de un dispositivo, puede utilizar los temas reservados de MQTT, el Device Shadow REST APIs que admite las `GET` operaciones con HTTP y `DELETE` las operaciones con HTTP y la AWS IoT CLI. `UPDATE`

En el ejemplo anterior, supongamos que desea cambiar el color `desired` a `yellow`. Para ello, envía una solicitud a la API UpdateThingShadow o publica un mensaje en el tema Actualizar, `$aws/things/THING_NAME/shadow/update`.

```
{
  "state": {
    "desired": {
      "color": yellow
    }
  }
}
```

Las actualizaciones afectan únicamente a los campos especificados en la solicitud. Tras actualizar correctamente Device Shadow, AWS IoT publica el nuevo `desired` estado del `delta` tema,`$aws/things/THING_NAME/shadow/delta`. El documento Shadow en este caso tiene este aspecto:

```
{
  "state": {
    "desired": {
      "color": yellow
    },
    "reported": {
      "color": green
    },
    "delta": {
      "color": yellow
      }
  }
}
```

A continuación, se informa del nuevo estado a AWS IoT Device Shadow mediante el `Update` tema `$aws/things/THING_NAME/shadow/update` con el siguiente mensaje JSON:

```
{
  "state": {
    "reported": {
      "color": yellow
```

```
    }
  }
}
```

Si desea obtener la información sobre el estado actual, envíe una solicitud a la API GetThingShadow o publique un mensaje MQTT en el tema Obtener, `$aws/things/THING_NAME/shadow/get`.

Para obtener más información sobre el uso del servicio Device Shadow, consulte AWS IoT Servicio Device Shadow.

Para obtener más información sobre el uso de Device Shadows en dispositivos, aplicaciones y servicios, consulte Uso de sombras en dispositivos y Uso de sombras en aplicaciones y servicios.

Para obtener información sobre cómo interactuar con AWS IoT las sombras, consulteInteracción con sombras.

Para obtener información sobre los temas reservados de MQTT y HTTP REST APIs, consulte Temas MQTT de sombra de dispositivo yAPI REST de sombra de dispositivo.

Tutorial: Cómo preparar la Raspberry Pi para ejecutar la aplicación shadow

Este tutorial muestra cómo instalar y configurar un dispositivo Raspberry Pi y cómo crear los AWS IoT recursos que un dispositivo necesita para conectarse e intercambiar mensajes MQTT.

> **Note**
>
> Si tiene previsto the section called “Crear un dispositivo virtual con Amazon EC2”, puede saltarse esta página y continuar en the section called “Configuración del dispositivo”. Creará estos recursos cuando cree su objeto virtual. Si desea utilizar un dispositivo diferente en lugar de la Raspberry Pi, puede intentar seguir estos tutoriales adaptándolos a un dispositivo de su elección.

En este tutorial, aprenderá a:

- Configura un dispositivo Raspberry Pi y configúralo para usarlo con AWS IoTél.
- Cree un documento AWS IoT de política que autorice a su dispositivo a interactuar con AWS IoT los servicios.
- Cree un recurso específico en AWS IoT los certificados de dispositivo X.509 y, a continuación, adjunte el documento de política.

Se trata de la representación virtual de su dispositivo en el registro AWS IoT . El certificado autentica el dispositivo ante AWS IoT Core y el documento de política autoriza al dispositivo a interactuar con él. AWS IoT

Cómo ejecutar este tutorial

Para ejecutar la aplicación de ejemplo `shadow.py` para Device Shadows, necesitará un dispositivo Raspberry Pi que se conecte a AWS IoT. Le recomendamos que siga este tutorial en el orden en que se presenta aquí, comenzando con la configuración de la Raspberry Pi y sus accesorios, y luego la creación de una política y la vinculación de la política a un recurso cosa que usted cree. A continuación, puede seguir este tutorial utilizando la interfaz gráfica de usuario (GUI) compatible con la Raspberry Pi para abrir la AWS IoT consola en el navegador web del dispositivo, lo que también facilita la descarga de los certificados directamente a su Raspberry Pi para conectarse a ella. AWS IoT

Antes de empezar este tutorial, asegúrese de que tiene:

- Un Cuenta de AWS. Si no la tiene, complete los pasos que se describen en Configurar Cuenta de AWS antes de continuar. Necesitarás una consola Cuenta de AWS y una AWS IoT consola para completar este tutorial.
- La Raspberry Pi y los accesorios necesarios. Necesitará lo siguiente:
  - Una Raspberry Pi 3 Modelo B o un modelo más reciente. Este tutorial puede funcionar en versiones anteriores de la Raspberry Pi, pero no lo hemos probado.
  - Sistema operativo Raspberry Pi (32 bits) o posterior. Recomendamos utilizar la última versión del sistema operativo Raspberry Pi. Este tutorial puede funcionar en versiones anteriores del sistema operativo, pero no lo hemos probado.
  - Una conexión Ethernet o wifi.
  - Teclado, ratón, monitor, cables y fuentes de alimentación.

Para completar este tutorial se necesitan aproximadamente 30 minutos.

Paso 1: Instalar y configurar el dispositivo Raspberry Pi

En esta sección, configuraremos un dispositivo Raspberry Pi para usarlo con AWS IoT.

> **Important**
>
> Adaptar estas instrucciones a otros dispositivos y sistemas operativos puede resultar difícil. Deberás entender el dispositivo lo suficientemente bien como para poder interpretar estas instrucciones y aplicarlas al dispositivo. Si encuentra dificultades, puede probar una de las otras opciones de dispositivos como alternativa, por ejemplo Crear un dispositivo virtual con Amazon EC2 o Utilice su PC o Mac con Windows o Linux como dispositivo AWS IoT.

Tendrá que configurar su Raspberry Pi de forma que pueda iniciar el sistema operativo (SO), conectarse a Internet y permitirle interactuar con ella en una interfaz de línea de comandos. También puedes usar la interfaz gráfica de usuario (GUI) compatible con la Raspberry Pi para abrir la AWS IoT consola y ejecutar el resto de este tutorial.

Para configurar el dispositivo Raspberry Pi:

1. Inserte la tarjeta SD en la ranura para tarjetas MicroSD de la Raspberry Pi. Algunas tarjetas SD vienen precargadas con un administrador de instalación que muestra un menú para instalar el sistema operativo después de arrancar la placa. También puede utilizar el generador de imágenes de la Raspberry Pi para instalar el sistema operativo en su tarjeta.
2. Conecte un televisor o monitor HDMI al cable HDMI que se conecta al puerto HDMI de la Raspberry Pi.
3. Conecte el teclado y el ratón a los puertos USB de la Raspberry Pi y, a continuación, enchufe el adaptador de corriente para arrancar la placa.

Una vez que se inicie la Raspberry Pi, si la tarjeta SD viene precargada con el administrador de instalación, aparecerá un menú para instalar el sistema operativo. Si tiene problemas para instalar el sistema operativo, pruebe los siguientes pasos. Para más información sobre la configuración de la Raspberry Pi, consulte Configuración de su Raspberry Pi.

Si tiene problemas para configurar la Raspberry Pi:

- Compruebe si ha insertado la tarjeta SD antes de arrancar la placa. Si conecta la tarjeta SD después de arrancar la placa, es posible que no aparezca el menú de instalación.
- Asegúrese de que el televisor o el monitor estén encendidos y de que se haya seleccionado la entrada correcta.
- Asegúrese de utilizar un software compatible con Raspberry Pi.

Una vez que haya instalado y configurado el sistema operativo Raspberry Pi, abra el navegador web de la Raspberry Pi y navegue hasta la AWS IoT Core consola para continuar con el resto de los pasos de este tutorial.

Si puedes abrir la AWS IoT Core consola, tu Raspberry Pi está lista y puedes continuar haciéndolothe section called “Aprovisionamiento de su dispositivo en AWS IoT ”.

Si tiene problemas o necesita ayuda adicional, consulte Cómo obtener ayuda para su Raspberry Pi.

Tutorial: Aprovisionar tu dispositivo en AWS IoT

En esta sección, se crean los AWS IoT Core recursos que utilizará el tutorial.

Pasos para aprovisionar el dispositivo AWS IoT

- Paso 1: Crear una AWS IoT política para Device Shadow
- Paso 2: Cree un recurso cosa y adjunte la política a la cosa
- Paso 3: Revisar los resultados y los siguientes pasos

Paso 1: Crear una AWS IoT política para Device Shadow

Los certificados X.509 autentican su dispositivo. AWS IoT Core AWS IoT se adjuntan políticas al certificado que permiten al dispositivo realizar AWS IoT operaciones, como suscribirse o publicar en temas reservados de MQTT utilizados por el servicio Device Shadow. El dispositivo presenta su certificado cuando se conecta y envía mensajes a. AWS IoT Core

En este procedimiento, creará una política que permita al dispositivo realizar las operaciones AWS IoT necesarias para ejecutar el programa de ejemplo. Le recomendamos que cree una política que conceda únicamente los permisos necesarios para realizar la tarea. Primero debe crear la AWS IoT política y, a continuación, adjuntarla al certificado del dispositivo que creará más adelante.

Para crear una AWS IoT política

1. En el panel de navegación de la izquierda, seleccione Seguridad y, a continuación, elija Políticas. Si su cuenta tiene pólizas existentes, elija Crear, de lo contrario, en la página Aún no tiene una política, elija Crear una política.
2. En la página Crear una política:

a. Introduzca un nombre para la política en el campo Nombre (por ejemplo, **`My_Device_Shadow_policy`**). No utilice información personalmente identificable en sus nombres de política.

b. En el documento de política, se describen las acciones de conexión, suscripción, recepción y publicación que dan permiso al dispositivo para publicar y suscribirse a los temas reservados de MQTT.

   Copie el siguiente ejemplo de política y péguelo en su documento de política. `thingname`Sustitúyala por el nombre de lo que vas a crear (por ejemplo,`My_light_bulb`), `region` por la AWS IoT región en la que utilizas los servicios y `account` por tu Cuenta de AWS número. Para obtener más información sobre AWS IoT las políticas, consulte[AWS IoT Core políticas](#).

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/
things/thingname/shadow/get",
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/
things/thingname/shadow/update"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/
things/thingname/shadow/get/accepted",
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/
things/thingname/shadow/get/rejected",
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/
things/thingname/shadow/update/accepted",
```

```
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/
things/thingname/shadow/update/rejected",
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/
things/thingname/shadow/update/delta"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/
things/thingname/shadow/get/accepted",
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/
things/thingname/shadow/get/rejected",
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/
things/thingname/shadow/update/accepted",
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/
things/thingname/shadow/update/rejected",
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/
things/thingname/shadow/update/delta"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "iot:Connect",
            "Resource": "arn:aws:iot:us-east-1:123456789012:client/test-*"
        }
    ]
}
```

Paso 2: Cree un recurso cosa y adjunte la política a la cosa

Los dispositivos conectados a AWS IoT pueden representarse mediante recursos de objetos en el AWS IoT registro. Un recurso de cosa representa un dispositivo específico o una entidad lógica, como la bombilla de este tutorial.

Para obtener información sobre cómo crear una cosa en AWS IoT, siga los pasos que se describen enCrear un objeto. He aquí algunas cosas clave que debe tener en cuenta mientras sigue los pasos de ese tutorial:

1. Elija Crear una sola cosa y, en el campo Nombre, introduzca un nombre para la cosa que sea igual a la `thingname` (por ejemplo, `My_light_bulb`) que especificó al crear la política anteriormente.

   No se puede cambiar el nombre de una cosa después de haberla creado. Si le dio un nombre diferente a `thingname`, cree una cosa nueva con el nombre `thingname` y borre la cosa antigua.

   > **Note**
   >
   > No utilice información personal identificable en el nombre de la cosa. El nombre de la cosa puede aparecer en comunicaciones e informes no cifrados.

2. Le recomendamos que descargue cada uno de los archivos de certificado de la página ¡Certificado creado! en un lugar donde pueda encontrarlos fácilmente. Necesitará instalar estos archivos para ejecutar la aplicación de ejemplo.

   Le recomendamos que descargue los archivos en un subdirectorio `certs` del directorio home de su Raspberry Pi y asigne un nombre más simple a cada uno de ellos, tal y como se sugiere en la siguiente tabla.

   Nombres de archivo de certificado

Archivos	Ruta de archivo
Certificado de entidad de certificación raíz	`~/certs/Amazon-root-CA-1.pem`
Certificado de dispositivo	`~/certs/device.pem.crt`
Clave privada	`~/certs/private.pem.key`

3. Después de activar el certificado para habilitar las conexiones AWS IoT, elija Adjuntar una política y asegúrese de adjuntar la política que creó anteriormente (por ejemplo,**`My_Device_Shadow_policy`**) a la cosa.

   Una vez que hayas creado una cosa, podrás ver tu recurso en la lista de cosas de la AWS IoT consola.

Paso 3: Revisar los resultados y los siguientes pasos

En este tutorial, ha aprendido a:

- Instalar y configurar el dispositivo Raspberry Pi.
- Crea un documento AWS IoT de política que autorice a tu dispositivo a interactuar con AWS IoT los servicios.
- Crear un recurso de cosas y un certificado de dispositivo X.509 asociado, y adjuntarle el documento de política.

Siguientes pasos

Ahora puede instalar el SDK del AWS IoT dispositivo para Python, ejecutar la aplicación de `shadow.py` ejemplo y usar Device Shadows para controlar el estado. Para más información sobre cómo ejecutar este tutorial, consulte Tutorial: Instalación del SDK del dispositivo y ejecución de la aplicación de ejemplo para Device Shadows.

Tutorial: Instalación del SDK del dispositivo y ejecución de la aplicación de ejemplo para Device Shadows

En esta sección se muestra cómo instalar el software necesario y el SDK de AWS IoT dispositivos para Python y ejecutar la aplicación de `shadow.py` ejemplo para editar el documento Shadow y controlar el estado de la sombra.

En este tutorial, aprenderá a:

- Usa el software instalado y el SDK de AWS IoT dispositivo para Python para ejecutar la aplicación de muestra.
- Descubre cómo al introducir un valor con la aplicación de ejemplo, se publica el valor deseado en la consola de AWS IoT .
- Revise la aplicación de ejemplo `shadow.py` y descubra cómo utiliza el protocolo MQTT para actualizar el estado de la sombra.

Antes de ejecutar este tutorial:

Debe haber configurado su Cuenta de AWS dispositivo Raspberry Pi y creado AWS IoT algo y una política que otorgue al dispositivo permisos para publicar y suscribirse a los temas reservados de

MQTT del servicio Device Shadow. Para obtener más información, consulte Tutorial: Cómo preparar la Raspberry Pi para ejecutar la aplicación shadow.

También debes haber instalado Git, Python y el SDK de AWS IoT dispositivos para Python. Este tutorial se basa en los conceptos presentados en el tutorial Conexión de una Raspberry Pi u otro dispositivo. Si no ha probado ese tutorial, le recomendamos que siga los pasos descritos en él para instalar los archivos de certificado y el SDK del dispositivo y, a continuación, vuelva a este tutorial para ejecutar la aplicación de ejemplo `shadow.py`.

En este tutorial, podrá:

- Paso 1: ejecutar la aplicación de ejemplo shadow.py
- Paso 2: Revisar la aplicación de ejemplo del SDK de dispositivos shadow.py
- Paso 3: Solucionar problemas con la aplicación de ejemplo shadow.py
- Paso 4: Revisar los resultados y los siguientes pasos

Para completar este tutorial se necesitan aproximadamente 20 minutos.

Paso 1: ejecutar la aplicación de ejemplo shadow.py

Antes de ejecutar la aplicación de ejemplo `shadow.py`, necesitará la siguiente información además de los nombres y la ubicación de los archivos de certificado que instaló.

Valores de los parámetros de la aplicación

Parámetro	Dónde encontrar el valor
your-iot-thing-name	Nombre del elemento AWS IoT que creaste anteriormente enthe section called “Paso 2: Cree un recurso cosa y adjunte la política a la cosa”. Para encontrar este valor, en la consola de AWS IoT, seleccione Administrar y, a continuación, seleccione Cosas.
your-iot-endpoint	El *your-iot-endpoint* valor tiene el formato:*endpoint_id* `-ats.iot.` *region* `.amazonaws.com` , por

Parámetro	Dónde encontrar el valor
	ejemplo,`a3qj468EXAMPLE-ats.iot.us-west-2.amazonaws.com` . Para encontrar este valor: 1. En la consola de AWS IoT, seleccione Administrar y después Cosas. 2. Elija la cosa IoT que creó para su dispositivo, My_light_bulb, que utilizó anteriormente, y después elija Interactuar. Su punto de conexión se muestra en la sección HTTPS.

Instalar y ejecutar la aplicación de ejemplo

1. Navegue hasta el directorio de la aplicación de ejemplo.

```
cd ~/aws-iot-device-sdk-python-v2/samples/service-clients
```

2. En la ventana de la línea de comandos, *your-iot-thing-name* sustituya *your-iot-endpoint* y, como se indica, y ejecute este comando.

```
python3 shadow.py --ca_file ~/certs/Amazon-root-CA-1.pem --cert ~/certs/
device.pem.crt --key ~/certs/private.pem.key --endpoint your-iot-endpoint --
thing_name your-iot-thing-name
```

3. Observe que la aplicación de ejemplo:

   1. Se conecta al servicio de AWS IoT de tu cuenta.
   2. Se suscribe a eventos `Delta` y respuestas `Update` y `Get`.
   3. Le pide que introduzca el valor deseado en el terminal.
   4. Muestra una salida similar a la siguiente:

```
Connecting to a3qEXAMPLEffp-ats.iot.us-west-2.amazonaws.com with client ID
 'test-0c8ae2ff-cc87-49d2-a82a-ae7ba1d0ca5a'...
Connected!
Subscribing to Delta events...
Subscribing to Update responses...
```

```
Subscribing to Get responses...
Requesting current shadow state...
Launching thread to read user input...
Finished getting initial shadow state.
Shadow contains reported value 'off'.
Enter desired value:
```

> **Note**
>
> Si tiene problemas para ejecutar la aplicación de ejemplo `shadow.py`, revise the section called “Paso 3: Solucionar problemas con la aplicación de ejemplo `shadow.py`”. Para obtener información adicional que pueda ayudarle a corregir el problema, añada el parámetro `--verbosity debug` a la línea de órdenes para que la aplicación de ejemplo muestre mensajes detallados sobre lo que está haciendo.

Introduzca los valores y observe las actualizaciones en el documento Shadow

Puede introducir valores en el terminal para especificar el valor `desired`, lo que también actualiza el valor `reported`. Supongamos que introduce el color `yellow` en el terminal. El valor `reported` también se actualiza al color `yellow`. A continuación se muestran los mensajes que se muestran en el terminal:

```
Enter desired value:
yellow
Changed local shadow value to 'yellow'.
Updating reported shadow value to 'yellow'...
Update request published.
Finished updating reported shadow value to 'yellow'.
```

Al publicar esta solicitud de actualización, AWS IoT crea una sombra clásica y predeterminada para el recurso. Puede observar la solicitud de actualización que publicó en `desired` los valores `reported` y de la AWS IoT consola consultando el documento Shadow del recurso de objetos que creó (por ejemplo,`My_light_bulb`). Para ver la actualización en el documento Shadow, siga estos pasos:

1. En la AWS IoT consola, selecciona Administrar y, a continuación, selecciona Cosas.

2. En la lista de cosas que se muestran, seleccione la cosa que ha creado, elija Sombras y, a continuación, Sombra clásica.

El documento Shadow debería tener un aspecto similar al siguiente y mostrar los valores `reported` y `desired` establecidos en el color `yellow`. Puede ver estos valores en la sección Estado de sombra del documento.

```
{
"desired": {
  "welcome": "aws-iot",
  "color": "yellow"
},
"reported": {
  "welcome": "aws-iot",
  "color": "yellow"
}
}
```

También verá una sección de metadatos que contiene la información de la marca de tiempo y el número de versión de la solicitud.

Puede utilizar la versión del documento de estado para asegurarse de que actualiza la versión más reciente del documento de sombra de un dispositivo. Si envía otra solicitud de actualización, el número de versión aumentará en 1. 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.

```
{
"metadata": {
  "desired": {
    "welcome": {
      "timestamp": 1620156892
    },
    "color": {
      "timestamp": 1620156893
    }
  },
  "reported": {
    "welcome": {
      "timestamp": 1620156892
    },
```

```
    "color": {
      "timestamp": 1620156893
    }
  }
},
"version": 10
}
```

Para obtener más información sobre el documento alternativo y observar los cambios en la información de estado, continúe con el siguiente tutorial Tutorial: Interactuar con Device Shadow mediante la aplicación de ejemplo y el cliente de pruebas MQTT, tal como se describe en la sección Paso 4: Revisar los resultados y los siguientes pasos de este tutorial. Si lo desea, también puede obtener información sobre el código de ejemplo `shadow.py` y cómo utiliza el protocolo MQTT en la siguiente sección.

Paso 2: Revisar la aplicación de ejemplo del SDK de dispositivos shadow.py

En esta sección se revisa la aplicación de ejemplo `shadow.py` del SDK de dispositivos AWS IoT v2 para Python utilizada en este tutorial. Aquí veremos cómo se conecta AWS IoT Core mediante los protocolos MQTT y MQTT over WSS. La biblioteca de tiempo de ejecución AWS común (AWS-CRT) proporciona compatibilidad con el protocolo de comunicación de bajo nivel y se incluye en el AWS IoT Device SDK v2 para Python.

Si bien este tutorial utiliza MQTT y MQTT sobre WSS, AWS IoT es compatible con los dispositivos que publican solicitudes HTTPS. Para ver un ejemplo de un programa de Python que envía un mensaje HTTP desde un dispositivo, consulta el ejemplo de código HTTPS con la biblioteca `requests` de Python.

Para obtener información sobre cómo tomar una decisión fundamentada sobre qué protocolo utilizar para las comunicaciones de su dispositivo, consulte la Elección de un protocolo de aplicación para la comunicación entre dispositivos.

MQTT

Las llamadas de ejemplo `shadow.py` `mtls_from_path` (mostradas aquí) en el `mqtt_connection_builder` para establecer una conexión con AWS IoT Core mediante el protocolo MQTT. `mtls_from_path` utiliza certificados X.509 y TLS v1.2 para autenticar el dispositivo. La biblioteca AWS-CRT gestiona los detalles de nivel inferior de esa conexión.

```
mqtt_connection = mqtt_connection_builder.mtls_from_path(
```

```
    endpoint=args.endpoint,
    cert_filepath=args.cert,
    pri_key_filepath=args.key,
    ca_filepath=args.ca_file,
    client_bootstrap=client_bootstrap,
    on_connection_interrupted=on_connection_interrupted,
    on_connection_resumed=on_connection_resumed,
    client_id=args.client_id,
    clean_session=False,
    keep_alive_secs=6
)
```

- `endpoint`es el AWS IoT punto final que ha ingresado desde la línea de comandos y `client_id` es el ID que identifica de forma exclusiva a este dispositivo en. Región de AWS
- `cert_filepath`, `pri_key_filepath` y `ca_filepath` son las rutas a los archivos de certificados y claves privadas del dispositivo y al archivo CA raíz.
- `client_bootstrap` es el objeto común en tiempo de ejecución que gestiona las actividades de comunicación del socket, y se instancia antes de la llamada a `mqtt_connection_builder.mtls_from_path`.
- `on_connection_interrupted` y `on_connection_resumed` son funciones de devolución de llamada para llamar cuando la conexión del dispositivo se interrumpe y se reanuda.
- `clean_session` es si se debe iniciar una nueva sesión persistente o, si ya existe una, reconectarse a una existente. `keep_alive_secs` es el valor de mantenimiento de vida, en segundos, que se debe enviar en la solicitud `CONNECT`. Se enviará automáticamente un ping en este intervalo. El servidor asume que la conexión se pierde si no recibe un ping después de 1,5 veces este valor.

La muestra `shadow.py` también llama a `websockets_with_default_aws_signing` en el `mqtt_connection_builder` para establecer una conexión con AWS IoT Core usando el protocolo MQTT sobre WSS. MQTT sobre WSS también utiliza los mismos parámetros que MQTT y toma estos parámetros adicionales:

- `regiones` la región de AWS firma utilizada en la autenticación de Signature V4 y `credentials_provider` son las AWS credenciales que se proporcionan para utilizarlas en la autenticación. La Región se pasa desde la línea de comandos, y el objeto `credentials_provider` se instancia justo antes de la llamada a `mqtt_connection_builder.websockets_with_default_aws_signing`.

- `websocket_proxy_options` son las opciones de proxy HTTP, si se utiliza un servidor proxy. En la aplicación de ejemplo `shadow.py`, este valor se crea justo antes de la llamada a `mqtt_connection_builder.websockets_with_default_aws_signing`.

Suscribirse a los temas y eventos de Sombra

El ejemplo `shadow.py` intenta establecer una conexión y espera a que esté completamente conectado. Si no está conectado, los comandos se ponen en cola. Una vez conectado, el ejemplo se suscribe a los eventos delta y actualiza y recibe mensajes, y publica los mensajes con un nivel de calidad de servicio (QoS) de 1 (`mqtt.QoS.AT_LEAST_ONCE`).

Cuando un dispositivo se suscribe a un mensaje con QoS de nivel 1, el agente de mensajes guarda los mensajes a los que está suscrito el dispositivo hasta que se puedan enviar al dispositivo. El agente de mensajes vuelve a enviar los mensajes hasta que recibe una respuesta PUBACK del dispositivo.

Para más información sobre el protocolo MQTT, consulte Revise el protocolo MQTT y MQTT.

Para obtener más información sobre cómo se utilizan MQTT, MQTT sobre WSS, las sesiones persistentes y los niveles de QoS que se utilizan en este tutorial, consulte Consulte la aplicación de ejemplo del SDK de dispositivo pubsub.py.

Paso 3: Solucionar problemas con la aplicación de ejemplo **`shadow.py`**

Cuando ejecute la aplicación de ejemplo `shadow.py`, debería ver algunos mensajes en el terminal y un aviso para introducir un valor `desired`. Si el programa arroja un error, para depurarlo puede empezar por comprobar si ha ejecutado el comando correcto para su sistema.

En algunos casos, el mensaje de error puede indicar problemas de conexión y tener un aspecto similar a: `Host name was invalid for dns resolution` o `Connection was closed unexpectedly`. En esos casos, puede comprobar lo siguiente:

- Compruebe la dirección del punto de conexión en el comando

  Revise el argumento `endpoint` del comando que introdujo para ejecutar la aplicación de ejemplo, (por ejemplo, `a3qEXAMPLEffp-ats.iot.us-west-2.amazonaws.com`) y compruebe este valor en la consola de AWS IoT .

  Para comprobar si ha utilizado el valor correcto:

1. En la consola de AWS IoT , seleccione Administrar y después Cosas.
2. Elija la cosa que creó para su aplicación de ejemplo (por ejemplo, My_light_bulb) y luego elija Interactuar.

Su punto de conexión se muestra en la sección HTTPS. También debería ver un mensaje que dice: `This thing already appears to be connected.`

- Compruebe la activación del certificado

  Los certificados autentican el dispositivo con AWS IoT Core.

  Para comprobar si su certificado está activo:

  1. En la consola de AWS IoT , seleccione Administrar y después Cosas.
  2. Elija la cosa que creó para su aplicación de ejemplo (por ejemplo, My_light_bulb) y luego elija Seguridad.
  3. Seleccione el certificado y, a continuación, en la página de detalles del certificado, seleccione Acciones.

  Si en la lista desplegable Activar no está disponible y solo puede elegir Desactivar, su certificado está activo. Si no es así, seleccione Activar y vuelva a ejecutar el programa de muestra.

  Si el programa sigue sin ejecutarse, compruebe los nombres de los archivos de certificado en la carpeta `certs`.

- Compruebe la política adjunta al recurso de la cosa

  Si bien los certificados autentican el dispositivo, AWS IoT las políticas permiten que el dispositivo realice AWS IoT operaciones, como suscribirse o publicar en temas reservados de MQTT.

  Para comprobar si se adjunta la política correcta:

  1. Busque el certificado tal y como se describió anteriormente y, a continuación, elija Políticas.
  2. Elija la política que se muestra y compruebe si describe las acciones `connect`, `subscribe`, `receive` y `publish` que permiten al dispositivo publicar y suscribirse a los temas reservados de MQTT.

     Para ver un ejemplo de política, consulte Paso 1: Crear una AWS IoT política para Device Shadow.

  Si ves mensajes de error que indican que tienes problemas para conectarte AWS IoT, es posible que se deba a los permisos que utilizas para la política. Si ese es el caso, le recomendamos que

comience con una política que proporcione acceso total a AWS IoT los recursos y, a continuación, vuelva a ejecutar el programa de ejemplo. Puede editar la política actual o elegir la política actual, seleccionar Separar y, a continuación, crear otra política que proporcione acceso completo y adjuntarla a su recurso principal. Más adelante, podrá restringir la política únicamente a las acciones y políticas que necesite para ejecutar el programa.

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

- Compruebe la instalación de su SDK del dispositivo

  Si el programa sigue sin ejecutarse, puede volver a instalar el SDK del dispositivo para asegurarse de que su instalación del SDK es completa y correcta.

Paso 4: Revisar los resultados y los siguientes pasos

En este tutorial, ha aprendido a:

- Instale el software, las herramientas y el SDK de AWS IoT dispositivo necesarios para Python.
- Entender cómo la aplicación de ejemplo, `shadow.py`, utiliza el protocolo MQTT para recuperar y actualizar el estado actual de la sombra.
- Ejecute la aplicación de ejemplo para Device Shadows y observe la actualización del documento Shadow en la AWS IoT consola. También aprendió a solucionar cualquier problema y a corregir errores al ejecutar el programa.

Siguientes pasos

Ahora puede ejecutar la aplicación de ejemplo `shadow.py` y usar Device Shadows para controlar el estado. Puede observar las actualizaciones del documento sombra en la consola de AWS IoT y

observar los eventos delta a los que responde la aplicación de muestra. Con el cliente de pruebas MQTT, puede suscribirse a los temas paralelos reservados y observar los mensajes que reciben los temas al ejecutar el programa de ejemplo. Para más información sobre cómo ejecutar este tutorial, consulte Tutorial: Interactuar con Device Shadow mediante la aplicación de ejemplo y el cliente de pruebas MQTT.

Tutorial: Interactuar con Device Shadow mediante la aplicación de ejemplo y el cliente de pruebas MQTT

Para interactuar con la aplicación de ejemplo `shadow.py`, introduzca un valor para el valor en el terminal `desired`. Por ejemplo, puede especificar colores que se parezcan a los semáforos AWS IoT y responder a la solicitud y actualizar los valores informados.

En este tutorial, aprenderá a:

- Usar la aplicación de ejemplo `shadow.py` para especificar los estados deseados y actualizar el estado actual de la sombra.
- Editar el documento Shadow para observar los eventos delta y cómo responde a ellos la aplicación de muestra `shadow.py`.
- Utilizar el cliente de prueba MQTT para suscribirse a temas sombra y observar las actualizaciones cuando ejecute el programa de ejemplo.

Antes de ejecutar este tutorial, debe disponer de lo siguiente:

Configura tu Cuenta de AWS, configura tu dispositivo Raspberry Pi y crea una AWS IoT cosa y una política. También debe haber instalado el software necesario, el SDK del dispositivo y los archivos de certificado necesarios y haber ejecutado el programa de muestra en la terminal. Para obtener más información, consulte los tutoriales anteriores Tutorial: Cómo preparar la Raspberry Pi para ejecutar la aplicación shadow y Paso 1: ejecutar la aplicación de ejemplo shadow.py. Si aún no lo ha hecho, debe completar estos tutoriales.

En este tutorial, podrá:

- Paso 1: Actualizar los valores deseados y notificados mediante la aplicación de ejemplo shadow.py
- Paso 2: Ver los mensajes de la aplicación de ejemplo shadow.py en el cliente de prueba MQTT
- Paso 3: solucionar los errores de las interacciones de Device Shadow
- Paso 4: Revisar los resultados y los siguientes pasos

Para completar este tutorial se necesitan aproximadamente 45 minutos.

Paso 1: Actualizar los valores deseados y notificados mediante la aplicación de ejemplo **`shadow.py`**

En el tutorial anterior[Paso 1: ejecutar la aplicación de ejemplo shadow.py](), aprendió a observar un mensaje publicado en el documento Shadow de la AWS IoT consola al introducir el valor deseado, tal y como se describe en la sección[Tutorial: Instalación del SDK del dispositivo y ejecución de la aplicación de ejemplo para Device Shadows]().

En el ejemplo anterior, configuramos el color deseado en `yellow`. Después de introducir cada valor, el terminal le pide que introduzca otro valor `desired`. Si vuelve a introducir el mismo valor (`yellow`), la aplicación lo reconoce y le pide que introduzca un nuevo valor `desired`.

```
Enter desired value:
yellow
Local value is already 'yellow'.
Enter desired value:
```

Ahora, supongamos que has introducido el color`green`. AWS IoT responde a la solicitud y actualiza el `reported` valor a`green`. Así es como se produce la actualización cuando el estado `desired` es diferente del estado `reported`, lo que provoca un delta.

Cómo simula la aplicación de ejemplo **`shadow.py`** las interacciones de Device Shadow:

1. Introduzca un valor `desired` (por ejemplo`yellow`) en el terminal para publicar el estado deseado.
2. Como el estado `desired` es diferente del estado `reported` (por ejemplo, el color `green`), se produce un delta y la aplicación que está suscrita al delta recibe este mensaje.
3. La aplicación responde al mensaje y actualiza su estado al valor `desired`, `yellow`.
4. A continuación, la aplicación publica un mensaje de actualización con el nuevo valor registrado del estado del dispositivo, `yellow`.

A continuación se muestran los mensajes que se muestran en el terminal y que muestran cómo se publica la solicitud de actualización.

```
Enter desired value:
green
Changed local shadow value to 'green'.
Updating reported shadow value to 'green'...
Update request published.
```

```
Finished updating reported shadow value to 'green'.
```

En la AWS IoT consola, el documento oculto refleja el valor actualizado `green` para los `desired` campos `reported` y, además, el número de versión se incrementa en 1. Por ejemplo, si el número de la versión anterior se mostraba como 10, el número de la versión actual aparecerá como 11.

> **Note**
>
> Borrar una sombra no restablece el número de versión a 0. Verá que la versión sombra se incrementa en 1 cuando publique una solicitud de actualización o cree otra sombra con el mismo nombre.

Editar el documento Shadow para observar los eventos delta

La aplicación de ejemplo `shadow.py` también está suscrita a los eventos `delta` y responde cuando se produce un cambio en el valor `desired`. Por ejemplo, puede cambiar el valor `desired` por el color `red`. Para ello, en la AWS IoT consola, edite el documento oculto haciendo clic en Editar y, a continuación, establezca el `desired` valor `red` en en el JSON, manteniendo el `reported` valor en. `green` Antes de guardar los cambios, mantén abierto el terminal de la Raspberry Pi, ya que verás mensajes en el terminal cuando se produzca el cambio.

```
{
"desired": {
  "welcome": "aws-iot",
  "color": "red"
},
"reported": {
  "welcome": "aws-iot",
  "color": "green"
}
}
```

Tras guardar el nuevo valor, la aplicación de ejemplo `shadow.py` responde a este cambio y muestra mensajes en el terminal que indican el delta. A continuación, verá los siguientes mensajes debajo de la solicitud para introducir el valor `desired`.

```
Enter desired value:
Received shadow delta event.
Delta reports that desired value is 'red'. Changing local value...
```

```
Changed local shadow value to 'red'.
Updating reported shadow value to 'red'...
Finished updating reported shadow value to 'red'.
Enter desired value:
Update request published.
Finished updating reported shadow value to 'red'.
```

Paso 2: Ver los mensajes de la aplicación de ejemplo **`shadow.py`** en el cliente de prueba MQTT

Puede utilizar el cliente de prueba de MQTT de la consola de AWS IoT para supervisar los mensajes de MQTT que se transmiten a su Cuenta de AWS. Al suscribirse a los temas MQTT reservados utilizados por el servicio Device Shadow, puede observar los mensajes recibidos por los temas al ejecutar la aplicación de ejemplo.

Si aún no ha utilizado el cliente de pruebas de MQTT, puede revisar Vea los mensajes MQTT con el cliente AWS IoT MQTT Esto le ayudará a aprender a utilizar el cliente de prueba de MQTT de la consola de AWS IoT para ver los mensajes MQTT a medida que pasan por el agente de mensajes.

1. Abra el cliente de prueba MQTT

   Abra el cliente de prueba de MQTT en la consola de AWS IoT en una ventana nueva para poder observar los mensajes recibidos por los temas MQTT sin perder la configuración de su cliente de prueba de MQTT. El cliente de prueba MQTT no conserva las suscripciones ni los registros de mensajes si lo abandona para ir a otra página de la consola. Para esta sección del tutorial, puedes hacer que el documento Shadow que AWS IoT prefieras y el cliente de pruebas de MQTT se abran en ventanas separadas para observar más fácilmente la interacción con Device Shadows.
2. Suscribirse a los temas reservados de MQTT Shadow

   Puede utilizar el cliente de pruebas de MQTT para introducir los nombres de los temas reservados de MQTT de Device Shadow y suscribirse a ellos para recibir actualizaciones cuando ejecute la aplicación de ejemplo `shadow.py`. Para suscribirse al tema:

   a. En el cliente de prueba MQTT de la consola de AWS IoT , seleccione Suscribirse a un tema.
   b. En la sección de filtros de temas, introduzca: ***`thingname`***$aws/things/ /shadow/update/ #. Aquí `thingname` es el nombre del recurso que creó anteriormente (por ejemplo, `My_light_bulb`).
   c. Mantenga los valores predeterminados para los ajustes de configuración adicionales y, a continuación, seleccione Suscribirse.

Si utiliza el comodín # en la suscripción al tema, podrá suscribirse a varios temas de MQTT al mismo tiempo y observar todos los mensajes que se intercambian entre el dispositivo y su Shadow en una sola ventana. Para obtener más información sobre los caracteres comodín y su uso, consulte Temas de MQTT.

3. Ejecute un programa de ejemplo **`shadow.py`** y observe los mensajes

   En la ventana de la línea de comandos de la Raspberry Pi, si ha desconectado el programa, vuelva a ejecutar la aplicación de ejemplo y mire los mensajes en el cliente de pruebas MQTT de la consola de AWS IoT .

   a. Ejecute la siguiente orden para reiniciar el programa de muestra. Sustituya *`your-iot-thing-name`* y por *`your-iot-endpoint`* los nombres de AWS IoT lo que creó anteriormente (por ejemplo,`My_light_bulb`) y del punto final para interactuar con el dispositivo.

   ```
   cd ~/aws-iot-device-sdk-python-v2/samples/service-clients
   python3 shadow.py --ca_file ~/certs/Amazon-root-CA-1.pem --cert ~/certs/
   device.pem.crt --key ~/certs/private.pem.key --endpoint your-iot-endpoint --
   thing_name your-iot-thing-name
   ```

   A continuación, la aplicación de ejemplo `shadow.py` se ejecuta y recupera el estado de sombra actual. Si ha eliminado la sombra o borrado los estados actuales, el programa fija el valor actual a `off` y luego le pide que introduzca un valor `desired`.

   ```
   Connecting to a3qEXAMPLEffp-ats.iot.us-west-2.amazonaws.com with client ID
    'test-0c8ae2ff-cc87-49d2-a82a-ae7ba1d0ca5a'...
   Connected!
   Subscribing to Delta events...
   Subscribing to Update responses...
   Subscribing to Get responses...
   Requesting current shadow state...
   Launching thread to read user input...
   Finished getting initial shadow state.
   Shadow document lacks 'color' property. Setting defaults...
   Changed local shadow value to 'off'.
   Updating reported shadow value to 'off'...
   Update request published.
   Finished updating reported shadow value to 'off'...
   ```

```
Enter desired value:
```

Por otro lado, si el programa se estaba ejecutando y lo reiniciaste, verás el último valor de color registrado en la terminal. En el cliente de pruebas de MQTT, verá una actualización de los temas $aws/things/ /shadow/get y ***thingname*** $aws/things//. ***thingname*** shadow/get/accepted

Supongamos que el último color registrado fue `green`. A continuación se muestra el contenido del archivo $aws/things//JSON. ***thingname*** shadow/get/accepted

```
{
"state": {
  "desired": {
    "welcome": "aws-iot",
    "color": "green"
  },
  "reported": {
    "welcome": "aws-iot",
    "color": "green"
  }
},
"metadata": {
  "desired": {
    "welcome": {
      "timestamp": 1620156892
    },
    "color": {
      "timestamp": 1620161643
    }
  },
  "reported": {
    "welcome": {
      "timestamp": 1620156892
    },
    "color": {
      "timestamp": 1620161643
    }
  }
},
"version": 10,
"timestamp": 1620173908
}
```

b. Introduzca un valor `desired` en la terminal, por ejemplo `yellow`. La aplicación de ejemplo `shadow.py` responde y muestra los siguientes mensajes en el terminal que muestran el cambio en el valor `reported` a `yellow`.

```
Enter desired value:
yellow
Changed local shadow value to 'yellow'.
Updating reported shadow value to 'yellow'...
Update request published.
Finished updating reported shadow value to 'yellow'.
```

En la opción Cliente de prueba de MQTT de la consola de AWS IoT , en Suscripciones, verá que los siguientes temas han recibido un mensaje:

- $aws/things/ ***thingname*** /shadow/update: muestra que ambos valores cambian de color. `desired` `updated` `yellow`
- $aws/things/***thingname***/shadow/update/accepted: muestra los valores actuales de los estados y y sus metadatos e información de versión. `desired` `reported`
- $aws/things/***thingname***/shadow/update/documents: muestra los valores anteriores y actuales de los estados y sus metadatos e información de versión. `desired` `reported`

Como el documento $aws/things/***thingname***/shadow/update/documentstambién contiene información incluida en los otros dos temas, podemos revisarlo para ver la información sobre el estado. El estado anterior muestra el valor informado establecido en `green`, sus metadatos e información de versión, y el estado actual, que muestra el valor informado actualizado a `yellow`.

```
{
"previous": {
  "state": {
    "desired": {
      "welcome": "aws-iot",
      "color": "green"
    },
    "reported": {
      "welcome": "aws-iot",
      "color": "green"
```

```
    }
  },
  "metadata": {
    "desired": {
      "welcome": {
        "timestamp": 1617297888
      },
      "color": {
        "timestamp": 1617297898
      }
    },
    "reported": {
      "welcome": {
        "timestamp": 1617297888
      },
      "color": {
        "timestamp": 1617297898
      }
    }
  },
  "version": 10
},
"current": {
  "state": {
    "desired": {
      "welcome": "aws-iot",
      "color": "yellow"
    },
    "reported": {
      "welcome": "aws-iot",
      "color": "yellow"
    }
  },
  "metadata": {
    "desired": {
      "welcome": {
        "timestamp": 1617297888
      },
      "color": {
        "timestamp": 1617297904
      }
    },
    "reported": {
      "welcome": {
```

```
        "timestamp": 1617297888
      },
      "color": {
        "timestamp": 1617297904
      }
    }
  },
  "version": 11
},
"timestamp": 1617297904
}
```

c. Ahora, si introduce otro valor `desired`, verá más cambios en los valores `reported` y en las actualizaciones de los mensajes que se reciben en estos temas. El número de versión también se incrementa en 1. Por ejemplo, si introduce el valor `green`, el estado anterior informa del valor `yellow` y el estado actual informa del valor `green`.

4. Editar el documento Shadow para observar los eventos delta

Para observar los cambios en el tema delta, edite el documento paralelo en la consola de AWS IoT . Por ejemplo, puede cambiar el valor `desired` por el color `red`. Para ello, en la AWS IoT consola, selecciona Editar y, a continuación, establece el `desired` valor en rojo en el JSON, manteniendo el valor establecido en. `reported` `green` Antes de guardar el cambio, mantenga el terminal abierto ya que verá el mensaje delta informado en el terminal.

```
{
"desired": {
  "welcome": "aws-iot",
  "color": "red"
},
"reported": {
  "welcome": "aws-iot",
  "color": "green"
}
}
```

La aplicación de ejemplo `shadow.py` responde a este cambio y muestra mensajes en el terminal que indican el delta. En el cliente de pruebas de MQTT, los temas `update` habrán recibido un mensaje que muestra los cambios en los valores `desired` y `reported`.

También verás que el tema $aws/things//***thingname***ha recibido un mensaje. shadow/update/delta Para ver el mensaje, seleccione este tema, que aparece en Suscripciones.

```
{
"version": 13,
"timestamp": 1617318480,
"state": {
  "color": "red"
},
"metadata": {
  "color": {
    "timestamp": 1617318480
  }
}
}
```

Paso 3: solucionar los errores de las interacciones de Device Shadow

Al ejecutar la aplicación de ejemplo Shadow, es posible que surjan problemas al observar las interacciones con el servicio Device Shadow.

Si el programa se ejecuta correctamente y le pide que introduzca un valor `desired`, debería poder observar las interacciones de Device Shadow mediante el documento Shadow y el cliente de pruebas MQTT, tal y como se ha descrito anteriormente. Sin embargo, si no puede ver las interacciones, puede comprobar lo siguiente:

- Comprueba el nombre de la cosa y su sombra en la consola AWS IoT

  Si no ve los mensajes en el documento Shadow, revise el comando y asegúrese de que coincide con el nombre de la cosa en la consola de AWS IoT . También puede comprobar si tiene una sombra clásica eligiendo su recurso cosa y luego eligiendo Sombras. Este tutorial se centra principalmente en las interacciones con la sombra clásica.

  También puede confirmar que el dispositivo que utilizó está conectado a Internet. En la consola de AWS IoT , elija lo que ha creado anteriormente y, a continuación, seleccione Interactuar. En la página de detalles de la cosa, debería ver un mensaje que dice: `This thing already appears to be connected.`
- Compruebe los temas reservados de MQTT a los que se ha suscrito

  Si los mensajes no aparecen en el cliente de prueba de MQTT, compruebe si los temas a los que se ha suscrito tienen el formato correcto. Los temas de Device Shadow de MQTT tienen el formato $aws/things/ ***`thingname`*** /shadow/ y pueden tener `update`, `get`, o `delete` seguir este formato

en función de las acciones que desee realizar en la sombra. En este tutorial se utiliza el tema $aws/things/ ***thingname*** /shadow/ #, así que asegúrese de haberlo introducido correctamente al suscribirse al tema en la sección de filtro de temas del cliente de prueba.

Al introducir el nombre del tema, asegúrate de que *thingname* es el mismo que el nombre del tema que has creado anteriormente. AWS IoT También puede suscribirse a otros temas de MQTT para comprobar si la actualización se ha realizado correctamente. Por ejemplo, puedes suscribirte al tema $aws/things/***thingname***/shadow/update/rejectedpara recibir un mensaje cada vez que falle una solicitud de actualización y así poder depurar los problemas de conexión. Para más información sobre los temas reservados, consulte the section called “Temas de sombra” y Temas MQTT de sombra de dispositivo.

Paso 4: Revisar los resultados y los siguientes pasos

En este tutorial, ha aprendido a:

- Usar la aplicación de ejemplo `shadow.py` para especificar los estados deseados y actualizar el estado actual de la sombra.
- Editar el documento Shadow para observar los eventos delta y cómo responde a ellos la aplicación de muestra `shadow.py`.
- Utilizar el cliente de prueba MQTT para suscribirse a temas sombra y observar las actualizaciones cuando ejecute el programa de ejemplo.

Siguientes pasos

Puede suscribirse a otros temas reservados de MQTT para ver las actualizaciones de la aplicación oculta. Por ejemplo, si solo te suscribes al tema $aws/things/***thingname***/shadow/update/accepted, solo verás la información sobre el estado actual cuando la actualización se realice correctamente.

También puede suscribirse a temas sombra adicionales para depurar problemas o aprender más sobre las interacciones de la Sombra de Dispositivos y también depurar cualquier problema con las interacciones de la Sombra de Dispositivos. Para obtener más información, consulte the section called “Temas de sombra” y Temas MQTT de sombra de dispositivo.

También puedes optar por ampliar tu aplicación utilizando sombras con nombre o utilizando hardware adicional conectado a la Raspberry Pi para observar los cambios en su estado mediante los mensajes enviados desde el LEDs terminal.

Para obtener más información sobre el servicio Device Shadow y el uso del servicio en dispositivos, aplicaciones y servicios, consulte AWS IoT Servicio Device Shadow, Uso de sombras en dispositivos, y Uso de sombras en aplicaciones y servicios.

Tutorial: Crear un autorizador personalizado para AWS IoT Core

En este tutorial se muestran los pasos para crear, validar y utilizar la autenticación personalizada mediante la AWS CLI. Si lo desea, con este tutorial, puede utilizar Postman para enviar datos a AWS IoT Core mediante la API de publicación HTTP.

En este tutorial se muestra cómo crear una función de Lambda de ejemplo que implemente la lógica de autorización y autenticación y un autorizador personalizado mediante la llamada a create-authorizer con la firma de token habilitada. A continuación, se valida el autorizador mediante la API de publicación HTTP ytest-invoke-authorizer, por último, se pueden enviar datos a él AWS IoT Core mediante la API de publicación HTTP a un tema de prueba de MQTT. La solicitud de ejemplo especificará el autorizador que se va a invocar mediante el `x-amz-customauthorizer-name` encabezado y pasará los encabezados de la solicitud y pasará los encabezados de la token-key-name solicitud. `x-amz-customauthorizer-signature`

Lo que aprenderá en este tutorial:

- Cómo crear una función de Lambda para que sea un controlador de autorizador personalizado.
- ¿Cómo crear un autorizador personalizado con la firma de token habilitada AWS CLI
- Cómo probar su autorizador personalizado con el comando test-invoke-authorizer.
- Cómo publicar un tema de MQTT con Postman y validar la solicitud con su autorizador personalizado.

Para completar este tutorial se necesitan aproximadamente 60 minutos.

En este tutorial, recorrerá los siguientes pasos:

- Paso 1: crear una función de Lambda para su autorizador personalizado
- Paso 2: crear un par de claves pública y privada para su autorizador personalizado
- Paso 3: crear un recurso autorizador personalizado y su autorización
- Paso 4: Pruebe el autorizador llamando test-invoke-authorizer
- Paso 5: Probar la publicación del mensaje MQTT utilizando Postman
- Paso 6: Visualizar los mensajes en el cliente de prueba MQTT

- Paso 7: Revisar los resultados y los siguientes pasos
- Paso 8: Eliminación

Antes de empezar este tutorial, asegúrese de que tiene:

- Configurar Cuenta de AWS

  Necesitarás tu AWS IoT consola Cuenta de AWS y tu consola para completar este tutorial.

  La cuenta que utiliza para este tutorial funciona mejor cuando incluye al menos estas políticas administradas de AWS :

  - `IAMFullAccess`
  - `AWSIoTFullAccess`
  - `AWSLambda_FullAccess`

  > **Important**
  >
  > Las políticas de IAM utilizadas en este tutorial son más permisivas de lo que deberían en una implementación de producción. En un entorno de producción, asegúrese de que sus políticas de cuentas y recursos concedan solo los permisos necesarios.
  > Cuando cree políticas de IAM para la producción, determine qué acceso necesitan los usuarios y los roles y, a continuación, diseñe las políticas que les permitan realizar solo esas tareas.
  > Para obtener más información, consulte Prácticas recomendadas de seguridad en IAM.

- Instaló la AWS CLI

  Para obtener información acerca de cómo instalar el AWS CLI, consulte Instalación de la AWS CLI. Este tutorial requiere una AWS CLI versión `aws-cli/2.1.3 Python/3.7.4 Darwin/18.7.0 exe/x86_64` o posterior.

- Herramientas de OpenSSL

  Los ejemplos de este tutorial utilizan LibreSSL 2.6.5. También puede usar las herramientas de OpenSSL v1.1.1i para este tutorial.

- Ha revisado la descripción general de AWS Lambda

Si no lo ha utilizado AWS Lambda antes, consulte AWS LambdaIntroducción a Lambda para conocer sus términos y conceptos.

- Ha revisado cómo crear solicitudes en Postman

  Para obtener más información, consulte Creación de solicitudes.

- Ha eliminado los autorizadores personalizados del tutorial anterior

  Solo Cuenta de AWS puede configurar un número limitado de autorizadores personalizados a la vez. Para obtener información sobre la eliminación de un autorizador personalizado, consulte the section called “Paso 8: Eliminación”.

Paso 1: crear una función de Lambda para su autorizador personalizado

La autenticación personalizada AWS IoT Core utiliza los recursos de autorización que usted crea para autenticar y autorizar a los clientes. La función que creará en esta sección autentica y autoriza a los clientes a medida que se conectan a los recursos y acceden a AWS IoT Core ellos. AWS IoT

La función de Lambda hace lo siguiente:

- Si una solicitud procede de test-invoke-authorizer, devuelve una política de IAM con una acción `Deny`.
- Si una solicitud procede de Postman mediante HTTP y el parámetro `actionToken` tiene un valor de `allow`, devuelve una política de IAM con una acción `Allow`. En caso contrario, devuelve una política de IAM con una acción `Deny`.

Para crear la función de Lambda para su autorizador personalizado

1. En la consola de Lambda, abra Funciones.
2. Seleccione Creación de función.
3. Confirme que se ha seleccionado Crear desde cero.
4. En Basic information:
   a. Bajo Function name (Nombre de función), escriba **`custom-auth-function`**.
   b. En Tiempo de ejecución, confirme Node.js 18.x.
5. Elija Create function (Crear función).

Lambda crea una función Node.js y un rol de ejecución que otorga a la función permiso para cargar los registros. La función Lambda asume la función de ejecución cuando se invoca la función y la utiliza para crear credenciales para el AWS SDK y leer los datos de las fuentes de eventos.

6. Para ver el código y la configuración de la función en el AWS Cloud9editor, elija custom-auth-functionen la ventana del diseñador y, a continuación, elija index.js en el panel de navegación del editor.

   Para lenguajes de scripting como Node.js, Lambda incluye una función básica que devuelve una respuesta de éxito. Puede utilizar el editor de AWS Cloud9 para editar su función siempre que su código fuente no supere los 3 MB.

7. Reemplace el código index.js en el editor por el siguiente código:

```
// A simple Lambda function for an authorizer. It demonstrates
// How to parse a CLI and Http password to generate a response.

export const handler = async (event, context, callback) => {

    //Http parameter to initiate allow/deny request
    const HTTP_PARAM_NAME='actionToken';
    const ALLOW_ACTION = 'Allow';
    const DENY_ACTION = 'Deny';

    //Event data passed to Lambda function
    var event_str = JSON.stringify(event);
    console.log('Complete event :'+ event_str);

    //Read protocolData from the event json passed to Lambda function
    var protocolData = event.protocolData;
    console.log('protocolData value---> ' + protocolData);

    //Get the dynamic account ID from function's ARN to be used
    // as full resource for IAM policy
    var ACCOUNT_ID = context.invokedFunctionArn.split(":")[4];
    console.log("ACCOUNT_ID---"+ACCOUNT_ID);

    //Get the dynamic region from function's ARN to be used
    // as full resource for IAM policy
    var REGION = context.invokedFunctionArn.split(":")[3];
    console.log("REGION---"+REGION);
```

```
    //protocolData data will be undefined if testing is done via CLI.
    // This will help to test the set up.
    if (protocolData === undefined) {

        //If CLI testing, pass deny action as this is for testing purpose only.
        console.log('Using the test-invoke-authorizer cli for testing only');
        callback(null, generateAuthResponse(DENY_ACTION,ACCOUNT_ID,REGION));

    } else{

        //Http Testing from Postman
        //Get the query string from the request
        var queryString = event.protocolData.http.queryString;
        console.log('queryString values -- ' + queryString);
        /*         global URLSearchParams       */
        const params = new URLSearchParams(queryString);
        var action = params.get(HTTP_PARAM_NAME);

        if(action!=null && action.toLowerCase() === 'allow'){

            callback(null, generateAuthResponse(ALLOW_ACTION,ACCOUNT_ID,REGION));

        }else{

            callback(null, generateAuthResponse(DENY_ACTION,ACCOUNT_ID,REGION));

        }

    }

};

// Helper function to generate the authorization IAM response.
var generateAuthResponse = function(effect,ACCOUNT_ID,REGION) {

    var full_resource = "arn:aws:iot:"+ REGION + ":" + ACCOUNT_ID + ":*";
    console.log("full_resource---"+full_resource);

    var authResponse = {};
    authResponse.isAuthenticated = true;
    authResponse.principalId = 'principalId';

    var policyDocument = {};
```

```
    policyDocument.Version = '2012-10-17       ';
    policyDocument.Statement = [];
    var statement = {};
    statement.Action = 'iot:*';
    statement.Effect = effect;
    statement.Resource = full_resource;
    policyDocument.Statement[0] = statement;
    authResponse.policyDocuments = [policyDocument];
    authResponse.disconnectAfterInSeconds = 3600;
    authResponse.refreshAfterInSeconds = 600;

    console.log('custom auth policy function called from http');
    console.log('authResponse --> ' + JSON.stringify(authResponse));
    console.log(authResponse.policyDocuments[0]);

    return authResponse;
}
```

8. Elija Implementar.
9. Después de que los cambios implementados aparezcan encima del editor:
    a. Desplácese hasta la sección Información general de la función situada encima del editor.
    b. Copie el ARN de la función y guárdelo para utilizarlo más adelante en este tutorial.
10. Comprobación de la función de .
    a. Elija la pestaña Prueba.
    b. Utilizando la configuración predeterminada de la prueba, seleccione Invocar.
    c. Si la prueba se ha realizado correctamente, abra la vista Detalles en Resultados de la ejecución. Debería ver el documento de política que devolvió la función.

       Si la prueba falló o no ve ningún documento de política, revise el código para encontrar y corregir los errores.

Paso 2: crear un par de claves pública y privada para su autorizador personalizado

Su autorizador personalizado requiere una clave pública y privada para autenticarlo. Los comandos de esta sección utilizan herramientas OpenSSL para crear este par de claves.

Para crear un par de claves pública y privada para su autorizador personalizado

1. Cree el archivo de clave privada.

```
openssl genrsa -out private-key.pem 4096
```

2. Verifique el archivo de clave privada que acaba de crear.

```
openssl rsa -check -in private-key.pem -noout
```

   Si el comando no muestra ningún error, el archivo de clave privada es válido.

3. Cree el archivo de clave pública.

```
openssl rsa -in private-key.pem -pubout -out public-key.pem
```

4. Verifique el archivo de clave pública.

```
openssl pkey -inform PEM -pubin -in public-key.pem -noout
```

   Si el comando no muestra ningún error, el archivo de clave pública es válido.

Paso 3: crear un recurso autorizador personalizado y su autorización

El autorizador AWS IoT personalizado es el recurso que une todos los elementos creados en los pasos anteriores. En esta sección, creará un recurso de autorizador personalizado y le dará permiso para ejecutar la función de Lambda que creó anteriormente. Puede crear un recurso de autorización personalizado mediante la AWS IoT consola AWS CLI, la o la AWS API.

Para este tutorial, solo necesita crear un autorizador personalizado. En esta sección se describe cómo crear mediante la AWS IoT consola y el AWS CLI, de modo que pueda utilizar el método que le resulte más conveniente. No hay diferencia entre los recursos del autorizador personalizado creados por uno u otro método.

Creación de un recurso de autorizador personalizado

Elija una de estas opciones para crear su recurso de autorizador personalizado

- Cree un autorizador personalizado mediante la consola AWS IoT
- Creación de un autorizador personalizado utilizando la consola de AWS CLI

Para crear un autorizador personalizado (consola)

1. Abra la página del autorizador personalizado de la AWS IoT consola y elija Crear autorizador.
2. En Crear un autorizador:
   a. En Nombre del autorizador, introduzca **`my-new-authorizer`**.
   b. En Estado del autorizador, marque Activo.
   c. En Función del autorizador, elija la función de Lambda que creó anteriormente.
   d. En Validación del token - opcional:
      i. Active Validación del token.
      ii. En Nombre de la clave del token, introduzca **`tokenKeyName`**.
      iii. Elija Agregar clave.
      iv. En Nombre de la clave, introduzca **`FirstKey`**.
      v. En Clave pública, introduzca el contenido del archivo `public-key.pem`. Asegúrese de incluir las líneas del archivo con `-----BEGIN PUBLIC KEY-----` y `-----END PUBLIC KEY-----`, y no agregue ni elimine ningún salto de línea, retorno de carro u otros caracteres del contenido del archivo. La cadena que introduzca debe parecerse a la de este ejemplo.

```
-----BEGIN PUBLIC KEY-----
MIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEAvEBzOk4vhN+3LgslvEWt
sLCqNmt5Damas3bmiTRvq2gjRJ6KXGTGQChqArAJwL1a9dkS9+maaXC3vc6xzx9z
QPu/vQOe5tyzz1MsKdmtFGxMqQ3qjEXAMPLEOmqyUKPP5mff58k6ePSfXAnzBH0q
lg2HioefrpU5OSAnpuRAjYKofKjbc2Vrn6N2G7hV+IfTBvCElf0csalS/Rk4phD5
oa4Y0GHISRnevypg5C8n9Rrz91PWGqP6M/q5DNJJXjMyleG92hQgu1N696bn5Dw8
FhedszFa6b2x6xrItZFzewNQkPMLMFhNrQIIyvshtT/F1LVCS5+v8AQ8UGGDfZmv
QeqAMAF7WgagDMXcfgKSVU8yid2sIm56qsCLMvD2Sq8Lgzpey9N5ON1o1Cvldwvc
KrJJtgwW6hVqRGuShnownLpgG86M6neZ5sRMbVNZO8OzcobLngJ0Ibw9KkcUdklW
gvZ6HEJqBY2XE70iEXAMPLETPHzhqvK6Ei1HGxpHsXx6BNft582J1VpgYjXha8oa
/NN7l7Zbj/euAb41IVtmX8JrD9z613d1iM5L8HluJlUzn62Q+VeNV2tdA7MfPfMC
8btGYladFAnitThaz6+F0VSBJPu7pZQoLnqyEp5zLMtF+kFl2yOBmGAP0RBivRd9
JWBUCG0bqcLQPeQyjbXSOfUCAwEAAQ==
-----END PUBLIC KEY-----
```

3. Elija Crear autorizador.

4. Si se creó el recurso de autorizador personalizado, verá la lista de autorizadores personalizados y su nuevo autorizador personalizado debería aparecer en la lista y podrá continuar con la siguiente sección para probarlo.

   Si ve un error, revíselo e intente crear de nuevo su autorizador personalizado y vuelva a comprobar las entradas. Tenga en cuenta que cada recurso de autorizador personalizado debe tener un nombre único.

Para crear un autorizador personalizado (AWS CLI)

1. Sustituya sus valores por `authorizer-function-arn` y `token-signing-public-keys`, y ejecute el siguiente comando:

```
aws iot create-authorizer \
--authorizer-name "my-new-authorizer" \
--token-key-name "tokenKeyName" \
--status ACTIVE \
--no-signing-disabled \
--authorizer-function-arn "arn:aws:lambda:Region:57EXAMPLE833:function:custom-auth-
function" \
--token-signing-public-keys FirstKey="-----BEGIN PUBLIC KEY-----
MIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEAvEBzOk4vhN+3LgslvEWt
sLCqNmt5Damas3bmiTRvq2gjRJ6KXGTGQChqArAJwL1a9dkS9+maaXC3vc6xzx9z
QPu/vQOe5tyzz1MsKdmtFGxMqQ3qjEXAMPLEOmqyUKPP5mff58k6ePSfXAnzBH0q
lg2HioefrpU5OSAnpuRAjYKofKjbc2Vrn6N2G7hV+IfTBvCElf0csalS/Rk4phD5
oa4Y0GHISRnevypg5C8n9Rrz91PWGqP6M/q5DNJJXjMyleG92hQgu1N696bn5Dw8
FhedszFa6b2x6xrItZFzewNQkPMLMFhNrQIIyvshtT/F1LVCS5+v8AQ8UGGDfZmv
QeqAMAF7WgagDMXcfgKSVU8yid2sIm56qsCLMvD2Sq8Lgzpey9N5ON1o1Cvldwvc
KrJJtgwW6hVqRGuShnownLpgG86M6neZ5sRMbVNZO8OzcobLngJ0Ibw9KkcUdklW
gvZ6HEJqBY2XE70iEXAMPLETPHzhqvK6Ei1HGxpHsXx6BNft582J1VpgYjXha8oa
/NN7l7Zbj/euAb41IVtmX8JrD9z613d1iM5L8HluJlUzn62Q+VeNV2tdA7MfPfMC
8btGYladFAnitThaz6+F0VSBJPu7pZQoLnqyEp5zLMtF+kFl2yOBmGAP0RBivRd9
JWBUCG0bqcLQPeQyjbXSOfUCAwEAAQ==
-----END PUBLIC KEY-----"
```

   Donde:

   - El valor `authorizer-function-arn` es el nombre de recurso de Amazon (ARN) de la función de Lambda que creó para su autorizador personalizado.

- El valor `token-signing-public-keys` incluye el nombre de la clave, **`FirstKey`**, y el contenido del archivo `public-key.pem`. Asegúrese de incluir las líneas del archivo con `-----BEGIN PUBLIC KEY-----` y `-----END PUBLIC KEY-----`, y no agregue ni elimine ningún salto de línea, retorno de carro u otros caracteres del contenido del archivo.

  Nota: Tenga cuidado al introducir la clave pública, ya que cualquier alteración de su valor la inutilizará.

2. Si se crea el autorizador personalizado, el comando devuelve el nombre y el ARN del nuevo recurso, como se muestra a continuación.

```
{
    "authorizerName": "my-new-authorizer",
    "authorizerArn": "arn:aws:iot:Region:57EXAMPLE833:authorizer/my-new-authorizer"
}
```

   Guarde el valor `authorizerArn` para utilizarlo en el siguiente paso.

   Recuerde que cada recurso de autorizador personalizado debe tener un nombre único.

Autorización del recurso autorizador personalizado

En esta sección, concederá permiso al recurso de autorizador personalizado que acaba de crear para ejecutar la función de Lambda. Para conceder el permiso, puede utilizar el comando de la CLI add-permission.

Conceda permiso a su función Lambda mediante el AWS CLI

1. Después de insertar sus valores, introduzca el siguiente comando. Tenga en cuenta que el valor `statement-id` debe ser único. Reemplace *`Id-1234`* por otro valor si ha ejecutado este tutorial antes o si obtiene un error `ResourceConflictException`.

```
aws lambda add-permission  \
--function-name "custom-auth-function" \
--principal "iot.amazonaws.com" \
--action "lambda:InvokeFunction" \
--statement-id "Id-1234" \
--source-arn authorizerArn
```

2. Si el comando tiene éxito, devuelve una declaración de permiso, como la de este ejemplo. Puede continuar con la siguiente sección para probar el autorizador personalizado.

```
{
    "Statement": "{\"Sid\":\"Id-1234\",\"Effect\":\"Allow\",\"Principal
\":{\"Service\":\"iot.amazonaws.com\"},\"Action\":\"lambda:InvokeFunction
\",\"Resource\":\"arn:aws:lambda:Region:57EXAMPLE833:function:custom-
auth-function\",\"Condition\":{\"ArnLike\":{\"AWS:SourceArn\":
\"arn:aws:lambda:Region:57EXAMPLE833:function:custom-auth-function\"}}}"
}
```

Si el comando no tiene éxito, devuelve un error, como en este ejemplo. Tendrá que revisar y corregir el error antes de continuar.

```
An error occurred (AccessDeniedException) when calling the AddPermission operation:
 User: arn:aws:iam::57EXAMPLE833:user/EXAMPLE-1 is not authorized to perform:
 lambda:AddPer
mission on resource: arn:aws:lambda:Region:57EXAMPLE833:function:custom-auth-
function
```

Paso 4: Pruebe el autorizador llamando test-invoke-authorizer

Con todos los recursos definidos, en esta sección llamarás test-invoke-authorizer desde la línea de comandos para probar la aprobación de la autorización.

Tenga en cuenta que al invocar el autorizador desde la línea de comandos, `protocolData` no está definido, por lo que el autorizador siempre devolverá un documento DENY. Sin embargo, esta prueba confirma que su autorizador personalizado y la función de Lambda están configurados correctamente, aunque no pruebe por completo la función de Lambda.

Para probar su autorizador personalizado y su función Lambda mediante el AWS CLI

1. En el directorio que tiene el archivo `private-key.pem` que creó en un paso anterior, ejecute el siguiente comando.

```
echo -n "tokenKeyValue" | openssl dgst -sha256 -sign private-key.pem | openssl
 base64 -A
```

Este comando crea una cadena de firma que se utilizará en el siguiente paso. La cadena de firma tiene este aspecto:

```
dBwykzlb+fo+JmSGdwoGr8dyC2qB/IyLefJJr+rbCvmu9Jl4KHAA9DG+V
+MMWu09YSA86+64Y3Gt4tOykpZqn9mn
VB1wyxp+0bDZh8hmqUAUH3fwi3fPjBvCa4cwNuLQNqBZzbCvsluv7i2IMjEg
+CPY0zrWt1jr9BikgGPDxWkjaeeh
bQHHTo357TegKs9pP30Uf4TrxypNmFswA5k7QIc01n4bIyRTm90OyZ94R4bdJsHNig1JePgnuOBvMGCEFE09jGjj
szEHfgAUAQIWXiVGQj16BU1xKpTGSiTAwheLKUjITOEXAMPLECK3aHKYKY
+d1vTvdthKtYHBq8MjhzJ0kggbt29V
QJCb8RilN/P5+vcVniSXWPplyB5jkYs9UvG08REoy64AtizfUhvSul/r/F3VV8ITtQp3aXiUtcspACi6ca
+tsDuX
f3LzCwQQF/YSUy02u5XkWn
+sto6KCkpNlkD0wU8gl3+kOzxrthnQ8gEajd5Iylx230iqcXo3osjPha7JDyWM5o+K
EWckTe91I1mokDr5sJ4JXixvnJTVSx1li49IalW4en1DAkc1a0s2U2UNm236EXAMPLELotyh7h
+flFeloZlAWQFH
xRlXsPqiVKS1ZIUClaZWprh/orDJplpiWfBgBIOgokJIDGP9gwhXIIk7zWrGmWpMK9o=
```

Copie esta cadena de firma para utilizarla en el siguiente paso. Tenga cuidado de no incluir ningún carácter de más ni omitir ninguno.

2. En este comando, reemplace el valor `token-signature` por la cadena de firma del paso anterior y ejecute este comando para probar su autorizador.

```
aws iot test-invoke-authorizer \
--authorizer-name my-new-authorizer \
--token tokenKeyValue \
--token-signature dBwykzlb+fo+JmSGdwoGr8dyC2qB/IyLefJJr
+rbCvmu9Jl4KHAA9DG+V+MMWu09YSA86+64Y3Gt4tOykpZqn9mnVB1wyxp
+0bDZh8hmqUAUH3fwi3fPjBvCa4cwNuLQNqBZzbCvsluv7i2IMjEg
+CPY0zrWt1jr9BikgGPDxWkjaeehbQHHTo357TegKs9pP30Uf4TrxypNmFswA5k7QIc01n4bIyRTm90OyZ94R4bdJsH
+d1vTvdthKtYHBq8MjhzJ0kggbt29VQJCb8RilN/
P5+vcVniSXWPplyB5jkYs9UvG08REoy64AtizfUhvSul/r/F3VV8ITtQp3aXiUtcspACi6ca
+tsDuXf3LzCwQQF/YSUy02u5XkWn
+sto6KCkpNlkD0wU8gl3+kOzxrthnQ8gEajd5Iylx230iqcXo3osjPha7JDyWM5o
+KEWckTe91I1mokDr5sJ4JXixvnJTVSx1li49IalW4en1DAkc1a0s2U2UNm236EXAMPLELotyh7h
+flFeloZlAWQFHxRlXsPqiVKS1ZIUClaZWprh/orDJplpiWfBgBIOgokJIDGP9gwhXIIk7zWrGmWpMK9o=
```

Si el comando tiene éxito, devuelve la información generada por su función de autorización personalizada, como en este ejemplo.

```
{
    "isAuthenticated": true,
    "principalId": "principalId",
    "policyDocuments": [
        "{\"Version\":\"2012-10-17\",       \"Statement\":[{\"Action\":\"iot:*\",
\"Effect\":\"Deny\",\"Resource\":\"arn:aws:iot:Region:57EXAMPLE833:*\"}]}"
    ],
    "refreshAfterInSeconds": 600,
    "disconnectAfterInSeconds": 3600
}
```

Si el comando devuelve un error, revise el error y vuelva a comprobar los comandos que ha utilizado en esta sección.

Paso 5: Probar la publicación del mensaje MQTT utilizando Postman

1. Para obtener el punto de conexión de datos de su dispositivo desde la línea de comandos, llame a describe-endpoint, tal y como se muestra aquí.

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

   Guarde esta dirección para utilizarla como *device_data_endpoint_address* en un paso posterior.

2. Abra una nueva ventana de Postman y cree una nueva petición HTTP POST.

   a. Desde su ordenador, abra la aplicación Postman.

   b. En Postman, en el menú File, seleccione New....

   c. En el cuadro de diálogo New, seleccione Request.

   d. En Save Request,

      i. En Request Name, introduzca **Custom authorizer test request**.

      ii. En Select a collection or folder to save to: elija o cree una colección en la que guardar esta solicitud.

      iii. Selecciona Guardar en ***collection_name***.

3. Cree la solicitud POST para probar su autorizador personalizado.

a. En el selector de método de solicitud situado junto al campo URL, elija POST.

b. En el campo URL, cree la URL de su solicitud utilizando la siguiente URL con el comando *device_data_endpoint_address* from the describe-endpoint en un paso anterior.

```
https://device_data_endpoint_address:443/topics/test/cust-auth/topic?
qos=0&actionToken=allow
```

Tenga en cuenta que esta URL incluye el parámetro de consulta `actionToken=allow` que indicará a su función de Lambda que devuelva un documento normativo que permita el acceso a AWS IoT. Después de introducir la URL, los parámetros de consulta también aparecen en la pestaña Params de Postman.

c. En la pestaña Auth, en el campo Type, elija No Auth.

d. En la pestaña Headers:

i. Si hay alguna clave de host que esté marcada, desmárquela.

ii. Al final de la lista de encabezados, agregue estos nuevos y confirme que están marcados. Sustituya el **`Host`** valor por el *device_data_endpoint_address* suyo y el **`x-amz-customauthorizer-signature`** valor por la cadena de firma que utilizó con el test-invoke-authorize comando en la sección anterior.

Clave	Valor
`x-amz-customauthorizer-name`	**`my-new-authorizer`**
`Host`	*device_data_endpoint_address*
`tokenKeyName`	**`tokenKeyValue`**

Clave	Valor
x-amz-customauthorizer-signature	*dBwykzlb+fo+JmSGdwoGr8dyC2qB/IyLefJJr+rbCvmu9Jl4KHAA9DG+V+MMWu09YSA86+64Y3Gt4tOykpZqn9mnVB1wyxp+0bDZh8hmqUAUH3fwi3fPjBvCa4cwNuLQNqBZzbCvsluv7i2IMjEg+CPY0zrWt1jr9BikgGPDxWkjaeehbQHHTo357TegKs9pP30Uf4TrxypNmFswA5k7QIc01n4bIyRTm90OyZ94R4bdJsHNig1JePgnu0BvMGCEFE09jGjjszEHfgAUAQIWXiVGQj16BU1xKpTGSiTAwheLKUjITOEXAMPLECK3aHKYKY+d1vTvdthKtYHBq8MjhzJ0kggbt29VQJCb8RilN/P5+vcVniSXWPplyB5jkYs9UvG08REoy64AtizfUhvSul/r/F3VV8ITtQp3aXiUtcspACi6ca+tsDuXf3LzCwQQF/YSUy02u5XkWn+sto6KCkpNlkD0wU8gl3+kOzxrthnQ8gEajd5Iylx230iqcXo3osjPha7JDyWM5o+KEWckTe91I1mokDr5sJ4JXixvnJTVSx1li49IalW4en1DAkc1a0s2U2UNm236EXAMPLELotyh7h+flFeloZlAWQFHxRlXsPqiVKS1ZIUClaZWprh/orDJplpiWfBgBIOgokJIDGP9gwhXIIk7zWrGmWpMK9o=*

e. En la pestaña Body:

i. En el cuadro de opciones de formato de datos, elija Raw.

ii. En la lista de tipos de datos, elija JavaScript.

iii. En el campo de texto, introduzca esta carga de mensaje JSON para su mensaje de prueba:

```
{
```

```
    "data_mode": "test",
    "vibration": 200,
    "temperature": 40
}
```

4. Elija Send para enviar la solicitud.

   Si la solicitud se ha realizado correctamente, devuelve:

   ```
   {
       "message": "OK",
       "traceId": "ff35c33f-409a-ea90-b06f-fbEXAMPLE25c"
   }
   ```

   Si la respuesta es correcta, indica que su autorizador personalizado permitió la conexión AWS IoT y que el mensaje de prueba se entregó a Broker In AWS IoT Core.

   Si devuelve un error, revisa el mensaje de error*device_data_endpoint_address*, la cadena de firma y los demás valores del encabezado.

Guarde esta solicitud en Postman para utilizarla en la siguiente sección.

Paso 6: Visualizar los mensajes en el cliente de prueba MQTT

En el paso anterior, enviaste mensajes de un dispositivo AWS IoT simulado a través de Postman. La respuesta correcta indicó que su autorizador personalizado permitió la conexión a AWS IoT y que el mensaje de prueba fue entregado al agente en AWS IoT Core. En esta sección, utilizará el cliente de prueba MQTT de la AWS IoT consola para ver el contenido del mensaje de la misma forma que lo harían otros dispositivos y servicios.

Para ver los mensajes de prueba autorizados por su autorizador personalizado

1. En la AWS IoT consola, abra el cliente de prueba de MQTT.
2. En la pestaña Suscribirse al tema, en Filtro de temas, introduzca **`test/cust-auth/topic`**, que es el tema de mensaje utilizado en el ejemplo de Postman de la sección anterior.
3. Elija Suscribirse.

   Mantenga esta ventana visible para el siguiente paso.
4. En Postman, en la solicitud que creó para la sección anterior, elija Send.

Revise la respuesta para asegurarse de que se ha realizado correctamente. Si no es así, solucione el error, tal y como se describe en la sección anterior.

5. En el cliente de prueba de MQTT debería ver una nueva entrada que muestra el tema del mensaje y, si está expandido, la carga del mensaje de la solicitud que envió desde Postman.

   Si no ve sus mensajes en el cliente de prueba MQTT, he aquí algunas cosas que debe comprobar:

   - Asegúrese de que su solicitud de Postman ha sido devuelta correctamente. Si AWS IoT rechaza la conexión y devuelve un error, el mensaje de la solicitud no se pasa al intermediario de mensajes.
   - Asegúrese de que la Cuenta de AWS URL Región de AWS utilizada para abrir la AWS IoT consola sea la misma que la que utiliza en la URL de Postman.
   - Asegúrese de usar el punto de conexión adecuado para el autorizador personalizado. Es posible que el punto de conexión predeterminado de IoT no admita el uso de autorizadores personalizados con funciones de Lambda. En su lugar, puede usar las configuraciones de dominio para definir un nuevo punto de conexión y, a continuación, especificarlo para el autorizador personalizado.
   - Asegúrese de que ha introducido correctamente el tema en el cliente de prueba MQTT. El filtro del tema distingue entre mayúsculas y minúsculas. En caso de duda, también puedes suscribirte al **#** tema, que contiene todos los mensajes de MQTT que pasan por el intermediario de mensajes Cuenta de AWS y que Región de AWS se utilizan para abrir la consola. AWS IoT

Paso 7: Revisar los resultados y los siguientes pasos

En este tutorial:

- Ha creado una función de Lambda para que sea un controlador de autorizador personalizado.
- ha creado un autorizador personalizado con la firma de token habilitada.
- Ha probado su autorizador personalizado con el comando test-invoke-authorizer.
- Ha publicado un tema de MQTT con Postman y validar la solicitud con su autorizador personalizado.
- Ha utilizado el cliente de prueba de MQTT para ver los mensajes enviados desde su prueba de Postman.

Siguientes pasos

Después de enviar algunos mensajes desde Postman para verificar que el autorizador personalizado funciona, pruebe a experimentar para ver cómo afecta a los resultados cambiar diferentes aspectos de este tutorial. He aquí algunos ejemplos para empezar.

- Cambie la cadena de firma para que deje de ser válida y vea cómo se gestionan los intentos de conexión no autorizados. Debería obtener una respuesta de error, como esta, y el mensaje no debería aparecer en el cliente de prueba MQTT.

```
{
    "message": "Forbidden",
    "traceId": "15969756-a4a4-917c-b47a-5433e25b1356"
}
```

- Para obtener más información sobre cómo detectar los errores que pueden producirse al desarrollar y utilizar AWS IoT reglas, consulte. Monitorización AWS IoT

Paso 8: Eliminación

Si desea repetir este tutorial, puede que tenga que eliminar algunos de sus autorizadores personalizados. Solo Cuenta de AWS puede configurar un número limitado de autorizadores personalizados a la vez y puede obtener uno `LimitExceededException` si intenta añadir uno nuevo sin eliminar uno existente.

Para eliminar un autorizador personalizado (consola)

1. Abre la página de autorizadores personalizados de la AWS IoT consola y, en la lista de autorizadores personalizados, busca el autorizador personalizado que deseas eliminar.
2. Abra la página de detalles del autorizador personalizado y, en el menú Acciones, seleccione Editar.
3. Desmarque la casilla Activar autorizador y, a continuación, seleccione Actualizar.

   No puede eliminar un autorizador personalizado mientras esté activo.
4. En la página de detalles del autorizador personalizado, abra el menú Acciones y seleccione Eliminar.

Para eliminar un autorizador personalizado (AWS CLI)

1. Enumere los autorizadores personalizados que tiene instalados y busque el nombre del autorizador personalizado que desea eliminar.

   ```
   aws iot list-authorizers
   ```

2. Establezca el autorizador personalizado como `inactive` ejecutando este comando después de sustituir *`Custom_Auth_Name`* por el `authorizerName` del autorizador personalizado para eliminar.

   ```
   aws iot update-authorizer --status INACTIVE --authorizer-name Custom_Auth_Name
   ```

3. Elimine el autorizador personalizado ejecutando este comando después de sustituir *`Custom_Auth_Name`* por el `authorizerName` del autorizador personalizado para eliminar.

   ```
   aws iot delete-authorizer --authorizer-name Custom_Auth_Name
   ```

Tutorial: Monitorización de la humedad del suelo con una AWS IoT Raspberry Pi

En este tutorial se muestra cómo utilizar una Raspberry Pi, un sensor de humedad, y cómo AWS IoT controlar el nivel de humedad del suelo de una planta de interior o de un jardín. La Raspberry Pi ejecuta un código que lee el nivel de humedad y la temperatura del sensor y, a continuación, envía los datos a AWS IoT. Creas una regla AWS IoT que envía un correo electrónico a una dirección suscrita a un tema de Amazon SNS cuando el nivel de humedad cae por debajo de un umbral.

> **Note**
>
> Es posible que este tutorial no esté actualizado. Es posible que algunas referencias hayan sido reemplazadas desde que se publicó originalmente este tema.

Contenido

- Requisitos previos
- Con AWS IoT figuración
  - Paso 1: Crear la AWS IoT política

- Paso 2: Crea la AWS IoT cosa, el certificado y la clave privada
- Paso 3: Crear un tema y una suscripción a Amazon SNS
- Paso 4: Crea una AWS IoT regla para enviar un correo
- Configuración del dispositivo Raspberry Pi y el sensor de humedad

Requisitos previos

Para completar este tutorial, se necesita lo siguiente:

- Y. Cuenta de AWS
- Un usuario de IAM con permisos de administrador.
- Un equipo de desarrollo con Windows, macOS, Linux o Unix para obtener acceso a la consola de AWS IoT.
- Una Raspberry Pi 3B o 4B ejecutando el último Raspberry Pi OS. Para ver instrucciones de instalación, consulte Instalación de un sistema operativo en el sitio web de Rasberry Pi.
- Un monitor, teclado, ratón y conexión de red wifi o Ethernet para su Raspberry Pi.
- Un sensor de humedad compatible con Raspberry Pi. El sensor utilizado en este tutorial es un sensor de humedad capacitivo Adafruit STEMMA I2C con un conector hembra de 4 clavijas JST.

Con AWS IoT figuración

Para completar este tutorial, debe crear los siguientes recursos. Para conectar un dispositivo AWS IoT, se crea una cosa de IoT, un certificado de dispositivo y una AWS IoT política.

- Cualquier AWS IoT cosa.

  Un objeto representa un dispositivo físico (en este caso, su Rasberry Pi) e incluye metadatos estáticos sobre el dispositivo.
- Un certificado de dispositivo.

  Todos los dispositivos deben tener un certificado de dispositivo para conectarse a AWS IoT y autenticarse con el mismo.
- Una AWS IoT política.

  Cada certificado de dispositivo tiene una o más AWS IoT políticas asociadas. Estas políticas determinan a qué AWS IoT recursos puede acceder el dispositivo.

- Un certificado de CA AWS IoT raíz.

  Los dispositivos y otros clientes utilizan un certificado de CA AWS IoT raíz para autenticar el AWS IoT servidor con el que se comunican. Para obtener más información, consulte Autenticación del servidor.
- Una AWS IoT regla.

  Una regla incluye una consulta y una o varias acciones de regla. La consulta extrae datos de los mensajes del dispositivo para determinar si los datos de los mensajes deben procesarse. La acción de regla especifica qué hacer si los datos coinciden con la consulta.
- Un tema de Amazon SNS y una suscripción a un tema.

  La regla escucha los datos de humedad del dispositivo Raspberry Pi. Si el valor está por debajo de un umbral, envía un mensaje al tema de Amazon SNS. Amazon SNS envía ese mensaje a todas las direcciones de correo electrónico suscritas al tema.

Paso 1: Crear la AWS IoT política

Crea una AWS IoT política que permita a tu Raspberry Pi conectarse y enviar mensajes a AWS IoT.

1. En la consola de AWS IoT, si aparece un botón Empezar, elíjalo. De lo contrario, en el panel de navegación, expanda Seguridad y, a continuación, elija Políticas.
2. Si aparece el cuadro de diálogo Aún no tiene ninguna política, elija Crear una política. De lo contrario, seleccione Crear.
3. Introduzca un nombre para la AWS IoT política (por ejemplo,**`MoistureSensorPolicy`**).
4. En la sección Añadir instrucciones, sustituya la política existente por el siguiente JSON. Sustituya *`region`* y *`account`* por su Cuenta de AWS número Región de AWS and.

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

```
            "Effect": "Allow",
            "Action": "iot:Publish",
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/
RaspberryPi/shadow/update",
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/
RaspberryPi/shadow/delete",
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/
RaspberryPi/shadow/get"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "iot:Receive",
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/
RaspberryPi/shadow/update/accepted",
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/
RaspberryPi/shadow/delete/accepted",
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/
RaspberryPi/shadow/get/accepted",
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/
RaspberryPi/shadow/update/rejected",
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/
RaspberryPi/shadow/delete/rejected"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "iot:Subscribe",
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/things/
RaspberryPi/shadow/update/accepted",
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/things/
RaspberryPi/shadow/delete/accepted",
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/things/
RaspberryPi/shadow/get/accepted",
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/things/
RaspberryPi/shadow/update/rejected",
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/things/
RaspberryPi/shadow/delete/rejected"
            ]
        },
        {
```

```
            "Effect": "Allow",
            "Action": [
                "iot:GetThingShadow",
                "iot:UpdateThingShadow",
                "iot:DeleteThingShadow"
            ],
            "Resource": "arn:aws:iot:us-east-1:123456789012:thing/RaspberryPi"
        }
    ]
}
```

5. Seleccione Crear.

Paso 2: Crea la AWS IoT cosa, el certificado y la clave privada

Crea una cosa en el AWS IoT registro que represente tu Raspberry Pi.

1. En la consola de AWS IoT, en el panel de navegación, elija Administrar y, a continuación, seleccione Objetos.
2. Si aparece el cuadro de diálogo Aún no tiene ningún objeto, elija Registrar un objeto. De lo contrario, seleccione Crear.
3. En la página Crear AWS IoT cosas, selecciona Crear una sola cosa.
4. En la página Añadir su dispositivo al registro de dispositivos, escriba un nombre para el objeto de IoT (por ejemplo, **`RaspberryPi`**) y, a continuación, elija Siguiente. No puede modificar el nombre de un objeto una vez creado. Para cambiar el nombre de una cosa, debe crear otra nueva, asignarle el nuevo nombre y eliminar después la anterior.
5. En la página Agregar un certificado para el objeto, elija Crear certificado.
6. Elija los enlaces Descargar para descargar el certificado, la clave privada y el certificado de CA raíz.

   > **Important**
   >
   > Esta es la única vez que puede descargar el certificado y la clave privada.

7. Para activar el certificado, seleccione Activar. El certificado debe estar activo para que un dispositivo pueda conectarse a AWS IoT.
8. Elija Attach a policy (Asociar una política).

9. En Añadir una política para tu cosa, selecciona y MoistureSensorPolicy, a continuación, selecciona Registrar cosa.

Paso 3: Crear un tema y una suscripción a Amazon SNS

Cree un tema y una suscripción de Amazon SNS.

1. En la consola SNS de AWS, en el panel de navegación, seleccione Temas y, a continuación, seleccione Crear tema.
2. Elija el tipo Estándar e introduzca un nombre para el tema (por ejemplo, **`MoistureSensorTopic`**).
3. Escriba un nombre de visualización para el tema (por ejemplo, **`Moisture Sensor Topic`**). Este es el nombre que se muestra para el tema en la consola de Amazon SNS.
4. Seleccione Crear tema.
5. En la página de detalles del tema de Amazon SNS, seleccione Crear suscripción.
6. En Protocolo, elija Correo electrónico.
7. Para punto de conexión, introduzca su dirección de correo electrónico.
8. Seleccione Crear subscripción.
9. Abra su cliente de correo electrónico y busque un mensaje con el asunto **`MoistureSensorTopic`**. Abra el correo electrónico y elija el enlace Confirmar suscripción.

> **Important**
>
> No recibirá ninguna alerta por correo electrónico de este tema de Amazon SNS hasta que confirme la suscripción.

Debería recibir un mensaje de correo electrónico con el texto que escribió.

Paso 4: Crea una AWS IoT regla para enviar un correo

Una AWS IoT regla define una consulta y una o más acciones que se deben realizar cuando se recibe un mensaje de un dispositivo. El motor de AWS IoT reglas escucha los mensajes enviados por los dispositivos y utiliza los datos de los mensajes para determinar si se debe realizar alguna acción. Para obtener más información, consulte Reglas para AWS IoT.

En este tutorial el dispositivo Raspberry Pi publica mensajes en `aws/things/RaspberryPi/shadow/update`. Se trata de un tema de MQTT interno utilizado por los dispositivos y el servicio Thing Shadow. El Raspberry Pi publica mensajes que tienen el siguiente formato:

```
{
    "reported": {
        "moisture" : moisture-reading,
        "temp" : temperature-reading
    }
}
```

Puede crear una consulta que extraiga los datos de humedad y temperatura del mensaje entrante. También creará una acción de Amazon SNS que tome los datos y los envíe a los suscriptores de temas de Amazon SNS si la lectura de humedad está por debajo de un valor umbral.

Crear una regla de Amazon SNS

1. En la consola de AWS IoT, seleccione Enrutamiento de mensajes y, a continuación, selecciona Reglas. Si aparece el cuadro de diálogo You don't have any rules yet (Aún no tiene ninguna regla), elija Create a rule (Crear una regla). De lo contrario, seleccione Crear regla.
2. En la página de propiedades de la regla, introduzca un nombre de regla, por ejemplo**`MoistureSensorRule`**, y proporcione una breve descripción de la regla, por ejemplo **`Sends an alert when soil moisture level readings are too low`**.
3. Seleccione Siguiente y configure su sentencia SQL. Elija la versión SQL como 23 de marzo de 2016 e introduzca la siguiente AWS IoT sentencia de consulta SQL:

   ```
   SELECT * FROM '$aws/things/RaspberryPi/shadow/update/accepted' WHERE
    state.reported.moisture < 400
   ```

   Esta instrucción activa la acción de la regla cuando la lectura de `moisture` es menor que `400`.

   > **Note**
   >
   > Es posible que tenga que utilizar un valor diferente. Una vez que el código se ejecute en el dispositivo Raspberry Pi, si toca el sensor, lo coloca en agua o en una maceta, podrá ver los valores que se obtienen del sensor.

4. Elija Siguiente y adjunte las acciones de las reglas. Para la acción 1, elija Servicio de notificación simple. La descripción de esta acción de regla es Enviar un mensaje como una notificación push de SNS.
5. Para el tema de SNS, elija el tema en el que creó y deje el Paso 3: Crear un tema y una suscripción a Amazon SNS formato MoistureSensorTopicdel mensaje en RAW. En Rol de IAM, elija Crear un nuevo rol. Especifique un nombre para el rol (por ejemplo, **`LowMoistureTopicRole`**) y elija Crear rol.
6. Seleccione Siguiente para revisarla y, a continuación, seleccione Crear para crear la regla.

Configuración del dispositivo Raspberry Pi y el sensor de humedad

Inserte la tarjeta micro SD en el Raspberry Pi, conecte el monitor, el teclado, el ratón y, si no utiliza una conexión wifi, el cable Ethernet. No conecte aún el cable de alimentación.

Conecte el cable puente JST al sensor de humedad. El otro lado del puente tiene cuatro cables:

- Verde: I2C SCL
- Blanco: I2C SDA
- Rojo: alimentación (3,5 V)
- Negro: conexión a tierra

Sujete el dispositivo Raspberry Pi con el enchufe hembra Ethernet a la derecha. En esta orientación hay dos filas de clavijas GPIO en la parte superior. Conecte los cables del sensor de humedad a la fila inferior de clavijas en el orden que se indica a continuación. Comenzando por la clavija del extremo izquierdo, conecte el cable rojo (alimentación), el cable blanco (SDA) y el cable verde (SCL). Omita una clavija y, a continuación, conecte el cable negro (conexión a tierra). Para obtener más información, consulte Cableado de equipos Python.

Conecte el cable de alimentación al dispositivo Raspberry Pi y conecte el otro extremo a una toma de corriente para encenderlo.

Configuración del dispositivo Raspberry Pi

1. En Welcome to Raspberry Pi (Bienvenido a Raspberry Pi), elija Siguiente.
2. Elija el país, el idioma, la zona horaria y la distribución del teclado. Elija Siguiente.
3. Escriba una contraseña para el dispositivo Raspberry Pi y, a continuación, elija Siguiente.

4. Elija una red wifi y, a continuación, elija Siguiente. Si no utiliza una red wifi, elija Skip (Omitir).
5. Elija Siguiente para comprobar si hay actualizaciones de software. Cuando se completen las actualizaciones, elija Restart (Reiniciar) para reiniciar el dispositivo Raspberry Pi.

Una vez que se inicie el dispositivo Raspberry Pi, habilite la interfaz de I2C.

1. En la esquina superior izquierda del escritorio de Raspbian, elija el icono de Raspberry, elija Preferencias y, a continuación, elija Configuración de Raspberry Pi.
2. En la pestaña Interfaces, en I2C, elija Habilitar.
3. Seleccione Aceptar.

Las bibliotecas del sensor de humedad STEMMA de Adafruit están diseñadas para ello. CircuitPython Para ejecutarlas en un dispositivo Raspberry Pi, debe instalar la última versión de Python 3.

1. Ejecute los siguientes comandos desde un símbolo del sistema para actualizar el software del dispositivo Raspberry Pi:

   ```
   sudo apt-get update
   ```

   ```
   sudo apt-get upgrade
   ```
2. Ejecute el siguiente comando para actualizar la instalación de Python 3:

   ```
   sudo pip3 install --upgrade setuptools
   ```
3. Ejecute el siguiente comando para instalar las bibliotecas de GPIO de Raspberry Pi:

   ```
   pip3 install RPI.GPIO
   ```
4. Ejecute el siguiente comando para instalar las bibliotecas de Adafruit Blinka:

   ```
   pip3 install adafruit-blinka
   ```

   Para obtener más información, consulte Instalación de CircuitPython bibliotecas en Raspberry Pi.
5. Ejecute el siguiente comando para instalar las bibliotecas de Adafruit Seesaw:

   ```
   sudo pip3 install adafruit-circuitpython-seesaw
   ```
6. Ejecute el siguiente comando para instalar el SDK de AWS IoT dispositivos para Python:

```
pip3 install AWSIoTPythonSDK
```

El dispositivo Raspberry Pi ahora tiene todas las bibliotecas necesarias. Cree un archivo denominado **`moistureSensor.py`** y copie el siguiente código Python en el archivo:

```
from adafruit_seesaw.seesaw import Seesaw
from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTShadowClient
from board import SCL, SDA

import logging
import time
import json
import argparse
import busio

# Shadow JSON schema:
#
# {
#   "state": {
#       "desired":{
#           "moisture":<INT VALUE>,
#           "temp":<INT VALUE>
#       }
#   }
# }

# Function called when a shadow is updated
def customShadowCallback_Update(payload, responseStatus, token):

    # Display status and data from update request
    if responseStatus == "timeout":
        print("Update request " + token + " time out!")

    if responseStatus == "accepted":
        payloadDict = json.loads(payload)
        print("~~~~~~~~~~~~~~~~~~~~~~")
        print("Update request with token: " + token + " accepted!")
        print("moisture: " + str(payloadDict["state"]["reported"]["moisture"]))
        print("temperature: " + str(payloadDict["state"]["reported"]["temp"]))
        print("~~~~~~~~~~~~~~~~~~~~~~\n\n")
```

```
    if responseStatus == "rejected":
        print("Update request " + token + " rejected!")

# Function called when a shadow is deleted
def customShadowCallback_Delete(payload, responseStatus, token):

     # Display status and data from delete request
    if responseStatus == "timeout":
        print("Delete request " + token + " time out!")

    if responseStatus == "accepted":
        print("~~~~~~~~~~~~~~~~~~~~~~~")
        print("Delete request with token: " + token + " accepted!")
        print("~~~~~~~~~~~~~~~~~~~~~~~\n\n")

    if responseStatus == "rejected":
        print("Delete request " + token + " rejected!")


# Read in command-line parameters
def parseArgs():

    parser = argparse.ArgumentParser()
    parser.add_argument("-e", "--endpoint", action="store", required=True, dest="host",
 help="Your device data endpoint")
    parser.add_argument("-r", "--rootCA", action="store", required=True,
 dest="rootCAPath", help="Root CA file path")
    parser.add_argument("-c", "--cert", action="store", dest="certificatePath",
 help="Certificate file path")
    parser.add_argument("-k", "--key", action="store", dest="privateKeyPath",
 help="Private key file path")
    parser.add_argument("-p", "--port", action="store", dest="port", type=int,
 help="Port number override")
    parser.add_argument("-n", "--thingName", action="store", dest="thingName",
 default="Bot", help="Targeted thing name")
    parser.add_argument("-id", "--clientId", action="store", dest="clientId",
 default="basicShadowUpdater", help="Targeted client id")

    args = parser.parse_args()
    return args


# Configure logging
# AWSIoTMQTTShadowClient writes data to the log
```

```
def configureLogging():

    logger = logging.getLogger("AWSIoTPythonSDK.core")
    logger.setLevel(logging.DEBUG)
    streamHandler = logging.StreamHandler()
    formatter = logging.Formatter('%(asctime)s - %(name)s - %(levelname)s -
 %(message)s')
    streamHandler.setFormatter(formatter)
    logger.addHandler(streamHandler)


# Parse command line arguments
args = parseArgs()

if not args.certificatePath or not args.privateKeyPath:
    parser.error("Missing credentials for authentication.")
    exit(2)

# If no --port argument is passed, default to 8883
if not args.port:
    args.port = 8883


# Init AWSIoTMQTTShadowClient
myAWSIoTMQTTShadowClient = None
myAWSIoTMQTTShadowClient = AWSIoTMQTTShadowClient(args.clientId)
myAWSIoTMQTTShadowClient.configureEndpoint(args.host, args.port)
myAWSIoTMQTTShadowClient.configureCredentials(args.rootCAPath, args.privateKeyPath,
 args.certificatePath)

# AWSIoTMQTTShadowClient connection configuration
myAWSIoTMQTTShadowClient.configureAutoReconnectBackoffTime(1, 32, 20)
myAWSIoTMQTTShadowClient.configureConnectDisconnectTimeout(10) # 10 sec
myAWSIoTMQTTShadowClient.configureMQTTOperationTimeout(5) # 5 sec

# Initialize Raspberry Pi's I2C interface
i2c_bus = busio.I2C(SCL, SDA)

# Intialize SeeSaw, Adafruit's Circuit Python library
ss = Seesaw(i2c_bus, addr=0x36)

# Connect to AWS IoT
myAWSIoTMQTTShadowClient.connect()
```

```
# Create a device shadow handler, use this to update and delete shadow document
deviceShadowHandler =
 myAWSIoTMQTTShadowClient.createShadowHandlerWithName(args.thingName, True)

# Delete current shadow JSON doc
deviceShadowHandler.shadowDelete(customShadowCallback_Delete, 5)

# Read data from moisture sensor and update shadow
while True:

    # read moisture level through capacitive touch pad
    moistureLevel = ss.moisture_read()

    # read temperature from the temperature sensor
    temp = ss.get_temp()

    # Display moisture and temp readings
    print("Moisture Level: {}".format(moistureLevel))
    print("Temperature: {}".format(temp))

    # Create message payload
    payload = {"state":{"reported":{"moisture":str(moistureLevel),"temp":str(temp)}}}

    # Update shadow
    deviceShadowHandler.shadowUpdate(json.dumps(payload), customShadowCallback_Update,
 5)
    time.sleep(1)
```

Guarde el archivo en un lugar donde pueda encontrarlo. Ejecute `moistureSensor.py` desde la línea de comandos con los siguientes parámetros:

punto de conexión

Tu AWS IoT punto final personalizado. Para obtener más información, consulte API REST de sombra de dispositivo.

rootCA

La ruta completa a su certificado de CA AWS IoT raíz.

cert

La ruta completa al certificado de su AWS IoT dispositivo.

clave

La ruta completa a la clave privada AWS IoT del certificado de su dispositivo.

thingName

El nombre de la cosa (en este caso, `RaspberryPi`).

clientId

El ID de cliente de MQTT. Utilice `RaspberryPi`.

La línea de comandos debería tener este aspecto:

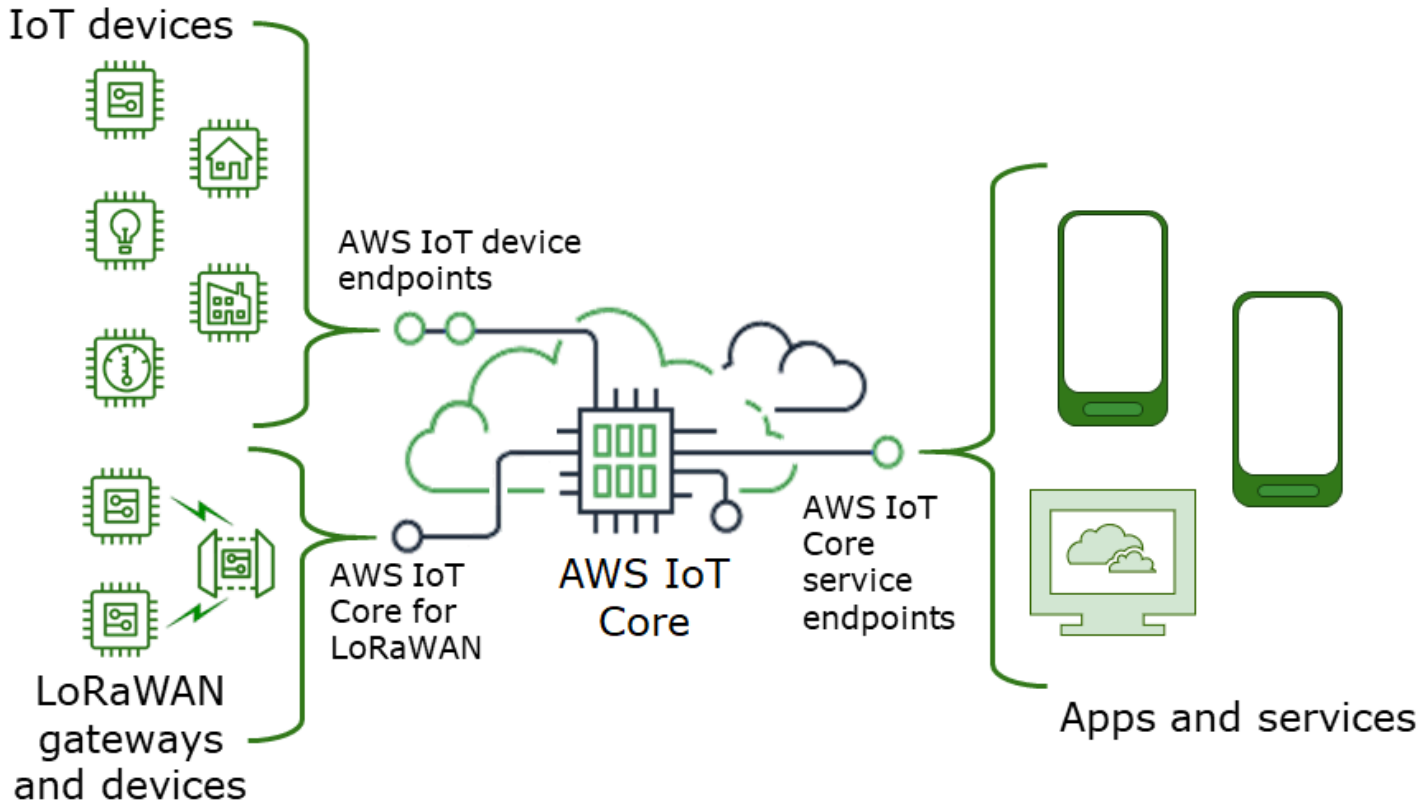
```
python3 moistureSensor.py --endpoint your-endpoint --rootCA ~/certs/
AmazonRootCA1.pem --cert ~/certs/raspberrypi-certificate.pem.crt --key
~/certs/raspberrypi-private.pem.key --thingName RaspberryPi --clientId
RaspberryPi
```

Pruebe a tocar el sensor, colocarlo en una maceta o ponerlo en un vaso de agua para ver cómo responde a distintos niveles de humedad. Si es necesario, puede cambiar el valor del umbral en la opción `MoistureSensorRule`. Cuando la lectura del sensor de humedad es inferior al valor especificado en la declaración de consulta SQL de la regla, AWS IoT publica un mensaje en el tema Amazon SNS. Debería recibir un mensaje de correo electrónico que incluya los datos de humedad y temperatura.

Una vez que haya verificado la recepción de los mensajes de correo electrónico de Amazon SNS, pulse CTRL+C para detener el programa Python. Es poco probable que el programa Python envíe suficientes mensajes para incurrir en gastos, pero se recomienda detener el programa cuando haya terminado.

Connect to AWS IoT Core

AWS IoT Core admite conexiones con dispositivos de IoT, pasarelas inalámbricas, servicios y aplicaciones. Los dispositivos se conectan para AWS IoT Core poder enviar y recibir datos de AWS IoT servicios y otros dispositivos. Las aplicaciones y otros servicios también se conectan AWS IoT Core para controlar y administrar los dispositivos de IoT y procesar los datos de su solución de IoT. En esta sección se describe cómo elegir la mejor forma de conectarse y comunicarse AWS IoT Core para cada aspecto de su solución de IoT.



Hay varias formas de interactuar con. AWS IoT Las aplicaciones y los servicios pueden utilizar las regiones AWS IoT Core: puntos de conexión del plano de control y puntos finales de la LoRa WAN y a las que se pueden conectar AWS IoT Core los dispositivos AWS IoT Core mediante AWS IoT puntos finales del dispositivo o desde ellas.

AWS IoT Core: puntos de conexión del plano de control

Los AWS IoT Core puntos finales del plano de control proporcionan acceso a las funciones que controlan y gestionan la solución. AWS IoT

- Puntos de conexión

  Los puntos de conexión del plano de control de AWS IoT Core y del plano de control de AWS IoT Core Device Advisor son específicos de cada región y se enumeran en puntos de conexión y cuotas de AWS IoT Core. Los formatos de los puntos de conexión son los siguientes.

Finalidad del punto de conexión	Formato de punto de conexión	Da servicio a
Plano de control de AWS IoT Core	AWS IoT Puntos finales del plano de control	AWS IoT API de control Plane
AWS IoT Core Device Advisor: plano de control	`api.iotdeviceadvisor.` *`aws-region`* `.amazonaws.com`	AWS IoT Core API de plano de control de Device Advisor

  - IPv4 puntos finales: los IPv4 puntos finales solo admiten IPv4 tráfico y están disponibles en todas las regiones.

    IPv4 los puntos finales utilizan la siguiente convención de nomenclatura:

    ```
    iot.aws-region.amazonaws.com
    ```

    Por ejemplo, el nombre del IPv4 punto final de la región us-east-1 es. `iot.us-east-1.amazonaws.com`

  - Puntos finales de doble pila (IPv4 y IPv6): los puntos finales de doble pila admiten tanto el tráfico como el tráfico. IPv4 IPv6 Cuando se realiza una solicitud a un punto final de doble pila, la URL del punto final pasa a ser una IPv6 o una IPv4 dirección, según el protocolo utilizado por la red y el cliente.

    Los puntos de conexión de doble pila utilizan la siguiente convención de nomenclatura:

    ```
    iot.aws-region.api.aws
    ```

    Por ejemplo, el nombre del punto de conexión de doble pila para la región us-east-1 es `iot.us-east-1.api.aws`.

- SDKs y herramientas

AWS SDKsProporcionan soporte en un idioma específico para los AWS IoT Core APIs demás AWS servicios y para APIs ellos. The AWS Mobile SDKs proporciona a los desarrolladores de aplicaciones soporte específico de la plataforma para la AWS IoT Core API y otros AWS servicios en los dispositivos móviles.

AWS CLIProporciona acceso desde la línea de comandos a las funciones proporcionadas por los puntos finales del servicio. AWS IoT AWS Tools for PowerShell proporciona herramientas para administrar los AWS servicios y recursos en el entorno de secuencias de PowerShell comandos.

- Autenticación

  Los puntos finales del servicio utilizan los usuarios y las AWS credenciales de IAM para autenticar a los usuarios.

- Más información

  Para obtener más información y enlaces a las referencias de los SDK, consulte the section called "Conéctese a puntos finales AWS IoT Core de servicio".

AWS IoT puntos finales del dispositivo

Los puntos finales del AWS IoT dispositivo admiten la comunicación entre sus dispositivos de IoT y AWS IoT.

- Puntos de conexión

  Los puntos finales del dispositivo son compatibles AWS IoT Core y funcionan. AWS IoT Device Management Son específicos para usted Cuenta de AWS y puede ver cuáles son con el describe-endpoint comando.

Finalidad del punto de conexión	Formato de punto de conexión	Da servicio a
Plano de datos de AWS IoT Core	Consulte ???.	AWS IoT API de Data Plane
Datos de trabajos de AWS IoT Device Management	Consulte ???.	AWS IoT API de plano de datos de Jobs

Finalidad del punto de conexión	Formato de punto de conexión	Da servicio a
AWS IoT Device Advisor: plano de datos	Consulte ???.	No aplicable
- tunelización segura de AWS IoT Device Management	`api.tunneling.iot.` *`aws-region`* `.amazonaws.com`	AWS IoT API de tunelización segura

Si utiliza puntos de enlace de doble pila (IPv4 y IPv6) para las operaciones del plano de datos, utilice el tipo de punto final. `iot:Data-ATS` `iot:Jobs`solo se puede usar para. IPv4 Para obtener más información sobre estos puntos de conexión y las funciones que admiten, consulte the section called “AWS IoT datos del dispositivo y puntos finales de servicio”.

- SDKs

  El AWS IoT dispositivo SDKs admite los protocolos Message Queueing Telemetry Transport (MQTT) y WebSocket Secure (WSS), con los que se comunican los dispositivos, en función de un idioma específico. AWS IoTAWS Aplicaciones SDKstambién admiten las comunicaciones de los dispositivos MQTT y otros servicios de los dispositivos AWS IoT APIs móviles. APIs AWS

- Autenticación

  Los puntos finales del dispositivo utilizan certificados X.509 o usuarios de AWS IAM con credenciales para autenticar a los usuarios.

- Más información

  Para obtener más información y enlaces a las referencias de los SDK, consulte the section called “AWS IoT Dispositivo SDKs”.

AWS IoT Core para dispositivos y puertas de enlace WAN LoRa

AWS IoT Core para LoRa WAN conecta puertas de enlace y dispositivos inalámbricos a. AWS IoT Core

- Puntos de conexión

AWS IoT Core para LoRa WAN administra las conexiones de la puerta de enlace a los puntos finales específicos de la cuenta y la región. AWS IoT Core Las pasarelas se pueden conectar al punto final del servidor de configuración y actualización (CUPS) de su cuenta que AWS IoT Core proporciona la WAN. LoRa

Finalidad del punto de conexión	Formato de punto de conexión	Da servicio a
Servidor de configuración y actualización (CUPS)	*account-specific-prefix* .cups.lorawan. *aws-region* .amazonaws.com:443	Comunicación de puerta de enlace con el servidor de configuración y actualización proporcionada AWS IoT Core por la WAN LoRa
LoRaServidor de red WAN (LNS)	*account-specific-prefix* .gateway.lorawan. *aws-region* .amazonaws.com:443	Comunicación de puerta de enlace con el servidor de red LoRa WAN proporcionada AWS IoT Core por LoRa WAN

- SDKs

  El AWS SDK admite AWS IoT Core la API AWS IoT inalámbrica en la que se basa la LoRa WAN. Para obtener más información, consulte AWS SDKs los kits de herramientas.

- Autenticación

  AWS IoT Core para las comunicaciones LoRa entre dispositivos WAN, utilice certificados X.509 para proteger las comunicaciones con. AWS IoT

- Más información

  Para obtener más información sobre la configuración y la conexión de dispositivos inalámbricos, consulte las regiones y AWS IoT Core los LoRa puntos finales de la WAN.

Conéctese a puntos finales AWS IoT Core de servicio

Puede acceder a las funciones del AWS IoT Core plano de control mediante el AWS CLI AWS SDK del idioma que prefiera o llamando directamente a la API REST. Te recomendamos que utilices

el SDK AWS CLI o un AWS SDK para interactuar con ellos, AWS IoT Core ya que incorporan las mejores prácticas para llamar a AWS los servicios. Puedes llamar APIs directamente al REST, pero debes proporcionar las credenciales de seguridad necesarias para poder acceder a la API.

> **Note**
>
> Los dispositivos IoT deberían usar AWS IoT Dispositivo SDKs. SDKs Los dispositivos están optimizados para su uso en dispositivos, admiten la comunicación MQTT y son compatibles con AWS IoT los dispositivos AWS IoT APIs más utilizados. Para obtener más información sobre el dispositivo SDKs y las funciones que ofrece, consulteAWS IoT Dispositivo SDKs. Los dispositivos móviles deberían usar AWS Aplicaciones SDKs. Los dispositivos móviles SDKs admiten las AWS IoT APIs comunicaciones de los dispositivos MQTT y otros AWS servicios APIs de los dispositivos móviles. Para obtener más información sobre el dispositivo móvil SDKs y las funciones que ofrecen, consulteAWS Aplicaciones SDKs.

Puede utilizar AWS Amplify las herramientas y los recursos de las aplicaciones web y móviles para conectarse más fácilmente AWS IoT Core. Para obtener más información sobre cómo conectarse AWS IoT Core mediante Amplify, consulte la documentación PubSubde Amplify.

En las siguientes secciones se describen las herramientas SDKs que puede utilizar para desarrollar e interactuar con otros AWS IoT AWS servicios. Para ver la lista completa de AWS herramientas y kits de desarrollo disponibles para crear y administrar aplicaciones AWS, consulte Herramientas sobre las que construir AWS.

AWS CLI para AWS IoT Core

AWS CLI Proporciona acceso desde la línea de comandos a. AWS APIs

- Instalación

  Para obtener información acerca de cómo instalar el AWS CLI, consulte Instalación del. AWS CLI
- Autenticación

  AWS CLI Utiliza las credenciales de su Cuenta de AWS.
- Referencia

  Para obtener información sobre los AWS CLI comandos de estos AWS IoT Core servicios, consulte:

- AWS CLI Referencia de comandos para IoT
- AWS CLI Referencia de comandos para datos de IoT
- AWS CLI Referencia de comandos para datos de trabajos de IoT
- AWS CLI Referencia de comandos para túneles seguros de IoT

Para ver las herramientas para administrar AWS los servicios y los recursos en el entorno de PowerShell secuencias de comandos, consulte AWS Herramientas para. PowerShell

AWS SDKs

Con AWS SDKs, sus aplicaciones y dispositivos compatibles pueden realizar llamadas AWS IoT APIs y APIs los demás AWS servicios. En esta sección se proporcionan enlaces a AWS SDKs la documentación de referencia de la API para cada APIs uno de los AWS IoT Core servicios.

Son AWS SDKs compatibles con estos AWS IoT Core APIs

- AWS IoT
- AWS IoT Plano de datos
- AWS IoT Plano de datos de Jobs
- AWS IoT Tunelización segura
- AWS IoT inalámbrico

C++

Para instalar el AWS SDK para C++ y usarlo para conectarse a AWS IoT:

1. Siga las instrucciones de Introducción al uso del AWS SDK para C++

   Estas instrucciones describen cómo:

   - Instalar y compilar el SDK a partir de archivos de origen.
   - Proporcionar las credenciales para usar el SDK con su Cuenta de AWS.
   - Inicializar y cerrar el SDK en su aplicación o servicio.
   - Crea un CMake proyecto para crear tu aplicación o servicio
2. Crear y ejecutar una aplicación de ejemplo. Para ver aplicaciones de ejemplo que usan el SDK de AWS para C++, consulte Ejemplos de código de AWS SDK para C++.

Documentación de los AWS IoT Core servicios que AWS SDK para C++ admite

- AWS: :IoTClient»: documentación de referencia
- Documentación de referencia de Aws: :Io TData Plane: TData PlaneClient :Io
- Documentación de referencia de Aws: :Io TJobsDataPlane: :Io TJobs DataPlaneClient
- Documentación de referencia de Aws: :Io TSecure Tunneling: :Io TSecure TunnelingClient

Go

Para instalar el AWS SDK for Go y usarlo para conectarse a AWS IoT:

1. Siga las instrucciones que se indican en Cómo empezar con AWS SDK for Go

   Estas instrucciones describen cómo:

   - Instale el AWS SDK for Go
   - Obtener las claves de acceso del SDK para acceder a su Cuenta de AWS.
   - Importar paquetes en el código fuente de nuestras aplicaciones o servicios.
2. Crear y ejecutar una aplicación de ejemplo. Para ver aplicaciones de ejemplo que usan AWS SDK for Go, consulte Ejemplos de código de AWS SDK for Go.

Documentación de los AWS IoT Core servicios que AWS SDK for Go admite

- Documentación de referencia de IoT
- Documentación de referencia de Io TData Plane
- Documentación de TJobs DataPlane referencia de Io
- Documentación de referencia TSecure sobre Io Tunneling

Java

Para instalar el AWS SDK for Java y usarlo para conectarse a AWS IoT:

1. Siga las instrucciones de Primeros pasos con AWS SDK for Java 2.x

   Estas instrucciones describen cómo:

   - Regístrese AWS y cree un usuario de IAM
   - Descargar el SDK

- Configure AWS las credenciales y la región
- Usar el SDK con Apache Maven.
- Usar el SDK con Gradle.

2. Cree y ejecute una aplicación de ejemplo con uno de los ejemplos de código de AWS SDK for Java 2.x.
3. Consulte la documentación de referencia de la API del SDK.

Documentación de los AWS IoT Core servicios que AWS SDK for Java admite

- IotClient documentación de referencia
- IotDataPlaneClient documentación de referencia
- IotJobsDataPlaneClient documentación de referencia
- Documentación de TSecure TunnelingClient referencia de Io

JavaScript

Para instalar el AWS SDK para JavaScript y usarlo para conectarse a AWS IoT:

1. Siga las instrucciones de configuración del AWS SDK para JavaScript. Estas instrucciones se aplican al uso de AWS SDK para JavaScript en el navegador y con Node.JS. Asegúrese de seguir las instrucciones que se aplican a la instalación.

   Estas instrucciones describen cómo:

   - Comprobar si se cumplen los requisitos previos.
   - Instale el SDK para JavaScript
   - Cargue el SDK para JavaScript
2. Cree y ejecute una aplicación de ejemplo para empezar a utilizar el SDK, tal y como se describe en la opción de introducción de su entorno.
   - Comience con el AWS SDK para JavaScript el navegador, o
   - Comience con el AWS SDK para JavaScript Node.js

Documentación de los AWS IoT Core servicios que AWS SDK para JavaScript admite

- `AWS.Iot reference documentation`

- `AWS.IotData reference documentation`
- `AWS.IotJobsDataPlane reference documentation`
- `AWS.IotSecureTunneling reference documentation`

.NET

Para instalar el AWS SDK for .NET y usarlo para conectarse a AWS IoT:

1. Siga las instrucciones de Configuración del AWS SDK for .NET entorno
2. Siga las instrucciones de Configuración AWS SDK for .NET del proyecto

   Estas instrucciones describen cómo:

   - Iniciar un nuevo proyecto.
   - Obtenga y configure AWS las credenciales
   - Instale los paquetes AWS del SDK
3. Cree y ejecute uno de los programas de ejemplo de Cómo trabajar con AWS los servicios del AWS SDK para .NET
4. Consulte la documentación de referencia de la API del SDK.

Documentación de los AWS IoT Core servicios que AWS SDK for .NET admite

- Documentación de referencia de Amazon.IoT.Model
- Amazon. IotData.Documentación de referencia del modelo
- Amazon.IO TJobs DataPlane .Documentación de referencia para modelos
- Amazon.IO Tunneling.Documentación TSecure de referencia sobre el modelo

PHP

Para instalar el AWS SDK for PHP y usarlo para conectarse a AWS IoT:

1. Siga las instrucciones de Introducción a la versión 3 AWS SDK for PHP

   Estas instrucciones describen cómo:

   - Comprobar si se cumplen los requisitos previos.
   - Instalar el SDK.

- Aplicar el SDK a un script de PHP.

2. Cree y ejecute una aplicación de ejemplo con uno de los ejemplos de código de AWS SDK for PHP (versión 3).

Documentación de los AWS IoT Core servicios que AWS SDK for PHP admite

- Documentación de TClient referencia de IO
- Documentación de TData PlaneClient referencia de Io
- Documentación de TJobs DataPlaneClient referencia de Io
- Documentación de TSecure TunnelingClient referencia de Io

Python

Para instalar el AWS SDK for Python (Boto3) y usarlo para conectarse a AWS IoT:

1. Siga las instrucciones que se detallan en Inicio rápido de AWS SDK for Python (Boto3).

   Estas instrucciones describen cómo:

   - Instalar el SDK
   - Configurar el SDK.
   - Utilizar el SDK en el código.

2. Cree y ejecute un programa de ejemplo que utilice el AWS SDK for Python (Boto3).

   Este programa muestra las opciones de registro configuradas actualmente en la cuenta. Tras instalar el SDK y configurarlo para su cuenta, debería poder ejecutar este programa.

```
import boto3
import json

# initialize client
iot = boto3.client('iot')

# get current logging levels, format them as JSON, and write them to stdout
response = iot.get_v2_logging_options()
print(json.dumps(response, indent=4))
```

Para obtener más información sobre la función que se utiliza en este ejemplo, consulte the section called “Configure el AWS IoT registro”.

Documentación de los AWS IoT Core servicios que AWS SDK for Python (Boto3) soporta

- Documentación de referencia de IoT
- Documentación de referencia de Io TData Plane
- Documentación de TJobs DataPlane referencia de Io
- Documentación de referencia TSecure sobre Io Tunneling

Ruby

Para instalar el AWS SDK for Ruby y usarlo para conectarse a AWS IoT:

- Siga las instrucciones de Primeros pasos con el AWS SDK for Ruby

  Estas instrucciones describen cómo:

  - Instalar el SDK
  - Configurar el SDK.
- Cree y ejecute el tutorial de Hello World.

Documentación de los AWS IoT Core servicios compatibles con el AWS SDK for Ruby

- Documentación de referencia de Aws::IoT::Client
- Documentación de referencia de Aws: :Io TData Plane: :Client
- Documentación de referencia de Aws: :Io TJobsDataPlane: :Client
- Documentación de referencia de Aws: :Io TSecure Tunneling: :Client

AWS Aplicaciones SDKs

The AWS Mobile SDKs proporciona a los desarrolladores de aplicaciones móviles soporte específico para la plataforma APIs de los servicios AWS IoT Core , la comunicación de dispositivos IoT mediante MQTT y otros servicios. APIs AWS

Android

AWS Mobile SDK para Android

AWS Mobile SDK para Android Contiene una biblioteca, ejemplos y documentación para que los desarrolladores puedan crear aplicaciones móviles conectadas con ellas. AWS Este SDK también incluye soporte para las comunicaciones y las llamadas a los APIs AWS IoT Core servicios desde dispositivos MQTT. Para obtener más información, consulte los siguientes temas:

- AWS SDK móvil para Android en adelante GitHub
- AWS Léame el SDK móvil para Android
- AWS Ejemplos de SDK para Android para dispositivos móviles
- AWS Referencia de la API de SDK for Android
- AWSIoTClient Documentación de referencia de clases

iOS

AWS Mobile SDK for iOS

AWS Mobile SDK for iOS Se trata de un kit de desarrollo de software de código abierto, distribuido bajo una licencia de código abierto de Apache. El SDK para iOS proporciona una biblioteca, ejemplos de código y documentación para ayudar a los desarrolladores a crear aplicaciones móviles conectadas mediante AWS. Este SDK también incluye soporte para las comunicaciones y las llamadas a los APIs AWS IoT Core servicios desde dispositivos MQTT. Para obtener más información, consulte los siguientes temas:

- AWS Mobile SDK for iOS activado GitHub
- AWS SDK for iOS (Léame)
- AWS Ejemplos de SDK for iOS
- AWS IoT Documentos de referencia de clases en el AWS SDK para iOS

RESTO APIs de los AWS IoT Core servicios

Se puede llamar directamente al RESTO APIs de los AWS IoT Core servicios mediante solicitudes HTTP.

- URL del punto de conexión

Los puntos de enlace del servicio que exponen el resto APIs de los AWS IoT Core servicios varían según la región y se enumeran en AWS IoT Core puntos de enlace y cuotas. Debe usar el punto final de la región que tiene los AWS IoT recursos a los que desea acceder, ya que AWS IoT los recursos son específicos de la región.

- Autenticación

  El RESTO APIs de los AWS IoT Core servicios utilizan las credenciales de AWS IAM para la autenticación. Para obtener más información, consulte Firmar solicitudes de AWS API en la Referencia AWS general.

- Referencia de la API

  Para obtener información sobre las funciones específicas que proporciona el REST APIs de los AWS IoT Core servicios, consulte:

  - Referencia de la API para IoT.
  - Referencia de la API para datos de IoT.
  - Referencia de la API para datos de trabajos de IoT.
  - Referencia de la API para tunelización segura de IoT.

Connect dispositivos a AWS IoT

Los dispositivos se conectan a otros servicios AWS IoT y a través de ellos AWS IoT Core. De este modo AWS IoT Core, los dispositivos envían y reciben mensajes mediante puntos de conexión específicos de su cuenta. Los the section called “AWS IoT Dispositivo SDKs” admiten comunicaciones entre dispositivos mediante los protocolos MQTT y WSS. Para obtener más información acerca de los protocolos que pueden usar los dispositivos, consulte the section called “Protocolos de comunicación de dispositivos”.

Agente de mensajes

AWS IoT gestiona la comunicación de los dispositivos a través de un intermediario de mensajes. Los dispositivos y los clientes publican los mensajes en el agente de mensajes y también se suscriben a los mensajes que publica el agente de mensajes. Los mensajes se identifican mediante un tema definido por la aplicación. Cuando el agente de mensajes recibe un mensaje publicado por un dispositivo o un cliente, lo vuelve a publicar en los dispositivos y clientes que estén suscritos a ese tema. El agente de mensajes también reenvía los mensajes al motor de AWS IoT reglas, que puede actuar sobre el contenido del mensaje.

AWS IoT seguridad de los mensajes

Conexiones de dispositivos que se AWS IoT van a utilizar the section called “Certificados de cliente X.509” y AWS firma V4 para la autenticación. Las comunicaciones de los dispositivos están protegidas por la versión 1.3 de TLS y AWS IoT requieren que los dispositivos envíen la extensión de indicación del nombre del servidor (SNI) al conectarse. Para obtener más información, consulte Seguridad del transporte en. AWS IoT

AWS IoT datos del dispositivo y puntos finales de servicio

> **Important**
>
> Puede almacenar o almacenar en caché los puntos de conexión de su dispositivo. Esto significa que no tendrá que consultar la API de `DescribeEndpoint` cada vez que conecte un dispositivo nuevo. Los puntos de conexión no cambiarán después de AWS IoT Core crearlos para tu cuenta.

Cada cuenta tiene varios puntos de conexión de dispositivo que son exclusivos de la cuenta y admiten funciones de IoT específicas. Los puntos finales de datos del AWS IoT dispositivo admiten un publish/subscribe protocolo diseñado para las necesidades de comunicación de los dispositivos de IoT; sin embargo, otros clientes, como aplicaciones y servicios, también pueden usar esta interfaz si su aplicación requiere las funciones especializadas que proporcionan estos puntos finales. Los puntos finales de servicio de los AWS IoT dispositivos permiten el acceso centrado en los dispositivos a los servicios de seguridad y administración.

Para conocer el punto de conexión de datos del dispositivo de tu cuenta, puedes encontrarlo en la página de configuración de la consola. AWS IoT Core

Si quiere saber cuál es el punto de conexión del dispositivo de su cuenta para una finalidad concreta, incluido el punto de conexión de datos del dispositivo, utilice el comando describe-endpoint de la CLI que se muestra aquí o la API de REST `DescribeEndpoint` y proporcione el valor del parámetro *`endpointType`* de la siguiente tabla.

```
aws iot describe-endpoint --endpoint-type endpointType
```

Este comando devuelve un *`iot-endpoint`* en el siguiente formato:*`account-specific-prefix`*`.iot.`*`aws-region`*`.amazonaws.com`.

Cada cliente tiene un punto de conexión `iot:Data-ATS` y `iot:Data`. Cada punto de conexión utiliza un certificado X.509 para autenticar al cliente. Recomendamos a los clientes que utilicen el tipo de punto de conexión `iot:Data-ATS` más reciente para evitar problemas relacionados con la desconfianza generalizada en las entidades de certificación de Symantec. Proporcionamos el `iot:Data` punto final para que los dispositivos recuperen datos de puntos finales antiguos que utilizan VeriSign certificados de compatibilidad con versiones anteriores. Para obtener más información, consulte Autenticación del servidor.

AWS IoT puntos finales para dispositivos

Finalidad del punto de conexión	Valor de ***endpointType***	Description (Descripción)
Operaciones de plano de datos de AWS IoT Core	`iot:Data-ATS`	Se utiliza para enviar y recibir datos desde y hacia los componentes agente de mensajes, Device Shadow y Rules Engine de AWS IoT. `iot:Data-ATS` devuelve un punto de conexión de datos firmados por ATS.
Operaciones de plano de datos de AWS IoT Core(here dadas)	`iot:Data`	`iot:Data`devuelve un punto final de datos VeriSign firmado que se proporciona para garantizar la compatibilidad con versiones anteriores. Los puntos de conexión de Symantec (`iot:Data`) no admiten MQTT 5.
AWS IoT Core acceso mediante credenciales	`iot:CredentialProvider`	Se utiliza para intercambiar el certificado X.509 integrado en un dispositivo por credencia les temporales con el fin de conectarse directamente con otros servicios de AWS . Para obtener más informaci

Finalidad del punto de conexión	Valor de *`endpointType`*	Description (Descripción)
		ón sobre cómo conectarse a otros AWS servicios, consulte Autorización de llamadas directas a AWS servicios.
Operaciones de datos de trabajos de AWS IoT Device Management	`iot:Jobs`	Se utiliza para permitir que los dispositivos interactúen con el servicio AWS IoT Jobs mediante el HTTPS APIs del dispositivo Jobs. `iot:Jobs` IPv4 solo se puede usar para. Si utiliza puntos de enlace de doble pila (IPv4 y IPv6), utilice el tipo de `iot:Data-ATS` punto final.
AWS IoT Operaciones de Device Advisor	`iot:DeviceAdvisor`	Un tipo de punto de conexión de prueba que se utiliza para probar dispositivos con Device Advisor. Para obtener más información, consulte ???.
AWS IoT Core versión beta de datos (versión preliminar)	`iot:Data-Beta`	Un tipo de punto de conexión reservado para las versiones beta. Para obtener informaci ón sobre su uso actual, consulte ???.

También puede utilizar su propio nombre de dominio completo (FQDN), por ejemplo*`example.com`*, y el certificado de servidor asociado, para conectar los dispositivos AWS IoT mediante. the section called “Configuraciones de dominio”

AWS IoT Dispositivo SDKs

El AWS IoT dispositivo le SDKs ayuda a conectar sus dispositivos de IoT AWS IoT Core y es compatible con los protocolos MQTT y MQTT sobre WSS.

El AWS IoT dispositivo se SDKs diferencia del dispositivo AWS SDKs en que el AWS IoT dispositivo SDKs admite las necesidades de comunicación especializadas de los dispositivos de IoT, pero no es compatible con todos los servicios compatibles con el AWS SDKs. El AWS IoT dispositivo SDKs es compatible con los AWS SDKs que admiten todos los AWS servicios; sin embargo, utilizan diferentes métodos de autenticación y se conectan a diferentes puntos finales, lo que podría hacer que su uso no fuera AWS SDKs práctico en un dispositivo de IoT.

Dispositivos móviles

Son the section called “AWS Aplicaciones SDKs” compatibles tanto con las comunicaciones de los dispositivos MQTT como con algunos de los AWS IoT servicios APIs y con otros servicios. APIs AWS Si está desarrollando en un dispositivo móvil compatible, revise el SDK para ver si es la mejor opción para desarrollar su solución de IoT.

C++

AWS IoT SDK para dispositivos C++

El SDK para dispositivos de AWS IoT C++ permite a los desarrolladores crear aplicaciones conectadas utilizando AWS y APIs los AWS IoT Core servicios. En concreto, este SDK se diseñó para los dispositivos que no tienen limitación de recursos y requieren características avanzadas, como la puesta en cola de mensajes, la compatibilidad con varios procesos y las características de idioma más actualizadas. Para obtener más información, consulte los siguientes temas:

- AWS IoT El SDK del dispositivo C++ v2 está activado GitHub
- AWS IoT SDK de dispositivo C++ v2 Léame
- AWS IoT Ejemplos de Device SDK C++ v2
- AWS IoT Documentación de la API C++ v2 del SDK de dispositivos

Python

AWS IoT SDK de dispositivo para Python

El SDK de AWS IoT dispositivos para Python permite a los desarrolladores escribir scripts de Python para usar sus dispositivos para acceder a la AWS IoT plataforma a través de MQTT

o MQTT a través del protocolo WebSocket Secure (WSS). Al conectar sus dispositivos a los AWS IoT Core servicios, los usuarios pueden trabajar APIs de forma segura con el agente de mensajes, las reglas y el servicio Device Shadow que AWS IoT Core proporciona AWS Lambda, así como con otros AWS servicios como Amazon Kinesis y Amazon S3, entre otros.

- AWS IoT SDK de dispositivo para Python v2 en GitHub
- AWS IoT SDK de dispositivo para Python v2 Readme
- AWS IoT Ejemplos de SDK de dispositivos para Python v2
- AWS IoT Documentación de la API del SDK de dispositivos para Python v2

JavaScript

AWS IoT SDK de dispositivo para JavaScript

El SDK de AWS IoT dispositivos JavaScript permite a los desarrolladores escribir JavaScript aplicaciones que accedan APIs a ellas AWS IoT Core mediante MQTT o MQTT a través del WebSocket protocolo. Se puede utilizar en entornos de Node.js y aplicaciones de navegador. Para obtener más información, consulte los siguientes temas:

- AWS IoT SDK de dispositivo para JavaScript la versión 2 en adelante GitHub
- AWS IoT Lea el SDK del dispositivo para JavaScript la versión 2
- AWS IoT Ejemplos de SDK de dispositivo para la JavaScript versión 2
- AWS IoT Documentación sobre la API del SDK de JavaScript dispositivos para la versión 2

Java

AWS IoT SDK de dispositivo para Java

El AWS IoT Device SDK for Java permite a los desarrolladores de Java acceder a él a AWS IoT Core través APIs de MQTT o MQTT a través del WebSocket protocolo. El SDK admite el servicio Device Shadow. Puede tener acceso a las sombras mediante los métodos GET, UPDATE y DELETE de HTTP. El SDK es también compatible con un modelo de acceso a sombras simplificado, lo que permite a los desarrolladores intercambiar datos con las sombras utilizando métodos getter y setter, sin tener que serializar ni deserializar documentos JSON. Para obtener más información, consulte los siguientes temas:

- AWS IoT SDK de dispositivo para Java v2 activado GitHub

- AWS IoT SDK de dispositivo para Java v2 (Léame)
- AWS IoT Ejemplos de SDK de dispositivos para Java v2
- AWS IoT Documentación de la API de Device SDK for Java v2

Embedded C

AWS IoT SDK de dispositivo para C integrado

> **Important**
>
> Este SDK está diseñado para que lo utilicen desarrolladores de software incrustado con experiencia.

El AWS IoT Device SDK para Embedded C (C-SDK) es una colección de archivos fuente en C bajo la licencia de código abierto del MIT que se puede usar en aplicaciones integradas para conectar de forma segura dispositivos de IoT a AWS IoT Core. Incluye las bibliotecas MQTT, JSON Parser y AWS IoT Device Shadow, entre otras. Se distribuye como código fuente y está diseñado para integrarse en el firmware cliente junto con código de aplicación, otras bibliotecas y, opcionalmente, RTOS (Real Time Operating System).

Por lo general, AWS IoT Device SDK para Embedded C está dirigido a dispositivos con recursos limitados que requieren un tiempo de ejecución optimizado en lenguaje C. Puede usar el SDK en cualquier sistema operativo y alojarlo en cualquier tipo de procesador (por ejemplo, MCUs y MPUs). Si su dispositivo tiene suficientes recursos de memoria y procesamiento disponibles, le recomendamos que utilice uno de los otros AWS IoT dispositivos y dispositivos móviles SDKs, como el SDK de AWS IoT dispositivos para C++ JavaScript, Java o Python.

Para obtener más información, consulte los siguientes temas:

- AWS IoT SDK de dispositivo para C integrado en GitHub
- AWS IoT SDK de dispositivo para C integrado: readme
- AWS IoT Ejemplos de SDK de dispositivo para C integrado

Protocolos de comunicación de dispositivos

AWS IoT Core admite dispositivos y clientes que utilizan los protocolos MQTT y MQTT over WebSocket Secure (WSS) para publicar mensajes y suscribirse a ellos, y dispositivos y clientes que utilizan el protocolo HTTPS para publicar mensajes. Todos los protocolos admiten y IPv4 . IPv6 En esta sección se describen las diferentes opciones de conexión para dispositivos y clientes.

Versiones del protocolo TLS

AWS IoT Core utiliza la versión 1.2 de TLS y la versión 1.3 de TLS para cifrar todas las comunicaciones. Puede configurar versiones adicionales de la política de TLS para su punto de conexión mediante el establecimiento de los ajustes de TLS en las configuraciones de dominio. Al conectar los dispositivos a AWS IoT Core, los clientes pueden enviar la extensión de indicación de nombre de servidor (SNI), que es necesaria para características como el registro de varias cuentas, los puntos de conexión configurables, los dominios personalizados y los puntos de conexión de VPC. Para obtener más información, consulte Seguridad de transporte en AWS IoT.

Los AWS IoT Dispositivo SDKs admiten MQTT y MQTT a través de WSS y cumplen los requisitos de seguridad de las conexiones de los clientes. Recomendamos utilizar los AWS IoT Dispositivo SDKs para conectar los clientes a AWS IoT.

Protocolos, asignaciones de puertos y autenticación

La forma en que un dispositivo o cliente se conecta al agente de mensajes se puede configurar con un tipo de autenticación. De forma predeterminada o cuando no se envía ninguna extensión SNI, el método de autenticación se basa en el protocolo de aplicación, el puerto y la extensión TLS de negociación del protocolo de capa de aplicación (ALPN) que utilizan los dispositivos. En la siguiente tabla se muestra la autenticación esperada en función del puerto y la ALPN.

Protocolos, autenticación y asignaciones de puertos

Protocolo	Operaciones admitidas	Autenticación	Puerto	Nombre del protocolo ALPN
MQTT ha terminado WebSocket	Publicar, suscribirse	Signature Version 4	443	N/A
Se acabó MQTT WebSocket	Publicar, suscribirse	Autenticación personalizada	443	N/A

Protocolo	Operaciones admitidas	Autenticación	Puerto	Nombre del protocolo ALPN
MQTT	Publicar, suscribirse	Certificado de cliente X.509	443†	`x-amzn-mqtt-ca`
MQTT	Publicar, suscribirse	Certificado de cliente X.509	8883	N/A
MQTT	Publicar, suscribirse	Autenticación personalizada	443†	`mqtt`
HTTPS	Solo publicar	Signature Version 4	443	N/A
HTTPS	Solo publicar	Certificado de cliente X.509	443†	`x-amzn-http-ca`
HTTPS	Solo publicar	Certificado de cliente X.509	8443	N/A
HTTPS	Solo publicar	Autenticación personalizada	443	N/A

Negociación de protocolo de capa de aplicación (ALPN)

† Al utilizar las configuraciones de punto final predeterminadas, los clientes que se conecten al puerto 443 con la autenticación mediante certificado de cliente X.509 deben implementar la extensión TLS de negociación del protocolo de capa de aplicación (ALPN) y utilizar el nombre del protocolo ALPN que aparece en el ALPN ProtocolNameList enviado por el cliente como parte del mensaje. `ClientHello`
En el puerto 443, el punto final IoT:data-ATS admite HTTP x-amzn-http-ca ALPN, pero el punto final IoT:Jobs no.
En el puerto 8443 HTTPS y el puerto 443 MQTT con ALPN, no se puede usar la autenticación personalizada. x-amzn-mqtt-ca

Los clientes se conectan a los puntos finales Cuenta de AWS de sus dispositivos. Consulte the section called “AWS IoT datos del dispositivo y puntos finales de servicio” para obtener información sobre cómo encontrar los puntos de conexión de los dispositivos de su cuenta.

> **Note**
>
> AWS SDKs no requieren la URL completa. Solo requieren el nombre de host del punto final, como en el `pubsub.py`ejemplo de AWS IoT Device SDK for Python on GitHub. Pasar la URL completa como se indica en la siguiente tabla puede generar un error (como un nombre de host no válido).

Conectándose a AWS IoT Core

Protocolo	Punto de enlace o URL
MQTT	*iot-endpoint*
MQTT a través de WSS	wss://*iot-endpoint* /mqtt
HTTPS	https://*iot-endpoint* /topics

Elección de un protocolo de aplicación para la comunicación entre dispositivos

Para la mayoría de las comunicaciones de los dispositivos de IoT a través de los puntos finales del dispositivo, querrá utilizar los protocolos Secure MQTT o MQTT over WebSocket Secure (WSS); sin embargo, los puntos finales del dispositivo también admiten HTTPS.

En la siguiente tabla se compara cómo se AWS IoT Core utilizan los dos protocolos de alto nivel (MQTT y HTTPS) para la comunicación entre dispositivos.

AWS IoT protocolos de dispositivo (MQTT y HTTPS) side-by-side

Característica	MQTT	HTTPS
Soporte para publicación/suscripción	Publicar y suscribirse	Solo publicar

Característica	MQTT	HTTPS
Soporte para SDK	AWS El dispositivo SDKs admite los protocolos MQTT y WSS	No compatible con el SDK, pero es posible usar métodos de lenguajes específicos para realizar solicitudes HTTPS
Soporte para calidad del servicio	Niveles 0 y 1 de calidad del servicio de MQTT	Se admite la calidad del servicio al pasar un parámetro de cadena de consulta `?qos=qos` donde el valor puede ser 0 o 1. Puede agregar esta cadena de consulta para publicar un mensaje con el valor de calidad del servicio que desee.
¿Se pueden perder los mensajes recibidos mientras el dispositivo estaba sin conexión?	Sí	No
Soporte de campo de `clientId`	Sí	No
Detección de desconexión de dispositivos	Sí	No
Comunicaciones seguras	Sí Consulte ???	Sí Consulte ???
Definiciones de temas	Definido por la aplicación	Definido por la aplicación
Formato de datos de mensajes	Definido por la aplicación	Definido por la aplicación
Sobrecarga de protocolo	Más baja	Más alta
Consumo eléctrico	Más bajo	Más alto

Elección de un tipo de autenticación para la comunicación entre dispositivos

Puede configurar el tipo de autenticación para su punto de conexión de IoT mediante puntos de conexión configurables. Como alternativa, utilice la configuración predeterminada y determine cómo se autentican sus dispositivos con una combinación de protocolo de aplicación, puerto y extensión TLS de ALPN. El tipo de autenticación que elija determina cómo se autenticarán sus dispositivos cuando se conecten a ellos. AWS IoT Core Hay cinco tipos de autenticación:

Certificado X.509

Autentica los dispositivos mediante certificados de cliente X.509, que AWS IoT Core validan la autenticación del dispositivo. Este tipo de autenticación es compatible con los protocolos Secure MQTT (MQTT a través de TLS) y HTTPS.

Certificado X.509 con autorizador personalizado

Autentique los dispositivos con certificados de cliente X.509 y realice acciones de autenticación adicionales con un autorizador personalizado, que recibirá la información del certificado de cliente X.509. Este tipo de autenticación es compatible con los protocolos Secure MQTT (MQTT a través de TLS) y HTTPS. Este tipo de autenticación solo es posible mediante puntos de conexión configurables con autenticación personalizada X.509. No hay ninguna opción ALPN.

AWS Firma, versión 4 (SigV4)

Autentique los dispositivos mediante Cognito o su servicio de backend, que admiten la federación social y empresarial. Este tipo de autenticación funciona con los protocolos MQTT over WebSocket Secure (WSS) y HTTPS.

Autorizador personalizado

Autentique los dispositivos configurando una función de Lambda para procesar la información de autenticación personalizada enviada a AWS IoT Core. Este tipo de autenticación funciona con los protocolos MQTT seguro (MQTT sobre TLS), HTTPS y MQTT sobre WebSocket seguro (WSS).

Predeterminado

Autentique los dispositivos en función de la extensión de negociación del protocolo de capa de and/or aplicación (ALPN) de puertos que utilizan los dispositivos. No se admiten algunas opciones de autenticación adicionales. Para obtener más información, consulte ???.

La siguiente tabla muestra todas las combinaciones compatibles de tipos de autenticación y protocolos de aplicación.

Combinaciones compatibles de tipos de autenticación y protocolos de aplicación

Tipo de autenticación	Secure MQTT (MQTT a través de TLS)	MQTT a través de WebSocket Secure (WSS)	HTTPS	Predeterminado
Certificado X.509	✓		✓	
Certificado X.509 con autorizador personalizado	✓		✓	
AWS Firma versión 4 (SiGv4)		✓	✓	
Autorizador personalizado	✓	✓	✓	
Predeterminado				✓

Límites de duración de la conexión

No se garantiza que las conexiones HTTPS duren más del tiempo necesario para recibir las solicitudes y responder.

La duración de la conexión MQTT depende de la característica de autenticación que utilice. La siguiente tabla muestra la duración máxima de la conexión en condiciones ideales para cada característica.

Duración de la conexión MQTT por característica de autenticación

Característica	Duración máxima*
Certificado de cliente X.509	De 1 a 2 semanas
Autenticación personalizada	De 1 a 2 semanas

Característica	Duración máxima*
Signature Version 4	Hasta 24 horas

* No garantizado

Con los certificados X.509 y la autenticación personalizada, la duración de la conexión no tiene un límite estricto, pero puede durar incluso solo unos minutos. Las interrupciones en la conexión pueden producirse por diversos motivos. A continuación, se presentan algunos de los casos más comunes.

- Interrupciones en la disponibilidad de la conexión wifi
- Interrupciones de conexión del proveedor de servicios de Internet (ISP)
- Parches de servicio
- Implementaciones de servicios
- Escalado automático de servicios
- Host de servicio no disponible
- Problemas y actualizaciones del equilibrador de carga
- Errores del cliente

Los dispositivos deben implementar estrategias para detectar las desconexiones y volver a conectarse. Para obtener información sobre los eventos de desconexión y saber cómo gestionarlos, consulte??? en ???.

MQTT

MQTT (Transporte de telemetría de colas de mensajes) es un protocolo de mensajería ligero y ampliamente adoptado que está diseñado para dispositivos restringidos. La compatibilidad de AWS IoT Core con MQTT se basa en la especificación MQTT v3.1.1 y en la especificación MQTT v5.0, con algunas diferencias, tal y como se documenta en the section called “AWS IoT diferencias con las especificaciones de MQTT”. Al ser la última versión del estándar, MQTT 5 presenta varias características clave que hacen que un sistema basado en MQTT sea más robusto, incluidas nuevas mejoras de escalabilidad, informes de errores mejorados con respuestas de códigos de motivo, temporizadores de caducidad de mensajes, y sesiones y encabezados de mensajes de usuario personalizados. Para obtener más información sobre las funciones compatibles con MQTT 5, consulte Funciones AWS IoT Core compatibles con MQTT 5. AWS IoT Core también admite la

comunicación entre versiones de MQTT (MQTT 3 y MQTT 5). Un editor de MQTT 3 puede enviar un mensaje de MQTT 3 a un suscriptor de MQTT 5 que recibirá un mensaje de publicación de MQTT 5, y viceversa.

AWS IoT Core admite conexiones de dispositivos que utilizan el protocolo MQTT y el protocolo MQTT sobre el protocolo WSS y que se identifican mediante un ID de cliente. AWS IoT Dispositivo SDKs es compatible con ambos protocolos y son las formas recomendadas de conectar los dispositivos a AWS IoT Core. El AWS IoT dispositivo SDKs admite las funciones necesarias para que los dispositivos y los clientes se conecten a los servicios y accedan AWS IoT a ellos. El dispositivo SDKs admite los protocolos de autenticación que requieren los AWS IoT servicios y los requisitos de identificación de conexión que requieren el protocolo MQTT y los protocolos MQTT sobre WSS. Para obtener información sobre cómo conectarse AWS IoT mediante el AWS dispositivo SDKs y enlaces a ejemplos AWS IoT en los idiomas compatibles, consulte. the section called “Conexión con MQTT mediante el dispositivo AWS IoT SDKs” Para obtener más información acerca de los métodos de autenticación y las asignaciones de puertos para mensajes MQTT, consulte ???.

Si bien recomendamos usar el AWS IoT dispositivo SDKs para conectarse AWS IoT, no son obligatorios. Sin embargo SDKs, si no utiliza el AWS IoT Dispositivo, debe proporcionar la seguridad de conexión y comunicación necesaria. Los clientes también deben enviar la extensión TLS de Indicación de nombre de servidor (SNI) en la solicitud de conexión. Se rechazan los intentos de conexión que no incluyan la SNI. Para obtener más información, consulte Seguridad en el transporte en AWS IoT. Los clientes que utilizan usuarios y AWS credenciales de IAM para autenticar a los clientes deben proporcionar la autenticación correcta de la versión 4 de Signature.

Una vez que sus clientes estén conectados, puede supervisar y gestionar sus conexiones de clientes MQTT mediante. APIs Para obtener más información, consulte the section called “Administración de conexiones de MQTT”.

En este tema:

- Conexión con MQTT mediante el dispositivo AWS IoT SDKs
- Opciones de calidad de servicio (QoS) de MQTT
- Sesiones persistentes de MQTT
- Mensajes retenidos de MQTT
- Mensajes Last Will and Testament (LWT) de MQTT
- Uso de connectAttributes
- Características compatibles con MQTT 5

- Propiedades de MQTT 5
- Códigos de motivo de MQTT
- AWS IoT diferencias con las especificaciones de MQTT
- Administración de conexiones de MQTT

Conexión con MQTT mediante el dispositivo AWS IoT SDKs

Esta sección contiene enlaces al AWS IoT dispositivo SDKs y al código fuente de programas de muestra que ilustran cómo conectar un dispositivo al AWS IoT mismo. Las aplicaciones de muestra enlazadas aquí muestran cómo conectarse AWS IoT mediante el protocolo MQTT y MQTT a través de WSS.

> **Note**
>
> The AWS IoT Device SDKs ha lanzado un cliente MQTT 5.

C++

Uso del SDK de dispositivos AWS IoT C++ para conectar dispositivos

- Código fuente de una aplicación de muestra que ilustra un ejemplo de conexión MQTT en C++
- AWS IoT SDK de dispositivo para C++ v2 en GitHub

Python

Uso del SDK de AWS IoT dispositivos para Python para conectar dispositivos

- Código fuente de una aplicación de muestra que ilustra un ejemplo de conexión MQTT en Python
- AWS IoT SDK de dispositivo v2 para Python activado GitHub

JavaScript

Uso del SDK de AWS IoT dispositivos JavaScript para conectar dispositivos

- Código fuente de una aplicación de muestra que muestra un ejemplo de conexión MQTT en JavaScript

- AWS IoT SDK de dispositivo para la versión 2 en JavaScript adelante GitHub

Java

Uso del AWS IoT Device SDK for Java para conectar dispositivos

> **Note**
>
> El AWS IoT Device SDK for Java v2 ahora es compatible con el desarrollo de Android. Para obtener más información, consulte AWS IoT Device SDK for Android.

- Código fuente de una aplicación de muestra que ilustra un ejemplo de conexión MQTT en Java
- AWS IoT SDK de dispositivo para Java v2 activado GitHub

Embedded C

Uso del SDK de AWS IoT dispositivos para C integrado para conectar dispositivos

> **Important**
>
> Este SDK está diseñado para que lo utilicen desarrolladores de software incrustado con experiencia.

- Código fuente de una aplicación de muestra que ilustra un ejemplo de conexión MQTT en Embedded C
- AWS IoT SDK de dispositivo para C integrado en GitHub

Opciones de calidad de servicio (QoS) de MQTT

AWS IoT y el AWS IoT dispositivo SDKs admiten los niveles de calidad de servicio (QoS) `0` de MQTT y. 1 El protocolo MQTT define un tercer nivel de QoS, el 2 nivel, AWS IoT pero no lo admite. Solo el protocolo MQTT admite la característica QoS. HTTPS admite QoS al pasar un parámetro de cadena de consulta `?qos=qos` donde el valor puede ser 0 o 1.

En esta tabla se describe cómo afecta cada nivel de QoS a los mensajes publicados en el agente de mensajes y por este.

Con un nivel de QoS de...	El mensaje es…	Comentarios
QoS nivel 0	Enviado cero o más veces	Este nivel debe usarse para los mensajes que se envían a través de enlaces de comunicación fiables o que pueden perderse sin problemas.
QoS nivel 1	Se envía al menos una vez y, a continuación, varias veces hasta recibir una respuesta `PUBACK`	El mensaje no se considera completo hasta que el remitente reciba una respuesta `PUBACK` que indique que se ha entregado correctamente.

Sesiones persistentes de MQTT

Las sesiones persistentes almacenan las suscripciones y los mensajes de un cliente, con una calidad de servicio (QoS) de 1, que el cliente no ha reconocido. Cuando el dispositivo se vuelve a conectar a una sesión persistente, la sesión se reanuda, las suscripciones se restablecen y los mensajes suscritos no confirmados recibidos y almacenados antes de la reconexión se envían al cliente.

El procesamiento de los mensajes almacenados se registra en CloudWatch métricas y registros. CloudWatch Para obtener información sobre las entradas escritas CloudWatch y los CloudWatch registros, consulte Métricas del agente de mensajes yEntrada de registro en cola.

Creación de una sesión persistente

En MQTT 3, puede crear una sesión persistente de MQTT enviando un mensaje `CONNECT`, que establezca la marca `cleanSession` en `0`. Si no existe ninguna sesión para el cliente que envía el mensaje `CONNECT`, se crea una sesión persistente nueva. Si ya existe una sesión para el cliente, el cliente reanuda la sesión existente. Para crear una sesión limpia, se envía un mensaje `CONNECT` y se establece la marca `cleanSession` en `1`, y el agente no almacenará ningún estado de la sesión cuando el cliente se desconecte.

En MQTT 5, las sesiones persistentes se gestionan configurando la marca `Clean Start` y `Session Expiry Interval`. Clean Start controla el inicio de la sesión de conexión y el final de la

sesión anterior. Si se establece `Clean Start` = `1`, se crea una nueva sesión y, si existe, se termina la sesión anterior. Si se establece `Clean Start` = `0`, la sesión de conexión reanuda la sesión anterior, si ya existe. El intervalo de caducidad de la sesión controla el final de la sesión de conexión. El intervalo de caducidad de la sesión especifica el tiempo, en segundos (entero de 4 bytes), que durará una sesión tras la desconexión. La configuración `Session Expiry interval` = `0` hace que la sesión finalice inmediatamente después de la desconexión. Si el intervalo de caducidad de la sesión no se especifica en el mensaje CONNECT, el valor predeterminado es 0.

Inicio limpio y caducidad de la sesión de MQTT 5

Valor de la propiedad	Description (Descripción)
`Clean Start= 1`	Crea una nueva sesión y termina una sesión anterior, si existe.
`Clean Start= 0`	Reanuda una sesión si existe una sesión anterior.
`Session Expiry Interval> 0`	Persiste una sesión.
`Session Expiry interval= 0`	No persiste una sesión.

En MQTT 5, si establece `Clean Start` = `1` y `Session Expiry Interval` =`0`, equivale a una sesión limpia de MQTT 3. Si establece `Clean Start` = `0` y `Session Expiry Interval` >`0`, equivale a una sesión persistente de MQTT 3.

> **Note**
>
> No se admiten sesiones persistentes entre versiones MQTT (MQTT 3 y MQTT 5). Una sesión persistente de MQTT 3 no se puede reanudar como una sesión de MQTT 5 y viceversa.

Operaciones durante una sesión persistente

Los clientes utilizan el atributo `sessionPresent` del mensaje de confirmación de conexión (`CONNACK`) para determinar si existe una sesión persistente. Si `sessionPresent` es `1`, hay una sesión persistente y todos los mensajes almacenados para el cliente se envían al cliente después de que el cliente reciba el `CONNACK`, tal y como se describe en Tráfico de mensajes tras la reconexión a una sesión persistente. Si `sessionPresent` es `1`, el cliente no necesita volver a suscribirse.

Sin embargo, si `sessionPresent` está establecido en `0`, significa que no existe ninguna sesión persistente y que el cliente debe volver a suscribirse a sus filtros de temas.

Una vez que el cliente se une a una sesión persistente, puede publicar mensajes y suscribirse a filtros de temas sin ningún indicador adicional en cada operación.

Tráfico de mensajes tras volver a conectarse a una sesión persistente

Una sesión persistente representa una conexión en curso entre un cliente y un agente de mensajes de MQTT. Cuando un cliente se conecta al agente de mensajes mediante una sesión persistente, el agente de mensajes guarda todas las suscripciones que el cliente realiza durante la conexión. Cuando el cliente se desconecta, el agente de mensajes almacena los mensajes QoS 1 sin confirmar y los mensajes QoS 1 nuevos publicados en los temas a los que el cliente está suscrito. Los mensajes se almacenan según el límite de la cuenta. Mensajes que superen el límite se eliminarán. Para más información acerca de los límites de mensajes persistentes, consulte Puntos de conexión y cuotas de AWS IoT Core. Cuando el cliente vuelve a conectarse la sesión persistente, todas las suscripciones se restablecen y todos los mensajes almacenados se envían al cliente a una velocidad máxima de 10 mensajes por segundo. En MQTT 5, si un QoS 1 saliente con un intervalo de caducidad de mensajes caduca cuando un cliente está desconectado, una vez que se reanude la conexión, el cliente no recibirá el mensaje caducado.

Tras la reconexión, los mensajes almacenados se envían al cliente a una velocidad limitada a 10 mensajes almacenados por segundo, junto con el tráfico de mensajes actual hasta que se alcance el límite de `Publish requests per second per connection`. Como la velocidad de entrega de los mensajes almacenados es limitada, se tardarán varios segundos en entregar todos los mensajes almacenados si una sesión tiene más de 10 mensajes almacenados que entregar después de la reconexión.

En el caso de los suscriptores compartidos, los mensajes se pondrán en cola si al menos un suscriptor de un grupo utiliza una sesión persistente y no hay ningún suscriptor conectado para recibir el mensaje de QoS 1. La eliminación de la cola de mensajes se realiza a una velocidad máxima de 20 mensajes por segundo por cada suscriptor activo de un grupo. Para obtener más información, consulte Cola de mensajes de suscripciones compartidas.

Finalizar una sesión persistente

Las sesiones persistentes pueden finalizar de una de las siguientes formas:

- Transcurre el tiempo de caducidad de la sesión persistente. El temporizador de caducidad de la sesión persistente se inicia cuando el agente de mensajes detecta que un cliente se ha

desconectado, ya sea porque el cliente se ha desconectado o porque se ha agotado el tiempo de espera de la conexión.

- El cliente envía un mensaje `CONNECT` en el que se establece la marca `cleanSession` en `1`.
- El cliente se desconecta manualmente y se borra la sesión mediante la API `DeleteConnection`. Para obtener más información, consulte the section called “Administración de conexiones de MQTT”.

En MQTT 3, el valor predeterminado del tiempo de caducidad de las sesiones persistentes es de una hora, y esto se aplica a todas las sesiones de la cuenta.

En MQTT 5, puede establecer el intervalo de caducidad de la sesión para cada sesión en los paquetes CONNECT y DISCONNECT.

Para el intervalo de caducidad de la sesión en el paquete DISCONNECT:

- Si la sesión actual tiene un intervalo de caducidad de sesión igual a 0, no puede establecer el intervalo de caducidad de la sesión en un valor superior a 0 en el paquete DISCONNECT.
- Si la sesión actual tiene un intervalo de caducidad de sesión superior a 0 y se establece el intervalo de caducidad de la sesión en 0 en el paquete DISCONNECT, la sesión finalizará en DISCONNECT.
- De lo contrario, el intervalo de caducidad de la sesión del paquete DISCONNECT actualizará el intervalo de caducidad de la sesión actual.

> **Note**
>
> Los mensajes guardados en espera de ser enviados al cliente al finalizar la sesión se descartan; sin embargo, se siguen facturando según la tarifa de mensajería estándar, aunque no se hayan podido enviar. Para obtener más información sobre los precios de los mensajes, consulte Precios de AWS IoT Core. Puede configurar el intervalo de tiempo de caducidad.

Reconexión después de que haya caducado una sesión persistente

Si un cliente no se vuelve a conectar a su sesión persistente antes de que caduque, la sesión finaliza y los mensajes almacenados se descartan. Cuando un cliente se vuelve a conectar después de que la sesión haya caducado con una marca `cleanSession` establecida en `0`, el servicio crea

una nueva sesión persistente. Las suscripciones o los mensajes de la sesión anterior no están disponibles para esta sesión porque se descartaron al expirar la sesión anterior.

Cargos por mensajes de sesión persistentes

Los mensajes se cobran Cuenta de AWS cuando el agente de mensajes envía un mensaje a un cliente o a una sesión persistente fuera de línea. Cuando un dispositivo desconectado con una sesión persistente se vuelve a conectar y reanuda la sesión, los mensajes almacenados se envían al dispositivo y se vuelven a cargar a su cuenta. Para obtener más información sobre los precios de los mensajes, consulte los Precios de AWS IoT Core : Mensajería.

El tiempo de caducidad predeterminado de la sesión persistente, de una hora, se puede aumentar mediante el proceso de aumento de límites estándar. Tenga en cuenta que aumentar el tiempo de caducidad de la sesión puede aumentar los cargos por mensajes, ya que el tiempo adicional podría permitir almacenar más mensajes para el dispositivo sin conexión y esos mensajes adicionales se cargarían en su cuenta según la tarifa de mensajería estándar. El tiempo de caducidad de la sesión es aproximado y una sesión puede prolongarse hasta 30 minutos más que el límite de la cuenta; sin embargo, una sesión no superará el límite de la cuenta. Para obtener más información acerca de los límites de sesión, consulte Cuotas de servicio de AWS.

Mensajes retenidos de MQTT

AWS IoT Core admite el `RETAIN` indicador descrito en el protocolo MQTT. Cuando un cliente establece la `RETAIN` marca en un mensaje MQTT que publica, AWS IoT Core guarda el mensaje. A continuación, puede enviarse a los nuevos suscriptores, recuperarse mediante una llamada a la operación `GetRetainedMessage` y visualizarse en la consola de AWS IoT.

Ejemplos del uso de mensajes retenidos en MQTT

- Como mensaje de configuración inicial

  Los mensajes retenidos de MQTT se envían a un cliente después de que el cliente se suscriba a un tema. Si desea que todos los clientes que se suscriban a un tema reciban el mensaje retenido de MQTT inmediatamente después de su suscripción, puede publicar un mensaje de configuración con el indicador `RETAIN` activado. Los clientes que se suscriben también reciben actualizaciones de esa configuración cada vez que se publica un mensaje de configuración nuevo.
- Como último mensaje de estado conocido

  Los dispositivos pueden configurar el indicador `RETAIN` en los mensajes en estado actual para que AWS IoT Core los guarde. Cuando las aplicaciones se conectan o se vuelven a conectar,

pueden suscribirse a este tema y obtener el último estado registrado inmediatamente después de suscribirse al tema del mensaje retenido. De esta forma, pueden evitar tener que esperar hasta el siguiente mensaje del dispositivo para ver el estado actual.

En esta sección:

- Las tareas habituales con MQTT retienen los mensajes en AWS IoT Core
- Facturación y mensajes retenidos
- Comparación de los mensajes retenidos de MQTT y las sesiones persistentes de MQTT
- Los mensajes retenidos por MQTT y Device Shadows AWS IoT

Las tareas habituales con MQTT retienen los mensajes en AWS IoT Core

AWS IoT Core guarda los mensajes MQTT con la `RETAIN` bandera puesta. Estos mensajes retenidos se envían a todos los clientes que se han suscrito al tema, como un mensaje MQTT normal, y también se almacenan para enviarlos a los nuevos suscriptores del tema.

Los mensajes MQTT retenidos requieren políticas específicas para autorizar a los clientes a acceder a ellos. Para ver ejemplos del uso de políticas de mensajes retenidos, consulte Ejemplos de políticas de mensajes retenidos.

En esta sección se describen las operaciones comunes que implican los mensajes retenidos.

- Creación de mensajes retenidos

  El cliente determina si un mensaje se conserva cuando publica un mensaje MQTT. Los clientes pueden establecer el indicador `RETAIN` al publicar un mensaje mediante un SDK de dispositivos. Las aplicaciones y los servicios pueden establecer el indicador `RETAIN` cuando utilizan la acción `Publish` para publicar un mensaje de MQTT.

  Solo se conserva un mensaje por nombre de tema. Un mensaje nuevo con la marca RETAIN activada y publicada en un tema reemplaza cualquier mensaje retenido existente que se haya enviado anteriormente al tema.

  > **Note**
  >
  > No puede publicar un tema reservado con el indicador `RETAIN` activado.

- Suscripción a un tema de mensajes retenido

Los clientes se suscriben a los temas de mensajes retenidos como lo harían con cualquier otro tema de mensajes MQTT. Los mensajes retenidos que se reciben al suscribirse a un tema de mensajes retenidos tienen el indicador `RETAIN` activado.

Los mensajes retenidos se eliminan AWS IoT Core cuando un cliente publica un mensaje retenido con una carga útil de 0 bytes en el tema del mensaje retenido. Los clientes que se hayan suscrito al tema del mensaje retenido también recibirán el mensaje de 0 bytes.

Si se suscribe a un filtro de temas con formato comodín que incluye un tema de mensaje retenido, el cliente puede recibir los mensajes subsiguientes publicados en el tema del mensaje retenido, pero no entrega el mensaje retenido al suscribirse.

> **Note**
>
> Para recibir un mensaje retenido al suscribirse, el filtro de temas de la solicitud de suscripción debe coincidir exactamente con el tema del mensaje retenido.

Los mensajes retenidos que se reciben al suscribirse a un tema de mensajes retenidos tienen el indicador `RETAIN` activado. Los mensajes retenidos que recibe un cliente suscriptor después de la suscripción, no lo hacen.

- Recuperación de un mensaje retenido

  Los mensajes retenidos se entregan a los clientes automáticamente cuando se suscriben al tema que contiene el mensaje retenido. Para que un cliente reciba el mensaje retenido al suscribirse, debe suscribirse al nombre exacto del tema del mensaje retenido. Si se suscribe a un filtro de temas con formato comodín que incluye un tema de mensaje retenido, el cliente puede recibir los mensajes posteriores publicados en el tema del mensaje retenido, pero no entrega el mensaje retenido al suscribirse.

  Los servicios y las aplicaciones pueden enumerar y recuperar los mensajes retenidos llamando a `ListRetainedMessages` y `GetRetainedMessage`.

  No se impide que un cliente publique mensajes en un tema de mensaje retenido sin configurar el indicador `RETAIN`. Esto podría provocar resultados inesperados, como que el mensaje retenido no coincida con el mensaje recibido al suscribirse al tema.

Con MQTT 5, si un mensaje retenido tiene establecido el intervalo de caducidad del mensaje y el mensaje retenido caduca, el nuevo suscriptor que se suscriba a ese tema no recibirá el mensaje retenido si se suscribe correctamente.

- Generación de una lista con los temas de los mensajes retenidos

  Puede generar una lista de los mensajes retenidos llamando a `ListRetainedMessages` y los mensajes retenidos se pueden ver en la consola de AWS IoT.

- Obtención de los detalles de los mensajes retenidos

  Puede obtener los detalles de los mensajes retenidos llamando a `GetRetainedMessage` y los puede ver en la consola de AWS IoT.

- Retención de un mensaje Will

  Los mensajes Will de MQTT que se crean cuando se conecta un dispositivo se pueden conservar configurando la marca `Will Retain` en el campo `Connect Flag bits`.

- Eliminación de mensajes retenidos

  Los dispositivos, las aplicaciones y los servicios pueden eliminar un mensaje retenido publicando un mensaje con el indicador `RETAIN` activado y una carga útil de mensaje vacía (0 bytes) junto al nombre del tema del mensaje retenido que se va a eliminar. Estos mensajes eliminan el mensaje retenido AWS IoT Core y se envían a los clientes con una suscripción al tema, pero no se conservan. AWS IoT Core

  Los mensajes retenidos también se pueden eliminar de forma interactiva accediendo al mensaje retenido en la consola de AWS IoT. Los mensajes retenidos que se eliminan mediante la consola de AWS IoT también envían un mensaje de 0 bytes a los clientes que se hayan suscrito al tema del mensaje retenido.

  Los mensajes retenidos no se pueden restaurar una vez eliminados. Un cliente tendría que publicar un nuevo mensaje retenido para sustituir al mensaje eliminado.

- Depuración y solución de problemas de los mensajes retenidos

  La consola de AWS IoT proporciona varias herramientas para ayudarle a solucionar los problemas de los mensajes retenidos:

  - La página de mensajes retenidos

La página de mensajes retenidos de la consola de AWS IoT proporciona una lista paginada de los mensajes retenidos que su cuenta ha almacenado en la región actual. En esta página puede hacer lo siguiente:

- Consulte los detalles de cada mensaje retenido, como la carga del mensaje, la QoS y la hora en que se recibió.
- Actualice el contenido de un mensaje retenido.
- Elimine un mensaje retenido.

- El cliente de pruebas MQTT

  La página del cliente de pruebas de MQTT de la consola de AWS IoT permite suscribirse a temas de MQTT y publicarlos. La opción de publicación le permite establecer la marca RETAIN en los mensajes que publique para simular el comportamiento de sus dispositivos. También puede utilizar el cliente de prueba de MQTT para supervisar mensajes de clientes conectados que administra a través de la interfaz de conexiones de clientes. Para obtener más información sobre la administración de conexiones de clientes, consulte the section called “Administración de conexiones de MQTT”.

Algunos resultados inesperados pueden deberse a estos aspectos de la forma en que se implementan los mensajes retenidos en AWS IoT Core.

- Límites de mensajes retenidos

  Cuando una cuenta ha almacenado el número máximo de mensajes retenidos, AWS IoT Core devuelve una respuesta limitada a los mensajes publicados con el comando RETAIN activado y con cargas útiles superiores a 0 bytes, hasta que se eliminen algunos mensajes retenidos y el recuento de mensajes retenidos caiga por debajo del límite.

- Pedido de entrega de mensajes retenidos

  No se garantiza la secuencia de entrega de mensajes retenidos y mensajes suscritos.

Facturación y mensajes retenidos

La publicación de mensajes con el indicador `RETAIN` activado desde un cliente, mediante la consola de AWS IoT o mediante una llamada a `Publish` conlleva los gastos de mensajería adicionales descritos en la sección Precios de mensajería de AWS IoT Core.

La recuperación de los mensajes retenidos por parte de un cliente, mediante una AWS IoT consola o mediante una llamada `GetRetainedMessage`conlleva gastos de mensajería además de los cargos

por uso normal de la API. Los cargos adicionales se describen en Precios de mensajería de AWS IoT Core.

Los mensajes Will que se publiquen cuando un dispositivo se desconecte inesperadamente incurrirán en los cargos de mensajería descritos en el apartado Precios de mensajería de AWS IoT Core.

Para obtener más información sobre los costes de mensajería, consulte Precios de mensajería de AWS IoT Core.

Comparación de los mensajes retenidos de MQTT y las sesiones persistentes de MQTT

Los mensajes retenidos y las sesiones persistentes son características estándar de MQTT que permiten a los dispositivos recibir los mensajes que se publicaron sin conexión a Internet. Los mensajes retenidos se pueden publicar desde sesiones persistentes. En esta sección se describen los aspectos clave de estas características y la forma en que funcionan juntas.

	Mensajes retenidos	Sesiones persistentes
Características principales	Los mensajes retenidos se pueden usar para configurar grandes grupos de dispositivos después de que se conecten o enviarles notificaciones. Los mensajes retenidos también se pueden usar cuando desee que los dispositivos reciban solo el último mensaje publicado en un tema tras una reconexión.	Las sesiones persistentes son útiles para los dispositivos que tienen una conectividad intermitente y pueden pasar por alto varios mensajes importantes. Los dispositivos se pueden conectar con una sesión persistente para recibir los mensajes enviados mientras están desconectados.
Ejemplos	Los mensajes retenidos pueden proporcionar a los dispositivos información de configuración sobre su entorno cuando se conectan a Internet. La configuración	Los dispositivos que se conectan a través de una red móvil con conectividad intermitente pueden utilizar sesiones persistentes para evitar perder los mensajes

	Mensajes retenidos	Sesiones persistentes
	inicial podría incluir una lista de otros temas de mensajes a los que debería suscribirse o información sobre cómo debe configurar su zona horaria local.	importantes que se envían cuando un dispositivo está fuera de la cobertura de la red o necesita apagar la radio móvil.
Mensajes recibidos al suscribir se por primera vez a un tema	Tras suscribirse a un tema con un mensaje retenido, se recibe el mensaje retenido más reciente.	Tras suscribirse a un tema sin un mensaje retenido, no se recibe ningún mensaje hasta que se publique uno en el tema.
Temas suscritos tras la reconexión	Sin una sesión persistente, el cliente debe suscribirse a los temas tras la reconexión.	Los temas suscritos se restauran tras la reconexión.
Mensajes recibidos tras la reconexión	Tras suscribirse a un tema con un mensaje retenido, se recibe el mensaje retenido más reciente.	Todos los mensajes publicado s con un QOS igual a 1 y a los que se haya suscrito con un QOS igual a 1 mientras el dispositivo estaba desconect ado se envían cuando el dispositivo se vuelve a conectar.

	Mensajes retenidos	Sesiones persistentes
Caducidad de los datos/sesiones	En MQTT 3, los mensajes retenidos no caducan. Se almacenan hasta que se sustituyan o eliminen. En MQTT 5, los mensajes retenidos caducan después del intervalo de caducidad establecido. Para obtener más información, consulte Caducidad de los mensajes.	Las sesiones persistentes caducan si el cliente no se vuelve a conectar dentro del periodo de espera. Cuando caduca una sesión persistente, se eliminan las suscripciones del cliente y los mensajes guardados que se publicaron con una QOS = 1 y a los que se suscribió con una QOS = 1 mientras el dispositivo estaba desconectado. Los mensajes caducados no se entregarán. Para obtener más información acerca de la caducidad de las sesiones con sesiones persistentes, consulte the section called “Sesiones persistentes de MQTT”.

Para obtener más información acerca del almacenamiento persistente, consulte the section called “Sesiones persistentes de MQTT”.

Con mensajes retenidos, el cliente de publicación determina si un mensaje debe conservarse y entregarse a un dispositivo después de que se conecte, independientemente de que haya tenido una sesión anterior o no. La elección de almacenar un mensaje la realiza el publicador y el mensaje almacenado se entrega a todos los clientes actuales y futuros que se suscriban con suscripciones QoS 0 o QoS 1. Los mensajes retenidos guardan solo un mensaje sobre un tema determinado cada vez.

Cuando una cuenta ha almacenado el número máximo de mensajes retenidos, AWS IoT Core devuelve una respuesta limitada a los mensajes publicados con la marca RETAIN activada y con cargas útiles superiores a 0 bytes hasta que se eliminen algunos mensajes retenidos y el recuento de mensajes retenidos caiga por debajo del límite.

Los mensajes retenidos por MQTT y Device Shadows AWS IoT

Tanto los mensajes retenidos como las sombras de dispositivo retienen los datos de un dispositivo, pero se comportan de manera diferente y tienen diferentes propósitos. En esta sección se describen sus similitudes y diferencias.

	Mensajes retenidos	Sombras del dispositivo
La carga de los mensajes tiene una estructura o un esquema predefinidos	Según lo definido por la implementación. MQTT no especifica una estructura o esquema para la carga de sus mensajes.	AWS IoT admite una estructur a de datos específica.
La actualización de la carga del mensaje genera mensajes de eventos	Al publicar un mensaje retenido, se envía el mensaje a los clientes suscritos, pero no se generan mensajes de actualización adicionales.	La actualización de un a sombra de dispositivo produce los mensajes de actualización que describen el cambio.
Las actualizaciones de los mensajes están numeradas	Los mensajes retenidos no se numeran automáticamente.	Los documentos de sombra de dispositivo tienen números de versión y marcas de tiempo automáticos.
La carga del mensaje está adjunta a un recurso	Los mensajes retenidos no se asocian a un recurso de objeto.	Las sombras de dispositivos se asocian a un recurso de objetos.
Actualización de los elementos individuales de la carga del mensaje	Los elementos individuales del mensaje no se pueden cambiar sin actualizar toda la carga del mensaje.	Los elementos individuales de un documento de sombra de dispositivo se pueden actualizar sin necesidad de actualizar todo el documento de sombra de dispositivo.
El cliente recibe los datos del mensaje al suscribirse	El cliente recibe automátic amente un mensaje retenido después de suscribirse a	Los clientes pueden suscribir se a las actualizaciones de sombra de dispositivo, pero

	Mensajes retenidos	Sombras del dispositivo
	un tema con un mensaje retenido.	deben solicitar el estado actual de forma deliberada.
Indexación y capacidad de búsqueda	Los mensajes retenidos no se indexan para su búsqueda.	La indexación de flotas indexa los datos de sombra de dispositivo para su búsqueda y agregación.

Mensajes Last Will and Testament (LWT) de MQTT

Last Will and Testament (LWT) es una característica de MQTT. Con LWT, los clientes pueden especificar un mensaje que el agente publicará en un tema definido por el cliente y lo enviará a todos los clientes que se hayan suscrito al tema cuando se produzca una desconexión no iniciada. El mensaje que especifican los clientes se denomina mensaje LWT o mensaje Will, y el tema que definen los clientes se denomina tema Will. Puede especificar un mensaje LWT cuando un dispositivo se conecte al agente. Estos mensajes se pueden conservar configurando la marca `Will Retain` en el campo `Connect Flag bits` durante la conexión. Por ejemplo, si la marca `Will Retain` está establecido en `1`, el mensaje Will se almacenará en el agente en el tema Will asociado. Para obtener más información, consulte Plantillas de mensaje.

Al administrar las conexiones de los clientes, puede controlar si los mensajes de LWT se publican al desconectar un cliente. Para obtener más información, consulte the section called “Administración de conexiones de MQTT”.

El agente almacenará los mensajes Will hasta que se produzca una desconexión no iniciada. Cuando eso suceda, el agente publicará los mensajes para todos los clientes que se hayan suscrito al tema Will para notificarles la desconexión. Si el cliente se desconecta del agente con una desconexión iniciada por el cliente mediante el mensaje MQTT DISCONNECT, el agente no publicará los mensajes LWT almacenados. En el resto de los casos, se enviarán los mensajes de LWT. Dada la naturaleza asíncrona del procesamiento de desconexiones, no se garantiza que los mensajes de LWT se envíen en orden durante la reconexión. Le recomendamos que utilice los eventos del ciclo de vida para mejorar la precisión de la detección del estado de la conectividad, ya que estos eventos proporcionan atributos como las marcas de tiempo y los números de versión para gestionar out-of-order los eventos. Para obtener una lista completa de los escenarios de desconexión en los que el agente enviará los mensajes de LWT, consulte Eventos de conexión y desconexión.

Uso de connectAttributes

`ConnectAttributes` le permite especificar qué atributos quiere usar en su mensaje de conexión en sus políticas de IAM, como `PersistentConnect` y `LastWill`. Con `ConnectAttributes`, puede crear políticas que no otorguen a los dispositivos acceso a nuevas características de forma predeterminada, lo que puede resultar útil si un dispositivo se ve comprometido.

`connectAttributes` admite las siguientes características:

`PersistentConnect`

Utilice la característica `PersistentConnect` para guardar todas las suscripciones que el cliente contrate durante la conexión cuando se interrumpa la conexión entre el cliente y el agente.

`LastWill`

Utilice la característica `LastWill` para publicar un mensaje `LastWillTopic` cuando un cliente se desconecte inesperadamente.

De forma predeterminada, su política tiene una conexión no persistente y no se transfieren atributos para esta conexión. Debe especificar una conexión persistente en su política de IAM si desea tener una.

Al administrar las conexiones de los clientes, puede ver los atributos de conexión y la configuración de la sesión de los clientes conectados. Para obtener más información, consulte the section called “Administración de conexiones de MQTT”.

Para ver ejemplos de `ConnectAttributes`, consulte Ejemplos de políticas de conexión.

Características compatibles con MQTT 5

AWS IoT Core La compatibilidad con MQTT 5 se basa en la especificación MQTT v5.0, con algunas diferencias, tal como se documenta en. the section called “AWS IoT diferencias con las especificaciones de MQTT”

AWS IoT Core es compatible con las siguientes funciones de MQTT 5:

- Suscripciones compartidas
- Inicio limpio y caducidad de la sesión
- Código de motivo en todos ACKs
- Alias de temas

- Caducidad del mensaje
- Otras características de MQTT 5

Suscripciones compartidas

AWS IoT Core admite suscripciones compartidas tanto para MQTT 3.1.1 como para MQTT 5. Las suscripciones compartidas permiten que varios clientes compartan una suscripción a un tema y solo un cliente recibirá los mensajes publicados sobre ese tema mediante una distribución aleatoria. Las suscripciones compartidas pueden equilibrar eficazmente la carga de los mensajes de MQTT entre varios suscriptores. Por ejemplo, supongamos que tiene 1000 dispositivos que publican sobre el mismo tema y 10 aplicaciones de segundo plano que procesan esos mensajes. En ese caso, las aplicaciones de backend pueden suscribirse al mismo tema y cada una de ellas recibiría de forma aleatoria los mensajes publicados por los dispositivos en el tema compartido. De hecho, se trata de "compartir" la carga de esos mensajes. Las suscripciones compartidas también ofrecen una mayor resiliencia. Cuando una aplicación de segundo plano se desconecta, el agente distribuye la carga entre los suscriptores restantes del grupo. Cuando todos los suscriptores se desconectan, los mensajes se ponen en cola.

Las funciones de cola de mensajes están disponibles para las suscripciones compartidas en las conexiones MQTT 3.1.1 y MQTT 5 con el fin de mejorar la fiabilidad de la entrega de mensajes.

Para usar las suscripciones compartidas, los clientes se suscriben a un filtro de temas de una suscripción compartida de la siguiente manera:

```
$share/{ShareName}/{TopicFilter}
```

- `$share` es una cadena literal que indica el filtro de temas de una suscripción compartida, que debe empezar por `$share`.
- `{ShareName}` es una cadena de caracteres para especificar el nombre compartido utilizado por un grupo de suscriptores. Un filtro de temas de una suscripción compartida debe incluir un `ShareName` e ir seguido por el carácter `/`. `{ShareName}` no debe incluir los siguientes caracteres: `/`, + o #. El tamaño máximo para `{ShareName}` es de 128 caracteres UTF-8.
- `{TopicFilter}` sigue la misma sintaxis de filtro de temas que una suscripción no compartida. El tamaño máximo para `{TopicFilter}` es de 256 caracteres UTF-8.
- Las dos barras diagonales obligatorias (`/`) de `$share/{ShareName}/{TopicFilter}` no se incluyen en el número máximo de barras diagonales por tema y en el límite de filtros de temas.

Las suscripciones que tienen el mismo `{ShareName}/{TopicFilter}` pertenecen al mismo grupo de suscripciones compartidas. Puede crear varios grupos de suscripciones compartidas y no superar el límite de suscripciones compartidas por grupo. Para obtener más información, consulte Puntos de conexión y cuotas de AWS IoT Core en la Referencia general de AWS .

En las siguientes tablas se comparan las suscripciones no compartidas y las suscripciones compartidas:

Suscripción	Description (Descripción)	Ejemplos de filtros de temas
Suscripciones no compartidas	Cada cliente crea una suscripción independiente para recibir los mensajes publicados. Cuando se publica un mensaje en un tema, todos los suscriptores de ese tema reciben una copia del mensaje.	`sports/tennis` `sports/#`
Suscripciones compartidas	Varios clientes pueden compartir una suscripción a un tema y solo un cliente recibirá los mensajes publicados sobre ese tema de forma aleatoria.	`$share/consumer/sports/tennis` `$share/consumer/sports/#`

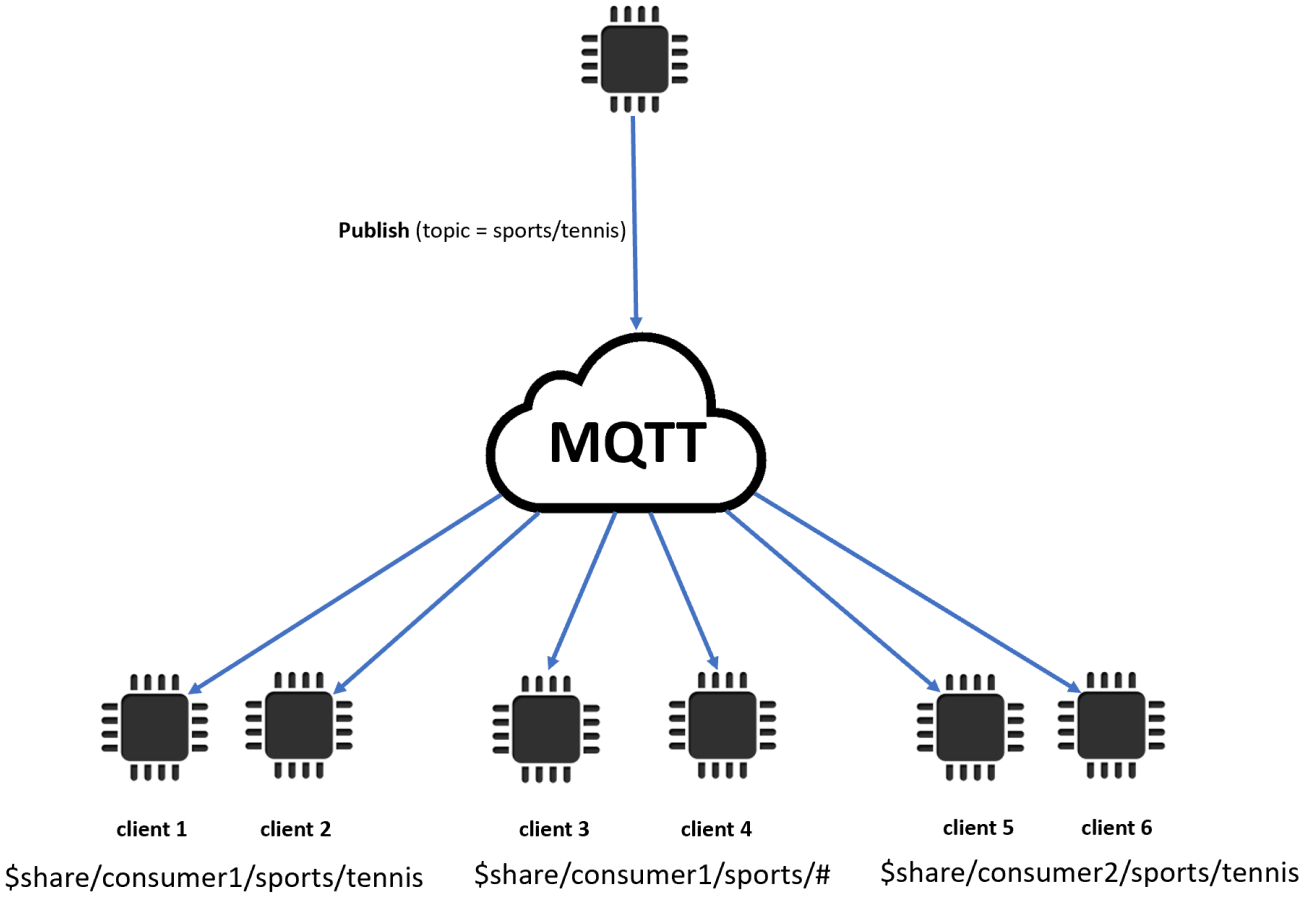
Flujo de suscripciones no compartidas	Flujo de suscripciones compartidas		
			

Notas importantes sobre el uso de las suscripciones compartidas

- Si el grupo compartido de suscriptores está formado por algún suscriptor de sesión persistente, si todos los suscriptores del grupo compartido están desconectados o si algún suscriptor infringe

el límite de solicitudes de publicación por segundo por conexión, se pondrá en cola todos los mensajes de QoS 1 no confirmados y los mensajes de QoS 1 no entregados publicados en un grupo de suscripciones compartidas. Para obtener más información, consulte Cola de mensajes de suscripciones compartidas.

- Los mensajes de QoS 0 publicados en un grupo de suscripciones compartidas se eliminarán en caso de que se produzca un error.
- Las suscripciones compartidas no reciben mensajes retenidos cuando se suscriben a patrones de temas como parte de un grupo compartido de suscriptores. Los mensajes que se publican sobre temas que tienen suscriptores compartidos y que tienen indicador `RETAIN` activado se envían a los suscriptores compartidos como cualquier otro mensaje publicado.
- Cuando las suscripciones compartidas incluyen caracteres comodín (# o +), es posible que haya varias suscripciones compartidas que coincidan con un tema. Si eso ocurre, el agente de mensajes copia el mensaje de publicación y lo envía a un cliente aleatorio en cada suscripción compartida que coincida. En el siguiente diagrama se explica el comportamiento de caracteres comodín de las suscripciones compartidas.



En este ejemplo, hay tres suscripciones compartidas que coinciden con el tema de publicación de MQTT `sports/tennis`. El agente de mensajes copia el mensaje publicado y lo envía a un cliente aleatorio de cada grupo coincidente.

El cliente 1 y el cliente 2 comparten la suscripción: `$share/consumer1/sports/tennis`

El cliente 3 y el cliente 4 comparten la suscripción: `$share/consumer1/sports/#`

El cliente 5 y el cliente 6 comparten la suscripción: `$share/consumer2/sports/tennis`

Para obtener más información sobre los límites de las suscripciones compartidas, consulte Puntos de conexión y las cuotas de AWS IoT Core en la Referencia general de AWS . Para probar las suscripciones compartidas mediante el cliente AWS IoT MQTT de la AWS IoT consola, consulte. ??? También puede ver a qué temas están suscritos los clientes conectados, incluidas las suscripciones compartidas, mediante las características de administración de conexiones de clientes. Para obtener más información, consulte the section called “Administración de conexiones de MQTT”. Para obtener más información sobre las suscripciones compartidas, consulte Suscripciones compartidas de la especificación MQTTv5 .0.

Cola de mensajes de suscripciones compartidas

Para mejorar la fiabilidad de la entrega de mensajes, las suscripciones compartidas incluyen funciones de cola de mensajes que almacenan los mensajes cuando no hay suscriptores en línea disponibles. Cuando un grupo de suscripciones compartidas incluye al menos un miembro con una sesión persistente, la característica de cola estará habilitada para el grupo. Al distribuir los mensajes, se seleccionarán los miembros en línea como destinatarios. Los mensajes de QoS 1 se ponen en cola cuando no se encuentra ningún miembro en línea o cuando los suscriptores superan el límite de `Publish requests per second per connection`. Los mensajes en cola se entregan cuando los miembros existentes reanudan sus sesiones persistentes o cuando se unen al grupo nuevos miembros. Los mensajes en cola se envían a una velocidad máxima de 20 mensajes en cola por segundo por cada suscriptor activo del grupo, junto con cualquier otro mensaje que se entregue al suscriptor según las suscripciones.

De forma predeterminada, la retención de mensajes en cola sigue la cuota de `Persistent Session expiry period`. Sin embargo, si se establece un intervalo de caducidad de mensajes (MEI) en el mensaje de publicación entrante, el MEI tiene prioridad. Cuando el MEI está presente, determina el periodo de retención del mensaje, independientemente del periodo de caducidad de la sesión persistente.

Las tasas de cola de mensajes están limitadas en función de la cuota de `Queued messages per second per account` y la cantidad de mensajes está limitada por la cuota de `Maximum number of queued messages per shared subscription group`. Para ver y administrar sus cuotas, acceda a la consola de Service Quotas.

Puede supervisar la cola CloudWatch buscando `ApproximateQueueDepth` en el espacio de `AWS/Usage` nombres o puede usar el siguiente comando de CLI para enumerar las métricas asociadas a la profundidad de cola de cada grupo de suscripciones compartidas.

```
aws cloudwatch list-metrics --namespace "AWS/Usage" --dimensions
 Name=Resource,Value='ApproximateQueueDepth'
```

Cuando se superan estos límites, solo se conservan los mensajes que ya estaban en cola antes de alcanzar el límite. Se eliminan los nuevos mensajes entrantes que superen los límites. El sistema no reemplaza los mensajes antiguos en cola por otros más nuevos.

La cola de mensajes se registra en métricas y registros. CloudWatch CloudWatch Para obtener información sobre las entradas escritas CloudWatch y los CloudWatch registros, consulte Métricas del agente de mensajes yEntrada de registro en cola. Los mensajes en cola se siguen facturando según la tasa estándar de mensajes. Para obtener más información sobre los precios de los mensajes, consulte Precios de AWS IoT Core.

Ciclo de vida de la sesión en un grupo de suscripciones compartidas

Cuando una sesión válida se suscribe a un grupo, se convierte en miembro en línea del grupo. Cuando se cancela la suscripción o se desconecta, la sesión válida abandona el grupo.

Cuando una sesión persistente se suscribe a un grupo, se convierte en miembro en línea del grupo. Cuando se desconecta, permanece en el grupo, pero pasa a ser un miembro sin conexión del grupo. Cuando se vuelve a conectar, se convierte de nuevo en un miembro en línea. La sesión persistente abandona el grupo cuando se anula la suscripción de forma explícita o cuando caduca tras la desconexión.

Inicio limpio y caducidad de la sesión

Puede usar las funciones de inicio limpio y caducidad de la sesión para gestionar sus sesiones persistentes con más flexibilidad. Un marca de inicio limpio indica si la sesión debe iniciarse sin utilizar una sesión existente. Un intervalo de caducidad de la sesión indica cuánto tiempo se debe conservar la sesión tras una desconexión. El intervalo de caducidad de la sesión se puede modificar

al desconectarse. Para obtener más información, consulte the section called “Sesiones persistentes de MQTT”.

Código de motivo en todos ACKs

Puede depurar o procesar los mensajes de error más fácilmente utilizando los códigos de motivo. El agente de mensajes devuelve los códigos de motivo en función del tipo de interacción con el agente (suscribirse, publicar o confirmar). Para obtener más información, consulte Códigos de motivo de MQTT. Para obtener una lista completa de los códigos de motivo de MQTT, consulte la especificación de MQTT v5.

Alias de temas

Puede sustituir el nombre de un tema por un alias de tema, que es un entero de dos bytes. El uso de alias de temas puede optimizar la transmisión de los nombres de los temas y reducir potencialmente los costos de datos en los servicios de datos medidos. AWS IoT Core tiene un límite predeterminado de 8 alias de temas. Para obtener más información, consulte Puntos de conexión y cuotas de AWS IoT Core en la Referencia general de AWS .

Caducidad del mensaje

Puede agregar valores de caducidad de los mensajes a los mensajes publicados. Estos valores representan el intervalo de caducidad de los mensajes (MEI) en segundos. Si los mensajes no se envían a los suscriptores en ese intervalo, los mensajes caducan y se eliminan. Si no establece el valor de caducidad del mensaje, el mensaje no caducará.

Al salir, el suscriptor recibirá un mensaje con el tiempo restante del intervalo de caducidad. Por ejemplo, si un mensaje de publicación entrante tiene una caducidad de 30 segundos y se envía al suscriptor después de 20 segundos, el campo de caducidad del mensaje se actualizará a 10. Es posible que el mensaje recibido por el suscriptor tenga un MEI actualizado de 0. Esto se debe a que tan pronto como el tiempo restante sea de 999 ms o menos, se actualizará a 0.

En AWS IoT Core, el MEI mínimo es 1. Si el intervalo se establece en 0 desde el lado del cliente, se ajustará a 1. El intervalo máximo de caducidad de mensajes es 604800 (7 días). Cualquier valor superior a este valor se ajustará al valor máximo.

En la comunicación entre versiones, el comportamiento de caducidad del mensaje lo decide la versión MQTT del mensaje de publicación entrante. Por ejemplo, un mensaje con fecha de caducidad enviado por una sesión a través de la cual se ha conectado MQTT5 puede caducar en el caso de los dispositivos suscritos con MQTT3 sesiones. En la siguiente tabla se muestra cómo la caducidad de los mensajes admite los siguientes tipos de mensajes de publicación:

Tipo de mensaje de publicación	Intervalo de caducidad de los mensajes
Publicación regular	Si un servidor no entrega el mensaje en el tiempo especificado, el mensaje caducado se eliminará y el suscriptor no lo recibirá. Esto incluye situaciones como cuando un dispositivo no publica sus mensajes de QoS 1.
Retener	Si un mensaje retenido caduca y un cliente nuevo se suscribe al tema, el cliente no recibirá el mensaje al suscribirse.
Última voluntad	El intervalo de los mensajes de última voluntad comienza después de que el cliente se desconecte y el servidor intente entregar el mensaje de última voluntad a sus suscriptores.
Mensajes en cola	Si un QoS 1 saliente con intervalo de caducidad de mensajes caduca cuando un cliente está desconectado, cuando se reanude la sesión persistente, el cliente no recibirá el mensaje caducado.

Otras características de MQTT 5

Desconexión del servidor

Cuando se produce una desconexión, el servidor puede enviar de forma proactiva al cliente una desconexión para notificar el cierre de la conexión con un código de motivo de la desconexión.

Solicitud/respuesta

Los editores pueden solicitar que el destinatario envíe una respuesta a un tema especificado por el editor en el momento de la recepción.

Tamaño máximo de paquete

El cliente y el servidor pueden especificar de forma independiente el tamaño máximo de paquete que admiten.

Formato de carga y tipo de contenido

Puede especificar el formato de carga (binario, texto) y el tipo de contenido al publicar un mensaje. Se reenvían al destinatario del mensaje.

Propiedades de MQTT 5

Las propiedades de MQTT 5 son adiciones importantes al estándar MQTT para admitir las nuevas funciones de MQTT 5, como la caducidad de la sesión y el patrón. Request/Response En AWS IoT Core, puede crear reglas que reenvíen las propiedades de los mensajes salientes o utilizar HTTP Publish para publicar los mensajes MQTT con algunas de las nuevas propiedades.

En la siguiente tabla se enumeran todas las propiedades de MQTT 5 compatibles. AWS IoT Core

Propiedad	Description (Descripción)	Tipo de entrad	Paquete
Indicador de formato de carga	Un valor booleano que indica si la carga útil tiene formato UTF-8.	Byte	PUBLISH, CONNECT
Tipo de contenido	Una cadena UTF-8 que describe el contenido de la carga.	Caden UTF-8	PUBLISH, CONNECT
Tema de respuesta	Una cadena UTF-8 que describe el tema en el que el receptor debe publicar como parte del flujo de solicitud-respuesta. El tema no debe contener caracteres comodín.	Caden UTF-8	PUBLISH, CONNECT
Datos de correlación	Los datos binarios que utiliza el remitente del mensaje de solicitud para identificar a qué solicitud corresponde el mensaje de respuesta.	Binario	PUBLISH, CONNECT
Propiedades del usuario	Un par de cadenas UTF-8. Esta propiedad puede aparecer varias veces en un paquete. Los receptores recibirán los pares clave-valor en el mismo orden en que se enviaron.	Par de caden UTF-8	CONNECT, PUBLISH, Propiedades Will, SUBSCRIBE, DISCONNECT, UNSUBSCRIBE
Intervalo de caducidad de los mensajes	Un entero de 4 bytes que representa el intervalo de caducidad del mensaje en segundos. Si no existe, el mensaje no vence.	Enterc de 4 byte	PUBLISH, CONNECT

Propiedad	Description (Descripción)	Tipo de entrad	Paquete
Intervalo de caducidad de la sesión	Un entero de 4 bytes que representa el intervalo de caducidad de la sesión en segundos. AWS IoT Core admite un máximo de 7 días, con un máximo predeterminado de una hora. Si el valor que ha estableci do supera el máximo de su cuenta, AWS IoT Core devolverá el valor ajustado en el CONNACK.	Enterc de 4 byte	CONNECT, CONNACK, DISCONNECT
Identificador de cliente asignado	Un ID de cliente aleatorio que se genera AWS IoT Core cuando los dispositivos no especific an un ID de cliente. El ID de cliente aleatorio debe ser un identificador de cliente nuevo que no utilice ninguna otra sesión gestionada actualmente por el agente.	Caden UTF-8	CONNACK
Servidor Keep Alive	Un entero de 2 bytes que representa el tiempo de mantenimiento activo asignado por el servidor. El servidor desconectará al cliente si el cliente permanece inactivo durante más tiempo que el tiempo de mantenimiento activo.	Enterc de 2 byte	CONNACK
Solicitar informaci ón sobre el problema	Un valor booleano que indica si la cadena de motivo o las propiedades del usuario se envían en caso de errores.	Byte	CONNECT
Recibir máximo	Un entero de 2 bytes que representa el número máximo de paquetes PUBLISH QOS > 0 que se pueden enviar sin recibir un PUBACK.	Enterc de 2 byte	CONNECT, CONNACK

Propiedad	Description (Descripción)	Tipo de entrad	Paquete
Máximo de alisas de tema	Este valor indica el valor más alto que se aceptará como alias de tema. El valor predeterminado es 0.	Enterc de 2 byte	CONNECT, CONNACK
QoS máxima	El valor máximo de QoS que AWS IoT Core admite. El valor predeterminado es 1. AWS IoT Core no admite QoS 2.	Byte	CONNACK
Retener disponible	Un valor booleano que indica si el agente de mensajes admite los AWS IoT Core mensajes retenidos. El valor predeterminado de es 1.	Byte	CONNACK
Tamaño máximo de paquete	El tamaño máximo de paquete que se AWS IoT Core acepta y envía. No puede superar los 128 KB.	Enterc de 4 byte	CONNECT, CONNACK
Suscripci ón comodín disponible	Un valor booleano que indica si el agente de AWS IoT Core mensajes admite la suscripci ón Wildcard disponible. El valor predeterm inado de es 1.	Byte	CONNACK
Identificador de suscripci ón disponible	Un valor booleano que indica si el agente de AWS IoT Core mensajes admite el identific ador de suscripción disponible. El valor predeterminado es 0.	Byte	CONNACK

Códigos de motivo de MQTT

MQTT 5 introduce una mejora en la notificación de errores con respuestas en códigos de motivo. AWS IoT Core puede devolver códigos de motivo, incluidos, entre otros, los siguientes agrupados por paquetes. Para obtener una lista completa de los códigos de motivo compatibles con MQTT 5, consulte las especificaciones de MQTT 5.

Códigos de motivo de CONNACK

Valor	Hex	Motivo: nombre en clave	Description (Descripción)
0	0x00	Success	Se acepta la conexión.
128	0x80	Error no especificado	El servidor no desea revelar el motivo del fallo o no se aplica ninguno de los otros códigos de motivo.
133	0x85	El identificador de cliente no es válido	El identificador del cliente es una cadena válida, pero el servidor no lo permite.
134	0x86	Nombre de usuario o contraseña incorrectos	El servidor no acepta el nombre de usuario o la contraseña especificados por el cliente.
135	0x87	No tiene autorización	El cliente no está autorizado a conectarse.
144	0x90	El nombre del tema no es válido	El nombre del tema Will está formado correctamente, pero el servidor no lo acepta.
151	0x97	Cuota excedida	Se ha superado un límite de implementación o impuesto por la administración.
155	0x9B	QoS no admitida	El servidor no admite la QoS establecida en Will QoS.

Códigos de motivo de PUBACK

Valor	Hex	Motivo: nombre en clave	Description (Descripción)
0	0x00	Success	Se acepta el mensaje. Continúa la publicación del mensaje de QoS 1.

Valor	Hex	Motivo: nombre en clave	Description (Descripción)
128	0x80	Error no especificado	El receptor no acepta la publicación, pero no quiere revelar el motivo o no coincide con ninguno de los otros valores.
135	0x87	No tiene autorización	La PUBLICACIÓN no está autorizada.
144	0x90	El nombre del tema no es válido	El nombre del tema no tiene un formato incorrecto, pero el cliente o el servidor no lo aceptan.
145	0x91	Identificador de paquetes en uso	El identificador del paquete ya está en uso. Esto puede indicar una discrepancia en el estado de la sesión entre el cliente y el servidor.
151	0x97	Cuota excedida	Se ha superado un límite de implementación o impuesto por la administración.

Códigos de motivo de desconexión

Valor	Hex	Motivo: nombre en clave	Description (Descripción)
129	0x81	Paquete con formato incorrecto	El paquete recibido no cumple con esta especificación.
130	0x82	Error de protocolo	Se recibió un paquete inesperado o fuera de servicio.
135	0x87	No tiene autorización	La solicitud no está autorizada.

Valor	Hex	Motivo: nombre en clave	Description (Descripción)
139	0x8B	El servidor se está apagando	El servidor se está apagando.
141	0x8D	Tiempo de espera de Keep Alive	La conexión está cerrada porque no se ha recibido ningún paquete durante 1,5 veces el tiempo de Keep Alive.
142	0x8E	Sesión sustituida	Se ha conectado otra conexión que utiliza el mismo ID de cliente, lo que provoca el cierre de esta conexión.
143	0x8F	Filtro de tema no válido	El filtro de temas está formado correctamente, pero el servidor no lo acepta.
144	0x90	El nombre del tema no es válido	El nombre del tema está formado correctamente, pero este cliente o servidor no lo acepta.
147	0x93	Límite de recepción excedido	El cliente o servidor ha recibido más publicaciones que la cantidad máxima de recepción para la que no ha enviado PUBACK o PUBCOMP.
148	0x94	Alias de tema no válido	El cliente o el servidor ha recibido un paquete PUBLISH que contiene un alias de tema superior al alias de tema máximo que envió en el paquete CONNECT o CONNACK.
151	0x97	Cuota excedida	Se ha superado un límite de implementación o impuesto por la administración.
152	0x98	Acción administrativa	La conexión se cierra debido a una acción administrativa.
155	0x9B	QoS no admitida	El cliente especificó una QoS superior a la QoS especific ada en una QoS máxima en el CONNACK.

Valor	Hex	Motivo: nombre en clave	Description (Descripción)
161	0xA1	No se admiten los identificadores de suscripción	El servidor no admite identificadores de suscripción; no se acepta la suscripción.

Códigos de motivo de SUBACK

Valor	Hex	Motivo: nombre en clave	Description (Descripción)
0	0x00	QoS 0 concedida	Se acepta la suscripción y la QoS máxima enviada será QoS 0. Esto podría ser una QoS inferior a la solicitada.
1	0x01	QoS 1 concedida	Se acepta la suscripción y la QoS máxima enviada será QoS 1. Esto podría ser una QoS inferior a la solicitada.
128	0x80	Error no especificado	No se acepta la suscripción y el servidor no quiere revelar el motivo o no se aplica ninguno de los demás códigos de motivo.
135	0x87	No tiene autorización	El cliente no está autorizado a realizar esta suscripción.
143	0x8F	Filtro de tema no válido	El filtro de temas está formado correctamente, pero no está permitido para este cliente.
145	0x91	Identificador de paquetes en uso	El identificador de paquete especificado ya está en uso.
151	0x97	Cuota excedida	Se ha superado un límite de implementación o impuesto por la administración.

Códigos de motivo de UNSUBACK

Valor	Hex	Motivo: nombre en clave	Description (Descripción)
0	0x00	Success	Se elimina la suscripción.
128	0x80	Error no especificado	No se ha podido cancelar la suscripción y el servidor no quiere revelar el motivo o no se aplica ninguno de los demás códigos de motivo.
143	0x8F	Filtro de tema no válido	El filtro de temas está formado correctamente, pero no está permitido para este cliente.
145	0x91	Identificador de paquetes en uso	El identificador de paquete especificado ya está en uso.

AWS IoT diferencias con las especificaciones de MQTT

La implementación del agente de mensajes se basa en la especificación MQTT v3.1.1 y la especificación MQTT v5.0, pero se diferencia de las especificaciones en los siguientes aspectos:

- AWS IoT no admite los siguientes paquetes para MQTT 3: PUBREC, PUBREL y PUBCOMP.
- AWS IoT no admite los siguientes paquetes para MQTT 5: PUBREC, PUBREL, PUBCOMP y AUTH.
- AWS IoT no admite la redirección de servidores MQTT 5.
- AWS IoT solo admite los niveles de calidad de servicio (QoS) 0 y 1 de MQTT. AWS IoT no admite la publicación ni la suscripción con el nivel 2 de QoS. El agente de mensajes no envía PUBACK o SUBACK cuando se solicita QoS nivel 2.
- En AWS IoT, suscribirse a un tema con un nivel de QoS 0 significa que un mensaje se entrega 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.
- 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.

- Antes de enviar paquetes de control adicionales o una solicitud de desconexión, el cliente debe esperar a que recibir el mensaje CONNACK del agente de mensajes de AWS IoT .
- 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.
- Cuando un cliente utiliza el carácter comodín # del filtro de temas para suscribirse a un tema, coinciden todas las cadenas que estén en su nivel o por debajo de él en la jerarquía de temas. Sin embargo, el tema principal no coincide. Por ejemplo, una suscripción al tema `sensor/#` recibe los mensajes publicados en los temas `sensor/`, `sensor/temperature` y `sensor/temperature/room1`, pero no los mensajes publicados en `sensor`. Para obtener más información acerca de cómo utilizar comodines, consulte Filtros de nombres de temas.
- 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 estar conectados simultáneamente al agente de mensajes. Cuando un cliente se conecta al agente de mensajes mediante un ID de cliente que otro cliente está utilizando, se acepta la nueva conexión de cliente y se desconecta el cliente previamente conectado. También puede desconectar los clientes manualmente mediante. APIs Para obtener más información, consulte the section called “Administración de conexiones de MQTT”.
- En raras ocasiones, el agente de mensajes reenviará el mismo mensaje PUBLISH lógico con otro ID de paquete.
- La suscripción a filtros de temas que contienen un carácter comodín no puede recibir los mensajes retenidos. Para recibir un mensaje retenido, la solicitud de suscripción debe contener un filtro de tema que coincida exactamente con el tema del mensaje retenido.
- El agente de mensajes no garantiza el orden de recepción de los mensajes y ACK.
- AWS IoT puede tener límites distintos de los especificados. Para obtener más información, consulte Límites y cuotas del agente de mensajes y del protocolo de AWS IoT Core en la Guía de referencia de AWS IoT .
- No se admite el indicador MQTT DUP.

Administración de conexiones de MQTT

AWS IoT Core proporciona ayuda APIs para gestionar las conexiones MQTT, incluida la posibilidad de desconectar los clientes y gestionar sus sesiones. Estas funciones le proporcionan un mayor control sobre su flota de AWS IoT clientes y le ayudan a solucionar problemas de conexión.

DeleteConnection API

Utilice la `DeleteConnection` API para desconectar los dispositivos MQTT AWS IoT Core especificando su cliente IDs. Al desconectar un cliente, lo AWS IoT Core desconecta del intermediario de mensajes y, si lo desea, puede limpiar el estado de la sesión y suprimir el AWS IoT Core mensaje de Last Will and Testament (LWT).

Cuando llamas a la `DeleteConnection` API, AWS IoT Core realiza varias acciones para garantizar una desconexión limpia. AWS IoT Core primero envía un mensaje de desconexión de MQTT al cliente para finalizar la sesión de MQTT. A continuación, el servicio cierra el conector subyacente TCP/TLS .

El agente de mensajes envía un paquete `DISCONNECT` al dispositivo y publica un evento del ciclo de vida con el motivo de la desconexión `API_INITIATED_DISCONNECT`. Esto le ayuda a identificar cuándo se inició una desconexión a través de la API en lugar de hacerlo a través del cliente o debido a problemas de red. Puede supervisar estos eventos con fines de visibilidad, resolución de problemas y auditoría. Por ejemplo, puede usar AWS IoT reglas para procesar estos eventos y realizar un seguimiento de cuándo y por qué se desconectaron los clientes.

Si establece el `cleanSession` parámetro en`true`, AWS IoT Core elimina el estado de la sesión del cliente, incluidas todas las suscripciones y los mensajes en cola. Si borra una sesión, la sesión persistente finalizará. Si el cliente era una sesión persistente y el parámetro `preventWillMessage` está establecido en `false`, el servicio enviará el mensaje de LWT si está disponible, lo que resulta útil durante las operaciones de mantenimiento planificadas.

Al llamar a la API `DeleteConnection`, el proceso de desconexión comienza inmediatamente, pero el momento exacto en que el cliente reconoce la desconexión puede variar en función de las condiciones de la red y de la implementación del cliente. La mayoría de los clientes detectarán la desconexión en unos segundos, pero en algunos casos con una conectividad de red deficiente, es posible que el cliente tarde más en reconocer que se ha desconectado.

> **Note**
>
> Al forzar la desconexión, el cliente debe volver a autenticarse y volver a autorizarse con un estado de sesión nuevo. La llamada a la API en sí misma no impide la reconexión de los clientes. Si lo desea, también debe modificar la credencial o política del cliente antes de emitir la llamada a la API `DeleteConnection`.

Para obtener información sobre precios, consulte Precios de AWS IoT Core.

Casos de uso

La API `DeleteConnection` es útil para administrar los clientes malintencionados y que muestran un comportamiento problemático o consumen un número excesivo de recursos. Al forzar la desconexión, puede asegurarse de que los clientes restablezcan sus conexiones con la autenticación y la autorización adecuadas, lo que puede ayudar a resolver los problemas de consumo de recursos.

Las situaciones de redireccionamiento de clientes también sacan partido de esta API. Cuando necesite redirigir a los clientes a distintos puntos de conexión o bien Regiones de AWS, puede desconectarlos mediante programación y hacer que se vuelvan a conectar a un AWS IoT Core punto final diferente cambiando la configuración de DNS. Esta API puede ayudar a resolver conexiones bloqueadas o eliminar estados de sesión problemáticos que podrían estar impidiendo el funcionamiento normal.

Parámetros de la API

La API `DeleteConnection` acepta los siguientes parámetros:

clientId (obligatorio)

El identificador único del cliente de MQTT que se va a desconectar. Se especifica en la ruta URL. El ID del cliente no puede comenzar por el símbolo de dólar ($).

> **Note**
>
> El cliente MQTT IDs puede contener caracteres que no son válidos en las solicitudes HTTP. Al utilizar la API `DeleteConnection`, debe codificar mediante URL (codificación porcentual) cualquier carácter del ID de cliente que sea válido en MQTT pero no en HTTP. Esto incluye caracteres especiales como espacios, barras diagonales (/) y caracteres UTF-8. Por ejemplo, un espacio se convierte en %20, una barra diagonal se convierte en %2F y el carácter UTF-8 ü se convierte en %C3%BC. La codificación adecuada garantiza que el cliente MQTT IDs se transmita correctamente en la llamada a la API basada en HTTP.

cleanSession (opcional)

Especifica si se debe eliminar el estado de sesión del cliente al desconectarse. Establezca en `true` para eliminar toda la información sobre la sesión, incluidos mensajes en cola y suscripciones. Establezca en `false` para conservar el estado de la sesión. De forma predeterminada, esta opción está establecida en `false` (conserva el estado de la sesión). En las sesiones válidas, se ignorará este parámetro.

preventWillMessage (opcional)

Controla si AWS IoT Core envía el mensaje de última voluntad y testamento (LWT) si está disponible en el momento de la desconexión. Establezca en `true` para evitar el envío del mensaje de LWT. Establezca en `false` para permitir el envío. De forma predeterminada, esta opción está establecida en `false` (envía LWT si está disponible).

Sintaxis de la API

La API `DeleteConnection` utiliza el siguiente formato de solicitud HTTP:

```
DELETE /connections/<clientId>?
cleanSession=<cleanSession>&preventWillMessage=<preventWillMessage> HTTP/1.1
```

Solicitudes de ejemplo:

```
// Basic disconnect (preserves session, allows LWT message)
DELETE /connections/myDevice123 HTTP/1.1

// Disconnect and clear session
DELETE /connections/myDevice123?cleanSession=TRUE HTTP/1.1

// Disconnect, clear session, and prevent LWT message
DELETE /connections/myDevice123?cleanSession=TRUE&preventWillMessage=TRUE HTTP/1.1
```

Las solicitudes correctas devuelven HTTP 200 OK sin cuerpo de la respuesta.

> **Note**
>
> El nombre del servicio que utiliza AWS Signature Version 4 para firmar las solicitudes es: iotdevicegateway. Puede encontrar el punto de conexión mediante el comando `aws iot describe-endpoint --endpoint-type iot:Data-ATS`.

Permisos necesarios

Para usar la API `DeleteConnection`, se requiere el siguiente permiso de IAM:

```
iot:DeleteConnection
```

Puede asignar este permiso a un cliente específico IDs mediante políticas basadas en recursos. Por ejemplo:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iot:DeleteConnection",
            "Resource": "arn:aws:iot:region:account:client/myDevice*"
        }
    ]
}
```

Consideraciones importantes

Los clientes desconectados pueden intentar volver a conectarse inmediatamente, a menos que haya implementado una lógica adicional para impedir la reconexión. La operación de desconexión solo termina la conexión actual y no impide que una conexión se vuelva a conectar. Si necesita impedir la reconexión, plantéese implementar una lógica de cliente o deshabilitar la credencial de un dispositivo.

Los límites de velocidad se aplican a la API como parte de la limitación de velocidad de la AWS IoT Core API estándar. Cuando planifique las operaciones de desconexión masiva, asegúrese de tener en cuenta estos límites e implementar la lógica de reintentos y las estrategias de procesamiento por lotes adecuadas para evitar las limitaciones. Para obtener más información, consulte Puntos de conexión y cuotas de AWS IoT Core.

Respuestas de error

La API `DeleteConnection` puede devolver las siguientes respuestas de error:

InvalidRequestException

La solicitud no es válida. Esto puede ocurrir si el formato del ID de cliente no es válido, incluye un prefijo con el signo de dólar ($) o si faltan los parámetros necesarios.

ResourceNotFoundException

El ID de cliente especificado no existe o no está conectado actualmente y no tiene una sesión persistente.

UnauthorizedException

No está autorizado para realizar esta operación. Asegúrese de que tenga el permiso `iot:DeleteConnection` necesario.

ForbiddenException

La persona que llama no está autorizada a realizar la solicitud. Esto puede deberse a la insuficiencia de los permisos de IAM o a las restricciones de las políticas basadas en los recursos.

ThrottlingException

El índice supera el límite. Reduzca la frecuencia de las llamadas a la API e implemente la lógica de reintentos adecuada con un retroceso exponencial.

InternalFailureException

Se ha producido un error inesperado. Vuelva a intentar la solicitud después de esperar unos segundos.

ServiceUnavailableException

El servicio no está disponible temporalmente. Vuelva a intentar la solicitud después de esperar unos segundos.

Publicación en HTTPS

Los clientes pueden publicar mensajes realizando solicitudes a la API REST utilizando los protocolos HTTP 1.0 o 1.1. Para obtener información sobre la autenticación y las asignaciones de puertos utilizados por las solicitudes HTTP, consulte ???.

> **Note**
>
> HTTPS no admite un valor `clientId` como lo hace MQTT. `clientId` está disponible cuando se usa MQTT, pero no cuando se usa HTTPS.

URL del mensaje HTTPS

Los dispositivos y los clientes publican sus mensajes con solicitudes POST a un punto de conexión específico del cliente y a una dirección URL específica del tema:

```
https://IoT_data_endpoint/topics/url_encoded_topic_name?qos=1
```

- *IoT_data_endpoint*es el punto final de los datos del AWS IoT dispositivo. Puedes encontrar el punto final en la AWS IoT consola, en la página de detalles del dispositivo, o en el cliente mediante el AWS CLI comando:

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

  El punto de conexión debería tener un aspecto similar al siguiente: `a3qjEXAMPLEffp-ats.iot.us-west-2.amazonaws.com`
- *url_encoded_topic_name*es el nombre completo del tema del mensaje que se envía.

Ejemplos de código de mensajes HTTPS

Estos son algunos ejemplos de cómo enviar un mensaje HTTPS a AWS IoT.

Python (port 8443)

```
import requests
import argparse

# define command-line parameters
parser = argparse.ArgumentParser(description="Send messages through an HTTPS
 connection.")
parser.add_argument('--endpoint', required=True, help="Your AWS IoT data custom
 endpoint, not including a port. " +
                                                       "Ex: \"abcdEXAMPLExyz-
ats.iot.us-east-1.amazonaws.com\"")
parser.add_argument('--cert', required=True, help="File path to your client
 certificate, in PEM format.")
parser.add_argument('--key', required=True, help="File path to your private key, in
 PEM format.")
parser.add_argument('--topic', required=True, default="test/topic", help="Topic to
 publish messages to.")
```

```
parser.add_argument('--message', default="Hello World!", help="Message to publish. "
 +
                                                        "Specify empty string to
 publish nothing.")

# parse and load command-line parameter values
args = parser.parse_args()

# create and format values for HTTPS request
publish_url = 'https://' + args.endpoint + ':8443/topics/' + args.topic + '?qos=1'
publish_msg = args.message.encode('utf-8')

# make request
publish = requests.request('POST',
            publish_url,
            data=publish_msg,
            cert=[args.cert, args.key])

# print results
print("Response status: ", str(publish.status_code))
if publish.status_code == 200:
        print("Response body:", publish.text)
```

Python (port 443)

```
import requests
import http.client
import json
import ssl

ssl_context = ssl.SSLContext(protocol=ssl.PROTOCOL_TLS_CLIENT)
ssl_context.minimum_version = ssl.TLSVersion.TLSv1_2

# note the use of ALPN
ssl_context.set_alpn_protocols(["x-amzn-http-ca"])
ssl_context.load_verify_locations(cafile="./<root_certificate>")

# update the certificate and the AWS endpoint
ssl_context.load_cert_chain("./<certificate_in_PEM_Format>",
 "<private_key_in_PEM_format>")
connection = http.client.HTTPSConnection('<the ats IoT endpoint>', 443,
 context=ssl_context)
message = {'data': 'Hello, I'm using TLS Client authentication!'}
```

```
json_data = json.dumps(message)
connection.request('POST', '/topics/device%2Fmessage?qos=1', json_data)

# make request
response = connection.getresponse()

# print results
print(response.read().decode())
```

CURL

Puede usar curl desde un cliente o dispositivo para enviar un mensaje a AWS IoT.

Para usar curl para enviar un mensaje desde un dispositivo AWS IoT cliente

1. Compruebe la versión de curl.

   a. En su cliente, ejecute este comando en un símbolo del sistema.

      curl --help

      En el texto de ayuda, busque las opciones TLS. Debería ver la opción `--tlsv1.2`.

   b. Si ve la opción `--tlsv1.2`, continúe.

   c. Si no ve la opción `--tlsv1.2` o aparece un error `command not found`, tal vez deba actualizar o instalar curl en el cliente o instalar `openssl` antes de continuar.

2. Instale los certificados en su cliente.

   Copie los archivos de certificado que creó al registrar su cliente (cosa) en la AWS IoT consola. Asegúrese de que tiene estos tres archivos de certificado en su cliente antes de continuar.

   - El archivo de certificado de entidad de certificación (*`Amazon-root-CA-1.pem`* en este ejemplo).
   - El archivo de certificado del cliente (*`device.pem.crt`* en este ejemplo).
   - El archivo de clave privada del cliente (*`private.pem.key`* en este ejemplo).

3. Cree la línea de comandos de curl y sustituya los valores reemplazables por los de su cuenta y sistema.

   ```
   curl --tlsv1.2 \
       --cacert Amazon-root-CA-1.pem \
   ```

```
    --cert device.pem.crt \
    --key private.pem.key \
    --request POST \
    --data "{ \"message\": \"Hello, world\" }" \
    "https://IoT_data_endpoint:8443/topics/topic?qos=1"
```

--tlsv1.2

Use TLS 1.2 (SSL).

--cacert *Amazon-root-CA-1.pem*

El nombre de archivo y la ruta de acceso, si es necesario, del certificado de entidad de certificación para verificar el par.

--cert *device.pem.crt*

El nombre y la ruta de acceso del archivo de certificado del cliente, si es necesario.

--clave *private.pem.key*

El nombre y la ruta del archivo de la clave privada del cliente, si es necesario.

--request POST

El tipo de solicitud HTTP (en este caso, POST).

--datos «» *{ \"message\": \"Hello, world\" }*

Los datos de HTTP POST que quiere publicar. En este caso, es una cadena JSON, con las comillas internas con el carácter de escape de barra invertida (\).

«https: //:8443/temas/? *IoT_data_endpoint topic* qos=1"

La URL del punto de conexión de datos del AWS IoT dispositivo de su cliente, seguida del puerto HTTPS`:8443`, seguido de la palabra clave `/topics/` y, en este caso, `topic` del nombre del tema. Especifique la calidad del servicio como parámetro de consulta, ? `qos=1`.

4. Abra el cliente de pruebas de MQTT en la AWS IoT consola.

   Siga las instrucciones Vea los mensajes MQTT con el cliente AWS IoT MQTT y configure la consola para suscribirse a los mensajes con el nombre del tema *topic* utilizado en su curl comando, o utilice el filtro de temas comodín de. #

5. Pruebe el comando.

Mientras monitoriza el tema en el cliente de prueba de la consola de AWS IoT , vaya a su cliente y ejecute la línea de comandos curl que creó en el paso 3. Debería ver los mensajes de su cliente en la consola.

Temas de MQTT

Los temas de MQTT identifican los AWS IoT mensajes. AWS IoT los clientes identifican los mensajes que publican asignando nombres a los temas de los mensajes. Los clientes identifican los mensajes a los que desean suscribirse (recibir) registrando un filtro de temas con AWS IoT Core. El agente de mensajes de utiliza nombres y filtros de temas para dirigir los mensajes de los clientes de publicación a los clientes de suscripción.

El agente de mensajes utiliza los temas para identificar los mensajes enviados mediante MQTT y enviados mediante HTTP al URL del mensaje HTTPS.

Aunque AWS IoT admite algunos temas de sistema reservados, la mayoría de los temas de MQTT los crea y gestiona usted, el diseñador del sistema. AWS IoT utiliza los temas para identificar los mensajes recibidos de los clientes de publicación y seleccionar los mensajes que se van a enviar a los clientes suscritos, tal y como se describe en las siguientes secciones. Antes de crear un espacio de nombres de temas para el sistema, consulte las características de los temas de MQTT para crear la jerarquía de nombres de temas que funcione mejor con su sistema de IoT.

Nombres de temas

Los nombres de los temas y los filtros de temas son cadenas codificadas por UTF-8. Pueden representar una jerarquía de información utilizando el carácter de barra diagonal (/) para separar los niveles de la jerarquía. Por ejemplo, este nombre de tema podría referirse a un sensor de temperatura en la sala 1:

- `sensor/temperature/room1`

En este ejemplo, también puede haber otros tipos de sensores en otras salas con nombres de temas como:

- `sensor/temperature/room2`
- `sensor/humidity/room1`
- `sensor/humidity/room2`

> **Note**
>
> Cuando considere los nombres de los temas para los mensajes de su sistema, tenga en cuenta:
>
> - Los nombres de temas y los filtros de temas distinguen entre mayúsculas y minúsculas.
> - Los nombres de los temas no deben contener información de identificación personal.
> - Los nombres de tema que comienzan por $ son temas reservados que debe utilizar únicamente AWS IoT Core.
> - AWS IoT Core no puedo enviar ni recibir mensajes entre Cuenta de AWS nosotros o regiones.

Para obtener más información sobre cómo diseñar los nombres de los temas y el espacio de nombres, consulte nuestro documento técnico, Diseño de temas MQTT para AWS IoT Core.

Para ver ejemplos de cómo las aplicaciones pueden publicar y suscribirse a mensajes, empiece por Cómo empezar con AWS IoT Core los tutoriales y SDK de dispositivos, SDK para móviles y cliente de dispositivo de AWS IoT.

> **Important**
>
> El espacio de nombres del tema está limitado a una región Cuenta de AWS y. Por ejemplo, el `sensor/temp/room1` tema utilizado por un usuario Cuenta de AWS de una región es distinto del `sensor/temp/room1` tema utilizado por la misma AWS cuenta en otra región o utilizado por cualquier otra cuenta Cuenta de AWS en cualquier región.

ARN de tema

Todos los temas ARNs (Amazon Resource Names) tienen el siguiente formulario:

```
arn:aws:iot:aws-region:AWS-account-ID:topic/Topic
```

Por ejemplo, `arn:aws:iot:us-west-2:123EXAMPLE456:topic/application/topic/device/sensor` es un ARN del tema `application/topic/device/sensor`.

Filtros de nombres de temas

Los clientes suscriptores registran filtros de nombres de temas con el agente de mensajes para especificar los temas de mensajes que el agente de mensajes debe enviarles. Un filtro de nombre de tema puede ser un único nombre de tema para suscribirse a un único nombre de tema o puede incluir caracteres comodín para suscribirse a varios nombres de tema a la vez.

Los clientes de publicación no pueden utilizar caracteres comodín en los nombres de tema que publican.

En la tabla siguiente se enumeran los caracteres comodín que se pueden utilizar en un filtro de temas.

Comodines de tema

Carácter comodín	Coincide	Notas
#	Todas las cadenas en y por debajo de su nivel en la jerarquía de temas.	Debe ser el último carácter del filtro de temas. Debe ser el único carácter en su nivel de jerarquía de temas. Se puede utilizar en un filtro de temas que también contiene el carácter comodín +.
+	Cualquier cadena en el nivel que contiene el carácter.	Debe ser el único carácter en su nivel de jerarquía de temas. Se puede utilizar en varios niveles de un filtro de tema.

Uso de comodines con los ejemplos de nombres de tema de sensor anteriores:

- Una suscripción a `sensor/#` recibe los mensajes publicados en `sensor/`, `sensor/temperature` y `sensor/temperature/room1`, pero no los mensajes publicados en `sensor`.

- Una suscripción a `sensor/+/room1` recibe mensajes publicados en `sensor/temperature/room1` y `sensor/humidity/room1`, pero no mensajes enviados a `sensor/temperature/room2` o `sensor/humidity/room2`.

ARN de filtro de temas

Todos los filtros de temas ARNs (Amazon Resource Names) tienen el siguiente formulario:

```
arn:aws:iot:aws-region:AWS-account-ID:topicfilter/TopicFilter
```

Por ejemplo, `arn:aws:iot:us-west-2:123EXAMPLE456:topicfilter/application/topic/+/sensor` es un ARN del filtro de tema `application/topic/+/sensor`.

Carga de mensajes MQTT

La carga útil de mensajes que se envía en tus mensajes MQTT no viene especificada por AWS IoT, a menos que sea para uno de los. the section called “Temas reservados” Para adaptarse a las necesidades de su aplicación, le recomendamos que defina la carga de mensajes para sus temas dentro de las restricciones de AWS IoT Core Service Quotas para Protocolos.

El uso de un formato JSON para la carga útil de los mensajes permite al motor de AWS IoT reglas analizar los mensajes y aplicarles consultas SQL. Si su aplicación no requiere que el motor de reglas aplique consultas SQL a las cargas útiles de sus mensajes, puede utilizar cualquier formato de datos que requiera su aplicación. Para obtener información sobre las limitaciones y los caracteres reservados de un documento JSON utilizado en las consultas SQL, consulte Extensiones JSON.

Para obtener más información sobre el diseño de los temas de MQTT y sus correspondientes cargas de mensajes, consulte Diseño de temas de MQTT para AWS IoT Core.

Si el límite de tamaño de un mensaje supera las cuotas de servicio, se producirá un `CLIENT_ERROR` con motivo `PAYLOAD_LIMIT_EXCEEDED` y “La carga útil del mensaje supera el límite de tamaño para el tipo de mensaje.” Para obtener más información sobre el límite de tamaño de los mensajes, consulte Límites y cuotas del agente de mensajes AWS IoT Core.

Temas reservados

Los temas que comienzan con un signo de dólar ($) están reservados para su uso exclusivo AWS IoT. Puede suscribirse y publicar en estos temas reservados según lo permitan; sin embargo, no puede crear nuevos temas que comiencen por un signo de dólar. Las operaciones de publicación o suscripción no admitidas a temas reservados pueden dar como resultado que se termine la conexión.

Temas del modelo de activos

Topic	Operaciones de cliente permitidas	Description (Descripción)
$ aws/sitewise/asset *assetModelId* -models/ / assets/ /properties/ *assetId* *propertyId*	Suscribirse	AWS IoT SiteWise publica notificaciones de propiedades de activos sobre este tema. Para obtener más información, consulte Interacción con otros AWS servicios en la Guía del AWS IoT SiteWise usuario.

AWS IoT Device Defender temas

Estos mensajes admiten búferes de respuesta en formato de representación concisa de objetos binarios (CBOR) y en notación de JavaScript objetos (JSON), según el *payload-format* tema. AWS IoT Device Defender los temas solo admiten la publicación en MQTT.

payload-format	Tipo de datos de formato de respuesta
cbor	Concise Binary Object Representation (Representación concisa de objetos binarios, CBOR)
json	JavaScript Notación de objetos (JSON)

Para obtener más información, consulte Envío de métricas desde dispositivos.

Topic	Operaciones permitidas	Description (Descripción)
$aws/things/ /defender /metrics/ *thingName* *payload-format*	Publish	AWS IoT Device Defender los agentes publican métricas sobre este tema. Para obtener más información, consulte Envío de métricas desde dispositivos.

Topic	Operaciones permitidas	Description (Descripción)
$aws/things/ /defender /metrics/ /accepted *thingName payload-format*	Suscribirse	AWS IoT publica en este tema después de que un agente publique correctamente un mensaje en $aws/things/ /defender /metrics/. AWS IoT Device Defender *thingName payload-format* Para obtener más información, consulte Envío de métricas desde dispositivos.
$aws/things/ /defender /metrics/ /rejected *thingName payload-format*	Suscribirse	AWS IoT publica en este tema después de que un agente publique un mensaje fallido en $aws/things/ /defender/metrics/ . AWS IoT Device Defender *thingName payload-format* Para obtener más información, consulte Envío de métricas desde dispositivos.

AWS IoT Core Temas de ubicación de dispositivos

AWS IoT Core La ubicación del dispositivo puede resolver los datos de medición de su dispositivo y proporcionar una ubicación estimada de sus dispositivos de IoT. Los datos de medición del dispositivo pueden incluir el GNSS, la conexión Wi-Fi, la red móvil y la dirección IP. AWS IoT Core A continuación, la ubicación del dispositivo elige el tipo de medición que proporciona la mejor precisión y resuelve la información de ubicación del dispositivo. Para obtener más información, consulte AWS IoT Core Ubicación del dispositivo y Resolución de la ubicación del dispositivo mediante los temas de ubicación AWS IoT Core del dispositivo (MQTT)..

Topic	Operaciones permitidas	Description (Descripción)
$aws/device_location/ / get_position_estimate *customer_device_id*	Publish	Un dispositivo publica sobre este tema para que los datos de medición sin procesar escaneados se resuelvan según

Topic	Operaciones permitidas	Description (Descripción)
		la ubicación del dispositivo. AWS IoT Core
$aws/device_location/ / get_position_estimate/ accepted *customer_device_id*	Suscribirse	AWS IoT Core Device Location publica este tema después de haber resuelto correctamente la ubicación del dispositivo.
$aws/device_location/ / get_position_estimate/ rejected *customer_device_id*	Suscribirse	AWS IoT Core La ubicación del dispositivo publica este tema cuando no puede resolver la ubicación del dispositivo correctamente debido a errores 4xx.

Temas de eventos

Los mensajes de eventos se publican cuando se producen determinados eventos. Por ejemplo, el registro genera eventos cuando se agregan, actualizan o eliminan objetos. En la tabla se muestran los distintos AWS IoT eventos y sus temas reservados.

Topic	Operaciones de cliente permitidas	Description (Descripción)
$aws/events/certificates/ registered/*caCertificateId*	Suscribirse	AWS IoT publica este mensaje cuando registra AWS IoT automáticamente un certificado y cuando un cliente presenta un certificado con el `PENDING_ACTIVATION` estado. Para obtener más información, consulte the section called “Configurar la primera conexión de un cliente para el registro automático”.
$aws/events/job/*jobID*/ cancelado	Suscribirse	AWS IoT publica este mensaje cuando se cancela un trabajo. Para obtener

Topic	Operaciones de cliente permitidas	Description (Descripción)
		más información, consulte Eventos de trabajos.
$aws/events/job/*jobID*/cancellation_in_progress	Suscribirse	AWS IoT publica este mensaje cuando hay una cancelación de trabajo en curso. Para obtener más información, consulte Eventos de trabajos.
$aws/events/job/*jobID*/completado	Suscribirse	AWS IoT publica este mensaje cuando se ha completado un trabajo. Para obtener más información, consulte Eventos de trabajos.
$aws/events/job/*jobID*/eliminado	Suscribirse	AWS IoT publica este mensaje cuando se elimina un trabajo. Para obtener más información, consulte Eventos de trabajos.
$aws/events/job/*jobID*/deletion_in_progress	Suscribirse	AWS IoT publica este mensaje cuando se está eliminando un trabajo. Para obtener más información, consulte Eventos de trabajos.
$aws/events/jobExecution/*jobID*/cancelado	Suscribirse	AWS IoT publica este mensaje cuando se cancela la ejecución de un trabajo. Para obtener más información, consulte Eventos de trabajos.
$aws/events/jobExecution/*jobID*/eliminado	Suscribirse	AWS IoT publica este mensaje cuando se elimina la ejecución de un trabajo. Para obtener más información, consulte Eventos de trabajos.

Topic	Operaciones de cliente permitidas	Description (Descripción)
$aws/events/jobExecution/*jobID*/failed	Suscribirse	AWS IoT publica este mensaje cuando se produce un error en la ejecución de un trabajo. Para obtener más información, consulte Eventos de trabajos.
$aws/events/jobExecution/*jobID*/rejected	Suscribirse	AWS IoT publica este mensaje cuando se rechaza la ejecución de un trabajo. Para obtener más información, consulte Eventos de trabajos.
$aws/events/jobExecution/*jobID*/removed	Suscribirse	AWS IoT publica este mensaje cuando se elimina una ejecución de trabajo. Para obtener más información, consulte Eventos de trabajos.
$aws/events/jobExecution/*jobID*/succeeded	Suscribirse	AWS IoT publica este mensaje cuando la ejecución de un trabajo se ha realizado correctamente. Para obtener más información, consulte Eventos de trabajos.
$aws/events/jobExecution/*jobID*/timed_out	Suscribirse	AWS IoT publica este mensaje cuando se agota el tiempo de espera de una ejecución de un trabajo. Para obtener más información, consulte Eventos de trabajos.
$aws/events/presence/connected/*clientId*	Suscribirse	AWS IoT publica este tema cuando se conecta un cliente MQTT con el ID de cliente especificado. AWS IoT Para obtener más información, consulte Eventos de conexión/desconexión.

Topic	Operaciones de cliente permitidas	Description (Descripción)
$aws/events/presence/disconnected/*clientId*	Suscribirse	AWS IoT publica este tema cuando un cliente MQTT con el ID de cliente especificado se desconecta a. AWS IoT Para obtener más información, consulte Eventos de conexión/desconexión.
$aws/events/subscriptions/subscribed/*clientId*	Suscribirse	AWS IoT publica en este tema cuando un cliente de MQTT con el ID de cliente especificado se suscribe a un tema de MQTT. Para obtener más información, consulte Eventos de suscripción/cancelación de suscripción.
$aws/events/subscriptions/unsubscribed/*clientId*	Suscribirse	AWS IoT publica en este tema cuando un cliente de MQTT con el ID de cliente especificado cancela su suscripción a un tema de MQTT. Para obtener más información, consulte Eventos de suscripción/cancelación de suscripción.
$//created aws/events/thing *thingName*	Suscribirse	AWS IoT publica en este tema cuando se crea la *thingName* cosa. Para obtener más información, consulte the section called “Eventos de registro”.
$aws/events/thing/ *thingName* /updated	Suscribirse	AWS IoT publica en este tema cuando *thingName* se actualiza. Para obtener más información, consulte the section called “Eventos de registro”.
$aws/events/thing/ *thingName* /eliminado	Suscribirse	AWS IoT publica en este tema cuando se elimina la *thingName* cosa. Para obtener más información, consulte the section called “Eventos de registro”.

Topic	Operaciones de cliente permitidas	Description (Descripción)
$aws/events/thingGroup/*thingGroupName*/created	Suscribirse	AWS IoT publica en este tema cuando *thingGroupName* se crea el grupo de cosas. Para obtener más información, consulte the section called “Eventos de registro”.
$aws/events/thingGroup/*thingGroupName*/updated	Suscribirse	AWS IoT publica en este tema cuando se *thingGroupName* actualiza el grupo de cosas. Para obtener más información, consulte the section called “Eventos de registro”.
$aws/events/thingGroup/*thingGroupName*/eliminado	Suscribirse	AWS IoT publica en este tema cuando *thingGroupName* se elimina un grupo de cosas. Para obtener más información, consulte the section called “Eventos de registro”.
$aws/events/thingType/*thingTypeName*/created	Suscribirse	AWS IoT publica en este tema cuando se crea el tipo de *thingTypeName* cosa. Para obtener más información, consulte the section called “Eventos de registro”.
$aws/events/thingType/*thingTypeName*/updated	Suscribirse	AWS IoT publica en este tema cuando se actualiza el tipo de *thingTypeName* cosa. Para obtener más información, consulte the section called “Eventos de registro”.
$aws/events/thingType/*thingTypeName*/eliminado	Suscribirse	AWS IoT publica en este tema cuando se elimina el tipo de *thingTypeName* cosa. Para obtener más información, consulte the section called “Eventos de registro”.

Topic	Operaciones de cliente permitidas	Description (Descripción)
$aws/events/thingTypeAssociation/thing/*thingName*/*thingTypeName*	Suscribirse	AWS IoT publica en este tema cuando algo *thingName* está asociado o disociado de un tipo *thingTypeName* de cosa. Para obtener más información, consulte the section called “Eventos de registro”.
$aws/events/thingGroupMembership/thingGroup//thing/*thingGroupName* /added *thingName*	Suscribirse	AWS IoT publica en este tema cuando *thingName* se agrega algo al grupo de cosas. *thingGroupName* Para obtener más información, consulte the section called “Eventos de registro”.
$aws/events/thingGroupMembership/thingGroup/*thingGroupName* /thing/ /removed *thingName*	Suscribirse	AWS IoT publica en este tema cuando *thingName* se elimina algo del grupo de cosas. *thingGroupName* Para obtener más información, consulte the section called “Eventos de registro”.
$aws/events/thingGroupHierarchy/thingGroup//*parentThingGroupName* childThingGroup/*childThingGroupName* /agregado	Suscribirse	AWS IoT publica en este tema cuando *childThingGroupName* se agrega un grupo de cosas al grupo *parentThingGroupName* de cosas. Para obtener más información, consulte the section called “Eventos de registro”.
$aws/events/thingGroupHierarchy/thingGroup/*parentThingGroupName* childThingGroup/*childThingGroupName* /eliminado	Suscribirse	AWS IoT publica en este tema cuando *childThingGroupName* se elimina un grupo de cosas del grupo *parentThingGroupName* de cosas. Para obtener más información, consulte the section called “Eventos de registro”.

Temas de aprovisionamiento de flotas

> **Note**
>
> Las operaciones de cliente indicadas como Recibir en esta tabla indican los temas que se AWS IoT publican directamente para el cliente que los solicitó, independientemente de que el cliente esté suscrito al tema o no. Los clientes deben esperar recibir estos mensajes de respuesta aunque no estén suscritos a ellos. Estos mensajes de respuesta no pasan por el agente de mensajes y otros clientes o reglas no pueden suscribirse a ellos.

Estos mensajes admiten búferes de respuesta en formato de representación concisa de objetos binarios (CBOR) y en notación *payload-format* de JavaScript objetos (JSON), según el tema.

payload-format	Tipo de datos de formato de respuesta
cbor	Concise Binary Object Representation (Representación concisa de objetos binarios, CBOR)
json	JavaScript Notación de objetos (JSON)

Para obtener más información, consulte API de MQTT de aprovisionamiento de dispositivos.

Topic	Operaciones de cliente permitidas	Description (Descripción)
$aws/certificates/create/*payload-format*	Publish	Publique en este tema para crear un certificado a partir de una solicitud de firma de certificado (CSR).
$aws/certificates/create/*payload-format* / aceptado	Suscribirse, recibir	AWS IoT publica en este tema tras una llamada exitosa a $aws/certificates/create/*payload-format* .

Topic	Operaciones de cliente permitidas	Description (Descripción)
$aws/certificates/create/*payload-format* / rechazado	Suscribirse, recibir	AWS IoT publica en este tema después de una llamada fallida a $aws/certificates/create/*payload-format* .
$ aws/certificates/create-from-csr/ *payload-format*	Publish	Publica en este tema para crear un certificado a partir de una CSR.
$ aws/certificates/create-from-csr/ /aceptado *payload-format*	Suscribirse, recibir	AWS IoT publica en este tema una llamada exitosa a $ -from-csr/. aws/certificates/create *payload-format*
$ -from-csr/ /rejected aws/certificates/create *payload-format*	Suscribirse, recibir	AWS IoT publica en este tema una llamada fallida a $ -from-csr/. aws/certificates/create *payload-format*
templateName $aws/provisioning-templates/ /provision/ *payload-format*	Publish	Publique en este tema para registrar un objeto.
$aws/provisioning-templates/ /provision/ *templateName* /accepted *payload-format*	Suscribirse, recibir	AWS IoT publica en este tema tras una llamada exitosa a $aws/provisioning-templates/ /provision/. *templateName* *payload-format*
$aws/provisioning-templates/ /provision/ / rejected *templateName* *payload-format*	Suscribirse, recibir	AWS IoT publica en este tema tras una llamada fallida a $aws/provisioning-templates/ /provision/. *templateName* *payload-format*

Temas de trabajos

> **Note**
>
> Las operaciones de cliente indicadas como Recibir en esta tabla indican los temas que se AWS IoT publican directamente para el cliente que los solicitó, independientemente de que el cliente esté suscrito al tema o no. Los clientes deben esperar recibir estos mensajes de respuesta aunque no estén suscritos a ellos.
> Estos mensajes de respuesta no pasan por el agente de mensajes y otros clientes o reglas no pueden suscribirse a ellos. Para suscribirse a los mensajes relacionados con la actividad de trabajos, utilice los temas `notify` y `notify-next`.
> Al suscribirse a los temas de tareas y eventos `jobExecution` de su solución de monitoreo de flota, primero debe habilitar los eventos de tareas y ejecución de tareas para recibir cualquier evento en la nube.
> Para obtener más información, consulte Operaciones de la API MQTT de dispositivos de Jobs.

Topic	Operaciones de cliente permitidas	Description (Descripción)
$aws/things/ /jobs/get *thingName*	Publish	Los dispositivos publican un mensaje en este tema para realizar una solicitud `GetPendingJobExecutions` . Para obtener más información, consulte Operaciones de la API MQTT de dispositivos de Jobs.
thingName $aws/cosas// jobs/get/accepted	Suscribirse, recibir	Los dispositivos se suscriben a este tema para recibir respuestas correctas de una solicitud `GetPendingJobExecutions` . Para obtener más información, consulte Operaciones de la API MQTT de dispositivos de Jobs.
thingName $aws/cosas// jobs/get/rejected	Suscribirse, recibir	Los dispositivos se suscriben a este tema para recibir una respuesta cuando

Topic	Operaciones de cliente permitidas	Description (Descripción)
		se rechaza una solicitud `GetPendingJobExecutions`. Para obtener más información, consulte Operaciones de la API MQTT de dispositivos de Jobs.
$aws/things/ /jobs/start-next *thingName*	Publish	Los dispositivos publican un mensaje en este tema para realizar una solicitud `StartNextPendingJobExecution`. Para obtener más información, consulte Operaciones de la API MQTT de dispositivos de Jobs.
thingName $aws/cosas// jobs/start-next/accepted	Suscribirse, recibir	Los dispositivos se suscriben a este tema para recibir respuestas correctas a una solicitud `StartNextPendingJobExecution`. Para obtener más información, consulte Operaciones de la API MQTT de dispositivos de Jobs.
thingName $aws/cosas// jobs/start-next/rejected	Suscribirse, recibir	Los dispositivos se suscriben a este tema para recibir una respuesta cuando se rechaza una solicitud `StartNextPendingJobExecution`. Para obtener más información, consulte Operaciones de la API MQTT de dispositivos de Jobs.
$aws/cosas/ /jobs/ /get *thingName* *jobId*	Publish	Los dispositivos publican un mensaje en este tema para realizar una solicitud `DescribeJobExecution`. Para obtener más información, consulte Operaciones de la API MQTT de dispositivos de Jobs.

Topic	Operaciones de cliente permitidas	Description (Descripción)
$aws/things/ /jobs/ *thingName* /get/accepted *jobId*	Suscribirse, recibir	Los dispositivos se suscriben a este tema para recibir respuestas correctas a una solicitud `DescribeJobExecution` . Para obtener más información, consulte Operaciones de la API MQTT de dispositivos de Jobs.
$aws/things/ /jobs/ *thingName* /get/rejected *jobId*	Suscribirse, recibir	Los dispositivos se suscriben a este tema para recibir una respuesta cuando se rechaza una solicitud `DescribeJobExecution` . Para obtener más información, consulte Operaciones de la API MQTT de dispositivos de Jobs.
$aws/things/ /jobs/ *thingName* /update *jobId*	Publish	Los dispositivos publican un mensaje en este tema para realizar una solicitud `UpdateJobExecution` . Para obtener más información, consulte Operaciones de la API MQTT de dispositivos de Jobs.
$aws/things/ /jobs/ *thingName* /update/accepted *jobId*	Suscribirse, recibir	Los dispositivos se suscriben a este tema para recibir respuestas correctas a una solicitud `UpdateJobExecution` . Para obtener más información, consulte Operaciones de la API MQTT de dispositivos de Jobs. **Nota** Solo el dispositivo que publica en $aws/things/ /jobs/ /update recibe mensajes sobre este tema. *thingName* *jobId*

Topic	Operaciones de cliente permitidas	Description (Descripción)
$aws/things/ /jobs/ /update/ rejected *thingName* *jobId*	Suscribirse, recibir	Los dispositivos se suscriben a este tema para recibir una respuesta cuando se rechaza una solicitud `UpdateJobExecution`. Para obtener más información, consulte Operaciones de la API MQTT de dispositivos de Jobs. **Nota** Solo el dispositivo que publica en $aws/things/ /jobs/ /update recibe mensajes sobre este tema. *thingName* *jobId*
$aws/things/ /jobs/notify *thingName*	Suscribirse, recibir	Los dispositivos se suscriben a este tema para recibir notificaciones cuando la ejecución de un trabajo se añade o elimina de la lista de ejecuciones pendientes de un objeto. Para obtener más información, consulte Operaciones de la API MQTT de dispositivos de Jobs.
$aws/things/ /jobs/notify-next *thingName*	Suscribirse, recibir	Los dispositivos se suscriben a este tema para recibir notificaciones cuando cambia la siguiente ejecución de un trabajo pendiente para el objeto. Para obtener más información, consulte Operaciones de la API MQTT de dispositivos de Jobs.
$/aws/events/job/completado *jobId*	Suscribirse	El servicio de trabajos publica un evento sobre este tema cuando se completa un trabajo. Para obtener más información, consulte Eventos de trabajos.

Topic	Operaciones de cliente permitidas	Description (Descripción)
$aws/events/job//cancelado *jobId*	Suscribirse	El servicio de trabajos publica un evento sobre este tema cuando se cancela un trabajo. Para obtener más información, consulte Eventos de trabajos.
$aws/events/job//eliminado *jobId*	Suscribirse	El servicio de trabajos publica un evento sobre este tema cuando se elimina un trabajo. Para obtener más información, consulte Eventos de trabajos.
$aws/events/job/*jobId*/ cancelación_en curso	Suscribirse	El servicio de trabajos publica un evento sobre este tema cuando se inicia la cancelación de un trabajo. Para obtener más información, consulte Eventos de trabajos.
$aws/events/job/*jobId*/ eliminación_en curso	Suscribirse	El servicio de trabajos publica un evento sobre este tema cuando se inicia la eliminación de un trabajo. Para obtener más información, consulte Eventos de trabajos.
$ aws/events/jobExec ution*jobId*//conseguido	Suscribirse	El servicio de trabajos publica un evento sobre este tema cuando la ejecución del trabajo se ejecuta satisfactoriamente. Para obtener más información, consulte Eventos de trabajos.
$aws/events/jobExecution// falló *jobId*	Suscribirse	El servicio de trabajos publica un evento sobre este tema cuando la ejecución de un trabajo produce un error. Para obtener más información, consulte Eventos de trabajos.

Topic	Operaciones de cliente permitidas	Description (Descripción)
$aws/events/jobExecution//rechazado *jobId*	Suscribirse	El servicio de trabajos publica un evento sobre este tema cuando se rechaza una ejecución de un trabajo. Para obtener más información, consulte Eventos de trabajos.
$aws/events/jobExecution//cancelado *jobId*	Suscribirse	El servicio de trabajos publica un evento sobre este tema cuando se cancela la ejecución de un trabajo. Para obtener más información, consulte Eventos de trabajos.
$aws/events/jobExecution//timed_out *jobId*	Suscribirse	El servicio de trabajos publica un evento sobre este tema cuando se agota el tiempo de espera de ejecución de un trabajo. Para obtener más información, consulte Eventos de trabajos.
$ aws/events/jobExecution*jobId*//eliminado	Suscribirse	El servicio de trabajos publica un evento sobre este tema cuando se elimina la ejecución de un trabajo. Para obtener más información, consulte Eventos de trabajos.
$aws/events/jobExecution//eliminado *jobId*	Suscribirse	El servicio de trabajos publica un evento sobre este tema cuando se elimina la ejecución de un trabajo. Para obtener más información, consulte Eventos de trabajos.

Temas de comandos

> **Note**
>
> Las operaciones de cliente indicadas como Recibir en esta tabla indican los temas que se AWS IoT publican directamente para el cliente que los solicitó, independientemente de que el cliente esté suscrito al tema o no. Los clientes deben esperar recibir estos mensajes de respuesta aunque no estén suscritos a ellos.
> Estos mensajes de respuesta no pasan por el agente de mensajes y otros clientes o reglas no pueden suscribirse a ellos.

Topic	Operaciones de cliente permitidas	Description (Descripción)
$aws/commands/// executions/ /request/ *<devices>* *<DeviceID>* *<ExecutionId>* *<PayloadFormat>* $aws/commands//*<devices>* *<DeviceID>* /executions/ /request *<ExecutionId>*	Suscribirse, recibir	Los dispositivos reciben un mensaje sobre este tema cuando se solicita iniciar la ejecución de un comando desde la consola o mediante la API `StartCommandExecution` . En este caso, *<devices>* pueden ser cosas de IoT o clientes MQTT, y *<DeviceID>* pueden ser el nombre de la cosa IoT o el ID de cliente MQTT.
$aws/commands/ // executions/ /response/ *<devices>* *<DeviceID>* *<ExecutionId>* *<PayloadFormat>*	Publish	Los dispositivos utilizan la API de MQTT `UpdateCommandExecution` para publicar un mensaje en este tema sobre la ejecución de comandos. El mensaje se publica como respuesta a la solicitud de iniciar la ejecución de un comando desde la consola o mediante la API `StartCommandExecution` . El mensaje publicado utilizará JSON o CBOR como. *<PayloadFormat>*

Topic	Operaciones de cliente permitidas	Description (Descripción)
$aws/commands///executions/ /response/accepted/ *<devices> <DeviceID> <ExecutionId> <PayloadFormat>* $aws/commands//*<devices> <DeviceID>* /executions/ /response/accepted *<ExecutionId>*	Suscribirse, recibir	Si el servicio en la nube procesó correctamente el resultado de la ejecución del comando, AWS IoT Device Management publicará una respuesta en el tema /accepted.
$aws/commands//*<devices> <DeviceID>* /executions/ *<ExecutionId>* /response/rechected/ *<PayloadFormat>* $aws/commands//*<devices> <DeviceID>* /executions/ /response/rejected *<ExecutionId>*	Publish	Si el servicio en la nube no pudo procesar el resultado de la ejecución del comando, AWS IoT Device Management publicará una respuesta en el tema /rejected.

Temas de reglas

Topic	Operaciones de cliente permitidas	Description (Descripción)
$aws/rules/ *ruleName*	Publish	Los dispositivos o las aplicaciones publican en este tema para activar reglas directamente. Para obtener más

Topic	Operaciones de cliente permitidas	Description (Descripción)
		información, consulte Reducción de los costes de mensajería con Basic Ingest.

Temas de tunelización segura

Topic	Operaciones de cliente permitidas	Description (Descripción)
$aws/things/ /tunnels/notify *thing-name*	Suscribirse	AWS IoT publica este mensaje para que un agente de IoT inicie un proxy local en el dispositivo remoto. Para obtener más información, consulte the section called "Fragmento de agente de IoT".

Temas de sombra

Las sombras con nombre y sin nombre utilizan los temas de esta sección. Los temas utilizados por cada uno solo difieren en el prefijo del tema. Esta tabla muestra el prefijo de tema utilizado por cada tipo de sombra.

Valor de ***ShadowTopicPrefix***	Tipo de sombra
$aws/things/ /shadow *thingName*	Sombra sin nombre (clásica)
thingName $aws/cosas/ /shadow/name/ *shadowName*	Sombra con nombre

Para crear un tema completo, seleccione el tipo de sombra al que desee hacer referencia, sustitúyala y, si procede, *ShadowTopicPrefix* por sus valores correspondientes *thingName* y *shadowName* añádala al código auxiliar del tema, tal y como se muestra en la siguiente tabla. Recuerde que los temas distinguen entre mayúsculas y minúsculas.

Topic	Operaciones de cliente permitidas	Description (Descripción)
ShadowTopicPrefix / eliminar	Publicar/suscribirse	Un dispositivo o una aplicación publica en este tema para eliminar una sombra. Para obtener más información, consulte /delete.
ShadowTopicPrefix / eliminar/aceptado	Suscribirse	El servicio Device Shadow envía mensajes a este tema cuando se elimina una sombra. Para obtener más informaci ón, consulte /delete/accepted.
ShadowTopicPrefix / eliminar/rechazado	Suscribirse	El servicio Device Shadow envía mensajes a este tema cuando se rechaza una solicitud para eliminar una sombra. Para obtener más información, consulte /delete/rejected.
ShadowTopicPrefix /get	Publicar/suscribirse	Una aplicación o un objeto publica un mensaje vacío en este tema para obtener una sombra. Para obtener más información, consulte Temas MQTT de sombra de dispositivo.
ShadowTopicPrefix / get/accepted	Suscribirse	El servicio Device Shadow envía mensajes a este tema cuando se realiza correctamente una solicitud de una sombra. Para obtener más información, consulte /get/accepted.
ShadowTopicPrefix / get/rejected	Suscribirse	El servicio Device Shadow envía mensajes a este tema cuando se rechaza una solicitud de una sombra. Para obtener más información, consulte /get/rejected.

Topic	Operaciones de cliente permitidas	Description (Descripción)
ShadowTopicPrefix / actualizar	Publicar/suscribirse	Un objeto o una aplicación publica en este tema para actualizar una sombra. Para obtener más información, consulte /update.
ShadowTopicPrefix / update/aceptado	Suscribirse	El servicio Device Shadow envía mensajes a este tema cuando se realiza correctamente una actualización en una sombra. Para obtener más información, consulte /update/accepted.
ShadowTopicPrefix / update/rechazado	Suscribirse	El servicio Device Shadow envía mensajes a este tema cuando se rechaza una actualización en una sombra. Para obtener más información, consulte /update/rejected.
ShadowTopicPrefix / update/delta	Suscribirse	El servicio Device Shadow envía mensajes a este tema cuando se detecta una diferencia entre las secciones para los estados reported y desired de una sombra. Para obtener más información, consulte /update/delta.
ShadowTopicPrefix / update/documents	Suscribirse	AWS IoT publica un documento estatal sobre este tema cada vez que se realiza correctamente una actualización de la sombra. Para obtener más información, consulte /update/documents.

Temas relacionados con la entrega de archivos basada en MQTT

> **Note**
>
> Las operaciones de cliente indicadas como Recibir en esta tabla indican los temas que se AWS IoT publican directamente para el cliente que los solicitó, independientemente de que el cliente esté suscrito al tema o no. Los clientes deben esperar recibir estos mensajes de respuesta aunque no estén suscritos a ellos. Estos mensajes de respuesta no pasan por el agente de mensajes y otros clientes o reglas no pueden suscribirse a ellos.

Estos mensajes admiten búferes de respuesta en formato de representación concisa de objetos binarios (CBOR) y en notación *payload-format* de JavaScript objetos (JSON), según el tema.

payload-format	Tipo de datos de formato de respuesta
cbor	Concise Binary Object Representation (Representación concisa de objetos binarios, CBOR)
json	JavaScript Notación de objetos (JSON)

Topic	Operaciones de cliente permitidas	Description (Descripción)
$aws/things/ /streams/ / data/ *ThingName StreamId payload-format*	Suscribirse, recibir	AWS La entrega de archivos basada en MQTT se publica en este tema si se acepta la solicitud «» de un dispositivo. GetStream La carga útil contiene los datos de la transmisión. Para obtener más información, consulte Uso de la entrega de archivos AWS IoT basada en MQTT en los dispositivos.

Topic	Operaciones de cliente permitidas	Description (Descripción)
$aws/things/ /streams/ /get/ *ThingName StreamId payload-format*	Publish	Un dispositivo publica en este tema para realizar una solicitud «». GetStream Para obtener más información, consulte Uso de la entrega de archivos AWS IoT basada en MQTT en los dispositivos.
$aws/things/ /streams/ / description/ *ThingName StreamId payload-format*	Suscribirse, recibir	AWS La entrega de archivos basada en MQTT se publica en este tema si se acepta la solicitud «» de un dispositivo. DescribeStream La carga útil contiene la descripción del flujo. Para obtener más informaci ón, consulte Uso de la entrega de archivos AWS IoT basada en MQTT en los dispositivos.
$aws/things/ /streams/ / describe/ *ThingName StreamId payload-format*	Publish	Un dispositivo publica en este tema para realizar una solicitud «». DescribeStream Para obtener más información, consulte Uso de la entrega de archivos AWS IoT basada en MQTT en los dispositivos.
$aws/things/ /streams/ / rejected/ *ThingName StreamId payload-format*	Suscribirse, recibir	AWS La entrega de archivos basada en MQTT se publica en este tema si se rechaza una solicitud «» o «» de un dispositivo. DescribeS tream GetStream Para obtener más información, consulte Uso de la entrega de archivos AWS IoT basada en MQTT en los dispositivos.

Temas reservados

Todos los temas reservados ARNs (Amazon Resource Names) tienen el siguiente formulario:

```
arn:aws:iot:aws-region:AWS-account-ID:topic/Topic
```

Por ejemplo, `arn:aws:iot:us-west-2:123EXAMPLE456:topic/$aws/things/thingName/jobs/get/accepted` es un ARN del tema reservado `$aws/things/thingName/jobs/get/accepted`.

Configuraciones de dominio

En AWS IoT Core, puede usar las configuraciones de dominio para configurar y administrar los comportamientos de sus puntos de enlace de datos. Con las configuraciones de dominio, puede generar varios puntos de enlace de AWS IoT Core datos, personalizarlos con sus propios nombres de dominio completos (FQDN) y los certificados de servidor asociados, y también asociar un autorizador personalizado. Para obtener más información, consulte Autenticación y autorización personalizada.

> **Note**
>
> Esta función no está disponible en. AWS GovCloud (US) Regiones de AWS

Temas de este capítulo:

- ¿Qué es una configuración de dominio?
- Creación y configuración de dominios AWS gestionados
- Creación y configuración de dominios administrados por el cliente
- Administración de configuraciones de dominio
- Configuración de los ajustes de TLS en las configuraciones de dominio
- Configuración del certificado de servidor para la asociación de OCSP

¿Qué es una configuración de dominio?

En AWS IoT Core, una configuración de dominio hace referencia a la instalación y configuración de un dominio (ya sea un dominio AWS gestionado o un dominio gestionado por el cliente) para los puntos finales de AWS IoT Core datos. AWS IoT Core también proporciona un punto final

predeterminado para su AWS cuenta (`iot:Data-ATS`) para que los dispositivos se comuniquen con AWS IoT Core ellos.

En este tema:

- Casos de uso
- Conceptos clave
- Notas importantes

Casos de uso

Puede utilizar configuraciones de dominio para simplificar tareas de la siguiente manera.

- Migre los dispositivos a AWS IoT Core.
- Admitir flotas de dispositivos heterogéneas manteniendo configuraciones de dominio diferentes para cada tipo de dispositivo
- Mantenga la identidad de la marca (por ejemplo, mediante el nombre de dominio) al migrar la infraestructura de aplicaciones a AWS IoT Core.

Conceptos clave

Los siguientes conceptos proporcionan detalles sobre las configuraciones de dominio y los conceptos relacionados.

- Configuración del dominio

  La instalación y configuración de un dominio para sus puntos AWS IoT Core finales.

- Dominio de punto de conexión predeterminado

  El dominio que AWS IoT proporciona el punto final predeterminado, como`iot:Data-ATS`. Para encontrar el punto final predeterminado, ejecute el comando describe-endpoint o CLI describe-domain-configuration. También puede ir a la consola de AWS IoT Core y elegir Configuraciones de dominio en Conectarse en el menú de navegación de la izquierda. El punto de conexión predeterminado aparece con el nombre `iot:Data-ATS`.

- AWS dominio gestionado

El dominio que AWS se administrará. Si eliges un dominio AWS gestionado, tus dispositivos se conectarán mediante un punto de conexión de datos proporcionado por AWS. AWS gestionará el dominio y los certificados.

- Dominio administrado por el cliente

  Es el dominio que administrará usted. También recibe el nombre de dominio personalizado. Si elige un dominio administrado por el cliente, sus dispositivos se conectarán mediante un punto de conexión de datos de dominio personalizado. Usted administrará el dominio y los certificados. El dominio gestionado por el cliente le permite personalizar el punto final URLs para que se adapte a sus necesidades. Por ejemplo, puede usar un nombre de dominio personalizado (`your-domain-name.com`) o aplicar políticas de acceso específicas.

- Tipo de autenticación

  Es el tipo de autenticación que elige para autenticar sus dispositivos al conectarse a AWS IoT Core. Al crear una configuración de dominio, debe especificar un tipo de autenticación. Para obtener más información, consulte ???.

- Protocolo de aplicación

  Son los protocolos de la capa de aplicación que utilizan sus dispositivos al conectarse a AWS IoT Core. Cuando cree una configuración de dominio, deberá especificar un protocolo de aplicación. Para obtener más información, consulte ???.

Notas importantes

AWS IoT Core utiliza la extensión TLS de indicación de nombre de servidor (SNI) para aplicar las configuraciones de dominio. Al conectar los dispositivos a AWS IoT Core, los clientes pueden enviar la extensión de indicación del nombre del servidor (SNI), que es necesaria para funciones como el registro de varias cuentas, los puntos de enlace configurables, los dominios personalizados y los puntos de enlace de VPC. También deben pasar un nombre de servidor que sea idéntico al nombre de dominio que especifique en la configuración del dominio. Para probar este servicio, utilice la versión v2 del dispositivo en.AWS IoT SDKs GitHub

Si crea varios puntos de conexión de datos en el suyo Cuenta de AWS, estos compartirán AWS IoT Core recursos como temas de MQTT, sombras de dispositivos y reglas.

Al proporcionar los certificados de servidor para una configuración de dominio AWS IoT Core personalizada, los certificados tienen un máximo de cuatro nombres de dominio. Para obtener más información, consulte Puntos de conexión y cuotas de AWS IoT Core.

Creación y configuración de dominios AWS gestionados

Puede crear un punto final configurable en un dominio AWS gestionado mediante la CreateDomainConfigurationAPI. La configuración de un dominio AWS administrado consta de lo siguiente:

- `domainConfigurationName`

  Un nombre definido por el usuario que identifique la configuración del dominio y el valor debe ser único para usted Región de AWS. No puede utilizar nombres de configuración de dominio que empiecen por `IoT:` porque están reservados para puntos de conexión predeterminados.
- `defaultAuthorizerName` (opcional)

  El nombre del autorizador personalizado que se utilizará en el punto de conexión.
- `allowAuthorizerOverride` (opcional)

  Un valor booleano que especifica si los dispositivos pueden anular el autorizador predeterminado especificando un autorizador diferente en el encabezado HTTP de la solicitud. Este valor es obligatorio si se especifica un valor para `defaultAuthorizerName`.
- `serviceType` (opcional)

  El tipo de servicio que ofrece el punto final. AWS IoT Core solo admite el tipo `DATA` de servicio. Si especifica `DATA`, AWS IoT Core devuelve un punto de conexión del tipo `iot:Data-ATS`. No puede crear un punto final configurable `iot:Data` (VeriSign).
- `TlsConfig` (opcional)

  Un objeto que especifica el servicio de autorización de un dominio. Para obtener más información, consulte ???.

El siguiente AWS CLI comando de ejemplo crea una configuración de dominio para un `Data` punto final.

```
aws iot create-domain-configuration --domain-configuration-name
 "myDomainConfigurationName" --service-type "DATA"
```

El resultado del comando puede tener un aspecto similar al siguiente.

```
{
    "domainConfigurationName": "myDomainConfigurationName",
    "domainConfigurationArn": "arn:aws:iot:us-east-1:123456789012:domainconfiguration/
myDomainConfigurationName/itihw"
}
```

Creación y configuración de dominios administrados por el cliente

Las configuraciones de dominio permiten especificar un nombre de dominio completo (FQDN) personalizado para conectarse a AWS IoT Core. El uso de dominios gestionados por el cliente (también conocidos como dominios personalizados) tiene muchas ventajas: puede exponer su propio dominio o el dominio de su empresa a los clientes con fines de marca; puede cambiar fácilmente su propio dominio para apuntar a un nuevo corredor; puede admitir el arrendamiento múltiple para atender a clientes con diferentes dominios dentro del mismo Cuenta de AWS; y puede administrar los detalles de sus propios certificados de servidor, como la autoridad de certificación (CA) raíz utilizada para firmar el certificado, el algoritmo de firma, la profundidad de la cadena de certificados y el ciclo de vida del certificado.

El flujo de trabajo para establecer una configuración de dominio con un dominio personalizado consta de las tres etapas siguientes.

1. Registrar certificados de servidor en AWS Certificate Manager
2. Creación de una configuración de dominio
3. Creación de registros DNS

Registrar los certificados de servidor en el administrador de AWS certificados

Antes de crear una configuración de dominio con un dominio personalizado, debe registrar la cadena de certificados de servidor en AWS Certificate Manager (ACM). Puede utilizar los siguientes tres tipos de certificados de servidor.

- Certificados públicos generados por ACM
- Certificados externos firmados por una entidad emisora de certificación pública

- Certificados externos firmados por una entidad emisora de certificación privada

> **Note**
>
> AWS IoT Core considera que un certificado está firmado por una CA pública si está incluido en el paquete ca de confianza de Mozilla.

Requisitos de certificados

Consulte Requisitos previos para la importación de certificados para conocer los requisitos para importar certificados a ACM. Además de estos requisitos, AWS IoT Core agrega los siguientes.

- El certificado de hoja debe incluir la extensión x509 v3 de uso extendido de claves con un valor de serverAuth (autenticación de servidor web TLS). Si solicita el certificado a ACM, esta extensión se agrega automáticamente.
- La profundidad máxima de la cadena de certificados es de 5 certificados.
- El tamaño máximo de la cadena de certificados es de 16 KB.
- Los algoritmos criptográficos y los tamaños de las claves compatibles son el RSA de 2048 bits (RSA_2048) y el ECDSA de 256 bits (EC_prime256v1).

Uso de un único certificado para varios dominios

Si tiene previsto utilizar un certificado para cubrir varios subdominios, utilice un dominio comodín en el campo Nombre común (CN) o Nombres alternativos del firmante (SAN). Por ejemplo, utilice **`*.iot.example.com`** para cubrir dev.iot.example.com, qa.iot.example.com y prod.iot.example.com. Cada FQDN requiere su propia configuración de dominio, pero varias configuraciones de dominio pueden usar el mismo valor comodín. El CN o el SAN deben cubrir el FQDN que desea utilizar como dominio personalizado. Si SANs están presentes, se omite la CN y la SAN debe cubrir el FQDN que desee utilizar como dominio personalizado. Esta cobertura puede ser una coincidencia exacta o una coincidencia de caracteres comodín. Una vez validado y registrado un certificado comodín en una cuenta, se impide que otras cuentas de la región creen dominios personalizados que coincidan con el certificado.

En las siguientes secciones se describe cómo obtener cada tipo de certificado. Cada recurso de certificado requiere un Nombre de recurso de Amazon (ARN) registrado en ACM que usted utiliza cuando crea la configuración de su dominio.

Certificados públicos generados por ACM

Puede generar un certificado público para su dominio personalizado mediante la RequestCertificateAPI. Cuando genera un certificado de esta manera, ACM valida que usted es el propietario del dominio personalizado. Para obtener más información, consulte Solicitud de un certificado público en la Guía del usuario de AWS Certificate Manager .

Certificados externos firmados por una entidad emisora de certificación pública

Si ya tienes un certificado de servidor firmado por una entidad emisora de certificados pública (una entidad emisora de certificados incluida en el paquete de certificados de confianza de Mozilla), puedes importar la cadena de certificados directamente a ACM mediante la ImportCertificateAPI. Para obtener más información sobre esta tarea, los requisitos previos y los requisitos de formato de certificado, consulte Importación de certificados.

Certificados externos firmados por una entidad emisora de certificación privada

Si ya tiene un certificado de servidor firmado por una entidad emisora de certificación privada o autofirmado, puede utilizar el certificado para crear la configuración de dominio, pero también debe crear un certificado público adicional en ACM para validar que usted es el propietario del dominio. Para ello, registre la cadena de certificados de su servidor en ACM mediante la API. ImportCertificate Para obtener más información sobre esta tarea, los requisitos previos y los requisitos de formato de certificado, consulte Importación de certificados.

Creación de un certificado de validación

Tras importar el certificado a ACM, genere un certificado público para su dominio personalizado mediante la RequestCertificateAPI. Cuando genera un certificado de esta manera, ACM valida que usted es el propietario del dominio personalizado. Para obtener más información, consulte Solicitar un certificado público. Cuando cree la configuración de dominio, utilice este certificado público como certificado de validación.

Creación de una configuración de dominio

Puede crear un punto final configurable en un dominio personalizado mediante la CreateDomainConfigurationAPI. Una configuración de dominio para un dominio personalizado consta de lo siguiente:

- `domainConfigurationName`

El nombre definido por el usuario que identifica la configuración del dominio. Los nombres de configuración de dominio que comienzan por `IoT:` están reservados para los puntos de conexión predeterminados y no se pueden usar. Además, este valor debe ser exclusivo de su Región de AWS.

- `domainName`

  El FQDN al que utilizan sus dispositivos para conectarse AWS IoT Core. AWS IoT Core utiliza la extensión TLS de indicación de nombre de servidor (SNI) para aplicar las configuraciones de dominio. Los dispositivos deben usar esta extensión al conectarse y pasar un nombre de servidor idéntico al nombre de dominio especificado en la configuración del dominio.

- `serverCertificateArns`

  El ARN de la cadena de certificados de servidor que registró en ACM. AWS IoT Core actualmente solo admite un certificado de servidor.

- `validationCertificateArn`

  El ARN del certificado público que generó en ACM para validar la propiedad de su dominio personalizado. Este argumento no es necesario si utiliza un certificado de servidor firmado públicamente o uno generado por ACM.

- `defaultAuthorizerName (optional)`

  El nombre del autorizador personalizado que se utilizará en el punto de conexión.

- `allowAuthorizerOverride`

  Un valor booleano que especifica si los dispositivos pueden anular el autorizador predeterminado especificando un autorizador diferente en el encabezado HTTP de la solicitud. Este valor es obligatorio si se especifica un valor para `defaultAuthorizerName`.

- `serviceType`

  AWS IoT Core actualmente solo admite el tipo `DATA` de servicio. Si lo especifica`DATA`, AWS IoT devuelve un punto final con un tipo de punto final de`iot:Data-ATS`.

- `TlsConfig` (opcional)

  Un objeto que especifica el servicio de autorización de un dominio. Para obtener más información, consulte ???.

- `serverCertificateConfig` (opcional)

Es un objeto que especifica la configuración del certificado de servidor para un dominio. Para obtener más información, consulte ???.

El siguiente AWS CLI comando crea una configuración de dominio para iot.example.com.

```
aws iot create-domain-configuration --domain-configuration-name
 "myDomainConfigurationName" --service-type "DATA"
--domain-name "iot.example.com" --server-certificate-arns serverCertARN --validation-
certificate-arn validationCertArn
```

> **Note**
>
> Tras crear la configuración de dominio, es posible que pasen hasta 60 minutos hasta que se entreguen los certificados de AWS IoT Core servidor personalizados.

Para obtener más información, consulte ???.

Creación de registros DNS

Después de registrar la cadena de certificados de servidor y crear la configuración de dominio, cree un registro DNS para que el dominio personalizado apunte a un dominio de AWS IoT . Este registro debe apuntar a un AWS IoT punto final de este tipo`iot:Data-ATS`. Puede obtener su punto final mediante la DescribeEndpointAPI.

El siguiente AWS CLI comando muestra cómo obtener el punto final.

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

Una vez que obtengas tu `iot:Data-ATS` punto de conexión, crea un `CNAME` registro desde tu dominio personalizado hasta este AWS IoT punto de conexión. Si crea varios dominios personalizados en el mismo dominio Cuenta de AWS, asígneles un alias a este mismo `iot:Data-ATS` punto final.

Resolución de problemas

Si tiene problemas para conectar los dispositivos a un dominio personalizado, asegúrese de que AWS IoT Core haya aceptado y aplicado su certificado de servidor. Puede comprobar si AWS IoT Core ha aceptado su certificado mediante la AWS IoT Core consola o el AWS CLI.

Para usar la AWS IoT Core consola, vaya a la página de configuraciones del dominio y seleccione el nombre de la configuración del dominio. En la sección Detalles del certificado del servidor, compruebe el estado y los detalles del estado. Si el certificado no es válido, reemplácelo en ACM por un certificado que cumpla los requisitos de certificados enumerados en la sección anterior. Si el certificado tiene el mismo ARN, lo AWS IoT Core recogerá y lo aplicará automáticamente.

Para comprobar el estado del certificado mediante el AWS CLI, llame a la DescribeDomainConfigurationAPI y especifique el nombre de configuración de su dominio.

> **Note**
>
> Si su certificado no es válido, AWS IoT Core seguirá emitiendo el último certificado válido.

Puede comprobar qué certificado se está sirviendo en su punto de conexión mediante el siguiente comando openssl.

```
openssl s_client -connect custom-domain-name:8883 -showcerts -servername
custom-domain-name
```

Administración de configuraciones de dominio

En este tema se describen las operaciones clave para administrar los recursos de configuración de su dominio. También puede administrar los ciclos de vida de las configuraciones existentes mediante lo siguiente APIs: ListDomainConfigurations, DescribeDomainConfigurationUpdateDomainConfiguration, y. DeleteDomainConfiguration

En este tema:

- Visualización de configuraciones de dominio
- Actualización de configuraciones de dominio
- Eliminación de configuraciones de dominio
- Certificados de rotación en dominios personalizados

Visualización de configuraciones de dominio

Para obtener una lista paginada de todas las configuraciones de dominio que tiene Cuenta de AWS, utilice la API. ListDomainConfigurations Puedes ver los detalles de una configuración de

dominio concreta mediante la DescribeDomainConfigurationAPI. Esta API toma un único parámetro `domainConfigurationName` y devuelve los detalles de la configuración especificada.

Ejemplo

Actualización de configuraciones de dominio

Para actualizar el estado o el autorizador personalizado de la configuración de tu dominio, usa la UpdateDomainConfigurationAPI. Puede establecer el estado en `ENABLED` o `DISABLED`. Si deshabilita la configuración del dominio, los dispositivos conectados a ese dominio recibirán un error de autenticación. Actualmente no puede actualizar el certificado de servidor en la configuración de su dominio. Para cambiar el certificado de una configuración de dominio, debe eliminarlo y volver a crearlo.

Ejemplo

Eliminación de configuraciones de dominio

Antes de eliminar una configuración de dominio, usa la UpdateDomainConfigurationAPI para establecer el estado en. `DISABLED` Esto le ayuda a evitar que se elimine el punto de conexión por error. Tras inhabilitar la configuración del dominio, elimínala mediante la DeleteDomainConfigurationAPI. Debes mantener los dominios AWS gestionados en `DISABLED` estado durante 7 días antes de poder eliminarlos. Puede establecer el estado `DISABLED` para los dominios personalizados y, a continuación, eliminarlos inmediatamente.

Ejemplo

Tras eliminar una configuración de dominio, ya AWS IoT Core no se publica el certificado de servidor asociado a ese dominio personalizado.

Certificados de rotación en dominios personalizados

Es posible que deba reemplazar periódicamente el certificado de servidor por un certificado actualizado. La velocidad a la que lo haga dependerá del período de validez del certificado. Si ha generado el certificado de servidor mediante AWS Certificate Manager (ACM), puede configurar el certificado para que se renueve automáticamente. Cuando ACM renueva el certificado, recoge AWS IoT Core automáticamente el nuevo certificado. No tiene que realizar ninguna acción adicional. Si ha importado el certificado de servidor desde un origen diferente, puede rotarlo volviéndolo a importar a ACM. Para obtener información sobre la reimportación de certificados, consulte Volver a importar un certificado.

> **Note**
>
> AWS IoT Core solo recoge las actualizaciones de los certificados en las siguientes condiciones.
>
> - El nuevo certificado tiene el mismo ARN que el anterior.
> - El nuevo certificado tiene el mismo algoritmo de firma, nombre común o nombre alternativo del asunto que el anterior.

Configuración de los ajustes de TLS en las configuraciones de dominio

AWS IoT Core proporciona políticas de seguridad predefinidas para que pueda personalizar la configuración de seguridad de la capa de transporte (TLS) para TLS 1.2 y TLS 1.3 en las configuraciones de dominio. Una política de seguridad es una combinación de protocolos TLS y sus cifrados que determina los protocolos y cifrados que se admiten durante las negociaciones de TLS entre un cliente y un servidor. Con las políticas de seguridad compatibles, puede administrar la configuración de TLS de sus dispositivos con mayor flexibilidad, aplicar la mayoría de las medidas de up-to-date seguridad al conectar nuevos dispositivos y mantener una configuración de TLS uniforme para los dispositivos existentes.

La siguiente tabla describe las políticas de seguridad, sus versiones TLS y las regiones admitidas:

Nombre de política de seguridad	Compatible Regiones de AWS
TSecurityPolítica de Io_ _1_3_2022_10 TLS13	Todos Regiones de AWS
TSecurityPolítica de Io_ _1_2_2022_10 TLS13	Todos Regiones de AWS
TSecurityPolítica de Io_ _1_2_2022_10 TLS12	Todos Regiones de AWS
TSecurityPolítica de Io_ _1_0_2016_01 TLS12	ap-east-1, ap-northeast-2, ap-south-1, ap-southeast-2, ca-central-1, cn-north-1, cn-northwest-1, eu-north-1, eu-west-2, eu-west-3, me-south-1, sa-east-1, us-east-2, us-west-1

Nombre de política de seguridad	Compatible Regiones de AWS
TSecurityPolítica de Io_ TLS12 _1_0_2015_01	ap-northeast-1, ap-southeast-1, eu-central-1, eu-west-1, us-east-1, us-west-2

Los nombres de las políticas de seguridad AWS IoT Core incluyen información sobre la versión según el año y el mes en que se publicaron. Si crea una nueva configuración de dominio, la política de seguridad se establecerá de forma predeterminada en `IoTSecurityPolicy_TLS13_1_2_2022_10`. Para obtener una tabla completa de políticas de seguridad con detalles sobre los protocolos, los puertos TCP y los cifrados, consulte Políticas de seguridad. AWS IoT Core no admite políticas de seguridad personalizadas. Para obtener más información, consulte ???.

Para configurar los ajustes de TLS en las configuraciones de dominio, puede usar la AWS IoT consola o el AWS CLI.

Contenido

- Configuración de los ajustes de TLS en las configuraciones de dominio (consola)
- Configuración de los ajustes de TLS en las configuraciones de dominio (CLI)

Configuración de los ajustes de TLS en las configuraciones de dominio (consola)

Para configurar los ajustes de TLS mediante la consola AWS IoT

1. Inicie sesión en la AWS IoT consola Consola de administración de AWS y ábrala.
2. Para configurar los ajustes de TLS al crear una nueva configuración de dominio, siga estos pasos.
    1. En el panel de navegación izquierdo, selecciona Configuraciones de dominio y, a continuación, selecciona Crear configuración de dominio.
    2. En la página Crear configuración de dominio, en la sección Configuración de dominio personalizada: (opcional), elige una política de seguridad en Seleccionar política de seguridad.
    3. Siga el widget y complete el resto de los pasos. Seleccione Crear configuración de dominio.

3. Para actualizar la configuración de TLS en una configuración de dominio existente, siga estos pasos.

   1. En el panel de navegación izquierdo, elija Configuraciones de dominio y, a continuación, elija una configuración de dominio.
   2. En la página Detalles de configuración del dominio, elija Editar. A continuación, en la sección Configuración de dominio personalizada: (opcional), en Seleccionar política de seguridad, elija una política de seguridad.
   3. Seleccione Actualizar la configuración del dominio.

Para obtener más información, consulte Crear una configuración de dominio y Administrar configuraciones de dominio.

Configuración de los ajustes de TLS en las configuraciones de dominio (CLI)

Puede utilizar los comandos de la CLI create-domain-configuration y update-domain-configuration para configurar los ajustes de TLS en las configuraciones de dominio.

1. Para especificar la configuración de TLS mediante el comando de la CLI create-domain-configuration:

   ```
   aws iot create-domain-configuration \
       --domain-configuration-name domainConfigurationName \
       --tls-config securityPolicy=IoTSecurityPolicy_TLS13_1_2_2022_10
   ```

   El resultado de este comando puede tener un aspecto similar al siguiente.

   ```
   {
   "domainConfigurationName": "test",
   "domainConfigurationArn": "arn:aws:iot:us-west-2:123456789012:domainconfiguration/
   test/34ga9"
   }
   ```

   Si crea una nueva configuración de dominio sin especificar la política de seguridad, el valor predeterminado será: `IoTSecurityPolicy_TLS13_1_2_2022_10`.

2. Para describir la configuración de TLS mediante el comando de la CLI describe-domain-configuration:

```
aws iot describe-domain-configuration \
    --domain-configuration-name domainConfigurationName
```

Este comando puede devolver los detalles de configuración del dominio que incluyen los ajustes de TLS, como los siguientes:

```
{
 "tlsConfig": {
 "securityPolicy": "IoTSecurityPolicy_TLS13_1_2_2022_10"
 },
 "domainConfigurationStatus": "ENABLED",
 "serviceType": "DATA",
 "domainType": "AWS_MANAGED",
 "domainName": "d1234567890abcdefghij-ats.iot.us-west-2.amazonaws.com",
 "serverCertificates": [],
 "lastStatusChangeDate": 1678750928.997,
 "domainConfigurationName": "test",
 "domainConfigurationArn": "arn:aws:iot:us-west-2:123456789012:domainconfiguration/
test/34ga9"
}
```

3. Para actualizar la configuración de TLS mediante el comando de la CLI update-domain-configuration:

```
aws iot update-domain-configuration \
    --domain-configuration-name domainConfigurationName \
    --tls-config securityPolicy=IoTSecurityPolicy_TLS13_1_2_2022_10
```

El resultado de este comando puede tener un aspecto similar al siguiente.

```
{
"domainConfigurationName": "test",
"domainConfigurationArn": "arn:aws:iot:us-west-2:123456789012:domainconfiguration/
test/34ga9"
}
```

4. Para actualizar la configuración de TLS de su punto de conexión ATS, ejecute el comando de la CLI update-domain-configuration. El nombre de la configuración de dominio de su punto de conexión ATS es `iot:Data-ATS`.

```
aws iot update-domain-configuration \
    --domain-configuration-name "iot:Data-ATS" \
    --tls-config securityPolicy=IoTSecurityPolicy_TLS13_1_2_2022_10
```

El resultado del comando puede tener un aspecto similar al siguiente:

```
{
"domainConfigurationName": "iot:Data-ATS",
"domainConfigurationArn": "arn:aws:iot:us-west-2:123456789012:domainconfiguration/
iot:Data-ATS"
}
```

Para obtener más información consulte CreateDomainConfiguration y UpdateDomainConfiguration en la Referencia de la API de AWS .

Configuración del certificado de servidor para la asociación de OCSP

AWS IoT Core admite el grapado OCSP (Online Certificate Status Protocol) para certificados de servidor, también conocido como grapado OCSP de certificados de servidor o grapado OCSP. Es un mecanismo de seguridad que se utiliza para comprobar el estado de revocación del certificado de servidor mediante un protocolo de enlace de seguridad de la capa de transporte (TLS). El grapado OCSP AWS IoT Core le permite añadir un nivel adicional de verificación a la validez del certificado de servidor de su dominio personalizado.

Puede habilitar el grapado OCSP del certificado de servidor AWS IoT Core para comprobar la validez del certificado consultando periódicamente al respondedor OCSP. La configuración de la asociación de OCSP forma parte del proceso de creación o actualización de una configuración de dominio con un dominio personalizado. La asociación de OCSP comprueba continuamente el estado de revocación en el certificado del servidor. Esto ayuda a comprobar que los clientes que se conectan a sus dominios personalizados ya no confíen en los certificados que hayan sido revocados por la CA. Para obtener más información, consulte ???.

La asociación de OCSP de los certificados de servidor permite comprobar el estado de la revocación en tiempo real, reduce la latencia asociada a la comprobación del estado de la revocación y mejora la privacidad y la fiabilidad de las conexiones seguras. Para obtener más información sobre las ventajas de utilizar la asociación de OCSP, consulte ???.

> **Note**
>
> Esta función no está disponible en. AWS GovCloud (US) Regions

En este tema:

- ¿Qué es el OCSP?
- Funcionamiento de la asociación de OCSP
- Habilitar el certificado OCSP del servidor en AWS IoT Core
- Configurar el certificado de servidor OCSP para puntos finales privados en AWS IoT Core
- Notas importantes sobre el uso del certificado de servidor (OCSP): grapado AWS IoT Core
- Solución de problemas al grapar el certificado de servidor OCSP AWS IoT Core

¿Qué es el OCSP?

El Protocolo de estado de certificado en línea (OCSP, por sus siglas en inglés) ayuda a proporcionar el estado de revocación de un certificado de servidor para un protocolo de enlace de seguridad de la capa de transporte (TLS).

Conceptos clave

Los siguientes conceptos clave proporcionan detalles sobre el Protocolo de estado de certificado en línea (OCSP).

OCSP

El OCSP se utiliza para comprobar el estado de revocación del certificado durante el protocolo de enlace de seguridad de la capa de transporte (TLS). El OCSP permite validar los certificados en tiempo real. De este modo, se confirma que el certificado no se ha revocado ni ha caducado desde que se emitió. El OCSP también es más escalable en comparación con las listas de revocación de certificados tradicionales ()CRLs. Las respuestas de OCSP son más pequeñas y se pueden generar de manera eficiente, lo que las hace más adecuadas para infraestructuras de clave privada a gran escala (). PKIs

Respondedor de OCSP

Un respondedor de OCSP (también conocido como servidor de OCSP) recibe y responde a las solicitudes de OCSP de los clientes que quieren verificar el estado de revocación de los certificados.

OCSP del cliente

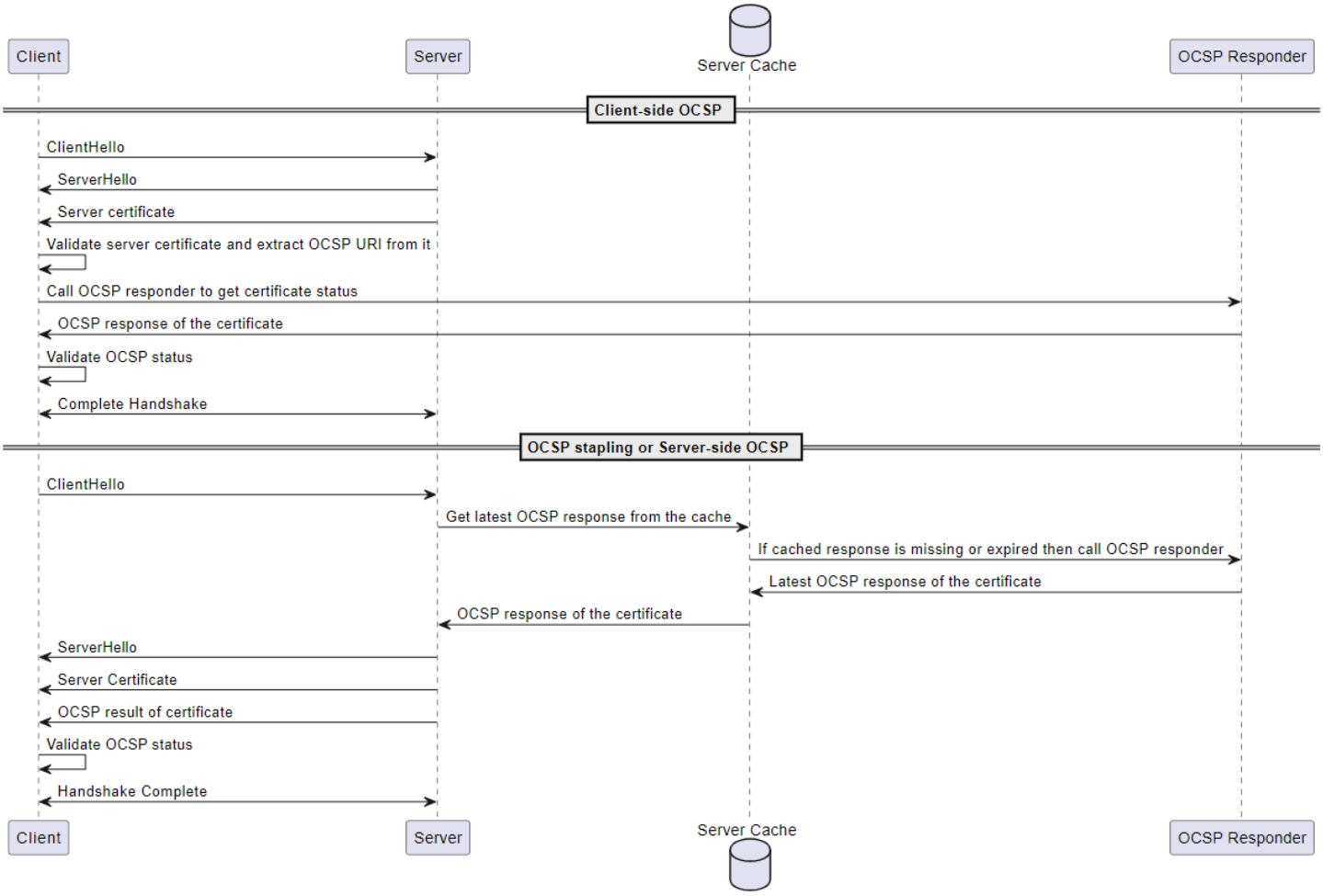
En el OCSP del cliente, este utiliza el OCSP para ponerse en contacto con un respondedor de OCSP y comprobar el estado de revocación del certificado durante el establecimiento de comunicación de TLS.

OCSP del servidor

En el OCSP del servidor (también conocido como asociación de OCSP), el servidor está activado (y no el cliente) para realizar la solicitud al respondedor de OCSP. El servidor asocia la respuesta de OCSP al certificado y la devuelve al cliente durante el protocolo de enlace TLS.

Diagramas del OSCP

En el siguiente diagrama se muestra cómo funcionan el OCSP del cliente y el OCSP del servidor.



OCSP del cliente

1. El cliente envía un mensaje `ClientHello` para iniciar el protocolo de enlace TLS con el servidor.
2. El servidor recibe el mensaje y responde con un mensaje `ServerHello`. El servidor también envía el certificado del servidor al cliente.
3. El cliente valida el certificado del servidor y extrae un URI de OCSP.
4. El cliente envía una solicitud de verificación de revocación del certificado al respondedor de OCSP.
5. El respondedor de OCSP envía una respuesta de OCSP.
6. El cliente valida el estado del certificado a partir de la respuesta de OCSP.
7. El protocolo de enlace TLS se completa.

OCSP del servidor

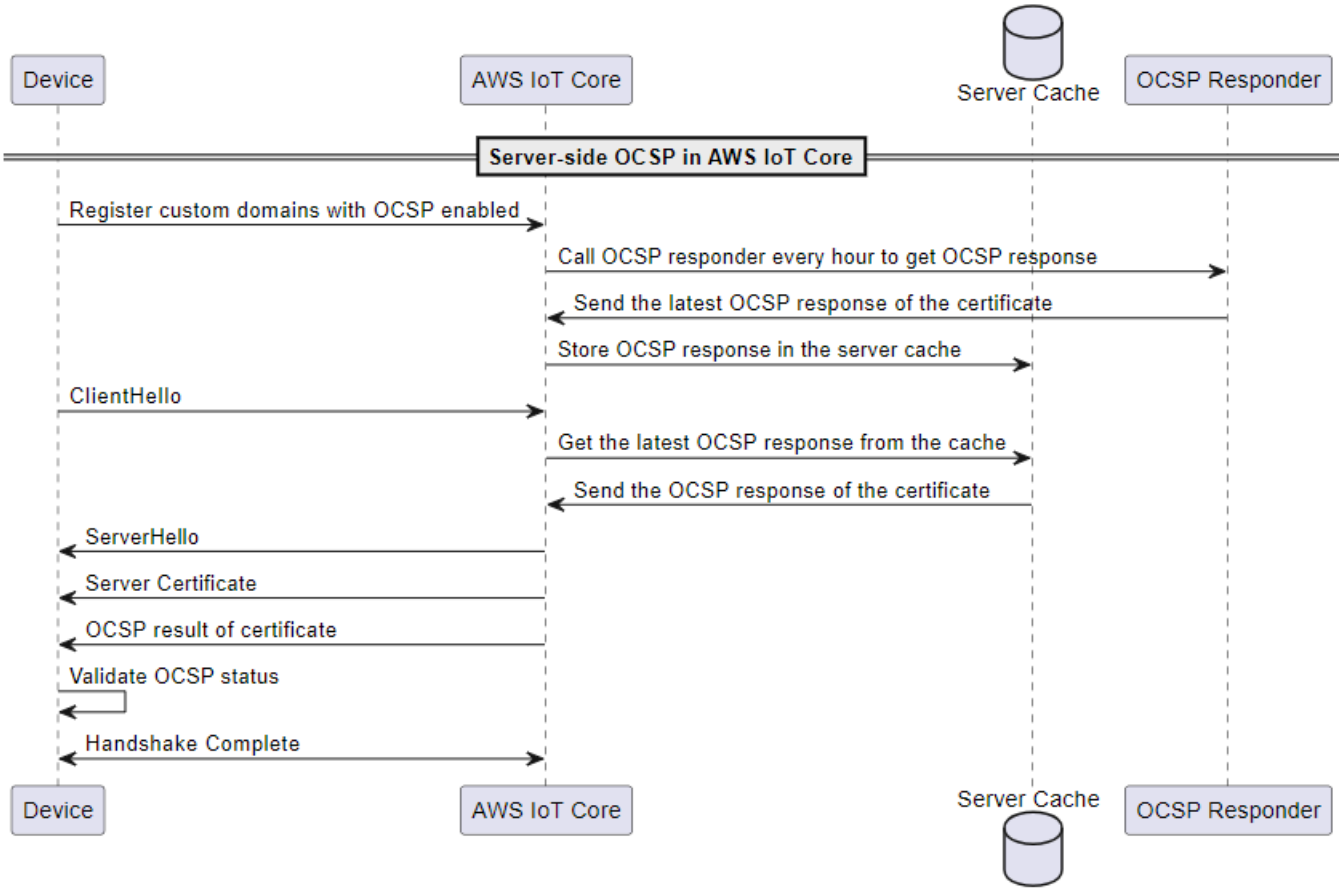
1. El cliente envía un mensaje `ClientHello` para iniciar el protocolo de enlace TLS con el servidor.
2. El servidor recibe el mensaje y obtiene la última respuesta de OCSP almacenada en caché. Si falta la respuesta en caché o ha caducado, el servidor llamará al respondedor de OCSP para obtener el estado del certificado.
3. El respondedor de OCSP envía una respuesta de OCSP al servidor.
4. El servidor envía un mensaje `ServerHello`. El servidor también envía el certificado de servidor y el estado del certificado al cliente.
5. El cliente valida el estado del certificado de OCSP.
6. El protocolo de enlace TLS se completa.

Funcionamiento de la asociación de OCSP

La asociación de OCSP se utiliza durante el establecimiento de comunicación de TLS entre el cliente y el servidor para comprobar el estado de revocación del certificado del servidor. El servidor realiza la solicitud de OCSP al respondedor de OCSP y asocia las respuestas del OCSP a los certificados devueltos al cliente. Al hacer que el servidor haga la solicitud al respondedor de OCSP, las respuestas se pueden almacenar en caché y, a continuación, se pueden utilizar varias veces para muchos clientes.

Cómo funciona el grapado OCSP en AWS IoT Core

El siguiente diagrama muestra cómo funciona la asociación de OCSP del servidor en AWS IoT Core.



1. El dispositivo debe estar registrado en dominios personalizados con la asociación de OCSP activada.
2. AWS IoT Core llama al respondedor OCSP cada hora para obtener el estado del certificado.
3. El respondedor de OCSP recibe la solicitud, envía la última respuesta de OCSP y almacena la respuesta de OCSP en caché.
4. El dispositivo envía un `ClientHello` mensaje para iniciar el protocolo de enlace TLS. AWS IoT Core
5. AWS IoT Core obtiene la última respuesta de OCSP de la memoria caché del servidor, que responde con una respuesta de OCSP del certificado.
6. El servidor envía un mensaje `ServerHello` al dispositivo. El servidor también envía el certificado de servidor y el estado del certificado al cliente.
7. El dispositivo valida el estado del certificado de OCSP.
8. El protocolo de enlace TLS se completa.

Ventajas de utilizar la asociación de OCSP en comparación con las comprobaciones de OCSP del cliente

A continuación se resumen algunas de las ventajas de utilizar la asociación de OCSP de certificados de servidor:

Mejora de la privacidad

Sin la asociación de OCSP, el dispositivo del cliente puede exponer la información a los respondedores de OCSP externos, lo que podría comprometer la privacidad del usuario. La asociación de OCSP mitiga este problema al hacer que el servidor obtenga la respuesta de OCSP y la entregue directamente al cliente.

Mejora de la fiabilidad

La asociación de OCSP puede mejorar la fiabilidad de las conexiones seguras porque reduce el riesgo de interrupciones del servidor OCSP. Cuando se asocian las respuestas de OCSP, el servidor incluye la respuesta más reciente en el certificado. Esto permite a los clientes acceder al estado de revocación incluso si el respondedor de OCSP no está disponible temporalmente. La asociación de OCSP ayuda a mitigar estos problemas, ya que el servidor recupera las respuestas de OCSP de forma periódica e incluye las respuestas en caché en el establecimiento de comunicación de TLS. Esto reduce la confianza en la disponibilidad en tiempo real de los respondedores de OCSP.

Reducción de la carga del servidor

La asociación de OCSP reduce la carga de responder a las solicitudes de OCSP de los respondedores de OCSP al servidor. Esto puede ayudarle a distribuir la carga de manera más uniforme, lo que hace que el proceso de validación de certificados sea más eficiente y escalable.

Reducción de la latencia

La asociación de OCSP reduce la latencia asociada a la comprobación del estado de revocación de un certificado durante el protocolo de enlace TLS. En lugar de que el cliente tenga que consultar un servidor OCSP distinto, el servidor envía la solicitud y asocia la respuesta de OCSP al certificado del servidor durante el protocolo de enlace.

Habilitar el certificado OCSP del servidor en AWS IoT Core

Para habilitar el engrapado OCSP del certificado de servidor AWS IoT Core, cree una configuración de dominio para un dominio personalizado o actualice una configuración de dominio personalizada

existente. Para obtener más información general sobre cómo crear una configuración de dominio con un dominio personalizado, consulte ???.

Siga las instrucciones siguientes para habilitar el grapado del servidor OCSP mediante o. Consola de administración de AWS AWS CLI

Consola

Para habilitar el grapado OCSP de certificados de servidor mediante la consola: AWS IoT

1. En el menú de navegación, seleccione Configuración y, a continuación, elija Crear configuración de dominio o seleccione una configuración de dominio existente para un dominio personalizado.
2. Si en el paso anterior opta por crear una nueva configuración de dominio, aparecerá la página Crear configuración de dominio. En la sección Propiedades de configuración de dominio, seleccione Dominio personalizado. Introduzca la información para crear una configuración de dominio.

   Si decide actualizar una configuración de dominio existente para un dominio personalizado, verá la página Detalles de configuración del dominio. Elija Edit (Edición de).
3. Para activar la asociación de servidores OCSP, seleccione Habilitar el grapado OCSP del certificado del servidor en la subsección Configuraciones de certificados de servidor.
4. Seleccione Crear configuración de dominio o Actualizar la configuración del dominio.

AWS CLI

Para activar la asociación de OCSP de certificado de servidor mediante la AWS CLI:

1. Si crea una nueva configuración de dominio para un dominio personalizado, el comando para activar la asociación del servidor OCSP puede tener el siguiente aspecto:

```
aws iot create-domain-configuration --domain-configuration-name
 "myDomainConfigurationName" \
        --server-certificate-arns arn:aws:iot:us-
east-1:123456789012:cert/
f8c1e5480266caef0fdb1bf97dc1c82d7ba2d3e2642c5f25f5ba364fc6b79ba3 \
        --server-certificate-config "enableOCSPCheck=true|false"
```

2. Si actualiza una configuración de dominio existente para un dominio personalizado, el comando para activar la asociación del servidor OCSP puede tener el siguiente aspecto:

```
aws iot update-domain-configuration --domain-configuration-name
 "myDomainConfigurationName" \
        --server-certificate-arns arn:aws:iot:us-
east-1:123456789012:cert/
f8c1e5480266caef0fdb1bf97dc1c82d7ba2d3e2642c5f25f5ba364fc6b79ba3 \
        --server-certificate-config "enableOCSPCheck=true|false"
```

Para obtener más información, consulte CreateDomainConfigurationy en la referencia UpdateDomainConfigurationde la AWS IoT API.

Configurar el certificado de servidor OCSP para puntos finales privados en AWS IoT Core

OCSP para puntos de enlace privados le permite utilizar sus recursos OCSP privados dentro de su Amazon Virtual Private Cloud (Amazon VPC) para las operaciones. AWS IoT Core El proceso implica configurar una función de Lambda que actúa como un respondedor de OCSP. La función Lambda puede usar sus recursos privados de OCSP para crear las respuestas de OCSP que utilizará. AWS IoT Core

Función de Lambda

Antes de configurar el OCSP del servidor para un punto de conexión privado, cree una función de Lambda que actúe como un respondedor del Protocolo de Verificación de Certificados en Línea (OCSP), compatible con el proceso de RFC (solicitud de comentarios) 6960 y que admita respuestas de OCSP básicas. La función de Lambda acepta una codificación en base64 de la solicitud de OCSP en el formato de reglas de codificación distinguidas (DER). La respuesta de la función de Lambda también es una respuesta de OCSP codificada en base64 en formato DER. El tamaño de la respuesta no debe superar los 4 kilobytes (KiB). La función Lambda debe estar en la misma configuración del dominio Cuenta de AWS y Región de AWS igual que ella. En el siguiente ejemplo se utilizan funciones de Lambda.

Ejemplo de función de Lambda

JavaScript

```
import * as pkijs from 'pkijs';
console.log('Loading function');

export const handler = async (event, context) => {
```

```
    const requestBytes = decodeBase64(event);
    const ocspRequest = pkijs.OCSPRequest.fromBER(requestBytes);

    console.log("Here is a better look at the OCSP request");
    console.log(ocspRequest.toJSON());

    const ocspResponse = getOcspResponse();

    console.log("Here is a better look at the OCSP response");
    console.log(ocspResponse.toJSON());

   const responseBytes = ocspResponse.toSchema().toBER();
   return encodeBase64(responseBytes);
};

function getOcspResponse() {
    const responseString = "MIIC/
woBAKCCAvgwggL0BgkrBgEFBQcwAQEEggLlMIIC4TCByqFkMGIxJzAlBgNVBAoMHlJpY2hhcmQncyBEaXNjb3VudCBMY
p5w7WOtPjp3otNtVgIBAYAAGA8yMDI0MDQyMzE4NTMyNVowDQYJKoZIhvcNAQELBQADggIBAJFRyjDAHfazNejo704Ra
+s82R1spDarr3k7Pzkod9jJhwsZ2YgushlS4Npfe4lHCdwFyZR75WXrW55aXFddy03KLz01ZLNYyxkleW3f5dgrUcRU3
DEBiyS7ZsyhKo6igWU/SY7YMSKgwBvFsqSDcOa/hRYQkxWKWJ19gcz8CIkWN7NvfIxCs6VrAdzEJwmE7y3v
+jdfhxW9JmI4xStE4K0tAR9vVOOfKs7NvxXj7oc9pCSG60xl96kaEE6PaY1YsfNTsKQ7pyCJ0s7/2q
+ieZ4AtNyzw1XBadPzPJNv6E0LvI24yQZqN5wACvtut5prMMRxAHbOy
+abLZR58wloFSELtGJ7UD96LFv1GgtC5s
+2QlzPc4bEEof7Lo1EISt3j2ibNch8LxhqTQ4ufrbhsMkpSOTFYEJVMJF6aKj/OGXBUUqgc0Jx6jjJXNQd
+l5KCY9pQFeb/wVUYC6mYqZOkNNMMJxPbHHbFnqb68yO+g5BE9011N44YXoPVJYoXxBLFX+OpRu9cqPkT9/
vlkKd+SYXQknwZ81agKzhf1HsBKabtJwNVMlBKaI8g5UGa7Bxi6ewH3ezdWiERRUK7F56OM53wto/";
    const responseBytes = decodeBase64(responseString);
    return pkijs.OCSPResponse.fromBER(responseBytes);
}

function decodeBase64(input) {
    const binaryString = atob(input);

    const byteArray = new Uint8Array(binaryString.length);
    for (var i = 0; i < binaryString.length; i++) {
        byteArray[i] = binaryString.charCodeAt(i);
    }

    return byteArray.buffer;
}

function encodeBase64(buffer) {
    var binary = '';
```

```
    const bytes = new Uint8Array( buffer );
    const len = bytes.byteLength;

    for (var i = 0; i < len; i++) {
        binary += String.fromCharCode( bytes[ i ] );
    }

    return btoa(binary);
}
```

Java

```
package com.example.ocsp.responder;
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.LambdaLogger;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import org.bouncycastle.cert.ocsp.OCSPReq;
import org.bouncycastle.cert.ocsp.OCSPResp;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.Base64;

public class LambdaResponderApplication implements RequestHandler<String, String> {
    @Override
    public String handleRequest(final String input, final Context context) {
        LambdaLogger logger = context.getLogger();

        byte[] decodedInput = Base64.getDecoder().decode(input);

        OCSPReq req;
        try {
            req = new OCSPReq(decodedInput);
        } catch (IOException e) {
            logger.log("Got an IOException creating the OCSP request: " +
 e.getMessage());
            throw new RuntimeException(e);
        }

        try {
            OCSPResp response = businessLogic.getMyResponse();
            String toReturn =
 Base64.getEncoder().encodeToString(response.getEncoded());
```

```
            return toReturn;
        } catch (Exception e) {
            logger.log("Got an exception creating the response: " + e.getMessage());
            return "";
        }
    }
}
```

Autorizar la invocación AWS IoT de la función Lambda

En el proceso de creación de la configuración del dominio con un respondedor OCSP de Lambda, debe conceder AWS IoT permiso para invocar la función Lambda una vez creada la función. Para conceder el permiso, puede utilizar el comando de la CLI add-permission.

Conceda permiso a su función Lambda mediante el AWS CLI

1. Después de insertar sus valores, introduzca el siguiente comando. Tenga en cuenta que el valor `statement-id` debe ser único. Reemplace *`Id-1234`* por el valor exacto que tiene; de lo contrario, podría producirse un error `ResourceConflictException`.

   ```
   aws lambda add-permission  \
   --function-name "ocsp-function" \
   --principal "iot.amazonaws.com" \
   --action "lambda:InvokeFunction" \
   --statement-id "Id-1234" \
   --source-arn arn:aws:iot:us-east-1:123456789012:domainconfiguration/<domain-config-name>/*
   --source-account 123456789012
   ```

   La configuración del dominio de IoT ARNs seguirá el siguiente patrón. El sufijo generado por el servicio no se conocerá antes del momento de la creación, por lo que debe reemplazar el sufijo por un *. Puede actualizar el permiso una vez que se haya creado la configuración del dominio y se conozca el ARN exacto.

   `arn:aws:iot:use-east-1:123456789012:domainconfiguration/domain-config-name/service-generated-suffix`

2. Si el comando tiene éxito, devuelve una declaración de permiso, como la de este ejemplo. Puede continuar con la siguiente sección para configurar la asociación de OCSP para puntos de conexión privados.

```
{
    "Statement": "{\"Sid\":\"Id-1234\",\"Effect\":\"Allow\",\"Principal
\":{\"Service\":\"iot.amazonaws.com\"},\"Action\":\"lambda:InvokeFunction
\",\"Resource\":\"arn:aws:lambda:us-east-1:123456789012:function:ocsp-
function\",\"Condition\":{\"ArnLike\":{\"AWS:SourceArn\":\"arn:aws:iot:us-
east-1:123456789012:domainconfiguration/domain-config-name/*\"}}}"
}
```

Si el comando no tiene éxito, devuelve un error, como en este ejemplo. Tendrá que revisar y corregir el error antes de continuar.

```
An error occurred (AccessDeniedException) when calling the AddPermission operation:
 User: arn:aws:iam::57EXAMPLE833:user/EXAMPLE-1 is not authorized to perform:
 lambda:AddPer
mission on resource: arn:aws:lambda:us-east-1:123456789012:function:ocsp-function
```

Configuración de la asociación de OCSP del servidor para puntos de conexión privados

Consola

Para configurar el grapado OCSP de certificados de servidor mediante la AWS IoT consola:

1. En el menú de navegación, seleccione Configuración y, a continuación, elija Crear configuración de dominio o seleccione una configuración de dominio existente para un dominio personalizado.
2. Si en el paso anterior opta por crear una nueva configuración de dominio, aparecerá la página Crear configuración de dominio. En la sección Propiedades de configuración de dominio, seleccione Dominio personalizado. Introduzca la información para crear una configuración de dominio.

   Si decide actualizar una configuración de dominio existente para un dominio personalizado, verá la página Detalles de configuración del dominio. Elija Edit (Edición de).
3. Para activar la asociación de servidores OCSP, seleccione Habilitar el grapado OCSP del certificado del servidor en la subsección Configuraciones de certificados de servidor.
4. Seleccione Crear configuración de dominio o Actualizar la configuración del dominio.

AWS CLI

Cómo configurar la asociación de OCSP del certificado de servidor mediante la AWS CLI:

1. Si crea una nueva configuración de dominio para un dominio personalizado, el comando para configurar el OCSP del certificado de servidor para puntos de conexión privados puede tener el siguiente aspecto:

```
aws iot create-domain-configuration --domain-configuration-name
 "myDomainConfigurationName" \
        --server-certificate-arns arn:aws:iot:us-
east-1:123456789012:cert/
f8c1e5480266caef0fdb1bf97dc1c82d7ba2d3e2642c5f25f5ba364fc6b79ba3 \
        --server-certificate-config "enableOCSPCheck=true,
 ocspAuthorizedResponderArn=arn:aws:acm:us-
east-1:123456789012:certificate/certificate_ID, ocspLambdaArn=arn:aws:lambda:us-
east-1:123456789012:function:my-function"
```

2. Si actualiza una configuración de dominio existente para un dominio personalizado, el comando para configurar el OCSP del certificado de servidor para puntos de conexión privados puede tener el siguiente aspecto:

```
aws iot update-domain-configuration --domain-configuration-name
 "myDomainConfigurationName" \
        --server-certificate-arns arn:aws:iot:us-
east-1:123456789012:cert/
f8c1e5480266caef0fdb1bf97dc1c82d7ba2d3e2642c5f25f5ba364fc6b79ba3 \
        --server-certificate-config "enableOCSPCheck=true,
 ocspAuthorizedResponderArn=arn:aws:acm:us-
east-1:123456789012:certificate/certificate_ID, ocspLambdaArn=arn:aws:lambda:us-
east-1:123456789012:function:my-function"
```

enableOCSPCheck

Se trata de un valor booleano que indica si la comprobación de asociación de OCSP del servidor está habilitada o no. Para habilitar la asociación de OCSP del certificado de servidor, este valor debe ser true (verdadero).

ocspAuthorizedResponderArn

Se trata de un valor de cadena del nombre de recurso de Amazon (ARN) de un certificado X.509 almacenado en AWS Certificate Manager (ACM). Si se proporciona, AWS IoT Core utilizará este certificado para validar la firma de la respuesta OCSP recibida. Si no se proporciona, AWS IoT Core utilizará el certificado emisor para validar las respuestas. El certificado debe estar en la misma configuración Cuenta de AWS y Región de AWS en el dominio. Para obtener más información sobre cómo registrar el certificado de respondedor autorizado, consulte Importar certificados a AWS Certificate Manager.

ocspLambdaArn

Se trata de un valor de cadena del nombre de recurso de Amazon (ARN) de una función de Lambda que actúa como un respondedor del Protocolo de Verificación de Certificados en Línea (OCSP), compatible con el proceso de RFC (solicitud de comentarios) 6960 y que admita respuestas de OCSP básicas. La función de Lambda acepta una codificación en base64 de la solicitud de OCSP que se codifica con el formato DER. La respuesta de la función de Lambda también es una respuesta de OCSP codificada en base64 en formato DER. El tamaño de la respuesta no debe superar los 4 kilobytes (KiB). La función Lambda debe estar en la misma configuración del dominio Cuenta de AWS y Región de AWS igual que ella.

Para obtener más información, consulte CreateDomainConfigurationy UpdateDomainConfigurationdesde la referencia de la AWS IoT API.

Notas importantes sobre el uso del certificado de servidor (OCSP): grapado AWS IoT Core

Cuando utilice el certificado de servidor OCSP AWS IoT Core, tenga en cuenta lo siguiente:

1. AWS IoT Core solo admite los respondedores OCSP a los que se puede acceder a través de direcciones públicas. IPv4
2. La función de grapado OCSP AWS IoT Core no admite respondedores autorizados. Todas las respuestas del OCSP deben estar firmadas por la CA que firmó el certificado y la CA debe formar parte de la cadena de certificados del dominio personalizado.
3. La función de grapado OCSP AWS IoT Core no admite dominios personalizados que se crean con certificados autofirmados.
4. AWS IoT Core llama a un respondedor OCSP cada hora y guarda la respuesta en caché. Si la llamada al respondedor falla, AWS IoT Core grabará la respuesta válida más reciente.

5. Si ya no `nextUpdateTime` es válido, AWS IoT Core eliminará la respuesta de la memoria caché y el protocolo de enlace TLS no incluirá los datos de respuesta del OCSP hasta la próxima llamada correcta al respondedor del OCSP. Esto puede ocurrir cuando la respuesta en caché caduca antes de que el servidor reciba una respuesta válida del respondedor de OCSP. El valor de `nextUpdateTime` sugiere que la respuesta de OCSP será válida hasta ese momento. Para obtener más información acerca de `nextUpdateTime`, consulte ???.
6. A veces, AWS IoT Core no recibe la respuesta del OCSP o elimina la respuesta OCSP existente porque ha caducado. Si se producen situaciones como estas, AWS IoT Core seguirá utilizando el certificado de servidor proporcionado por el dominio personalizado sin la respuesta del OCSP.
7. El tamaño de la respuesta de OCSP no puede superar los 4 KiB.

Solución de problemas al grapar el certificado de servidor OCSP AWS IoT Core

AWS IoT Core Emite la `RetrieveOCSPStapleData.Success` métrica y las entradas de `RetrieveOCSPStapleData` registro a. CloudWatch La métrica y las entradas de registro pueden ayudar a detectar problemas relacionados con la recuperación de las respuestas del OCSP. Para obtener más información, consulte ??? y ???.

Conéctese a puntos AWS IoT finales FIPS

AWS IoT proporciona puntos finales compatibles con el Estándar Federal de Procesamiento de Información (FIPS) 140-2. Los puntos finales que cumplen con el FIPS son diferentes de los puntos finales estándar. AWS Para interactuar con AWS IoT de forma compatible con FIPS, debe utilizar los puntos de conexión que se describen a continuación con su cliente compatible con FIPS. La AWS IoT consola no es compatible con FIPS.

En las siguientes secciones se describe cómo acceder a los AWS IoT puntos finales compatibles con FIPS mediante la API REST, un SDK o el. AWS CLI

Temas

- puntos de conexión del plano de control de AWS IoT Core
- puntos de conexión del plano de datos de AWS IoT Core
- Puntos de conexión de proveedores de credenciales de AWS IoT Core
- puntos de conexión de datos de trabajos de AWS IoT Device Management
- puntos de conexión de AWS IoT Device Management - Fleet Hub
- puntos de conexión de tunelización segura de AWS IoT Device Management

- AWS IoT Device Management - Puntos finales de integraciones gestionadas

puntos de conexión del plano de control de AWS IoT Core

Los puntos de conexión del plano de control de AWS IoT Core compatibles con FIPS que admiten operaciones de AWS IoT y sus comandos de la CLI relacionados se enumeran en puntos de conexión FIPS por servicio. En Puntos de enlace FIPS por servicio, busque el servicio de plano de control de AWS IoT Core y localice el punto de conexión correspondiente a su Región de AWS.

Para utilizar el punto de conexión compatible con FIPS al acceder a las AWS IoToperaciones, utilice el AWS SDK o la API REST con el punto de conexión adecuado para usted. Región de AWS

Para utilizar el punto de conexión compatible con FIPS al ejecutar comandos de la CLI aws iot, agregue al comando el parámetro --endpoint con el punto de conexión adecuado para su Región de AWS .

puntos de conexión del plano de datos de AWS IoT Core

Los puntos de conexión del plano de datos de AWS IoT Core compatibles con FIPS se enumeran en puntos de conexión FIPS por servicio. En Puntos de enlace FIPS por servicio, busque el servicio de plano de datos de AWS IoT Core y localice el punto de conexión correspondiente a su Región de AWS.

Para utilizar el terminal compatible con FIPS Región de AWS con un cliente compatible con FIPS, utilice el SDK del AWS IoT dispositivo y proporcione el punto final a la función de conexión del SDK en lugar del punto final predeterminado de su cuenta (el punto final del plano de datos AWS IoT Core). La función de conexión es específica del SDK del AWS IoT dispositivo. Para ver un ejemplo de una función de conexión, consulte la función de conexión en el SDK de AWS IoT dispositivos para Python.

> **Note**
>
> AWS IoT no admite puntos finales Cuenta de AWS de planos de datos específicos AWS IoT Core que sean compatibles con el FIPS. No se pueden utilizar las funciones de servicio que requieren un punto final Cuenta de AWS específico en la indicación del nombre del servidor (SNI). Los puntos de conexión del plano de datos de AWS IoT Core no admiten certificados de registro de varias cuentas, dominios personalizados, autorizadores personalizados ni puntos de conexión configurables (incluidas las políticas de TLS compatibles).

Puntos de conexión de proveedores de credenciales de AWS IoT Core

Los puntos de conexión de proveedores de credenciales de AWS IoT Core compatibles con FIPS se enumeran en Puntos de conexión FIPS por servicio. En Puntos de conexión FIPS por servicio, busque el servicio Proveedores de credenciales de AWS IoT Core y localice el punto de conexión correspondiente a su Región de AWS.

> **Note**
>
> AWS IoT no admite terminales de proveedores Cuenta de AWS de credenciales específicos AWS IoT Core que cumplan con el FIPS. No se pueden utilizar las funciones de servicio que requieren un punto final Cuenta de AWS específico en la indicación del nombre del servidor (SNI). Los puntos de conexión de proveedores de credenciales de AWS IoT Core no admiten certificados de registro de varias cuentas, dominios personalizados, autorizadores personalizados ni puntos de conexión configurables (incluidas las políticas de TLS compatibles).

puntos de conexión de datos de trabajos de AWS IoT Device Management

Los puntos de conexión datos de trabajos de AWS IoT Device Management compatibles con FIPS se enumeran en puntos de conexión FIPS por servicio. En Puntos de enlace FIPS por servicio, busque el servicio de datos de trabajos de AWS IoT Device Management y localice el punto de conexión correspondiente a su Región de AWS.

Para utilizar el punto de conexión de AWS IoT Device Management: datos de trabajos compatible con FIPS al ejecutar comandos de la CLI aws iot-jobs-data, agregue al comando el parámetro --endpoint con el punto de conexión adecuado para la Región de AWS . También puede usar la API de REST con este punto de conexión.

Recomendamos utilizarlas `Data-ATS` en lugar de. `iot:Jobs` `iot:Data-ATS`admite puntos finales de doble pila (IPv4 y IPv6), mientras que solo `iot:Jobs` admite. IPv4

Para utilizar el terminal compatible con FIPS Región de AWS con un cliente compatible con FIPS, utilice el SDK del AWS IoT dispositivo y proporcione el punto final a la función de conexión del SDK en lugar del punto final predeterminado de su cuenta (jobs data endpoint AWS IoT Device Management). La función de conexión es específica del SDK de dispositivos AWS IoT . Para ver un ejemplo de una función de conexión, consulte la función de conexión en el SDK de AWS IoT dispositivos para Python.

puntos de conexión de AWS IoT Device Management - Fleet Hub

Los puntos finales de Fleet Hub que cumplen AWS IoT Device Management con FIPS y que se utilizan con los comandos CLI de Fleet Hub para la administración de AWS IoT dispositivos aparecen en los puntos finales de FIPS por servicio. En Puntos de enlace FIPS por servicio, busque el servicio AWS IoT Device Management - Fleet Hub y localice el punto de conexión correspondiente a su Región de AWS.

Para utilizar el punto final Fleet Hub compatible con AWS IoT Device Management FIPS al ejecutar comandos de aws iotfleethub CLI, añada al comando el --endpoint parámetro con el punto final adecuado Región de AWS para usted. También puede usar la API de REST con este punto de conexión.

puntos de conexión de tunelización segura de AWS IoT Device Management

Los puntos de conexión de tunelización segura de AWS IoT Device Management compatibles con FIPS para la API de tunelización segura de AWS IoT y los comandos de la CLI correspondientes se enumeran en puntos de conexión FIPS por servicio. En Puntos de enlace FIPS por servicio, busque el servicio de tunelización segura de AWS IoT Device Management y localice el punto de conexión correspondiente a su Región de AWS.

Para utilizar el punto de conexión de AWS IoT Device Management: tunelización segura compatible con FIPS al ejecutar comandos de la CLI aws iotsecuretunneling, agregue al comando el parámetro --endpoint con el punto de conexión adecuado para la Región de AWS . También puede usar la API de REST con este punto de conexión.

AWS IoT Device Management - Puntos finales de integraciones gestionadas

Los puntos finales del plano de control compatibles con la FIPS que admiten las operaciones de integración gestionadas y sus AWS CLI comandos relacionados se enumeran en los puntos finales de la FIPS por servicio. En Puntos de conexión FIPS por servicio, busque el servicio AWS IoT Device Management : integraciones administradas y localice el punto de conexión correspondiente a la Región de AWS.

Para utilizar un punto de conexión compatible con FIPS al acceder a las operaciones de integración gestionadas, utilice el AWS SDK o la API REST con el punto de conexión adecuado para usted. Región de AWS

Para utilizar el punto de conexión compatible con FIPS al ejecutar comandos de la CLI de integraciones administradas, agregue al comando el parámetro --endpoint con el punto de conexión adecuado para la Región de AWS .

Administrar dispositivos con AWS IoT

AWS IoT proporciona un registro que le ayuda a administrar las cosas. Un objeto es una representación de un determinado dispositivo 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 entidad física que no se conecta a otros dispositivos que sí lo hacen, AWS IoT pero que está relacionada con ellos (por ejemplo, un automóvil que tiene sensores de motor o un panel de control).

La información sobre un objeto se almacena en el registro 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 de la cosa, como su número de serie o su fabricante.

En un caso de uso de dispositivo típico, se utiliza el nombre de la cosa como ID de cliente MQTT predeterminado. Aunque no exigimos una asignación entre el nombre de registro de una cosa y su uso del cliente MQTT IDs, los certificados o el estado oculto, le recomendamos que elija un nombre de cosa y lo utilice como ID de cliente MQTT tanto para el registro como para el servicio Device Shadow. 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.

No es necesario crear un objeto en el registro para conectar un dispositivo a AWS IoT. Al añadir objetos al registro, podrá administrar y buscar dispositivos con más facilidad.

Los datos del registro de cosas (incluidos los atributos, los tipos de cosas y la pertenencia a grupos) también se pueden recuperar dinámicamente en el motor de reglas para utilizarlos

en el procesamiento y el enrutamiento de mensajes. Para obtener más información, consulte `get_registry_data`.

Administración de objetos con el registro

Utiliza la AWS IoT consola, la AWS IoT API o la AWS CLI para interactuar con el registro. En las secciones siguientes se muestra cómo utilizar la CLI para trabajar con el registro.

Al nombrar sus objetos cosa:

- No utilice información personal identificable en el nombre del objeto. El nombre de la cosa puede aparecer en comunicaciones e informes no cifrados.

Temas

- Creación de un objeto
- Lista de objetos
- Describir las cosas
- Actualización de un objeto
- Eliminación de un objeto
- Asociar un principal a un objeto
- Enumeración de los objetos asociados a una entidad principal
- Enumeración de las entidades principales asociadas a un objeto
- Enumeración de los objetos asociados a una entidad principal V2
- Enumeración de las entidades principales asociadas a un objeto V2
- Desvincular un principal de un objeto

Creación de un objeto

El siguiente comando muestra cómo utilizar el AWS IoT CreateThing comando de la CLI para crear una cosa. No se puede modificar el nombre de un objeto una vez creado. Para cambiar el nombre de un objeto, debe crear un objeto nuevo, asignarle el nuevo nombre y eliminar el objeto antiguo.

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

El comando CreateThing muestra el nombre y el ARN (nombre de recurso de Amazon) del nuevo objeto:

```
{
    "thingArn": "arn:aws:iot:us-east-1:123456789012:thing/MyLightBulb",
    "thingName": "MyLightBulb",
    "thingId": "12345678abcdefgh12345678ijklmnop12345678"
}
```

> **Note**
>
> No es recomendable utilizar datos personales en los nombres de objeto.

Para obtener más información, consulte create-thing en la Referencia de comandos AWS CLI .

Lista de objetos

Puede utilizar el comando ListThings para enumerar todos los objetos en su cuenta:

```
$ aws iot list-things
```

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

```
}
```

Puede utilizar el comando ListThings para buscar todas las cosas de un tipo específico:

```
$  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 el comando ListThings para buscar todos los objetos que tienen un atributo con un valor específico. Este comando busca hasta tres atributos.

```
$  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"
        }
    ]
}
```

Para obtener más información, consulte list-things en la Referencia de comandos AWS CLI .

Describir las cosas

Puede utilizar el comando DescribeThing para mostrar información más detallada sobre una cosa:

```
$ aws iot describe-thing --thing-name "MyLightBulb"
{
    "version": 3,
    "thingName": "MyLightBulb",
    "thingArn": "arn:aws:iot:us-east-1:123456789012:thing/MyLightBulb",
    "thingId": "12345678abcdefgh12345678ijklmnop12345678",
    "defaultClientId": "MyLightBulb",
    "thingTypeName": "StopLight",
    "attributes": {
        "model": "123",
        "wattage": "75"
```

```
    }
}
```

También puede acceder a esta API dentro del motor de reglas mediante la función `get_registry_data()` en línea. Puede utilizar esta función para acceder y utilizar de forma dinámica la información del registro de objetos (incluidos los atributos, los tipos de elementos y la pertenencia a grupos) mediante llamadas y `ListThingGroupsForThing` APIs directamente dentro de AWS IoT las reglas, lo que permite procesar `DescribeThing` y enrutar los mensajes en tiempo real en función de los datos de registro del dispositivo. Para obtener más información, consulte `get_registry_data`.

Para obtener más información, consulte describe-thing en la Referencia de comandos. AWS CLI

Actualización de un objeto

Puede utilizar el comando UpdateThing para actualizar un objeto. Este comando solo actualiza los atributos del objeto. El nombre de un objeto no se puede modificar. Para cambiar el nombre de un objeto, debe crear un objeto nuevo, asignarle el nuevo nombre y eliminar el objeto antiguo.

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

El comando UpdateThing no genera una salida. Puede utilizar el comando de DescribeThing para ver el resultado:

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

Para obtener más información, consulte update-thing en la Referencia de comandos de AWS CLI .

Eliminación de un objeto

Puede utilizar el comando DeleteThing para eliminar un objeto:

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

Este comando se devuelve correctamente sin error si la eliminación se realiza correctamente o especifica un objeto que no existe.

Para obtener más información, consulte describe-thing en la Referencia de comandos de AWS CLI .

Asociar un principal a un objeto

Un dispositivo físico puede utilizar un principal para comunicarse con él. AWS IoT Una entidad principal puede ser un certificado X.509 o un ID de Amazon Cognito. Puede asociar un certificado o un ID de Amazon Cognito al elemento del registro que representa su dispositivo ejecutando el attach-thing-principalcomando.

Para adjuntar un certificado o un Amazon Cognito ID a tu dispositivo, usa el attach-thing-principalcomando:

```
$ aws iot attach-thing-principal \
    --thing-name "MyLightBulb1" \
    --principal "arn:aws:iot:us-
east-1:123456789012:cert/
a0c01f5835079de0a7514643d68ef8414ab739a1e94ee4162977b02b12842847"
```

Para adjuntar un certificado a su dispositivo con un tipo de adjunto (adjunto exclusivo o no exclusivo), utilice el attach-thing-principalcomando y especifique un tipo en el `--thing-principal-type` campo. Una asociación exclusiva significa que el objeto de IoT es lo único que se asocia al certificado y este certificado no se puede asociar a ningún otro objeto. Una asociación no exclusiva significa que el objeto de IoT se asocia al certificado y este certificado se puede asociar a otro objeto. Para obtener más información, consulte ???.

> **Note**
>
> Para la característica ???, solo puede usar el certificado X.509 como entidad principal.

```
$ aws iot attach-thing-principal \
    --thing-name "MyLightBulb2" \
    --principal "arn:aws:iot:us-
east-1:123456789012:cert/
a0c01f5835079de0a7514643d68ef8414ab739a1e94ee4162977b02b12842847" \
```

```
    --thing-principal-type "EXCLUSIVE_THING"
```

Si la asociación se realiza correctamente, el comando AttachThingPrincipal no producirá ningún resultado. Para describir el adjunto, utilice el comando list-thing-principals-v 2 CLI.

Para obtener más información, consulte AttachThingPrincipal y en la Referencia de la API de AWS IoT Core .

Enumeración de los objetos asociados a una entidad principal

Para enumerar los objetos asociados a la entidad principal especificada, ejecute el comando `list-principal-things`. Tenga en cuenta que este comando no indica el tipo de asociación entre el objeto y el certificado. Para enumerar el tipo de asociación, utilice el comando `list-principal-things-v2`. Para obtener más información, consulte ???.

```
$ aws iot list-principal-things \
    --principal "arn:aws:iot:us-
east-1:123456789012:cert/
2e1eb273792174ec2b9bf4e9b37e6c6c692345499506002a35159767055278e8"
```

El resultado puede ser similar al siguiente.

```
{
    "things": [
        "MyLightBulb1",
        "MyLightBulb2"
    ]
}
```

Para obtener más información, consulte ListPrincipalThings y en la Referencia de la API de AWS IoT Core .

Enumeración de las entidades principales asociadas a un objeto

Para enumerar las entidades principales asociadas al objeto especificado, ejecute el comando `list-thing-principals`. Tenga en cuenta que este comando no indica el tipo de asociación entre el objeto y el certificado. Para enumerar el tipo de asociación, utilice el comando `list-thing-principals-v2`. Para obtener más información, consulte ???.

```
$ aws iot list-thing-principals \
```

```
    --thing-name "MyLightBulb1"
```

El resultado puede ser similar al siguiente.

```
{
    "principals": [
        "arn:aws:iot:us-
east-1:123456789012:cert/
2e1eb273792174ec2b9bf4e9b37e6c6c692345499506002a35159767055278e8",
        "arn:aws:iot:us-
east-1:123456789012:cert/
1a234b39b4b68278f2e9d84bf97eac2cbf4a1c28b23ea29a44559b9bcf8d395b"
    ]
}
```

Para obtener más información, consulte ListThingPrincipals y en la Referencia de la API de AWS IoT Core .

Enumeración de los objetos asociados a una entidad principal V2

Para enumerar los objetos asociados al certificado especificado, junto con el tipo de asociación, ejecute el comando `list-principal-things-v2`. El tipo de asociación hace referencia a cómo se asocia el certificado al objeto.

```
$ aws iot list-principal-things-v2 \
    --principal "arn:aws:iot:us-
east-1:123456789012:cert/
2e1eb273792174ec2b9bf4e9b37e6c6c692345499506002a35159767055278e8"
```

El resultado puede ser similar al siguiente.

```
{
    "PrincipalThingObjects": [
        {
            "thingPrincipalType": "NON_EXCLUSIVE_THING",
            "thing": "arn:aws:iot:us-east-1:123456789012:thing/thing_1"
        },
        {
            "thingPrincipalType": "NON_EXCLUSIVE_THING",
            "thing": "arn:aws:iot:us-east-1:123456789012:thing/thing_2"
        }
```

```
    ]
}
```

Para obtener más información, consulte la ListPrincipalThingsversión 2 de la referencia de la AWS IoT Core API.

Enumeración de las entidades principales asociadas a un objeto V2

Para enumerar los certificados asociados al objeto especificado, junto con el tipo de asociación, ejecute el comando `list-thing-principals-V2`. El tipo de asociación hace referencia a cómo se asocia el certificado al objeto.

```
$ aws iot list-thing-principals-v2 \
    --thing-name "thing_1"
```

El resultado puede ser similar al siguiente.

```
{
    "ThingPrincipalObjects": [
        {
            "thingPrincipalType": "NON_EXCLUSIVE_THING",
            "principal": "arn:aws:iot:us-
east-1:123456789012:cert/
2e1eb273792174ec2b9bf4e9b37e6c6c692345499506002a35159767055278e8"
        },
        {
            "thingPrincipalType": "NON_EXCLUSIVE_THING",
            "principal": "arn:aws:iot:us-
east-1:123456789012:cert/
1a234b39b4b68278f2e9d84bf97eac2cbf4a1c28b23ea29a44559b9bcf8d395b"
        }
    ]
}
```

Para obtener más información, consulte la ListThingsPrincipalversión 2 de la referencia de la AWS IoT Core API.

Desvincular un principal de un objeto

Puede utilizar el comando `DetachThingPrincipal` para desvincular un certificado de un objeto:

```
$ aws iot detach-thing-principal \
    --thing-name "MyLightBulb" \
    --principal "arn:aws:iot:us-
east-1:123456789012:cert/
2e1eb273792174ec2b9bf4e9b37e6c6c692345499506002a35159767055278e8"
```

El comando DetachThingPrincipal no genera ninguna salida.

Para obtener más información, consulte detach-thing-principal y en la Referencia de la API de AWS IoT Core .

Tipos de cosas

Los tipos de objetos 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. Por ejemplo, puede definir un tipo de LightBulb cosa. Todos los elementos asociados al LightBulb tipo de objeto comparten un conjunto de atributos: número de serie, fabricante y potencia. Al crear un objeto de tipo LightBulb (o al cambiar el tipo de un objeto existente LightBulb), puede especificar valores para cada uno de los atributos definidos en el tipo de LightBulb objeto.

Aunque los tipos de objeto son opcionales, su uso facilita la detección de 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.
- Los objetos solo se pueden asociar a un tipo de objeto.
- El número de tipos de objetos que puede crear en su cuenta es ilimitado.

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.

Temas:

- Creación de un tipo de objeto
- Lista de los tipos de objeto
- Descripción de un tipo de objeto
- Asociación de un tipo de objeto a un objeto

- Actualización de un tipo de objeto
- Descartar un tipo de objeto
- Eliminación de un tipo de objeto

Creación de un tipo de objeto

Puede utilizar el comando CreateThingType 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 CreateThingType devuelve una respuesta que contiene el tipo de objeto y su ARN:

```
{
    "thingTypeName": "LightBulb",
    "thingTypeId": "df9c2d8c-894d-46a9-8192-9068d01b2886",
    "thingTypeArn": "arn:aws:iot:us-west-2:123456789012:thingtype/LightBulb"
}
```

Lista de los tipos de objeto

Puede utilizar el comando ListThingTypes para crear una lista de los tipos de objeto:

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

El ListThingTypes comando devuelve una lista de los tipos de cosas definidos en su Cuenta de AWS:

```
{
    "thingTypes": [
        {
            "thingTypeName": "LightBulb",
            "thingTypeProperties": {
                "searchableAttributes": [
                    "wattage",
                    "model"
                ],
                "thingTypeDescription": "light bulb type"
```

```
            },
            "thingTypeMetadata": {
                "deprecated": false,
                "creationDate": 1468423800950
            }
        }
    ]
}
```

Descripción de un tipo de objeto

Puede utilizar el comando DescribeThingType para obtener información acerca de un tipo de objeto:

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

El comando DescribeThingType devuelve información acerca del tipo especificado:

```
{
    "thingTypeProperties": {
        "searchableAttributes": [
            "model",
            "wattage"
        ],
        "thingTypeDescription": "light bulb type"
    },
    "thingTypeId": "df9c2d8c-894d-46a9-8192-9068d01b2886",
    "thingTypeArn": "arn:aws:iot:us-west-2:123456789012:thingtype/LightBulb",
    "thingTypeName": "LightBulb",
    "thingTypeMetadata": {
        "deprecated": false,
        "creationDate": 1544466338.399
    }
}
```

Asociación de un tipo de objeto a un objeto

Puede utilizar el comando CreateThing para especificar un tipo de objeto cuando crea un objeto:

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

Puede utilizar el comando UpdateThing en cualquier momento para cambiar el tipo de objeto asociado a un objeto:

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

También puede utilizar el comando UpdateThing para desvincular un objeto de un tipo de objeto.

Actualización de un tipo de objeto

Puede utilizar el comando UpdateThingType para actualizar un tipo de objeto cuando crea un objeto:

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

Puede utilizar el comando UpdateThing en cualquier momento para cambiar el tipo de objeto asociado a un objeto:

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

También puede utilizar el comando UpdateThing 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 estén asociados al tipo de objeto permanecen igual, sin cambios.

Para descartar un tipo de objeto, utilice el comando DeprecateThingType:

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

Puede utilizar el comando de DescribeThingType para ver el resultado:

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

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

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

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

Puede utilizar el comando de la CLI DescribeThingType para ver el resultado:

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

```
{
    "thingTypeName": "StopLight",
    "thingTypeArn": "arn:aws:iot:us-east-1:123456789012:thingtype/StopLight",
    "thingTypeId": "12345678abcdefgh12345678ijklmnop12345678"
    "thingTypeProperties": {
        "searchableAttributes": [
            "wattage",
            "numOfLights",
            "model"
        ],
        "thingTypeDescription": "traffic light type"
    },
    "thingTypeMetadata": {
        "deprecated": false,
        "creationDate": 1468425854308,
```

```
    }
}
```

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 el comando DeleteThingType:

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

> **Note**
>
> Para poder eliminar un tipo de objeto descartado debe esperar cinco minutos.

Grupos de objetos estáticos

Los grupos de objetos estáticos permiten administrar varios objetos a la vez clasificándolos en grupos. Los grupos de objetos estáticos contienen un grupo de objetos que se administran a través de la consola, la CLI o la API. Por otro lado, los grupos de objetos dinámicos contienen objetos que coinciden con una consulta especificada. Los grupos de cosas estáticas también pueden contener otros grupos de cosas estáticas - puede construir una jerarquía de grupos. Puede asociar una política a un grupo principal y sus grupos secundarios la heredarán, así como todos los objetos del grupo y los grupos secundarios. Esto facilita el control de los permisos para un gran número de objetos.

> **Note**
>
> Las políticas de grupos de cosas no permiten el acceso a las operaciones del plano de AWS IoT Greengrass datos. Para permitir que una cosa acceda a una operación del plano de AWS IoT Greengrass datos, añada el permiso a una AWS IoT política que adjunte al certificado de la cosa. Para obtener más información, vea Autenticación y autorización de dispositivos en la guía de desarrolladores de AWS IoT Greengrass .

Esto es lo que puede hacer con los grupos de objetos estáticos:

- Crear, describir o eliminar un grupo.

- Añadir un objeto a un grupo o a más de un grupo.
- Quitar un objeto de un grupo.
- Enumerar los grupos que ha creado.
- Enumerar todos los grupos secundarios de un grupo (sus descendientes directos e indirectos).
- Enumerar los objetos de un grupo, incluidos todos los objetos en sus grupos secundarios.
- Enumerar todos los grupos antecesores de un grupo (sus grupos principales directos e indirectos).
- Añadir, eliminar o actualizar los atributos de un grupo. (Los atributos son pares nombre-valor que puede usar para almacenar información acerca de un grupo).
- asociar una política a un grupo o separarla de él.
- Enumerar las directivas adjuntadas a un grupo.
- Enumerar las políticas heredadas por un objeto (en virtud de las políticas adjuntadas a su grupo o uno de sus grupos principales).
- Configurar opciones de registro para objetos de un grupo. Consulte Configure el AWS IoT registro.
- Crear trabajos que se enviarán y se ejecutarán en cada objeto del grupo y sus grupos secundarios. Consulte AWS IoT Empleos.

> **Note**
>
> Cuando una cosa está asociada a un grupo de cosas estáticas al que está asociada una AWS IoT Core política, el nombre de la cosa debe coincidir con el ID de cliente.

Los grupos de objetos estáticos tienen algunas limitaciones:

- Un grupo puede tener como máximo un grupo principal directo.
- Si un grupo es secundario de otro grupo, debe especificarlo en el momento en que se crea.
- No puede cambiar el principal de un grupo más adelante, así que asegúrese de planificar la jerarquía de grupos y crear un grupo principal antes de crear los grupos secundarios que contiene.
- El número de grupos a los que puede pertenecer un objeto es limitado.
- No puede añadir un objeto a más de un grupo en la misma jerarquía. (En otras palabras, no puede agregar un objeto a dos grupos que comparten un principal común).
- No se puede cambiar el nombre de un grupo.

- Los nombres de grupos de objetos no pueden contener caracteres internacionales, como û, é o ñ.
- No utilice información personal identificable en el nombre del grupo de objetos. El nombre del grupo de cosas puede aparecer en comunicaciones e informes no cifrados.

Asociar políticas a grupos y separarlas de ellos puede mejorar la seguridad de las operaciones de AWS IoT de una serie de formas significativas. El método por dispositivo de asociar una política a un certificado, que luego se asocia a un objeto, consume mucho tiempo y dificulta la actualización o cambio rápido de políticas en una flota de dispositivos. Disponer de una política adjunta al grupo de la cosa ahorra pasos cuando hay que rotar los certificados en un objeto. Además, las políticas se aplican dinámicamente a objetos cuando cambian la pertenencia a un grupo, por lo que no es necesario volver a crear un conjunto complejo de permisos cada vez que un dispositivo cambia la pertenencia en un grupo.

Crear un grupo de objetos estático

Utilice el comando CreateThingGroup para crear un grupo de objetos estático:

```
$ aws iot create-thing-group --thing-group-name LightBulbs
```

El comando CreateThingGroup devuelve una respuesta que contiene el nombre, el ID y el ARN del grupo de objetos estático:

```
{
    "thingGroupName": "LightBulbs",
    "thingGroupId": "abcdefgh12345678ijklmnop12345678qrstuvwx",
    "thingGroupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/LightBulbs"
}
```

> **Note**
>
> No es recomendable utilizar datos personales en los nombres de grupo de objetos.

A continuación, se muestra un ejemplo que especifica el grupo principal de un grupo de objetos estático durante su creación:

```
$ aws iot create-thing-group --thing-group-name RedLights --parent-group-name
 LightBulbs
```

Al igual que antes, el comando CreateThingGroup devuelve una respuesta que contiene el nombre, el ID y el ARN del grupo de objetos:

```
{
    "thingGroupName": "RedLights",
    "thingGroupId": "abcdefgh12345678ijklmnop12345678qrstuvwx",
    "thingGroupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RedLights",
}
```

Important

Tenga en cuenta las siguientes limitaciones cuando cree jerarquías de grupos de objetos:

- Un grupo de objetos solo puede tener un grupo principal directo.
- El número de grupos secundarios directos que puede tener un grupo de objetos es limitado.
- La profundidad máxima de una jerarquía de grupo es limitada.
- El número de atributos que puede tener un grupo de objetos es limitado. (Los atributos son pares nombre-valor que puede usar para almacenar información acerca de un grupo). La longitud de cada nombre de atributo y cada valor también es limitada.

Descripción de un grupo de objetos

Puede utilizar el comando DescribeThingGroup para obtener información acerca de un grupo de objetos:

```
$ aws iot describe-thing-group --thing-group-name RedLights
```

El comando DescribeThingGroup devuelve información acerca del grupo especificado:

```
{
    "thingGroupName": "RedLights",
    "thingGroupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RedLights",
    "thingGroupId": "12345678abcdefgh12345678ijklmnop12345678",
    "version": 1,
    "thingGroupMetadata": {
        "creationDate": 1478299948.882
```

```
        "parentGroupName": "Lights",
        "rootToParentThingGroups": [
            {
                "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/
ShinyObjects",
                "groupName": "ShinyObjects"
            },
            {
                "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/LightBulbs",
                "groupName": "LightBulbs"
            }
        ]
    },
    "thingGroupProperties": {
        "attributePayload": {
            "attributes": {
                "brightness": "3400_lumens"
            },
        },
        "thingGroupDescription": "string"
    },
}
```

Agregar un objeto a un grupo de objetos estático

Puede utilizar el comando AddThingToThingGroup para agregar un objeto a un grupo de objetos estático:

```
$ aws iot add-thing-to-thing-group --thing-name MyLightBulb --thing-group-name
 RedLights
```

El comando AddThingToThingGroup no genera una salida.

> **Important**
>
> Puede añadir un objeto a un máximo de 10 grupos. Sin embargo, no puede añadir un objeto a más de un grupo en la misma jerarquía. (En otras palabras, no puede añadir un objeto a dos grupos que comparten un grupo principal común).
> Si un objeto pertenece al número máximo de grupos de objetos posible y uno o varios de estos grupos es un grupo de objetos dinámico, puede utilizar la marca

overrideDynamicGroups para que los grupos estáticos tengan prioridad sobre los grupos dinámicos.

Eliminar un objeto de un grupo de objetos estático

Puede utilizar el comando RemoveThingFromThingGroup para quitar un objeto de un grupo:

```
$ aws iot remove-thing-from-thing-group --thing-name MyLightBulb --thing-group-name
 RedLights
```

El comando RemoveThingFromThingGroup no genera una salida.

Enumerar los objetos en un grupo de objetos

Puede utilizar el comando ListThingsInThingGroup para listar los objetos que pertenecen a un grupo:

```
$ aws iot list-things-in-thing-group --thing-group-name LightBulbs
```

El comando ListThingsInThingGroup devuelve una lista de los objetos en el grupo determinado:

```
{
    "things":[
        "TestThingA"
    ]
}
```

Con el parámetro `--recursive`, puede enumerar los objetos que pertenecen a un grupo y también los que están en alguno de sus grupos secundarios:

```
$ aws iot list-things-in-thing-group --thing-group-name LightBulbs --recursive
```

```
{
    "things":[
        "TestThingA",
        "MyLightBulb"
    ]
}
```

> **Note**
>
> Esta operación es a largo plazo coherente. En otras palabras, los cambios que se hagan en el grupo de objetos podrían no reflejarse inmediatamente.

Enumeración de grupos de objetos

Puede usar el comando ListThingGroups para mostrar los grupos de objetos de la cuenta:

```
$ aws iot list-thing-groups
```

El ListThingGroups comando devuelve una lista de los grupos de cosas de su Cuenta de AWS:

```
{
    "thingGroups": [
        {
            "groupName": "LightBulbs",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/LightBulbs"
        },
        {
            "groupName": "RedLights",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RedLights"
        },
        {
            "groupName": "RedLEDLights",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RedLEDLights"
        },
        {
            "groupName": "RedIncandescentLights",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/
RedIncandescentLights"
        }
        {
            "groupName": "ReplaceableObjects",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/
ReplaceableObjects"
        }
    ]
}
```

Utilice los filtros opcionales para enumerar los grupos que tienen un grupo determinado como grupo principal (`--parent-group`) o los grupos cuyo nombre comienza por un prefijo determinado (`--name-prefix-filter`). El parámetro `--recursive` le permite listar también todos los grupos secundarios, no los grupos secundarios directos de un grupo de objetos:

```
$ aws iot list-thing-groups --parent-group LightBulbs
```

En este caso, el ListThingGroups comando devuelve una lista de los grupos secundarios directos del grupo de cosas definido en su Cuenta de AWS:

```
{
    "childGroups":[
        {
            "groupName": "RedLights",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RedLights"
        }
    ]
}
```

Utilice el parámetro `--recursive` con el comando ListThingGroups para listar todos los grupos secundarios de un grupo de objetos, no solo un elemento secundario directo:

```
$ aws iot list-thing-groups --parent-group LightBulbs --recursive
```

El comando ListThingGroups devuelve una lista de todos los grupos secundarios de un grupo de objetos:

```
{
    "childGroups":[
        {
            "groupName": "RedLights",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RedLights"
        },
        {
            "groupName": "RedLEDLights",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RedLEDLights"
        },
        {
            "groupName": "RedIncandescentLights",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/
RedIncandescentLights"
```

```
        }
    ]
}
```

> **Note**
>
> Esta operación es a largo plazo coherente. En otras palabras, los cambios que se hagan en el grupo de objetos podrían no reflejarse inmediatamente.

Enumerar grupos para un objeto

Puede utilizar el comando ListThingGroupsForThing para listar los grupos directos a los que pertenece una cosa:

```
$ aws iot list-thing-groups-for-thing --thing-name MyLightBulb
```

El comando ListThingGroupsForThing devuelve una lista de los grupos de cosas directos a los que pertenece esta cosa:

```
{
    "thingGroups":[
        {
            "groupName": "LightBulbs",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/LightBulbs"
        },
        {
            "groupName": "RedLights",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RedLights"
        },
        {
            "groupName": "ReplaceableObjects",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/
ReplaceableObjects"
        }
    ]
}
```

También puedes acceder a esta API dentro del motor de reglas mediante la función `get_registry_data()` en línea. Puede utilizar esta función para acceder y utilizar de forma

dinámica la información del registro de objetos (incluidos los atributos, los tipos de elementos y la pertenencia a grupos) mediante llamadas y `ListThingGroupsForThing` APIs directamente dentro de AWS IoT las reglas, lo que permite procesar `DescribeThing` y enrutar los mensajes en tiempo real en función de los datos de registro del dispositivo. Para obtener más información, consulte `get_registry_data`.

Actualizar un grupo de objetos estático

Puede utilizar el comando UpdateThingGroup para actualizar los atributos de un grupo de objetos estático:

```
$ aws iot update-thing-group --thing-group-name "LightBulbs" --thing-group-properties
 "thingGroupDescription=\"this is a test group\", attributePayload=\"{\"attributes
\"={\"Owner\"=\"150\",\"modelNames\"=\"456\"}}"
```

El comando UpdateThingGroup devuelve una respuesta que contiene el número de versión del grupo después de la actualización:

```
{
    "version": 4
}
```

> **Note**
>
> El número de atributos que un objeto puede tener es limitado.

Eliminación de un grupo de objetos

Para eliminar un grupo de objetos, use el comando DeleteThingGroup:

```
$ aws iot delete-thing-group --thing-group-name "RedLights"
```

El comando DeleteThingGroup no genera una salida.

> **Important**
>
> Si intenta eliminar un grupo de objetos que tenga grupos secundarios de objetos, aparecerá un error:

```
A client error (InvalidRequestException) occurred when calling the
 DeleteThingGroup
operation: Cannot delete thing group : RedLights when there are still child
 groups attached to it.
```

Debe eliminar los grupos secundarios antes de eliminar el grupo.

Puede eliminar un grupo que tenga objetos secundarios, pero no se seguirán aplicando los permisos concedidos a los objetos por la pertenencia del grupo. Antes de eliminar un grupo que tenga una política adjunta, compruebe detenidamente que la eliminación de esos permisos no hará que los objetos del grupo no funcionen correctamente. Además, los comandos que muestran a qué grupos pertenece un objeto (por ejemplo, ListGroupsForThing) podrían seguir mostrando el grupo mientras se actualizan los registros en la nube.

Asociar una política a un grupo de objetos estático

Puede usar el comando AttachPolicy para asociar una política a un grupo de objetos estático y, por extensión, a todos los objetos de ese grupo y de sus grupos secundarios:

```
$ aws iot attach-policy \
  --target "arn:aws:iot:us-west-2:123456789012:thinggroup/LightBulbs" \
  --policy-name "myLightBulbPolicy"
```

El comando AttachPolicy no genera una salida

Important

Puede asociar un número máximo de dos políticas a un grupo.

Note

No es recomendable utilizar datos personales en los nombres de política.

El parámetro `--target` puede ser un ARN de grupo de objetos (como más arriba), un ARN del certificado o una identidad de Amazon Cognito. Para obtener más información acerca de las políticas, los certificados y la autenticación, consulte Autenticación.

Para obtener más información, consulte Políticas de AWS IoT Core.

Desconectar una política de un grupo de objetos estático

Puede usar el comando DetachPolicy para separar una política de un grupo de objetos y de esa forma, por extensión, a todos los objetos de ese grupo y a los objetos de cualquiera de sus grupos secundarios:

```
$ aws iot detach-policy --target "arn:aws:iot:us-west-2:123456789012:thinggroup/
LightBulbs" --policy-name "myLightBulbPolicy"
```

El comando DetachPolicy no genera una salida.

Mostrar las políticas asociadas a un grupo de objetos estático

Puede utilizar el comando ListAttachedPolicies para mostrar las políticas asociadas a un grupo de objetos estático:

```
$ aws iot list-attached-policies --target "arn:aws:iot:us-
west-2:123456789012:thinggroup/RedLights"
```

El parámetro `--target` puede ser un ARN de grupo de objetos (como más arriba), un ARN del certificado o una identidad de Amazon Cognito.

Agregue el parámetro `--recursive` opcional para incluir también todas las políticas asociadas a los grupos principales del grupo.

El comando ListAttachedPolicies devuelve una lista de políticas:

```
{
    "policies": [
        "MyLightBulbPolicy"
        ...
    ]
}
```

Enumeración de los grupos para una política

Puede utilizar el comando ListTargetsForPolicy para enumerar los destinos, incluidos los grupos, a los que se adjunta una política:

```
$ aws iot list-targets-for-policy --policy-name "MyLightBulbPolicy"
```

Añada el parámetro `--page-size` *`number`* opcional para especificar el número máximo de resultados que se va a devolver para cada consulta y el parámetro `--marker` *`string`* en las llamadas siguientes para recuperar el siguiente conjunto de resultados, si lo hubiera.

El comando ListTargetsForPolicy devuelve una lista de destinos y el token que usar para recuperar más resultados:

```
{
    "nextMarker": "string",
    "targets": [ "string" ... ]
}
```

Obtención de políticas en vigor para un objeto

Puede usar el comando GetEffectivePolicies para mostrar las políticas en vigor de un objeto, incluidas las políticas asociadas a grupos a los que pertenece el objeto (si el grupo es un grupo principal directo o un antecesor indirecto):

```
$ aws iot get-effective-policies \
  --thing-name "MyLightBulb" \
  --principal "arn:aws:iot:us-east-1:123456789012:cert/
a0c01f5835079de0a7514643d68ef8414ab739a1e94ee4162977b02b12842847"
```

Utilice el parámetro `--principal` para especificar el ARN del certificado adjunto al objeto. Si utiliza la autenticación de identidad de Amazon Cognito, utilice el parámetro `--cognito-identity-pool-id` y, opcionalmente, agregue el parámetro `--principal` para especificar una identidad de Amazon Cognito. Si especifica solo el `--cognito-identity-pool-id`, se devuelven las políticas asociadas a ese rol del grupo de identidades para usuarios sin autenticar. Si usa ambos, se devuelven las políticas asociadas a ese rol del grupo de identidades para los usuarios autenticados.

El parámetro `--thing-name` es opcional y puede usarse en lugar del parámetro `--principal`. Cuando se utiliza, se devolverán las políticas asociadas a cualquier grupo al que pertenece el objeto y las políticas asociadas a grupos principales de estos grupos (hasta el grupo raíz en la jerarquía).

El comando GetEffectivePolicies devuelve una lista de políticas:

```
{
    "effectivePolicies": [
        {
            "policyArn": "string",
            "policyDocument": "string",
            "policyName": "string"
        }
        ...
    ]
}
```

Prueba de autorización para acciones de MQTT

Puede utilizar el comando TestAuthorization para probar si se permite la acción de MQTT (`Publish`, `Subscribe`) para una cosa:

```
aws iot test-authorization \
    --principal "arn:aws:iot:us-east-1:123456789012:cert/
a0c01f5835079de0a7514643d68ef8414ab739a1e94ee4162977b02b12842847" \
    --auth-infos "{\"actionType\": \"PUBLISH\", \"resources\": [ \"arn:aws:iot:us-
east-1:123456789012:topic/my/topic\"]}"
```

Utilice el parámetro `--principal` para especificar el ARN del certificado adjunto al objeto. Si usa la autenticación de identidad de Amazon Cognito, especifique una identidad de Cognito como `--principal` o use el parámetro `--cognito-identity-pool-id`, o ambos. (Si especifica solo `--cognito-identity-pool-id`, se tienen en cuenta las políticas asociadas a ese rol del grupo de identidades para usuarios sin autenticar. Si usa ambos, se tienen en cuenta las políticas asociadas a ese rol del grupo de identidades para los usuarios autenticados.

Especifique una o más acciones de MQTT que desee probar mediante la enumeración de conjuntos de recursos y tipos de acción que siguen al parámetro `--auth-infos`. El campo `actionType` debería contener "PUBLISH", "SUBSCRIBE", "RECEIVE" o "CONNECT". El `resources` campo debe contener una lista de recursos ARNs. Para obtener más información, consulte AWS IoT Core políticas.

Puede probar los efectos de la adición de políticas mediante la especificación de estas con el parámetro `--policy-names-to-add`. También puede probar los efectos de la eliminación de políticas mediante ellas con el parámetro `--policy-names-to-skip`.

Puede usar el parámetro `--client-id` opcional para restringir los resultados.

El comando TestAuthorization devuelve detalles acerca de acciones que se permiten o deniegan para cada conjunto de consultas `--auth-infos` especificadas:

```
{
    "authResults": [
        {
            "allowed": {
                "policies": [
                    {
                        "policyArn": "string",
                        "policyName": "string"
                    }
                ]
            },
            "authDecision": "string",
            "authInfo": {
                "actionType": "string",
                "resources": [ "string" ]
            },
            "denied": {
                "explicitDeny": {
                    "policies": [
                        {
                            "policyArn": "string",
                            "policyName": "string"
                        }
                    ]
                },
                "implicitDeny": {
                    "policies": [
                        {
                            "policyArn": "string",
                            "policyName": "string"
                        }
                    ]
                }
            },
```

```
            "missingContextValues": [ "string" ]
        }
    ]
}
```

Grupos de objetos dinámicos

Los grupos de objetos dinámicos se crean a partir de consultas de búsqueda específicas en el registro. Los parámetros de las consultas de búsqueda, como la conectividad de los dispositivos, la creación de sombras de dispositivos y los datos sobre infracciones de AWS IoT Device Defender , lo respaldan. Los grupos de objetos dinámicos requieren que la indexación de flotas esté activada para indexar, buscar y agregar los datos de sus dispositivos. Puede obtener una vista previa de los objetos en un grupo de objetos dinámicos antes de crearlos realizando una consulta de búsqueda de indexación de flotas. Para obtener más información, consulte Indexación de flotas y Sintaxis de la consulta.

> **Note**
>
> Las operaciones de grupos de objetos dinámicos se miden en operaciones de registro. Para más información, consulte los detalles de medición adicionales de AWS IoT Core.

Los grupos de objetos dinámicos se diferencian de los grupos de objetos estáticos como sigue:

- La pertenencia de un objeto no se define de forma explícita. Para crear un grupo de objetos dinámicos, debe definir una cadena de consulta para determinar la pertenencia a un grupo.
- Los grupos de objetos dinámicos no pueden formar parte de una jerarquía.
- Los grupos de objetos dinámicos no pueden tener políticas aplicadas.
- Puede utilizar un conjunto diferente de comandos para crear, actualizar y eliminar grupos de objetos dinámicos. Para todas las demás operaciones, se utilizan los mismos comandos para ambos tipos de grupos de objetos.
- El número de grupos dinámicos por grupo Cuenta de AWS es limitado.

- No utilice información personal identificable en el nombre del grupo de objetos. El nombre del grupo de cosas puede aparecer en comunicaciones e informes no cifrados.

Para obtener más información acerca de los grupos de objetos estáticos, consulte Grupos de objetos estáticos.

Uso de casos de uso de grupos de objetos dinámicos

Puede utilizar grupos de objetos dinámicos para los siguientes casos de uso:

Especificación de un grupo de objetos dinámico como destino para un trabajo

La creación de un trabajo continuo con un grupo de objetos dinámico como destino le permite segmentar automáticamente los dispositivos si cumplen los criterios deseados. Los criterios pueden ser el estado de conectividad o cualquier criterio almacenado en el registro o en la sombra, como la versión o el modelo del software. Si un objeto no aparece en el grupo de objetos dinámico, no recibirá el documento del trabajo.

Por ejemplo, si su flota de dispositivos requiere una actualización del firmware para minimizar el riesgo de interrupciones durante el proceso de actualización y solo desea actualizar el firmware en dispositivos con una duración de la batería superior al 80 %. Puedes crear un grupo dinámico denominado 80 PercentBatteryLife que solo incluya dispositivos con una duración de batería superior al 80% y utilizarlo como objetivo para tu trabajo. Solo los dispositivos que cumplan con estos criterios de duración de la batería recibirán la actualización del firmware. A medida que los dispositivos alcancen el 80 % de duración de la batería, se irán añadiendo automáticamente al grupo de objetos dinámico y recibirán la actualización del firmware.

También es posible que tenga varios modelos de dispositivos con un firmware o un sistema operativo diferentes, por lo que deberán actualizarse las diferentes versiones del software. Este es el caso de uso más común de los grupos dinámicos con trabajos continuos, donde puede crear un grupo dinámico para cada combinación de modelo de dispositivo, firmware y sistema operativo. A continuación, puede configurar trabajos continuos en cada uno de estos grupos dinámicos para enviar las actualizaciones de software a medida que los dispositivos pasen a ser miembros automáticamente de estos grupos en función de los criterios definidos.

Para obtener más información sobre cómo especificar grupos de objetos como destinos de trabajos, consulte CreateJob.

Uso de los cambios dinámicos de pertenencia al grupo para realizar las acciones deseadas

Cada vez que se añade o elimina un dispositivo de un grupo dinámico, se envía una notificación a un tema de MQTT como parte de las actualizaciones de los eventos del registro. Puede configurar

AWS IoT Core reglas para interactuar con los AWS servicios en función de las actualizaciones dinámicas de pertenencia a los grupos y tomar las medidas que desee. Por ejemplo, escribir a Amazon DynamoDB, invocar una función de Lambda o enviar una notificación a Amazon SNS.

Adición de dispositivos a un grupo de objetos dinámico para detectar automáticamente las infracciones

AWS IoT Device Defender Los clientes de Detect pueden definir un perfil de seguridad en un grupo de cosas dinámico. El perfil de seguridad definido en el grupo detecta automáticamente las infracciones de los dispositivos del grupo de objetos dinámico.

Establecimiento de niveles de registro en grupos de objetos dinámicos para observar los dispositivos con un registro detallado

Puede especificar un nivel de registro en un grupo de objetos dinámico. Esto resulta útil si solo desea personalizar el nivel de registro y los detalles de los dispositivos que cumplen determinados criterios. Por ejemplo, si sospecha que los dispositivos con una determinada versión de firmware están causando errores en un tema publicado en una regla específica, puede configurar un registro detallado para depurar estos problemas. En este caso, puede crear un grupo dinámico para todos los dispositivos que tengan esta versión de firmware, que suponemos que se almacena como atributo de registro o en una sombra de dispositivo. A continuación, puede establecer un nivel de depuración, con el objetivo de registro definido como este grupo de objetos dinámico. Para obtener más información sobre el registro detallado, consulte Supervisar AWS IoT el uso de registros. CloudWatch Para obtener más información sobre cómo especificar un nivel de registro para un grupo de objetos específico, consulte Configure resource-specific logging in AWS IoT.

Creación de un grupo de objetos dinámico

Utilice el comando CreateDynamicThingGroup para crear un grupo de objetos dinámico. Para crear un grupo de cosas dinámico para el PercentBatteryLife escenario 80, utilice el comando create-dynamic-thing-group CLI:

```
$ aws iot create-dynamic-thing-group --thing-group-name "80PercentBatteryLife" --query-string "attributes.batterylife80"
```

> **Note**
>
> No se recomienda utilizar información personal identificable en los nombres de grupos de objetos dinámicos.

El comando CreateDynamicThingGroup no devuelve ninguna respuesta. La respuesta contiene el nombre del índice, la cadena de consulta, la versión de la consulta, el nombre del grupo de objetos y el Nombre de recurso de Amazon (ARN) de su grupo de objetos:

```
{
    "indexName": "AWS_Things",
    "queryVersion": "2017-09-30",
    "thingGroupName": "80PercentBatteryLife",
    "thingGroupArn": "arn:aws:iot:us-
west-2:123456789012:thinggroup/80PercentBatteryLife",
    "queryString": "attributes.batterylife80\n",
    "thingGroupId": "abcdefgh12345678ijklmnop12345678qrstuvwx"
}
```

La creación de grupos de objetos dinámicos no se produce al mismo tiempo. Reponer un grupo de objetos dinámico lleva tiempo. Cuando se crea un grupo de objetos dinámico, el estado del grupo se establece en `BUILDING`. Cuando termina el proceso de reposición, el estado cambia a `ACTIVE`. Para comprobar el estado del grupo de cosas dinámico, utilice el DescribeThingGroupcomando.

Describir un grupo de objetos dinámico

Utilice el comando DescribeThingGroup para obtener información acerca de un grupo de objetos dinámico:

```
$ aws iot describe-thing-group --thing-group-name "80PercentBatteryLife"
```

El comando DescribeThingGroup devuelve información acerca del grupo especificado:

```
{
    "status": "ACTIVE",
    "indexName": "AWS_Things",
    "thingGroupName": "80PercentBatteryLife",
    "thingGroupArn": "arn:aws:iot:us-
west-2:123456789012:thinggroup/80PercentBatteryLife",
```

```
    "queryString": "attributes.batterylife80\n",
    "version": 1,
    "thingGroupMetadata": {
        "creationDate": 1548716921.289
    },
    "thingGroupProperties": {},
    "queryVersion": "2017-09-30",
    "thingGroupId": "84dd9b5b-2b98-4c65-84e4-be0e1ecf4fd8"
}
```

Al ejecutar DescribeThingGroup en un grupo de objetos dinámico se devuelven atributos específicos de los grupos de objetos dinámicos. Algunos ejemplos de atributos de devolución son QueryString y el estado.

El estado de un grupo de objetos dinámico puede tomar los siguientes valores:

`ACTIVE`

El grupo de objetos dinámico está listo para usarse.

`BUILDING`

Se está creando el grupo de objetos dinámico y se está procesando la pertenencia de los objetos.

`REBUILDING`

Se está actualizando la pertenencia al grupo de objetos dinámico tras ajustar la consulta de búsqueda del grupo.

> **Note**
>
> Tras crear un grupo de objetos dinámico ya podrá utilizarlo, independientemente de su estado. Solo los grupos de objetos dinámicos con estado `ACTIVE` incluyen todos los objetos que coinciden con la consulta de búsqueda para ese grupo de objetos dinámico. Los grupos de objetos dinámicos con estado `BUILDING` y `REBUILDING` podrían no incluir todos los objetos coincidentes con la consulta de búsqueda.

Actualizar un grupo de objetos dinámico

Utilice el comando UpdateDynamicThingGroup para actualizar los atributos de un grupo de objetos dinámico, incluida la consulta de búsqueda del grupo. El siguiente comando actualiza dos atributos. Uno es la descripción del grupo de objetos y el otro es la cadena de consulta que cambia los criterios de pertenencia a una vida útil de la batería superior a 85:

```
$ aws iot update-dynamic-thing-group --thing-group-name "80PercentBatteryLife" --thing-
group-properties "thingGroupDescription=\"This thing group contains devices with a
 battery life greater than 85 percent.\"" --query-string "attributes.batterylife85"
```

El comando UpdateDynamicThingGroup devuelve una respuesta que contiene el número de versión del grupo después de la actualización:

```
{
    "version": 2
}
```

La actualización de un grupo de objetos dinámico no se produce simultáneamente. Reponer un grupo de objetos dinámico lleva tiempo. Cuando se actualiza un grupo de objetos dinámico, su estado cambia a `REBUILDING` mientras el grupo actualiza su pertenencia. Cuando termina el proceso de reposición, el estado cambia a `ACTIVE`. Para comprobar el estado del grupo de cosas dinámico, utilice el DescribeThingGroupcomando.

Eliminar un grupo de objetos dinámico

Para eliminar un grupo de objetos dinámico, use el comando DeleteDynamicThingGroup:

```
$ aws iot delete-dynamic-thing-group --thing-group-name "80PercentBatteryLife"
```

El comando DeleteDynamicThingGroup no genera ninguna salida.

Los comandos que muestran a qué grupos pertenece un objeto (por ejemplo, ListGroupsForThing) podrían seguir mostrando el grupo mientras se actualizan los registros en la nube.

Limitaciones de los grupos de objetos dinámicos y estáticos

Los grupos de objetos dinámicos y estáticos comparten las siguientes limitaciones:

- El número de atributos que puede tener un grupo de objetos es limitado.
- El número de grupos a los que puede pertenecer un objeto es limitado.
- No se puede cambiar el nombre de los grupos de objetos.
- Los nombres de grupos de objetos no pueden contener caracteres internacionales, como û, é o ñ.

Limitaciones de los grupos de objetos dinámicos

Los grupos de objetos dinámicos tienen las siguientes limitaciones:

Indexación de flota

Si el servicio de indexación de flotas está activado, puede realizar consultas de búsqueda en la flota de dispositivos. Puede crear y administrar grupos de objetos dinámicos tras completar el proceso de reposición de la indexación de la flota. El tiempo de finalización del proceso de reposición se ve afectado directamente por el tamaño de la flota de dispositivos registrada en la Nube de AWS. Tras habilitar el servicio de indexación de flotas para grupos de objetos dinámicos, no podrá deshabilitarlo hasta que se eliminen todos los grupos de objetos dinámicos.

> **Note**
>
> Si tiene los permisos para consultar el índice de la flota, podrá obtener acceso a los datos de los objetos en toda la flota.

El número de grupos de objetos dinámicos es limitado

El número de grupos de objetos dinámicos es limitado.

Comandos correctos pueden registrar errores

Cuando cree o actualice un grupo de objetos dinámico, es posible que algunos de los objetos se puedan incluir en un grupo de objetos dinámico, pero no se añaden. En ese caso, el comando de creación o actualización se ejecutará correctamente y, al mismo tiempo, registrará un error y generará una métrica `AddThingToDynamicThingGroupsFailed`. Una sola métrica puede representar varias entradas de registro.

Se crea una entrada en el CloudWatch registro de errores cuando ocurre lo siguiente:

- Un objeto apto no se puede agregar a un grupo de objetos dinámico.
- Se elimina un objeto de un grupo de objetos dinámico para añadirlo a otro grupo.

Cuando un objeto ya se pueda añadir a un grupo de objetos dinámico, tenga en cuenta lo siguiente:

- ¿Este objeto ya está en el número máximo de grupos posible? (Consulte los límites)
  - NO: el objeto se agrega al grupo de objetos dinámico.
  - SÍ: ¿el objeto es miembro de algún grupo de objetos dinámico?
    - NO: el objeto no se puede agregar al grupo de objetos dinámicos, se registra un error y se genera una métrica `AddThingToDynamicThingGroupsFailed`.
    - SÍ: ¿el grupo de objetos al que se va a unir es anterior a cualquier otro grupo de objetos dinámico del que el objeto ya sea miembro?
      - NO: el objeto no se puede agregar al grupo de objetos dinámicos, se registra un error y se genera una métrica `AddThingToDynamicThingGroupsFailed`.
      - SÍ: elimine el objeto del grupo de objetos dinámico más reciente, registre un error y agregue el objeto al grupo de objetos dinámico. Esto genera un error y una métrica `AddThingToDynamicThingGroupsFailed` en el grupo de objetos dinámicos del que se ha eliminado el objeto.

Cuando un objeto de un grupo de objetos dinámico ya no satisface la consulta de búsqueda, se elimina del grupo de objetos dinámico. Del mismo modo, cuando un objeto se actualiza para satisfacer una consulta de búsqueda de un grupo de objetos dinámico, se agrega al grupo tal y como se ha descrito anteriormente. Estas incorporaciones y eliminaciones son normales y no generan entradas en el registro de errores.

Con **`overrideDynamicGroups`** habilitado, los grupos estáticos disfrutan de prioridad sobre los grupos dinámicos.

El número de grupos a los que puede pertenecer un objeto es limitado. Cuando se utilizan los UpdateThingGroupsForThingcomandos AddThingToThingGroupo para actualizar la pertenencia a las cosas, al añadir el `--overrideDynamicGroups` parámetro se da prioridad a los grupos de cosas estáticos sobre los grupos de cosas dinámicos.

Si añade un objeto a un grupo de objetos estático, tenga en cuenta lo siguiente:

- ¿El objeto ya es miembro del número máximo de grupos?

- NO: el objeto se agrega al grupo de objetos estático.
- SÍ: ¿el objeto está en algún grupo dinámico?
  - NO: el objeto no se puede agregar al grupo de objetos. El comando genera una excepción.
  - SÍ: ¿se ha habilitado --overrideDynamicGroups?
    - NO: el objeto no se puede agregar al grupo de objetos. El comando genera una excepción.
    - SÍ: el objeto se elimina del último grupo de objetos dinámico que se creó, se registra un error y se genera una métrica `AddThingToDynamicThingGroupsFailed` para el grupo de objetos dinámico del que se eliminó el objeto. A continuación, el objeto se agrega al grupo de objetos estáticos.

Los grupos de objetos dinámicos más antiguos tienen prioridad sobre los más nuevos.

El número de grupos a los que puede pertenecer un objeto es limitado. Cuando una operación de creación o actualización crea un grupo adicional apto para un objeto y el objeto ha alcanzado el límite del grupo, se puede eliminar de otro grupo de objetos dinámico para que se pueda añadir. Si necesita más información sobre esto, consulte Comandos correctos pueden registrar errores y Con `overrideDynamicGroups` habilitado, los grupos estáticos disfrutan de prioridad sobre los grupos dinámicos. para ver ejemplos.

Cuando un objeto se elimina de un grupo de objetos dinámico, se registra un error y se genera un evento.

No se pueden aplicar políticas a grupos de cosas dinámicos

Si intenta aplicar una política a un grupo de objetos dinámico, se generará una excepción.

La pertenencia a grupos de objetos dinámicos es coherente a largo plazo.

Para el registro solo se evalúa el estado final de un objeto. Los estados intermedios se pueden omitir si se actualizan rápidamente los estados. Evite asociar una regla o un trabajo con un grupo de objetos dinámico cuya pertenencia dependa de un estado intermedio.

Asociar cualquier AWS IoT cosa a una conexión de cliente MQTT

Una asociación exclusiva a un objeto se produce cuando se asocia un certificado X.509 a un único objeto de AWS IoT . En este caso, el certificado no se puede usar con otros objetos. Al garantizar

que un certificado solo lo utilice un único objeto de IoT, ayuda a prevenir vulnerabilidades de seguridad.

En AWS IoT, el ID de cliente es un identificador único de una cosa o un dispositivo cuando se conecta al intermediario de AWS IoT Core MQTT. Si utiliza una asociación no exclusiva, se pueden asociar varios objetos al mismo certificado. Si existe una asociación no exclusiva a un objeto, para mantener una asociación clara y evitar posibles conflictos, debe hacer coincidir su ID de cliente con el nombre del objeto.

En este tema

- Casos de uso
- Cómo asociar un objeto a una conexión

Casos de uso

La asociación de un objeto a una conexión ofrece las siguientes funciones.

> **Note**
>
> Tenga en cuenta que si la conexión del objeto de IoT y del cliente tiene una asociación no exclusiva, puede utilizar todas las funciones siguientes, excepto la función de eventos del ciclo de vida. Para incluir el nombre del objeto en los mensajes de los eventos del ciclo de vida, la conexión del objeto de IoT y del cliente debe tener una asociación exclusiva.

Variables de política de cosas: puede utilizar variables de política de cosas para autorizar el acceso del dispositivo a las operaciones de la AWS IoT API. Estas variables te permiten escribir AWS IoT Core políticas que concedan o denieguen permisos en función de las propiedades de los objetos, como los nombres, los tipos y los valores de los atributos. Al utilizar variables de política de cosas, puede aplicar la misma política para controlar varios AWS IoT Core dispositivos. Esto le permite simplificar la administración de políticas y reducir la duplicación de recursos. Para obtener más información, consulte Variables de política de objeto.

Eventos del ciclo de vida: puede recibir el nombre del objeto en los eventos del ciclo de vida (por ejemplo, conectarse, desconectarse, suscribirse y cancelar la suscripción). Esto permite procesar el nombre del objeto incluido en los mensajes, como, por ejemplo, en las reglas. Para obtener más información, consulte Eventos del ciclo de vida.

Registro específico de recursos: puede configurar el registro específico de recursos para grupos de objetos y aplicar fácilmente la configuración del registro deseado a todos los objetos del grupo de objetos definido. Para obtener más información, consulte ???.

Asignación de costos: puede crear grupos de facturación con etiquetas personalizadas para la asignación de costos y agregar los objetos a estos grupos. Para obtener más información, consulte Grupos de facturación.

Cómo asociar un objeto a una conexión

Si tu ID de cliente coincide con el nombre de tu objeto en el registro, después de adjuntar un certificado X.509 a ese elemento de IoT, AWS IoT Core asociará la conexión del cliente al objeto. Si el ID de cliente no coincide con el nombre del objeto en el registro, puede asociar de forma exclusiva un certificado X.509 al objeto para establecer esta asociación. El objeto que tiene esta asociación exclusiva se denomina “objeto exclusivo”. De lo contrario, se denomina “objeto no exclusivo”. Cuando un certificado está asociado a un objeto exclusivo, este certificado solo se puede asociar a otros objetos si lo desasocia del objeto exclusivo. En esta sección, selecciona una opción Consola de administración de AWS o AWS CLI asociarla a una conexión.

Consola de administración de AWS

Cómo asociar un certificado a un objeto de forma exclusiva mediante la Consola de administración de AWS

1. Abre la página de AWS IoT inicio de la AWS IoT consola. En el panel de navegación de la izquierda, en Seguridad, elija Certificados.
2. En la página Certificados, seleccione el certificado al que desee asociar un objeto. A continuación, seleccione Adjuntar a objetos en Acciones en la esquina superior derecha de la página.

   También puede elegir un certificado y acceder a la página de detalles del certificado. Seleccione la pestaña Objetos y, a continuación, elija Adjuntar a objetos.
3. En la página Adjuntar certificado a objetos, marque la casilla de verificación Asociar objeto a una conexión. A continuación, elija el objeto al que desea asociar este certificado en la lista desplegable Objetos.
4. Seleccione Adjuntar objetos. Si la acción se realiza correctamente, aparecerá un banner que indica: “Se ha asociado correctamente un objeto al certificado” y el objeto se agregará a la pestaña Objetos.

Para separar un certificado de un elemento exclusivo mediante el Consola de administración de AWS

1. Abre la página de AWS IoT inicio de la AWS IoT consola. En el panel de navegación de la izquierda, en Seguridad, elija Certificados.
2. En la página Certificados, elija un certificado y diríjase a la página de detalles del certificado.
3. En la página de detalles del certificado, elija la pestaña Objetos. A continuación, seleccione el objeto del que desee desasociar el certificado. Elija Desasociar objetos.
4. En la ventana Desasociar objetos, confirme la acción. Elija Desasociar. Si la acción se realiza correctamente, aparecerá un banner que indica: “Se ha desasociado correctamente un objeto del certificado” y el objeto ya no aparecerá en la pestaña Objetos.

AWS CLI

1. Para adjuntar un certificado a cualquier cosa que utilice AWS CLI, ejecute el attach-thing-principalcomando. Para especificar el certificate-to-thing adjunto exclusivo, debe especificarlo `EXCLUSIVE_THING` en el `--thing-principal-type` campo. El siguiente comando es un ejemplo.

```
aws iot attach-thing-principal \
    --thing-name "thing_1" \
    --principal "arn:aws:iot:us-
east-1:123456789012:cert/
2e1eb273792174ec2b9bf4e9b37e6c6c692345499506002a35159767055278e8" \
    --thing-principal-type "EXCLUSIVE_THING"
```

   Este comando no proporciona ninguna salida. Para obtener más información, consulte ???.
2. Para enumerar los objetos asociados al certificado especificado, junto con el tipo de asociación, ejecute el comando `list-principal-things-v2`. El tipo de asociación hace referencia a cómo se asocia el certificado al objeto. El siguiente comando es un ejemplo.

```
$ aws iot list-principal-things-v2 \
    --principal "arn:aws:iot:us-
east-1:123456789012:cert/
2e1eb273792174ec2b9bf4e9b37e6c6c692345499506002a35159767055278e8"
```

   El resultado puede ser similar al siguiente.

```
{
```

```
    "PrincipalThingObjects": [
        {
            "thingPrincipalType": "EXCLUSIVE_THING",
            "thing": "arn:aws:iot:us-east-1:123456789012:thing/thing_1"
        }
    ]
}
```

Para obtener más información, consulte ???.

3. Para enumerar las entidades principales asociadas al objeto especificado, junto con el tipo de asociación, ejecute el comando `list-thing-principals-v2`. El tipo de asociación hace referencia a cómo se asocia el certificado al objeto. El siguiente comando es un ejemplo.

```
$ aws iot list-thing-principals-v2 \
    --thing-name "thing_1"
```

El resultado puede ser similar al siguiente.

```
{
    "ThingPrincipalObjects": [
        {
            "thingPrincipalType": "EXCLUSIVE_THING",
            "principal": "arn:aws:iot:us-
east-1:123456789012:cert/
2e1eb273792174ec2b9bf4e9b37e6c6c692345499506002a35159767055278e8"
        }
    ]
}
```

Para obtener más información, consulte ???.

4. Para separar un certificado de un objeto, ejecute el detach-thing-principalcomando.

```
aws iot detach-thing-principal \
    --principal "arn:aws:iot:us-
east-1:123456789012:cert/
2e1eb273792174ec2b9bf4e9b37e6c6c692345499506002a35159767055278e8" \
    --thing-name "thing_1"
```

Este comando no proporciona ninguna salida. Para obtener más información, consulte ???.

Agregación de atributos de propagación para enriquecer los mensajes

En él AWS IoT Core, puede enriquecer los mensajes MQTT de los dispositivos añadiendo atributos de propagación, que son metadatos contextuales a partir de atributos de objetos o detalles de conexión. Este proceso, conocido como enriquecimiento de mensajes, puede resultar útil en varios escenarios. Por ejemplo, puede enriquecer los mensajes para cada operación de publicación entrante sin realizar cambios en el dispositivo ni tener que usar reglas. Al sacar partido de los atributos de propagación, puede beneficiarse de una forma más eficiente y rentable de enriquecer los datos de IoT sin las complejidades de configurar reglas ni administrar las configuraciones de republicación.

La función de enriquecimiento de mensajes está disponible para AWS IoT Core los clientes que utilizan Basic Ingest and Message Broker. Es importante tener en cuenta que, si bien los dispositivos de publicación pueden utilizar cualquier versión de MQTT, los suscriptores (aplicaciones o servicios que consumen mensajes) deben ser compatibles con MQTT 5 para recibir los mensajes enriquecidos con atributos de propagación. Los mensajes enriquecidos se agregarán como propiedades de usuario de MQTT 5 a todos los mensajes que se publiquen desde los dispositivos. Si usa reglas, puede sacar partido de la función get_user_properties para recuperar los datos enriquecidos para el enrutamiento o procesamiento de los mensajes en función de los datos.

En AWS IoT Core, puede añadir atributos de propagación al crear o actualizar un tipo de cosa, utilizando el o el Consola de administración de AWS . AWS CLI

> **Important**
>
> Al agregar atributos de propagación, debe asegurarse de que el cliente que publica el mensaje se haya autenticado con un certificado. Para obtener más información, consulte Autenticación del cliente.

> **Note**
>
> Si intenta probar esta característica con el cliente de prueba de MQTT en la consola, es posible que no funcione, ya que esta característica requiere que los clientes de MQTT se hayan autenticado con un certificado asociado.

Consola de administración de AWS

Para añadir atributos de propagación para el enriquecimiento de los mensajes mediante el Consola de administración de AWS

1. Abra la página de AWS IoT inicio en la AWS IoT consola. En el panel de navegación de la izquierda, en Administrar, seleccione Todos los dispositivos. A continuación, seleccione Tipos de objetos.
2. En la página Tipos de objetos, seleccione Crear tipo de objeto.

   Para configurar el enriquecimiento de mensajes mediante la actualización de un tipo de objeto, elija un tipo de objeto. A continuación, en la página de detalles del tipo de objeto, seleccione Actualizar.
3. En la página Crear tipo de objeto, seleccione o introduzca la información del tipo de objeto en Propiedades del tipo de objeto.

   Si decide actualizar un tipo de objeto, aparecerá la opción Propiedades del tipo de objeto después de seleccionar Actualizar en el paso anterior.
4. En Configuración adicional, amplíe la opción Atributos de propagación. A continuación, elija el atributo de cosa e introduzca el atributo de cosa que desee rellenar en los MQTT5 mensajes publicados. Mediante la consola, puede agregar hasta tres atributos de objeto.

   En la sección Atributos de propagación, elija Atributo de conexión e introduzca el tipo de atributo y, si lo desea, su nombre.
5. Como opción, agregue etiquetas. A continuación, elija Crear tipo de objeto.

   Si decide actualizar un tipo de objeto, seleccione Actualizar tipo de objeto.

AWS CLI

1. Para añadir atributos de propagación para enriquecer los mensajes mediante la creación de un nuevo tipo de elemento mediante el AWS CLI, ejecute el comando. create-thing-type El siguiente comando es un ejemplo.

```
aws iot create-thing-type \
    --thing-type-name "LightBulb" \
```

```
    --thing-type-properties "{\"mqtt5Configuration\":{\"propagatingAttributes\":
[{\"userPropertyKey\":\"iot:ClientId\", \"connectionAttribute\":\"iot:ClientId\"},
 {\"userPropertyKey\":\"test\", \"thingAttribute\":\"A\"}]}}" \
```

El resultado del comando puede tener un aspecto similar al siguiente.

```
{
 "thingTypeName": "LightBulb",
 "thingTypeArn": "arn:aws:iot:us-west-2:123456789012:thingtype/LightBulb",
 "thingTypeId": "ce3573b0-0a3c-45a7-ac93-4e0ce14cd190"
}
```

2. Para configurar el enriquecimiento de mensajes mediante la actualización de un tipo de cosa mediante AWS CLI, ejecute el comando. update-thing-type Tenga en cuenta que solo puede actualizar `mqtt5Configuration` al ejecutar este comando. El siguiente comando es un ejemplo.

```
aws iot update-thing-type \
    --thing-type-name "MyThingType" \
    --thing-type-properties "{\"mqtt5Configuration\":{\"propagatingAttributes\":
[{\"userPropertyKey\":\"iot:ClientId\", \"connectionAttribute\":\"iot:ClientId\"},
 {\"userPropertyKey\":\"test\", \"thingAttribute\":\"A\"}]}}" \
```

Este comando no proporciona ninguna salida.

3. Para describir un tipo de objeto, ejecute el comando `describe-thing-type`. Este comando generará un resultado con la información de configuración del enriquecimiento de mensajes en el campo `thing-type-properties`. El siguiente comando es un ejemplo.

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

El resultado puede ser similar al siguiente.

```
{
 "thingTypeName": "LightBulb",
 "thingTypeId": "bdf72512-0116-4392-8d79-bf39b17ef73d",
 "thingTypeArn": "arn:aws:iot:us-east-1:123456789012:thingtype/LightBulb",
 "thingTypeProperties": {
  "mqtt5Configuration": {
   "propagatingAttributes": [
    {
```

```
      "userPropertyKey": "iot:ClientId",
      "connectionAttribute": "iot:ClientId"
     },
     {
      "userPropertyKey": "test",
      "thingAttribute": "attribute"
     }
    ]
   }
  },
  "thingTypeMetadata": {
   "deprecated": false,
   "creationDate": "2024-10-18T17:37:46.656000+00:00"
  }
}
```

Para obtener más información, consulte ???.

Etiquetar sus recursos AWS IoT

Para administrar y organizar más fácilmente grupos de objetos, tipos de objetos, reglas de temas, trabajos, auditorías programadas y perfiles de seguridad, dispone de la opción de asignar sus propios metadatos en forma de etiquetas a cada uno de estos recursos. En esta sección se describe qué son las etiquetas y cómo crearlas.

Para ayudarle a administrar los costos relacionados con los objetos puede crear grupos de facturación con objetos. A continuación, puede asignar etiquetas con sus metadatos a cada uno de estos grupos de facturación. En esta sección se explican también los grupos de facturación y los comandos disponibles para crearlos y administrarlos.

Conceptos básicos de etiquetas

Puede usar etiquetas para clasificar AWS IoT los recursos de diferentes maneras (por ejemplo, por propósito, propietario o entorno). Esto resulta útil cuando tiene muchos recursos del mismo tipo, ya que le permite identificar rápidamente un recurso específico utilizando las etiquetas asignadas. Cada etiqueta está formada por una clave y un valor opcional, ambos definidos por el usuario. Por ejemplo, puede definir un conjunto de etiquetas para los tipos de objetos de modo que le resulte más fácil hacer un seguimiento de los dispositivos con arreglo al tipo al que pertenecen. Le recomendamos que cree un conjunto de claves de etiqueta que cumpla sus necesidades para cada tipo de recurso. Mediante el uso de un conjunto coherente de claves de etiquetas, podrá administrar los recursos de más fácilmente.

Puede buscar y filtrar sus recursos en función de las etiquetas que añada o aplique. También puede utilizar etiquetas de grupo de facturación para categorizar y realizar un seguimiento de los costos. También puede utilizar etiquetas para controlar el acceso a los recursos según se describe en Uso de etiquetas con políticas de IAM.

Para facilitar su uso, el editor de etiquetas de la consola AWS de administración proporciona una forma centralizada y unificada de crear y administrar las etiquetas. Para obtener más información, consulte Trabajar con el editor de etiquetas en Trabajar con la consola AWS de administración.

También puede trabajar con etiquetas mediante la API AWS CLI y la AWS IoT API. Puede asociar etiquetas a grupos de objetos, tipos de objetos, reglas de temas, trabajos, perfiles de seguridad, políticas, grupos de facturación y las versiones y los paquetes asociados a los objetos cuando los cree; para ello, utilice el campo `Tags` en los siguientes comandos:

- CreateBillingGroup
- CreateDestination
- CreateDeviceProfile
- CreateDynamicThingGroup
- CreateJob
- CrearOTAUpdate
- CreatePolicy
- CreateScheduledAudit
- CreateSecurityProfile
- CreateServiceProfile
- CreateStream
- CreateThingGroup
- CreateThingType
- CreateTopicRule
- CreateWirelessGateway
- CreateWirelessDevice

Puede añadir, modificar o eliminar etiquetas para recursos existentes que admitan el uso de etiquetas con los siguientes comandos:

- TagResource
- ListTagsForResource
- UntagResource

Puedes editar las claves y los valores de las etiquetas y eliminar las etiquetas de un recurso en cualquier momento. Puede establecer el valor de una etiqueta como una cadena vacía, pero no puede asignarle un valor nulo. Si agrega una etiqueta que tiene la misma clave que una etiqueta existente en ese recurso, el nuevo valor sobrescribe el valor anterior. Si elimina un recurso, también se eliminarán todas las etiquetas que este tenga asociadas.

Restricciones y limitaciones en las etiquetas

Se aplican las siguientes restricciones básicas a las etiquetas de :

- Número máximo de etiquetas por recurso: 50
- Longitud máxima de la clave: 127 caracteres Unicode en UTF-8
- Longitud máxima del valor: 255 caracteres Unicode en UTF-8
- Las claves y los valores de las etiquetas distinguen entre mayúsculas y minúsculas.
- No utilice el prefijo `aws:` en los nombres o valores de las etiquetas. Está reservado para su AWS uso. Los nombres y valores de etiquetas que tienen este prefijo no se pueden editar ni eliminar. Las etiquetas que tengan este prefijo no cuentan para el límite de etiquetas por recurso.
- Si se utiliza su esquema de etiquetado en múltiples servicios y recursos de , recuerde que otros servicios podrían tener otras restricciones sobre caracteres permitidos. Los caracteres permitidos son: letras, espacios y números representables en UTF-8, además de los siguientes caracteres especiales: + - = . _ : / @.

Uso de etiquetas con políticas de IAM

Puede aplicar permisos de nivel de recurso basados en etiquetas en las políticas de IAM que utiliza con las acciones de la API de AWS IoT . Esto le ofrece un mejor control sobre los recursos que un usuario puede crear, modificar o utilizar. Puede utilizar el elemento `Condition` (también llamado bloque `Condition`) junto con las siguientes claves contextuales de condición y valores en una política de IAM para controlar el acceso del usuario (permiso) en función de las etiquetas de un usuario:

- Utilice `aws:ResourceTag/`*`tag-key`*`: `*`tag-value`* para permitir o denegar acciones de los usuarios en recursos con etiquetas específicas.
- Utilice `aws:RequestTag/`*`tag-key`*`: `*`tag-value`* para exigir (o impedir) el uso de una etiqueta específica al realizar una solicitud de API para crear o modificar un recurso que permita etiquetas.
- Utilice `aws:TagKeys: [`*`tag-key`*`, ...]` para exigir (o impedir) el uso de un conjunto de claves de etiquetas al realizar una solicitud de API para crear o modificar un recurso que permita etiquetas.

> **Note**
>
> Las condiciones, las claves y los valores contextuales de una política de IAM solo se aplican a aquellas AWS IoT acciones en las que el identificador de un recurso que se pueda etiquetar es un parámetro obligatorio. Por ejemplo, no DescribeEndpointse permite ni se deniega el uso de debido a las claves y valores del contexto de la condición, ya que en

esta solicitud no se hace referencia a ningún recurso etiquetable (grupos de cosas, tipos de cosas, reglas temáticas, trabajos o perfiles de seguridad). Para obtener más información sobre AWS IoT los recursos que se pueden etiquetar y las claves de condición que admiten, consulte Acciones, recursos y claves de condición. AWS IoT

Para obtener más información sobre el uso de etiquetas, consulte Control de acceso a los recursos de AWS mediante etiquetas, en la Guía del usuario de AWS Identity and Access Management . La sección de referencia de políticas JSON de IAM de esta guía incluye sintaxis, descripciones y ejemplos detallados de los elementos, variables y lógica de evaluación de las políticas JSON de IAM.

La siguiente política de ejemplo aplica dos restricciones basadas en etiquetas para las acciones de `ThingGroup`. Un usuario de IAM restringido por esta política:

- No se le puede dar a un grupo de objetos la etiqueta “env=prod” (en el ejemplo, consulte la línea `"aws:RequestTag/env" : "prod"`).
- No se puede modificar un grupo de objetos que tenga la etiqueta “env=prod”, y tampoco se puede acceder a dicho grupo (en el ejemplo, consulte la línea `"aws:ResourceTag/env" : "prod"`).

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "iot:CreateThingGroup",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/env": "prod"
        }
      }
    },
    {
      "Effect": "Deny",
      "Action": [
        "iot:CreateThingGroup",
        "iot:DeleteThingGroup",
        "iot:DescribeThingGroup",
        "iot:UpdateThingGroup"
```

```
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/env": "prod"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:CreateThingGroup",
        "iot:DeleteThingGroup",
        "iot:DescribeThingGroup",
        "iot:UpdateThingGroup"
      ],
      "Resource": "*"
    }
  ]
}
```

También puede especificar varios valores de etiqueta para una determinada clave de etiqueta encerrándola en una lista, tal y como se muestra a continuación:

```
          "StringEquals" : {
            "aws:ResourceTag/env" : ["dev", "test"]
          }
```

> **Note**
>
> Si permite o deniega a los usuarios acceso a recursos en función de etiquetas, debe considerar denegar explícitamente a los usuarios la posibilidad de agregar estas etiquetas o retirarlas de los mismos recursos. De lo contrario, es posible que un usuario eluda sus restricciones y obtenga acceso a un recurso modificando sus etiquetas.

Grupos de facturación

AWS IoT no te permite aplicar etiquetas directamente a cosas individuales, pero sí te permite colocar cosas en grupos de facturación y aplicarles etiquetas. AWS IoT En efecto, la asignación de los datos de coste y uso en función de las etiquetas se limita a los grupos de facturación.

AWS IoT Core ya que los recursos de LoRa WAN, como dispositivos inalámbricos y puertas de enlace, no se pueden agregar a los grupos de facturación. Sin embargo, se pueden asociar a AWS IoT cosas, que se pueden añadir a los grupos de facturación.

Dispone de los siguientes comandos:

- AddThingToBillingGroup añade un objeto a un grupo de facturación.
- CreateBillingGroup crea un grupo de facturación.
- DeleteBillingGroup elimina el grupo de facturación.
- DescribeBillingGroup devuelve información sobre un grupo de facturación.
- ListBillingGroups lista los grupos de facturación que ha creado.
- ListThingsInBillingGroup lista los objetos que ha agregado al grupo de facturación en cuestión.
- RemoveThingFromBillingGroup elimina el objeto en cuestión del grupo de facturación.
- UpdateBillingGroup actualiza información sobre el grupo de facturación.
- CreateThing le permite especificar un grupo de facturación para el objeto al crearlo.
- DescribeThing devuelve la descripción de un objeto incluido el grupo de facturación al que pertenece el objeto, si lo hay.

La API AWS IoT inalámbrica proporciona estas acciones para asociar dispositivos inalámbricos y pasarelas a AWS IoT cosas.

- AssociateWirelessDeviceWithThing
- AssociateWirelessGatewayWithThing

Visualización de datos de uso y asignación de costos

Puede utilizar etiquetas de grupo de facturación para categorizar y realizar un seguimiento de los costos. Cuando aplicas etiquetas a los grupos de facturación (y, por lo tanto, a las cosas que incluyen), AWS genera un informe de asignación de costes en forma de archivo de valores

separados por comas (CSV) con el uso y los costes agregados por etiquetas. Puede aplicar etiquetas que representen categorías de negocio (p. ej., centros de costos, nombres de aplicación o propietarios) para estructurar los costos entre diferentes servicios. Para obtener más información sobre el uso de etiquetas para la asignación de costes, consulte Uso de etiquetas de asignación de costos de AWS, en la guía de usuario ¿Qué es AWS Billing and Cost Management?

> **Note**
>
> Para asociar los datos de uso y costos de forma precisa con los objetos que ha colocado en grupos de facturación, cada dispositivo o cada aplicación debe:
>
> - Estar registrado como una cosa en. AWS IoT Para obtener más información, consulte Administrar dispositivos con AWS IoT.
> - Conéctese al agente de AWS IoT mensajes a través de MQTT utilizando solo el nombre de la cosa como ID de cliente. Para obtener más información, consulte the section called "Protocolos de comunicación de dispositivos". Si el ID de cliente no coincide con el nombre del objeto, puede habilitar la asociación de objeto exclusiva para establecer dicha asociación. Para obtener más información, consulte ???.
> - Autenticarse mediante un certificado de cliente asociado al objeto.

Dispone de las dimensiones de precios para los grupos de facturación (en función de la actividad de los objetos asociados al grupo de facturación):

- Conectividad (en función del nombre del objeto utilizado como ID de cliente para conectarse)
- Mensajes (en función de la entrada de mensajes procedentes de un objeto y la salida de mensajes destinados a un objeto; solo MQTT)
- Operaciones de sombra (en función del objeto cuyo mensaje activó una actualización de sombra)
- Reglas activadas (en función del objeto cuyo mensaje entrante activó la regla; no se aplica a las reglas activadas por los eventos del ciclo de vida de MQTT)
- Actualizaciones de índices de objetos (en función del objeto que se haya agregado al índice)
- Acciones remotas (en función del objeto actualizado)
- Informes de AWS IoT Device Defender (en función del objeto cuya actividad se notifica).

Los datos de costo y uso basados en etiquetas (y notificados para un grupo de facturación) no reflejan las siguientes actividades:

- Operaciones de registro de dispositivos (incluidas las actualizaciones de objetos, grupos de objetos y tipos de objetos). Para obtener más información, consulte Administrar dispositivos con AWS IoT).
- Actualizaciones de índices de grupos de objetos (al agregar un grupo de objetos)
- Consultas de búsqueda en índices.
- Aprovisionamiento de dispositivos.
- Informes de auditoría de AWS IoT Device Defender.

Seguridad en AWS IoT

La seguridad en la nube AWS es la máxima prioridad. Como AWS cliente, usted se beneficia de una arquitectura de centro de datos y red diseñada para cumplir con los requisitos de las organizaciones más sensibles a la seguridad.

La seguridad es una responsabilidad compartida entre usted AWS y usted. El modelo de responsabilidad compartida la describe como seguridad de la nube y seguridad en la nube:

- Seguridad de la nube: AWS es responsable de proteger la infraestructura que ejecuta AWS los servicios en la AWS nube. AWS también le proporciona servicios que puede utilizar de forma segura. Auditores independientes prueban y verifican periódicamente la eficacia de nuestra seguridad en el marco de los programas de conformidad de AWS. Para obtener más información sobre los programas de conformidad aplicables AWS IoT, consulte los AWS servicios incluidos en el ámbito de aplicación por programa de conformidad.
- Seguridad en la nube: su responsabilidad viene determinada por el AWS servicio que utilice. También es responsable de otros factores, incluida la confidencialidad de los datos, los requisitos de la empresa y la legislación y los reglamentos vigentes.

Esta documentación le ayuda a comprender cómo aplicar el modelo de responsabilidad compartida cuando se utiliza AWS IoT. Los siguientes temas muestran cómo configurarlo AWS IoT para cumplir sus objetivos de seguridad y conformidad. También aprenderá a utilizar otros AWS servicios que le ayudan a supervisar y proteger sus AWS IoT recursos.

Temas

- AWS IoT seguridad
- Autenticación
- Autorización
- Protección de datos en AWS IoT Core
- Gestión de identidad y acceso para AWS IoT
- Registro y supervisión
- Validación del cumplimiento de AWS IoT Core
- La resiliencia en el núcleo del AWS IoT
- Uso AWS IoT Core con puntos finales de VPC de interfaz

- Seguridad de la infraestructura en AWS IoT
- Supervisión de la seguridad de las flotas o dispositivos de producción con Core AWS IoT
- Mejores prácticas de seguridad en AWS IoT Core
- AWS formación y certificación

AWS IoT seguridad

Cada dispositivo o cliente conectado debe tener una credencial con la que interactuar con AWS IoT. Todo el tráfico entrante y saliente AWS IoT se envía de forma segura a través de Transport Layer Security (TLS). AWS Los mecanismos de seguridad en la nube protegen los datos cuando se mueven entre otros AWS servicios AWS IoT y otros.



- Es responsable de administrar las credenciales del dispositivo (certificados X.509, credenciales de AWS , identidades de Amazon Cognito, identidades federadas o tokens de autenticación personalizados) y las políticas de AWS IoT. También es el responsable de asignar identidades exclusivas a cada uno de los dispositivos y de administrar los permisos de cada dispositivo o un grupo de dispositivos.
- Sus dispositivos se conectan AWS IoT mediante certificados X.509 o identidades de Amazon Cognito a través de una conexión TLS segura. Durante la investigación y el desarrollo, y en el caso de algunas aplicaciones que utilizan o utilizan la API WebSockets, también puede autenticarse mediante usuarios y grupos de IAM o mediante tokens de autenticación personalizados. Para obtener más información, consulte Usuarios, grupos y roles de IAM.

- Al utilizar la AWS IoT autenticación, el agente de mensajes es responsable de autenticar tus dispositivos, ingerir de forma segura los datos de los dispositivos y conceder o denegar los permisos de acceso que especifiques para tus dispositivos mediante políticas. AWS IoT
- Al utilizar la autenticación personalizada, un autorizador personalizado se encarga de autenticar tus dispositivos y de conceder o denegar los permisos de acceso que especifiques para tus dispositivos mediante AWS IoT políticas de IAM.
- El motor de AWS IoT reglas reenvía los datos de los dispositivos a otros dispositivos u otros AWS servicios de acuerdo con las reglas que usted defina. Se utiliza AWS Identity and Access Management para transferir datos de forma segura a su destino final. Para obtener más información, consulte Gestión de identidad y acceso para AWS IoT.

Autenticación

La autenticación es un mecanismo en el que se verifica la identidad de un cliente o un servidor. La autenticación del servidor es el proceso en el que los dispositivos u otros clientes se aseguran de que se están comunicando con un AWS IoT punto final real. La autenticación de clientes es el proceso mediante el cual los dispositivos u otros clientes se autentican. AWS IoT

Información general del certificado 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 de CA, que utiliza para generar certificados X.509. Solo la autoridad de certificación tiene acceso a los certificados de CA. Las cadenas de certificados X.509 se utilizan tanto para la autenticación del servidor por parte de los clientes como para la autenticación del cliente por parte del servidor.

Autenticación del servidor

Cuando el dispositivo u otro cliente intente conectarse AWS IoT Core, el AWS IoT Core servidor enviará un certificado X.509 que el dispositivo utilizará para autenticar el servidor. La autenticación se lleva a cabo en la capa TLS mediante la validación de la cadena de certificados X.509. Este es el mismo método que utiliza el navegador cuando visita una URL HTTPS. Si desea utilizar certificados de su propia autoridad de certificación, consulte Administración de sus certificados de entidad de certificación.

Cuando sus dispositivos u otros clientes establecen una conexión TLS con un AWS IoT Core terminal, AWS IoT Core presenta una cadena de certificados que los dispositivos utilizan para comprobar que se están comunicando con otro servidor AWS IoT Core y no con otro servidor que se hace pasar por él. AWS IoT Core La cadena que se presenta depende de una combinación del tipo de terminal al que se conecta el dispositivo y del conjunto de cifrado que el cliente y el cliente AWS IoT Core negociaron durante el protocolo de enlace TLS.

Tipo de punto de conexión

AWS IoT Core admite. `iot:Data-ATS` `iot:Data-ATS`los puntos finales presentan un certificado de servidor firmado por una CA de Amazon Trust Services.

Los certificados presentados por los puntos de conexión de ATS están firmados por Starfield. Algunas implementaciones de cliente TLS requieren la validación de la raíz de confianza y requieren que los certificados de CA de Starfield estén instalados en los almacenes de confianza del cliente.

> **Warning**
>
> No se recomienda utilizar un método de fijación de certificados que aplica hash en todo el certificado (incluido el nombre del emisor, etc.) porque esto provocará un error en la verificación del certificado porque los certificados ATS que proporcionamos están firmados de forma cruzada por Starfield y tienen un nombre de emisor diferente.

> **Important**
>
> Utilice puntos de conexión `iot:Data-ATS`. Los certificados de Symantec y Verisign han quedado obsoletos y ya no son compatibles con. AWS IoT Core

Puede utilizar el comando `describe-endpoint` para crear el punto de conexión de ATS.

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

El comando `describe-endpoint` devuelve un punto de conexión en el formato siguiente.

```
account-specific-prefix.iot.your-region.amazonaws.com
```

> **Note**
>
> La primera vez que se llama a `describe-endpoint`, se crea un punto de conexión. Todas las llamadas posteriores a `describe-endpoint` devuelven el mismo punto de conexión.

> **Note**
>
> Para ver su **`iot:Data-ATS`** terminal en la AWS IoT Core consola, seleccione Configuración. La consola solo muestra el punto de conexión `iot:Data-ATS`.

Creación de un **`IotDataPlaneClient`** con el AWS SDK para Java

Para crear un `IotDataPlaneClient` que utilice un punto de conexión `iot:Data-ATS`, debe hacer lo siguiente.

- Cree un `iot:Data-ATS` punto final mediante la DescribeEndpointAPI.
- Especifique ese punto de conexión al crear el `IotDataPlaneClient`.

En el ejemplo siguiente se realizan ambas operaciones.

```
public void setup() throws Exception {
        IotClient client =
 IotClient.builder().credentialsProvider(CREDENTIALS_PROVIDER_CHAIN).region(Region.US_EAST_1).b
        String endpoint = client.describeEndpoint(r -> r.endpointType("iot:Data-
ATS")).endpointAddress();
        iot = IotDataPlaneClient.builder()
                                .credentialsProvider(CREDENTIALS_PROVIDER_CHAIN)
                                .endpointOverride(URI.create("https://" + endpoint))
                                .region(Region.US_EAST_1)
                                .build();
}
```

Certificados de entidad de certificación para autenticación de servidor

Según el tipo de punto final de datos que utilice y el conjunto de cifrado que haya negociado, los certificados de autenticación AWS IoT Core del servidor se firman con uno de los siguientes certificados de CA raíz:

Puntos de enlace de Amazon Trust Services (preferidos)

> **Note**
>
> Es posible que tenga que hacer clic con el botón derecho en estos enlaces y seleccionar Guardar enlace como... para guardar estos certificados como archivos.

- Clave RSA de 2048 bits: Amazon Root CA 1.
- Clave RSA de 4096 bits: Amazon Root CA 2. Reservado para uso futuro.
- Clave ECC de 256 bits: Amazon Root CA 3.
- Clave ECC de 384 bits: Amazon Root CA 4. Reservado para uso futuro.

Todos estos certificados tienen firma cruzada del Certificado Starfield Root CA. Todas AWS IoT Core las nuevas regiones, a partir del lanzamiento del 9 de mayo de 2018 AWS IoT Core en la región de Asia Pacífico (Bombay), solo ofrecen certificados ATS.

VeriSign Endpoints (heredado)

- Clave RSA de 2048 bits: certificado de CA raíz G5 primaria pública de VeriSign clase 3

Directrices de autenticación de servidores

Hay muchas variables que pueden afectar a la capacidad de un dispositivo para validar el certificado de autenticación del servidor de AWS IoT Core . Por ejemplo, los dispositivos pueden tener demasiada memoria limitada para contener todos los certificados de CA raíz posibles, o los dispositivos pueden implementar un método no estándar de validación de certificados. Por estas razones, sugerimos seguir estas directrices:

- Le recomendamos que utilice el punto de conexión ATS e instale todos los certificados de CA raíz de Amazon Root CA compatibles.
- Si no puede almacenar todos estos certificados en su dispositivo y si sus dispositivos no utilizan la validación basada en ECC, puede omitir los certificados Amazon Root CA 3 y Amazon Root CA 4 de ECC. Si sus dispositivos no implementan una validación de certificados basada en RSA, puede omitir los certificados Amazon Root CA 1 y Amazon Root CA 2 de RSA. Es posible que tenga que hacer clic con el botón derecho en estos enlaces y seleccionar Guardar enlace como... para guardar estos certificados como archivos.

- Si tiene problemas de validación de certificados de servidor al conectarse a su punto de conexión de ATS, intente agregar el certificado Amazon Root CA correspondiente con firma cruzada a su almacén de confianza. Es posible que tenga que hacer clic con el botón derecho en estos enlaces y seleccionar Guardar enlace como... para guardar estos certificados como archivos.
  - Amazon Root CA 1 con firma cruzada
  - Amazon Root CA 2 con firma cruzada: reservado para futura utilización.
  - Amazon Root CA 3 con firma cruzada
  - Amazon Root CA 4 con firma cruzada: reservado para futura utilización.
- Si experimenta problemas de validación de certificados de servidor, es posible que el dispositivo deba confiar explícitamente en la CA raíz. Intente agregar el Starfield Root CA Certificate a su almacén de confianza.
- Si sigue teniendo problemas después de ejecutar los pasos anteriores, póngase en contacto con AWS Developer Support.

> **Note**
>
> Los certificados de CA tienen una fecha de vencimiento posterior que no pueden usar para validar un certificado del servidor. Los certificados de CA podrían tener que reemplazarse antes de su fecha de vencimiento. Asegúrese de que puede actualizar los certificados de entidad de certificación raíz en todos sus dispositivos o clientes para asegurarse de que la conectividad se mantenga y esté al día de las prácticas recomendadas de seguridad.

> **Note**
>
> Cuando se conecte con AWS IoT Core el código de su dispositivo, pase el certificado a la API que está utilizando para conectarse. La API que use variará según el SDK. Para obtener más información, consulta el AWS IoT Core dispositivo SDKs.

Autenticación del cliente

AWS IoT admite tres tipos de principios de identidad para la autenticación de dispositivos o clientes:

- Certificados de cliente X.509

- Usuarios, grupos y roles de IAM
- Identidades de Amazon Cognito

Estas identidades se pueden usar con dispositivos, aplicaciones móviles, web o de escritorio. Incluso los puede usar un usuario que escribe AWS IoT comandos de la interfaz de línea de comandos (CLI). Por lo general, AWS IoT los dispositivos utilizan certificados X.509, mientras que las aplicaciones móviles utilizan las identidades de Amazon Cognito. Las aplicaciones web y de escritorio usan identidades federadas o de IAM. Los comandos de la AWS CLI utilizan IAM. Para obtener más información acerca de las identidades de IAM, consulte Gestión de identidad y acceso para AWS IoT.

Certificados de cliente X.509

Los certificados X.509 permiten AWS IoT autenticar las conexiones de clientes y dispositivos. Los certificados de cliente deben estar registrados AWS IoT antes de que un cliente pueda comunicarse con ellos. AWS IoT Un certificado de cliente se puede registrar en varios Cuenta de AWS s de la misma región Región de AWS para facilitar el traslado de dispositivos entre Cuenta de AWS los de la misma región. Para obtener más información, consulte Uso de certificados de cliente X.509 en varios Cuenta de AWS s con registro de varias cuentas.

Se recomienda que cada dispositivo o cliente reciba un certificado único para permitir acciones de administración de clientes precisas, incluida la revocación de certificados. Los dispositivos y los clientes deben ser compatibles con la rotación y la sustitución de certificados para garantizar un buen funcionamiento cuando los certificados caduquen.

Para obtener más información sobre el uso de certificados X.509 con el fin de admitir más de unos pocos dispositivos, consulte Aprovisionamiento de dispositivos para revisar las distintas opciones de aprovisionamiento y administración de certificados que admite AWS IoT .

AWS IoT admite los siguientes tipos de certificados de cliente X.509:

- Certificados X.509 generados por AWS IoT
- Certificados X.509 firmados por una entidad emisora de certificados registrada en. AWS IoT
- Certificados X.509 firmados por una entidad de certificación que no está registrada con AWS IoT.

En esta sección se describe cómo administrar certificados X.509 en AWS IoT. Puede utilizar la AWS IoT consola o AWS CLI realizar las siguientes operaciones de certificación:

- Cree certificados de AWS IoT cliente

- Creación de sus propios certificados de cliente
- Registrar un certificado de cliente
- Activar o desactivar un certificado de cliente
- Revocar un certificado de cliente

Para obtener más información sobre los AWS CLI comandos que realizan estas operaciones, consulte la referencia de AWS IoT CLI.

Uso de certificados de cliente X.509

Los certificados X.509 autentican las conexiones del cliente y del dispositivo a. AWS IoT Los certificados X.509 ofrecen varios beneficios con respecto a otros mecanismos de identificación y autenticación. Los certificados X.509 permiten usar claves asimétricas con los dispositivos. Por ejemplo, podría forzar claves privadas en un almacenamiento seguro en un dispositivo para que el material criptográfico confidencial nunca salga del dispositivo. Los certificados X.509 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 privada jamás abandona el dispositivo.

AWS IoT autentica los certificados de cliente mediante el modo de autenticación de cliente del protocolo TLS. La compatibilidad con TLS está disponible en numerosos lenguajes de programación y sistemas operativos, y se utiliza generalmente para cifrar datos. En la autenticación de clientes TLS, AWS IoT solicita un certificado de cliente X.509 y valida el estado del certificado y lo Cuenta de AWS compara con un registro de certificados. A continuación, pide al cliente que demuestre que es propietario de la clave privada que corresponde a la clave pública contenida en el certificado. AWS IoT exige que los clientes envíen la extensión de indicación del nombre del servidor (SNI) al protocolo Transport Layer Security (TLS). Para obtener más información sobre la configuración de la extensión SNI, consulte Seguridad del transporte en AWS IoT Core.

Para facilitar una conexión segura y coherente del cliente con el AWS IoT núcleo, un certificado de cliente X.509 debe tener lo siguiente:

- Registrado en AWS IoT Core. Para obtener más información, consulte Registrar un certificado de cliente.
- Tener un estado `ACTIVE`. Para obtener más información, consulte Activar o desactivar un certificado de cliente.
- No haber alcanzado aún la fecha de caducidad del certificado.

Puede crear certificados de cliente que utilicen la entidad de certificación Amazon Root y puede utilizar sus propios certificados de cliente firmados por otra entidad de certificación (CA). Para obtener más información sobre el uso de la AWS IoT consola para crear certificados que usen la CA raíz de Amazon, consulteCree certificados de AWS IoT cliente. Para obtener más información sobre el uso de sus propios certificados X.509, consulte Creación de sus propios certificados de cliente.

La fecha y hora de caducidad de los certificados firmados por un certificado de entidad de certificación se establecen en el momento de su creación. Los certificados X.509 generados por AWS IoT vencen a medianoche (UTC) del 31 de diciembre de 2049 (2049-12-31T 23:59:59 Z).

AWS IoT Device Defender puede realizar auditorías en sus dispositivos Cuenta de AWS y en sus dispositivos, de acuerdo con las mejores prácticas comunes de seguridad de IoT. Esto incluye administrar las fechas de caducidad de los certificados X.509 firmados por su entidad de certificación o la entidad de certificación de Amazon Root. Para obtener más información sobre la administración de la fecha de vencimiento de un certificado, consulte El certificado de dispositivo está caducando y El certificado de entidad de certificación está caducando.

En el AWS IoT blog oficial, se profundiza en la gestión de la rotación de certificados de dispositivos y las mejores prácticas de seguridad en Cómo gestionar la rotación de certificados de dispositivos de IoT mediante el uso AWS IoT.

Uso de certificados de cliente X.509 en varios Cuenta de AWS s con registro de varias cuentas

El registro de varias cuentas permite mover dispositivos entre sus Cuenta de AWS en la misma región o en regiones diferentes. Puede registrar, probar y configurar un dispositivo en una cuenta de preproducción y, a continuación, registrar y utilizar el mismo dispositivo y certificado de dispositivo en una cuenta de producción. También puede registrar el certificado de cliente en el dispositivo o los certificados del dispositivo sin una CA en la que esté registrada. AWS IoT Para obtener más información, consulte Registrar un certificado de cliente firmado por una entidad de certificación no registrada (CLI).

> **Note**
>
> Los certificados utilizados para el registro de varias cuentas son compatibles con los tipos de puntos de conexión `iot:Data-ATS`, `iot:Data` (heredados), `iot:Jobs` y `iot:CredentialProvider`. Para obtener más información sobre los puntos finales de los AWS IoT dispositivos, consulteAWS IoT datos del dispositivo y puntos finales de servicio.

Los dispositivos que utilizan el registro de varias cuentas deben enviar la extensión de indicación del nombre del servidor (SNI) al protocolo Transport Layer Security (TLS) y proporcionar la dirección completa del punto final en el `host_name` campo cuando se conecten a. AWS IoT AWS IoT utiliza la dirección del punto final `host_name` para enrutar la conexión a la cuenta correcta. AWS IoT Los dispositivos existentes que no envíen una dirección de punto de conexión válida en `host_name` seguirán funcionando, pero no podrán utilizar las características que requiere esta información. Para obtener más información acerca de la extensión SNI y para aprender a identificar la dirección del punto de conexión del campo `host_name`, consulte Seguridad del transporte en AWS IoT Core.

Para utilizar el registro de varias cuentas

1. Puede registrar los certificados de dispositivo con una entidad de certificación. Puede registrar la CA emisora de certificados en varias cuentas en modo `SNI_ONLY` y utilizar esa entidad de certificación para registrar el mismo certificado de cliente en varias cuentas. Para obtener más información, consulte Registro de un certificado de entidad de certificación en modo SNI_ONLY (CLI) - Recomendado.
2. Puede registrar los certificados de dispositivo sin una entidad de certificación. Consulte Registro de un certificado de cliente firmado por una entidad de certificación no registrada (CLI). El registro de una entidad de certificación es opcional. No es necesario que registre la CA con la que firmó los certificados del dispositivo AWS IoT.

Los algoritmos de firma de certificados son compatibles con AWS IoT

AWS IoT admite los siguientes algoritmos de firma de certificados:

- SHA256CON RSA
- SHA384CON RSA
- SHA512CON RSA
- SHA256WITHRSAANDMGF1 (RSASSA-PSS)
- SHA384WITHRSAANDMGF1 (RSASSA-PSS)
- SHA512WITHRSAANDMGF1 (RSASSA-PSS)
- DSA_CON_ SHA256
- ECDSA-CON- SHA256
- ECDSA-CON- SHA384
- ECDSA-CON- SHA512

Para obtener más información acerca de la autenticación y la seguridad de los certificados, consulte Calidad de la clave del certificado del dispositivoy.

> **Note**
>
> La solicitud de firma de certificado (CSR) debe incluir una clave pública. Puede tratarse de una clave RSA con una longitud de al menos 2048 bits o una clave ECC de curvas NIST P-256, NIST P-384 o NIST P-521. Para obtener más información, consulte CreateCertificateFromCsr en la Guía de referencia de la API de AWS IoT .

Algoritmos clave compatibles con AWS IoT

En la siguiente tabla se muestra cómo se admiten los algoritmos de clave:

Algoritmo clave	Algoritmo de firma de certificados	Versión de TLS	¿Se admite? Yes o No
RSA con un tamaño de clave de al menos 2048 bits	Todos	TLS 1.2 y TLS 1.3	Sí
ECC NIST P-256/P-384/ P-521	Todos	TLS 1.2 y TLS 1.3	Sí
RSA-PSS con un tamaño de clave de al menos 2048 bits	Todos	TLS 1.2	No
RSA-PSS con un tamaño de clave de al menos 2048 bits	Todos	TLS 1.3	Sí

Para crear un certificado mediante la CreateCertificateFromCSR, puede utilizar un algoritmo de clave compatible para generar una clave pública para su CSR. Para registrar su propio certificado mediante RegisterCertificateuna RegisterCertificateWithoutCA, puede utilizar un algoritmo de clave compatible para generar una clave pública para el certificado.

Para obtener más información, consulte Security policies.

Cree certificados de AWS IoT cliente

AWS IoT proporciona certificados de cliente firmados por la autoridad de certificación (CA) raíz de Amazon.

En este tema se describe cómo crear un certificado de cliente firmado por la entidad de certificación Amazon Root y descargar los archivos de certificado. Después de crear los archivos de certificado de cliente, debe instalarlos en el cliente.

> **Note**
>
> Cada certificado de cliente X.509 proporcionado por él AWS IoT contiene los atributos de emisor y sujeto que usted estableció en el momento de la creación del certificado. Los atributos del certificado solo son inmutables una vez creado el certificado.

Puede usar la AWS IoT consola o la AWS CLI para crear un AWS IoT certificado firmado por la autoridad de certificación raíz de Amazon.

Cree un AWS IoT certificado (consola)

Para crear un AWS IoT certificado mediante la AWS IoT consola

1. Inicie sesión en la AWS IoT consola Consola de administración de AWS y ábrala.
2. En el panel de navegación, elija Seguridad, Certificados y, a continuación, elija Crear.
3. Elija One-click certificate creation (recommended) [Creación de un certificado con un clic (recomendado)] - Create certificate (Crear certificado).
4. En la página Certificado creado, descargue los archivos de certificado del cliente para el objeto, la clave pública y la clave privada en una ubicación segura. Estos certificados generados por solo AWS IoT están disponibles para su uso con AWS IoT los servicios.

   Si necesita además el archivo de certificado de entidad de certificación de Amazon Root, esta página también tiene el enlace a la página desde la que puede descargarlo.
5. Ahora se ha creado y registrado un certificado de cliente con AWS IoT. Debe activar el certificado antes de usarlo en un cliente.

   Elija Activar para activar el certificado de cliente ahora. Si no desea activar el certificado ahora, consulte Activar un certificado de cliente (consola) para saber cómo activarlo más adelante.

6. Si desea asociar una política al certificado, elija Asociar una política.

   Si no desea asociar una política ahora, elija Hecho para finalizar. Puede asociar una política más adelante.

Después de completar el procedimiento, instale los archivos de certificado en el cliente.

Crear un AWS IoT certificado (CLI)

AWS CLI Proporciona el create-keys-and-certificate comando para crear certificados de cliente firmados por la autoridad de certificación raíz de Amazon. No obstante, este comando no descarga el archivo de certificado de entidad de certificación de Amazon Root. Puede descargar el archivo de certificado de entidad de certificación de Amazon Root desde Certificados de entidad de certificación para autenticación de servidor.

Este comando crea archivos de claves privadas, claves públicas y certificados X.509, y registra y activa el certificado con AWS IoT ellos.

```
aws iot create-keys-and-certificate \
    --set-as-active \
    --certificate-pem-outfile certificate_filename.pem \
    --public-key-outfile public_filename.key \
    --private-key-outfile private_filename.key
```

Si no desea activar el certificado al crearlo y registrarlo, este comando crea archivos de clave privada, clave pública y certificado X.509 y registra el certificado, pero no lo activa. Activar un certificado de cliente (CLI) describe cómo activar el certificado más adelante.

```
aws iot create-keys-and-certificate \
    --no-set-as-active \
    --certificate-pem-outfile certificate_filename.pem \
    --public-key-outfile public_filename.key \
    --private-key-outfile private_filename.key
```

Instale los archivos de certificado en el cliente.

Creación de sus propios certificados de cliente

AWS IoT admite certificados de cliente firmados por cualquier autoridad de certificación (CA) raíz o intermedia. AWS IoT utiliza los certificados de CA para comprobar la propiedad de los certificados. Para utilizar certificados de dispositivo firmados por una entidad emisora de certificados que no sea la entidad de certificación de Amazon, el certificado de la entidad emisora debe estar registrado en ella AWS IoT para que podamos comprobar la propiedad del certificado del dispositivo.

AWS IoT admite varias formas de traer tus propios certificados (BYOC):

- En primer lugar, registre la entidad de certificación que se utiliza para firmar los certificados de los clientes y, a continuación, registre los certificados de los clientes por separado. Si desea registrar el dispositivo o el cliente en su certificado de cliente cuando se conecte por primera vez AWS IoT (también conocido como aprovisionamiento justo a tiempo), debe registrar la CA firmante AWS IoT y activar el registro automático.
- Si no puede registrar la CA emisora de certificados, puede optar por registrar los certificados de cliente sin la CA. En el caso de los dispositivos registrados sin una CA, tendrá que presentar la indicación de nombre del servidor (SNI) al conectarlos a AWS IoT.

> **Note**
>
> Para registrar los certificados de cliente mediante una CA, debe registrar la CA firmante y no con ninguna otra AWS IoT CAs entidad jerárquica.

> **Note**
>
> Un certificado de entidad de certificación se puede registrar en modo `DEFAULT` solo mediante una cuenta en una región. Un certificado de entidad de certificación se puede registrar en modo `SNI_ONLY` mediante varias cuentas en una región.

Para obtener más información acerca del uso de certificados X.509 con el fin de admitir más de unos pocos dispositivos, consulte Aprovisionamiento de dispositivos para revisar las diferentes opciones de administración y aprovisionamiento de certificados que admite AWS IoT .

Temas

- Administración de sus certificados de entidad de certificación

- Crear un certificado de cliente mediante el certificado de entidad de certificación

Administración de sus certificados de entidad de certificación

En esta sección se describen las tareas comunes para administrar sus propios certificados de entidad de certificación.

Puede registrar su entidad emisora de certificados (CA) AWS IoT si utiliza certificados de cliente firmados por una entidad emisora de certificados que AWS IoT no los reconoce.

Si desea que los clientes registren automáticamente sus certificados de cliente AWS IoT cuando se conecten por primera vez, la CA que firmó los certificados de cliente debe estar registrada AWS IoT. De lo contrario, no es necesario registrar el certificado de entidad de certificación que firmó los certificados de cliente.

> **Note**
>
> Un certificado de entidad de certificación se puede registrar en modo `DEFAULT` solo mediante una cuenta en una región. Un certificado de entidad de certificación se puede registrar en modo `SNI_ONLY` mediante varias cuentas en una región.

Temas:

- Creación de un certificado de entidad de certificación
- Registro del certificado CA
- Desactivar un certificado de entidad de certificación

Creación de un certificado de entidad de certificación

Si no dispone de certificado de entidad de certificación, puede crear uno con las herramientas de OpenSSL v1.1.1i.

> **Note**
>
> No puede realizar este procedimiento en la AWS IoT consola.

Para crear un certificado de entidad de certificación con las herramientas de OpenSSL v1.1.1i

1. Genere un par de claves.

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

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

```
openssl req -x509 -new -nodes \
    -key root_CA_key_filename.key \
    -sha256 -days 1024 \
    -out root_CA_cert_filename.pem
```

Registro del certificado CA

Estos procedimientos describen cómo registrar un certificado de una autoridad de certificación (CA) que no es la CA de Amazon. AWS IoT Core utiliza certificados de CA para verificar la propiedad de los certificados. Para usar certificados de dispositivo firmados por una CA que no sea la CA de Amazon, debes registrar el certificado de CA con el AWS IoT Core fin de comprobar la propiedad del certificado del dispositivo.

Registro de un certificado de entidad de certificación (consola).

> **Note**
>
> Para registrar un certificado de entidad de certificación en la consola, comience en la consola en Registro de un certificado de entidad de certificación. Puede registrar su entidad emisora de certificados en el modo multicuenta y sin necesidad de proporcionar un certificado de verificación ni de acceder a la clave privada. Una entidad emisora de certificados se puede registrar en el modo multicuenta mediante varias Cuentas de AWS en la misma Región de AWS. Puede registrar su entidad emisora de certificados en el modo de cuenta única proporcionando un certificado de verificación y una prueba de propiedad de la clave privada de la entidad emisora de certificados.

Registro de un certificado de entidad de certificación (CLI)

Puede registrar un certificado de entidad de certificación en el modo `DEFAULT` o el modo `SNI_ONLY`. Una CA se puede registrar en `DEFAULT` modo uno Cuenta de AWS a uno Región de AWS. Una

CA se puede registrar en `SNI_ONLY` modo varias veces Cuentas de AWS en la misma modalidad Región de AWS. Para obtener más información acerca del modo de certificados de CA, consulte certificateMode.

> **Note**
>
> Se recomienda registrar una entidad emisora de certificados en el modo `SNI_ONLY`. No necesita proporcionar un certificado de verificación ni acceder a la clave privada, y puede registrar la CA varias veces Cuentas de AWS en la misma Región de AWS.

Registro de un certificado de entidad de certificación en modo SNI_ONLY (CLI) - Recomendado

Requisitos previos

Asegúrese de que dispone de lo siguiente en el ordenador antes de continuar:

- El archivo de certificado de entidad de certificación raíz (al que se hace referencia en el siguiente ejemplo como *`root_CA_cert_filename.pem`*).
- OpenSSL v1.1.1i o versiones posteriores.

Para registrar un certificado de CA en **`SNI_ONLY`** modo mediante el AWS CLI

1. Registre el certificado de CA con AWS IoT. Con el comando register-ca-certificate, introduzca el nombre del archivo del certificado de entidad de certificación. Para obtener más información, consulte register-ca-certificate en la Referencia de comandos de la AWS CLI .

   ```
   aws iot register-ca-certificate \
       --ca-certificate file://root_CA_cert_filename.pem \
       --certificate-mode SNI_ONLY
   ```

   Si se ejecuta correctamente, este comando devuelve el*`certificateId`*.
2. En este momento, el certificado de CA se ha registrado AWS IoT pero está inactivo. El certificado de entidad de certificación debe estar activo antes de poder registrar los certificados de cliente firmados por él.

   Este paso activa el certificado de entidad de certificación.

Para activar el certificado de entidad de certificación, utilice el comando update-certificate del modo siguiente. Para obtener más información, consulte update-certificate en la Referencia de comandos de la AWS CLI .

```
aws iot update-ca-certificate \
    --certificate-id certificateId \
    --new-status ACTIVE
```

Utilice el comando describe-ca-certificate para ver el estado del certificado de entidad de certificación. Para obtener más información, consulte describe-ca-certificate en la Referencia de comandos de la AWS CLI .

Registro de un certificado de entidad de certificación en modo **DEFAULT** (CLI)

Requisitos previos

Asegúrese de que dispone de lo siguiente en el ordenador antes de continuar:

- El archivo de certificado de entidad de certificación raíz (al que se hace referencia en el siguiente ejemplo como *root_CA_cert_filename.pem*).
- El archivo de clave privada del certificado de entidad de certificación raíz (al que se hace referencia en el siguiente ejemplo como *root_CA_key_filename.key*).
- OpenSSL v1.1.1i o versiones posteriores.

Para registrar un certificado de CA en **DEFAULT** modo mediante el AWS CLI

1. Para obtener un código de registro AWS IoT, utiliceget-registration-code. Guarde el `registrationCode` devuelto para usarlo como `Common Name` del certificado de verificación de clave privada. Para obtener más información, consulte get-registration-code en la Referencia de comandos de la AWS CLI .

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

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

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

3. Cree una solicitud de firma de certificado (CSR) para el certificado de verificación de clave privada. Establezca el campo `Common Name` del certificado en el `registrationCode` devuelto por get-registration-code.

```
openssl req -new \
    -key verification_cert_key_filename.key \
    -out verification_cert_csr_filename.csr
```

Se le solicita información, incluido el `Common Name` del certificado.

```
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
    State or Province Name (full name) []:
    Locality Name (for example, city) []:
    Organization Name (for example, company) []:
    Organizational Unit Name (for example, section) []:
    Common Name (e.g. server FQDN or YOUR name) []:your_registration_code
    Email Address []:

    Please enter the following 'extra' attributes
    to be sent with your certificate request
    A challenge password []:
    An optional company name []:
```

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

```
openssl x509 -req \
    -in verification_cert_csr_filename.csr \
    -CA root_CA_cert_filename.pem \
    -CAkey root_CA_key_filename.key \
    -CAcreateserial \
    -out verification_cert_filename.pem \
    -days 500 -sha256
```

5. Registre el certificado de CA con AWS IoT. Pase el nombre de archivo del certificado de entidad de certificación y el nombre de archivo del certificado de verificación de clave privada al

comando register-ca-certificate, tal como sigue: Para obtener más información, consulte register-ca-certificate en la Referencia de comandos de la AWS CLI .

```
aws iot register-ca-certificate \
    --ca-certificate file://root_CA_cert_filename.pem \
    --verification-cert file://verification_cert_filename.pem
```

Este comando devuelve el*certificateId*, si tiene éxito.

6. En este momento, el certificado de CA se ha registrado AWS IoT pero no está activo. El certificado de entidad de certificación debe estar activo antes de poder registrar los certificados de cliente firmados por él.

   Este paso activa el certificado de entidad de certificación.

   Para activar el certificado de entidad de certificación, utilice el comando update-certificate del modo siguiente. Para obtener más información, consulte update-certificate en la Referencia de comandos de la AWS CLI .

   ```
   aws iot update-ca-certificate \
       --certificate-id certificateId \
       --new-status ACTIVE
   ```

Utilice el comando describe-ca-certificate para ver el estado del certificado de entidad de certificación. Para obtener más información, consulte describe-ca-certificate en la Referencia de comandos de la AWS CLI .

Creación un certificado de verificación de CA para registrarlo en la consola

> **Note**
>
> Este procedimiento solo se utiliza si va a registrar un certificado de CA desde la AWS IoT consola.
> Si no ha realizado este procedimiento desde la AWS IoT consola, inicie el proceso de registro del certificado de CA en la consola en Registrar el certificado de CA.

Asegúrese de que dispone de lo siguiente en el mismo ordenador antes de continuar:

- El archivo de certificado de entidad de certificación raíz (al que se hace referencia en el siguiente ejemplo como *root_CA_cert_filename.pem*).
- El archivo de clave privada del certificado de entidad de certificación raíz (al que se hace referencia en el siguiente ejemplo como *root_CA_key_filename.key*).
- OpenSSL v1.1.1i o versiones posteriores.

Para usar la interfaz de línea de comandos con el fin de crear un certificado de verificación de CA para registrar su certificado de entidad de certificación en la consola:

1. Reemplace *verification_cert_key_filename.key* por el nombre del archivo de clave del certificado de verificación que quiera crear (por ejemplo, **verification_cert.key**). Luego ejecute este comando para generar un par de claves para el certificado de verificación de clave privada:

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

2. Reemplace *verification_cert_key_filename.key* por el nombre del archivo de clave que creó en el paso 1.

   Reemplace *verification_cert_csr_filename.csr* por el nombre del archivo de solicitud de firma de certificado (CSR) que quiera crear. Por ejemplo, **verification_cert.csr**.

   Ejecute el comando para crear el archivo CSR.

   ```
   openssl req -new \
       -key verification_cert_key_filename.key \
       -out verification_cert_csr_filename.csr
   ```

   El comando le solicita información adicional que se explica más adelante.
3. En la AWS IoT consola, en el contenedor del certificado de verificación, copie el código de registro.
4. La información que le solicita el comando openssl se muestra en el siguiente ejemplo. A excepción del campo `Common Name`, puede introducir sus propios valores o mantenerlos vacíos.

   En el campo `Common Name`, pegue el código de registro que copió en el paso anterior.

   ```
   You are about to be asked to enter information that will be incorporated
   into your certificate request.
   ```

```
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
    State or Province Name (full name) []:
    Locality Name (for example, city) []:
    Organization Name (for example, company) []:
    Organizational Unit Name (for example, section) []:
    Common Name (e.g. server FQDN or YOUR name) []:your_registration_code
    Email Address []:

    Please enter the following 'extra' attributes
    to be sent with your certificate request
    A challenge password []:
    An optional company name []:
```

Al terminar, el comando crea el archivo CSR.

5. Reemplace *verification_cert_csr_filename.csr* por el *verification_cert_csr_filename.csr* que utilizó en el paso anterior.

   Reemplace *root_CA_cert_filename.pem* por el nombre del archivo de del certificado de entidad de certificación que quiera registrar.

   Reemplace *root_CA_key_filename.key* por el nombre del archivo de clave privada del certificado de entidad de certificación.

   Reemplace *verification_cert_filename.pem* por el nombre del archivo de clave del certificado de verificación que quiera crear. Por ejemplo, **verification_cert.pem**.

   ```
   openssl x509 -req \
       -in verification_cert_csr_filename.csr \
       -CA root_CA_cert_filename.pem \
       -CAkey root_CA_key_filename.key \
       -CAcreateserial \
       -out verification_cert_filename.pem \
       -days 500 -sha256
   ```

6. Cuando se complete el comando OpenSSL, debería tener estos archivos listos para usarlos cuando regrese a la consola.

- Su archivo de certificado de entidad de certificación (*root_CA_cert_filename.pem* utilizado en el comando anterior).
- El certificado de verificación que creó en el paso anterior (*verification_cert_filename.pem*utilizado en el comando anterior)

Desactivar un certificado de entidad de certificación

Cuando un certificado de una entidad emisora de certificados (CA) está habilitado para el registro automático de certificados de cliente, AWS IoT comprueba el certificado de la CA para asegurarse de que la CA lo estáACTIVE. Si el certificado de CA lo estáINACTIVE, AWS IoT no permite registrar el certificado de cliente.

Al establecer el certificado de entidad de certificación como INACTIVE, impide que los certificados de cliente nuevos emitidos por la entidad de certificación se registren automáticamente.

> **Note**
>
> Todos los certificados de dispositivo registrados que haya firmado el certificado de entidad de certificación en riesgo siguen funcionando hasta que revoque explícitamente cada uno de ellos.

Desactivar un certificado de entidad de certificación (consola)

Para desactivar un certificado de CA mediante la consola AWS IoT

1. Inicie sesión en la AWS IoT consola Consola de administración de AWS y ábrala.
2. En el panel de navegación izquierdo, selecciona Seguro y elige CAs.
3. En la lista de autoridades de certificación, busque la que desea desactivar y elija el icono de los puntos suspensivos para abrir el menú de opciones.
4. En el menú de opciones, elija Deactivate (Desactivar).

La entidad de certificación debe mostrarse como Inactive (Inactiva) en la lista.

> **Note**
>
> La AWS IoT consola no proporciona una forma de enumerar los certificados firmados por la entidad emisora de certificados que ha desactivado. Para obtener una opción de AWS CLI para enumerar esos certificados, consulte Desactivar un certificado de entidad de certificación (CLI).

Desactivar un certificado de entidad de certificación (CLI)

AWS CLI Proporciona el update-ca-certificatecomando para desactivar un certificado de CA.

```
aws iot update-ca-certificate \
    --certificate-id certificateId \
    --new-status INACTIVE
```

Utilice el comando list-certificates-by-ca para obtener una lista de todos los certificados de cliente registrados firmados por la entidad de certificación especificada. Por cada certificado de cliente firmado por el certificado de entidad de certificación especificado, puede utilizar el comando update-certificate para revocar el certificado de cliente y evitar que este se use.

Utilice el comando describe-ca-certificate para ver el estado del certificado de entidad de certificación.

Crear un certificado de cliente mediante el certificado de entidad de certificación

Puede utilizar su propia entidad de certificación (CA) para crear certificados de cliente. El certificado de cliente debe estar registrado en él AWS IoT antes de usarlo. Para obtener información acerca de las opciones de registro de los certificados de cliente, consulte Registrar un certificado de cliente.

Crear un certificado de cliente (CLI)

> **Note**
>
> No puede realizar este procedimiento en la AWS IoT consola.

Para crear un certificado de cliente mediante AWS CLI

1. Genere un par de claves.

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

2. Cree una CSR para el certificado de cliente.

```
openssl req -new \
    -key device_cert_key_filename.key \
    -out device_cert_csr_filename.csr
```

Se le solicita que indique información, tal y como se muestra aquí:

```
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
    State or Province Name (full name) []:
    Locality Name (for example, city) []:
    Organization Name (for example, company) []:
    Organizational Unit Name (for example, section) []:
    Common Name (e.g. server FQDN or YOUR name) []:
    Email Address []:

    Please enter the following 'extra' attributes
    to be sent with your certificate request
    A challenge password []:
    An optional company name []:
```

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

```
openssl x509 -req \
    -in device_cert_csr_filename.csr \
    -CA root_CA_cert_filename.pem \
    -CAkey root_CA_key_filename.key \
    -CAcreateserial \
    -out device_cert_filename.pem \
    -days 500 -sha256
```

En este momento, se ha creado el certificado de cliente, pero aún no se ha registrado en él AWS IoT. Para obtener información acerca de cómo y cuándo registrar el certificado de cliente, consulte Registrar un certificado de cliente.

Registrar un certificado de cliente

Los certificados de cliente deben estar registrados AWS IoT para permitir las comunicaciones entre el cliente y AWS IoT. Puede registrar cada certificado de cliente manualmente o puede configurar los certificados de cliente para que se registren automáticamente cuando el cliente se conecte AWS IoT por primera vez.

Si desea que los clientes y dispositivos registren sus certificados de cliente cuando se conecten por primera vez, debe usar Registro del certificado CA para firmar el certificado de cliente con AWS IoT en las regiones en las que desea usarlo. La CA Amazon Root se registra automáticamente en AWS IoT.

Los certificados de cliente se pueden compartir Cuentas de AWS entre distintas regiones. Los procedimientos de estos temas deben realizarse en cada cuenta y región en la que desee utilizar el certificado de cliente. El registro de un certificado de cliente en una cuenta o región no es reconocido automáticamente por otra.

> **Note**
>
> Los clientes que utilizan el protocolo Transport Layer Security (TLS) para conectarse a AWS IoT deben admitir la extensión de indicación de nombre de servidor (SNI) para TLS. Para obtener más información, consulte Seguridad del transporte en AWS IoT Core.

Temas

- Registrar manualmente un certificado de cliente
- Registre un certificado de cliente cuando el cliente se conecte al AWS IoT just-in-time registro (JITR)

Registrar manualmente un certificado de cliente

Puede registrar un certificado de cliente manualmente mediante la AWS IoT consola y AWS CLI.

El procedimiento de registro que se utilice depende de si el certificado será compartido por Cuenta de AWS s y Regions. El registro de un certificado de cliente en una cuenta o región no es reconocido automáticamente por otra.

Los procedimientos de este tema deben realizarse en cada cuenta y región en la que desee utilizar el certificado de cliente. Los certificados de cliente se pueden compartir entre Cuenta de AWS sí y entre regiones.

Registro manual un certificado de cliente firmado por una entidad de certificación registrada (consola)

> **Note**
>
> Antes de realizar este procedimiento, asegúrese de tener el archivo.pem del certificado de cliente y de que el certificado de cliente ha sido firmado por una entidad emisora de certificados en la que se haya registrado. AWS IoT

Para registrar un certificado existente AWS IoT mediante la consola

1. Inicie sesión en la consola AWS de administración y abra la AWS IoT consola.
2. En el panel de navegación, en la sección Administrar, elija Seguridad y luego Certificados.
3. En la página Certificados del cuadro de diálogo Certificados, seleccione Agregar certificado y, a continuación, seleccione Registrar certificados.
4. En la página Registrar certificado del cuadro de diálogo Certificados para cargar, haga lo siguiente:

   - Seleccione La CA está registrada en AWS IoT.
   - En Elija un certificado de CA, seleccione su autoridad de certificación.
     - Seleccione Registrar una nueva CA para registrar una nueva autoridad de certificación en la que no esté registrada en AWS IoT.
     - Deje en blanco la opción Elija un certificado de CA si su autoridad de certificación es la autoridad de certificación Amazon Root.
   - Seleccione un máximo de 10 certificados para cargarlos y registrarlos AWS IoT.
     - Utilice los archivos de certificado que creó en Cree certificados de AWS IoT cliente y Crear un certificado de cliente mediante el certificado de entidad de certificación.
   - Elija Activar o Desactivar. Si selecciona Desactivar, Activar o desactivar un certificado de cliente explica cómo activar el certificado después de registrarlo.

- Elija Registrar.

En la página Certificados en el cuadro de diálogo Certificados, ahora aparecerán los certificados registrados.

Registro manual un certificado de cliente firmado por una entidad de certificación no registrada (consola)

> **Note**
>
> Antes de realizar este procedimiento, asegúrese de que tiene el archivo .pem del certificado de cliente.

Para registrar un certificado existente AWS IoT mediante la consola

1. Inicie sesión en la consola AWS de administración y abra la AWS IoT consola.
2. En el panel de navegación izquierdo, elija Security (Seguridad), elija Certificates (Certificados) y, a continuación, elija Crear.
3. En Crear un certificado, busque la entrada Usar mi certificado y elija Comenzar.
4. En Seleccionar una entidad de certificación, elija Siguiente.
5. En Registrar certificados de dispositivo existentes, elija Seleccionar certificados y seleccione hasta 10 archivos de certificado para registrar.
6. Después de cerrar el cuadro de diálogo de archivo, seleccione si desea activar o revocar los certificados de cliente cuando los registre.

   Si no activa un certificado cuando se registra, Activar un certificado de cliente (consola) describe cómo activarlo más adelante.

   Si un certificado se revoca cuando se registra, no se puede activar más tarde.

   Después de elegir los archivos de certificado que desea registrar y seleccionar las acciones que desea realizar después del registro, elija Registrar certificados.

Los certificados de cliente registrados correctamente aparecen en la lista de certificados.

Registro de un certificado de cliente firmado por una entidad de certificación registrada (CLI)

> **Note**
>
> Antes de realizar este procedimiento, asegúrese de que tiene el archivo .pem de la entidad de certificación y el archivo .pem del certificado de cliente. El certificado de cliente debe estar firmado por una entidad de certificación (CA) en la que se haya registrado AWS IoT.

Utilice el comando register-certificate para registrar, pero no activar, un certificado de cliente.

```
aws iot register-certificate \
    --certificate-pem file://device_cert_filename.pem \
    --ca-certificate-pem file://ca_cert_filename.pem
```

El certificado de cliente está registrado AWS IoT, pero aún no está activo. Consulte Activar un certificado de cliente (CLI) para obtener más información sobre cómo activarlo más adelante.

También puede activar el certificado de cliente cuando lo registre utilizando este comando.

```
aws iot register-certificate \
    --set-as-active \
    --certificate-pem file://device_cert_filename.pem \
    --ca-certificate-pem file://ca_cert_filename.pem
```

Para obtener más información sobre cómo activar el certificado para poder usarlo como conexión AWS IoT, consulte Activar o desactivar un certificado de cliente

Registro de un certificado de cliente firmado por una entidad de certificación no registrada (CLI)

> **Note**
>
> Antes de llevar a cabo este procedimiento, asegúrese de que tiene el archivo .pem del certificado.

Utilice el comando register-certificate-without-ca para registrar, pero no activar, un certificado de cliente.

```
aws iot register-certificate-without-ca \
```

```
    --certificate-pem file://device_cert_filename.pem
```

El certificado de cliente está registrado AWS IoT, pero aún no está activo. Consulte Activar un certificado de cliente (CLI) para obtener más información sobre cómo activarlo más adelante.

También puede activar el certificado de cliente cuando lo registre utilizando este comando.

```
aws iot register-certificate-without-ca \
    --status ACTIVE \
    --certificate-pem file://device_cert_filename.pem
```

Para obtener más información sobre cómo activar el certificado para poder utilizarlo como conexión AWS IoT, consulteActivar o desactivar un certificado de cliente.

Registre un certificado de cliente cuando el cliente se conecte al AWS IoT just-in-time registro (JITR)

Puede configurar un certificado de CA para permitir que los certificados de cliente con los que ha firmado se registren AWS IoT automáticamente la primera vez que el cliente se conecte. AWS IoT

Para registrar los certificados de cliente cuando un cliente se conecte AWS IoT por primera vez, debe habilitar el registro automático del certificado de CA y configurar la primera conexión del cliente para proporcionar los certificados necesarios.

Configurar un certificado de entidad de certificación para admitir el registro automático (consola)

Para configurar un certificado de CA para que admita el registro automático de certificados de cliente mediante la AWS IoT consola

1. Inicie sesión en la consola AWS de administración y abra la AWS IoT consola.
2. En el panel de navegación izquierdo, selecciona Seguro y elige CAs.
3. En la lista de entidades de certificación, busque aquella para la que desea habilitar el registro automático y abra el menú de opciones mediante el icono de puntos suspensivos.
4. En el menú de opciones, elija Enable auto-registration (Habilitar registro automático).

> **Note**
>
> El estado de registro automático no se muestra en la lista de entidades de certificación. Para ver el estado de registro automático de una entidad de certificación, debe abrir la página Details (Detalles) de la entidad de certificación.

Configurar un certificado de entidad de certificación para admitir el registro automático (CLI)

Si ya ha registrado su certificado de CA AWS IoT, utilice el update-ca-certificatecomando para establecer el certificado `autoRegistrationStatus` de CA en`ENABLE`.

```
aws iot update-ca-certificate \
--certificate-id caCertificateId \
--new-auto-registration-status ENABLE
```

Si desea habilitar `autoRegistrationStatus` al registrar el certificado de entidad de certificación, utilice el comando register-ca-certificate.

```
aws iot register-ca-certificate \
--allow-auto-registration  \
--ca-certificate file://root_CA_cert_filename.pem \
--verification-cert file://verification_cert_filename.pem
```

Utilice el comando describe-ca-certificate para ver el estado del certificado de entidad de certificación.

Configurar la primera conexión de un cliente para el registro automático

Cuando un cliente intenta conectarse AWS IoT por primera vez, el certificado de cliente firmado por su certificado de CA debe estar presente en el cliente durante el protocolo de enlace de Transport Layer Security (TLS).

Cuando el cliente se conecte AWS IoT, utilice el certificado de cliente que creó en Crear certificados de AWS IoT cliente o Crear sus propios certificados de cliente. AWS IoT reconoce el certificado de CA como un certificado de CA registrado, registra el certificado de cliente y establece su estado en`PENDING_ACTIVATION`. Esto significa que el certificado de cliente se registró automáticamente y que está a la espera de su activación. El estado del certificado de cliente debe ser `ACTIVE` antes de que se pueda usar para conectarse a AWS IoT. Consulte Activar o desactivar un certificado de cliente para obtener información sobre la activación de un certificado de cliente.

> **Note**
>
> Puede aprovisionar los dispositivos mediante la función de AWS IoT Core just-in-time registro (JITR) sin tener que enviar toda la cadena de confianza en el momento de la primera conexión de los dispositivos. AWS IoT Core La presentación del certificado de entidad de

certificación es opcional, pero es necesario que el dispositivo envíe la extensión Indicación del nombre del servidor (SNI) cuando se conecte.

Cuando registra AWS IoT automáticamente un certificado o cuando un cliente presenta un certificado en ese `PENDING_ACTIVATION` estado, AWS IoT publica un mensaje sobre el siguiente tema de MQTT:

`$aws/events/certificates/registered/`*`caCertificateId`*

Donde *`caCertificateId`* es el ID del certificado de entidad de certificación que generó el certificado de cliente.

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 de Lambda que verifique que el certificado de cliente 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 cliente. Si la política que va a crear requiere el ID de cliente de los dispositivos que se conectan, puede utilizar la función clientid() de la regla para recuperar el ID de cliente. Un ejemplo de definición de regla sería el siguiente:

```
SELECT *,
   clientid() as clientid
from $aws/events/certificates/registered/caCertificateId
```

En este ejemplo, la regla se suscribe al tema de JITR `$aws/events/certificates/registered/`*`caCertificateID`* y utiliza la función clientid() para recuperar el ID de cliente. A continuación, la regla agrega el ID de cliente a la carga útil de JITR. Para obtener más información sobre la función clientid() de la regla, consulte clientid().

Para obtener más información sobre cómo crear una regla Lambda que escuche el `$aws/events/certificates/registered/caCertificateID` tema y lleve a cabo estas acciones, consulte just-in-time Registro de certificados de cliente en. AWS IoT

Si se produce algún error o excepción durante el registro automático de los certificados de cliente, AWS IoT envía eventos o mensajes a sus CloudWatch registros en Logs. Para obtener más información sobre cómo configurar los registros de tu cuenta, consulta la CloudWatch documentación de Amazon.

Administración de certificados de cliente

AWS IoT proporciona funciones para gestionar los certificados de los clientes.

En este tema:

- Activar o desactivar un certificado de cliente
- Asociar un objeto o una política a un certificado de cliente
- Revocar un certificado de cliente
- Transferir un certificado a otra cuenta

Activar o desactivar un certificado de cliente

AWS IoT comprueba que un certificado de cliente esté activo cuando autentica una conexión.

Puede crear y registrar certificados de cliente sin activarlos para que no se puedan usar hasta que desee usarlos. También puede desactivar los certificados de cliente activos para deshabilitarlos temporalmente. Por último, puede revocar certificados de cliente para evitar que se utilicen en el futuro.

Activar un certificado de cliente (consola)

Para activar un certificado de cliente mediante la consola AWS IoT

1. Inicie sesión en la consola AWS de administración y abra la AWS IoT consola.
2. En el panel de navegación de la izquierda, elija Secure (Seguridad) y, a continuación, elija Certificates (Certificados).
3. En la lista de certificados, busque el certificado que desea activar y abra el menú de opciones mediante el icono de puntos suspensivos.
4. En el menú de opciones, elija Activate (Activar).

El certificado debe aparecer como Active (Activo) en la lista de certificados.

Desactivar un certificado de cliente (consola)

Para desactivar un certificado de cliente mediante la consola AWS IoT

1. Inicie sesión en la consola AWS de administración y abra la AWS IoT consola.
2. En el panel de navegación de la izquierda, elija Secure (Seguridad) y, a continuación, elija Certificates (Certificados).
3. En la lista de certificados, busque el certificado que desea desactivar y abra el menú de opciones mediante el icono de puntos suspensivos.
4. En el menú de opciones, elija Deactivate (Desactivar).

El certificado debe aparecer como Inactive (Inactivo) en la lista de certificados.

Activar un certificado de cliente (CLI)

AWS CLI Proporciona el update-certificatecomando para activar un certificado.

```
aws iot update-certificate \
    --certificate-id certificateId \
    --new-status ACTIVE
```

Si el comando se realizó correctamente, el estado del certificado será `ACTIVE`. Ejecute describe-certificate para ver el estado del certificado.

```
aws iot describe-certificate \
    --certificate-id certificateId
```

Desactivar un certificado de cliente (CLI)

AWS CLI Proporciona el update-certificatecomando para desactivar un certificado.

```
aws iot update-certificate \
    --certificate-id certificateId \
    --new-status INACTIVE
```

Si el comando se realizó correctamente, el estado del certificado será `INACTIVE`. Ejecute describe-certificate para ver el estado del certificado.

```
aws iot describe-certificate \
    --certificate-id certificateId
```

Asociar un objeto o una política a un certificado de cliente

Al crear y registrar un certificado independiente de AWS IoT un objeto, no tendrá políticas que autoricen ninguna AWS IoT operación ni estará asociado a ningún AWS IoT objeto. En esta sección se describe cómo agregar estas relaciones a un certificado registrado.

> **Important**
>
> Para completar estos procedimientos, debe haber creado ya el objeto o la política que desea asociar al certificado.

El certificado autentica un dispositivo AWS IoT para que pueda conectarse. Al asociar el certificado a un recurso de objeto, se establece la relación entre el dispositivo (mediante el certificado) y el recurso de objeto. Para autorizar al dispositivo a realizar AWS IoT acciones, como permitir que el dispositivo se conecte y publique mensajes, se debe adjuntar una política adecuada al certificado del dispositivo.

Asociar un objeto a un certificado de cliente (consola)

Necesitará el nombre de la cosa para completar este procedimiento.

Para asociar un objeto a un certificado registrado

1. Inicie sesión en la consola AWS de administración y abra la AWS IoT consola.
2. En el panel de navegación de la izquierda, elija Secure (Seguridad) y, a continuación, elija Certificates (Certificados).
3. En la lista de certificados, busque el certificado al que desea asociar una política, elija el icono de puntos suspensivos para abrir el menú de opciones del certificado y elija Attach thing (Asociar objeto).
4. En la ventana emergente, busque el nombre de la cosa que desea asociar al certificado, elija su casilla de verificación y elija Asociar.

El objeto debería aparecer ahora en la lista de objetos de la página de detalles del certificado.

Asociar una política a un certificado de cliente (consola)

Necesitará el nombre de la cosa de política para completar este procedimiento.

Para asociar un objeto de política a un certificado registrado

1. Inicie sesión en la consola AWS de administración y abra la AWS IoT consola.
2. En el panel de navegación de la izquierda, elija Secure (Seguridad) y, a continuación, elija Certificates (Certificados).
3. En la lista de certificados, busque el certificado al que desea asociar una política, elija el icono de puntos suspensivos para abrir el menú de opciones del certificado y elija Attach policy (Asociar política).
4. En la ventana emergente, busque el nombre de la política que desea asociar al certificado, elija su casilla de verificación y elija Asociar.

El objeto de política debería aparecer ahora en la lista de políticas de la página de detalles del certificado.

Asociar un objeto a un certificado de cliente (CLI)

AWS CLI Proporciona el attach-thing-principalcomando para adjuntar un objeto a un certificado.

```
aws iot attach-thing-principal \
    --principal certificateArn \
    --thing-name thingName
```

Asociar una política a un certificado de cliente (CLI)

AWS CLI Proporciona el attach-policycomando para adjuntar un objeto de política a un certificado.

```
aws iot attach-policy \
    --target certificateArn \
    --policy-name policyName
```

Revocar un certificado de cliente

Si detecta actividad sospechosa en un certificado de cliente registrado, puede revocarlo para que no se pueda volver a utilizar.

> **Note**
>
> Una vez revocado un certificado, su estado no se puede cambiar. Es decir, el estado del certificado no se puede cambiar a `Active` ni a ningún otro estado.

Revocar un certificado de cliente (consola)

Para revocar un certificado de cliente mediante la consola AWS IoT

1. Inicie sesión en la consola AWS de administración y abra la AWS IoT consola.
2. En el panel de navegación de la izquierda, elija Secure (Seguridad) y, a continuación, elija Certificates (Certificados).
3. En la lista de certificados, busque el certificado que desea revocar y abra el menú de opciones mediante el icono de puntos suspensivos.
4. En el menú de opciones, elija Revoke (Revocar).

Si el certificado se revocó correctamente, se mostrará como Revoked (Revocado) en la lista de certificados.

Revocar un certificado de cliente (CLI)

AWS CLI Proporciona el update-certificatecomando para revocar un certificado.

```
aws iot update-certificate \
    --certificate-id certificateId \
    --new-status REVOKED
```

Si el comando se realizó correctamente, el estado del certificado será `REVOKED`. Ejecute describe-certificate para ver el estado del certificado.

```
aws iot describe-certificate \
    --certificate-id certificateId
```

Transferir un certificado a otra cuenta

Los certificados X.509 que pertenecen a uno se Cuenta de AWS pueden transferir a otro. Cuenta de AWS

Para transferir un certificado X.509 de uno a otro Cuenta de AWS

1. the section called “Comenzar una transferencia de certificado”

   El certificado debe estar desactivado y separado de todas las políticas y elementos antes de iniciar la transferencia.
2. the section called “Aceptar o rechazar una transferencia de certificado”

   La cuenta receptora debe aceptar o rechazar explícitamente el certificado transferido. Una vez que la cuenta receptora acepte el certificado, este debe activarse antes de su uso.
3. the section called “Cancelar una transferencia de certificado”

   La cuenta de origen puede cancelar una transferencia si el certificado no ha sido aceptado.

Comenzar una transferencia de certificado

Puede empezar a transferir un certificado a otro Cuenta de AWS mediante la AWS IoT consola o el AWS CLI.

Comenzar una transferencia de certificado (consola)

Para completar este procedimiento, necesitará el ID del certificado que desea transferir.

Realice este procedimiento desde la cuenta con el certificado que desee transferir.

Para empezar a transferir un certificado a otro Cuenta de AWS

1. Inicie sesión en la consola AWS de administración y abra la AWS IoT consola.
2. En el panel de navegación de la izquierda, elija Secure (Seguridad) y, a continuación, elija Certificates (Certificados).

   Elija el certificado con el estado activo o inactivo que desee transferir y abra su página de detalles.
3. En la página Detalles del certificado, en el menú Acciones, si la opción Desactivar está disponible, elija la opción Desactivar para desactivar el certificado.
4. En la página Detalles del certificado, en el menú de la izquierda, seleccione Políticas.
5. En la página Políticas del certificado, si hay alguna política asociada al certificado, desasocie cada una de ellas abriendo el menú de opciones de la política y seleccionando Desasociar.

El certificado no debe tener ninguna política asociada para poder continuar.

6. En la página Políticas del certificado, en el menú de la izquierda, seleccione Objetos.
7. En la página Objetos del certificado, si hay algún objeto asociado al certificado, desasocie cada uno de ellos abriendo el menú de opciones de la cosa y seleccionando Desasociar.

   El certificado no debe tener ningún objeto asociado para poder continuar.
8. En la página Objetos del certificado, en el menú de la izquierda, seleccione Detalles.
9. En la página Detalles del certificado, en el menú Acciones, seleccione Iniciar transferencia para abrir el cuadro de diálogo Iniciar transferencia.
10. En el cuadro de diálogo Iniciar la transferencia, introduzca el Cuenta de AWS número de la cuenta que recibirá el certificado y un mensaje breve opcional.
11. Seleccione Iniciar transferencia para transferir el certificado.

La consola debería mostrar un mensaje que indique si la transferencia se ha realizado correctamente o no. Si se inició la transferencia, el estado del certificado se actualiza a Transferido.

Comenzar una transferencia de certificado (CLI)

Para completar este procedimiento, necesitará el certificado que desea transferir *certificateId* y el certificado. *certificateArn*

Realice este procedimiento desde la cuenta con el certificado que desee transferir.

Para empezar a transferir un certificado a otra AWS cuenta

1. Utilice el comando update-certificate para desactivar el certificado.

   ```
   aws iot update-certificate --certificate-id certificateId --new-status INACTIVE
   ```
2. Desasocie todas las políticas.

   1. Use el comando list-attached-policies para enumerar las políticas asociadas al certificado.

      ```
      aws iot list-attached-policies --target certificateArn
      ```
   2. Para cada política asociada, utilice el comando detach-policy para desasociar la política.

      ```
      aws iot detach-policy --target certificateArn --policy-name policy-name
      ```

3. Desasocie todos los objetos.

   1. Use el comando list-principal-things para enumerar los objetos asociados al certificado.

      ```
      aws iot list-principal-things --principal certificateArn
      ```

   2. Para cada objeto asociado, utilice el comando detach-thing-principal para desasociar el objeto.

      ```
      aws iot detach-thing-principal --principal certificateArn --thing-name thing-name
      ```

4. Use el comando transfer-certificate para iniciar la transferencia del certificado.

   ```
   aws iot transfer-certificate --certificate-id certificateId --target-aws-
   account account-id
   ```

Aceptar o rechazar una transferencia de certificado

Puede aceptar o rechazar un certificado que le haya sido transferido Cuenta de AWS desde otra persona Cuenta de AWS mediante la AWS IoT consola o el AWS CLI.

Aceptar o rechazar una transferencia de certificado (consola)

Para completar este procedimiento, necesitará el ID del certificado que se le transfirió a su cuenta.

Realice este procedimiento desde la cuenta receptora a la que se transfirió el certificado.

Para aceptar o rechazar un certificado que se ha transferido a su Cuenta de AWS

1. Inicie sesión en la consola AWS de administración y abra la AWS IoT consola.
2. En el panel de navegación de la izquierda, elija Secure (Seguridad) y, a continuación, elija Certificates (Certificados).

   Elija el certificado con el estado Pendiente de transferencia que desee aceptar o rechazar y abra su página de detalles.
3. En la página Detalles del certificado, en el menú Acciones:

   - Para aceptar el certificado, elija Aceptar transferencia.
   - Si no quiere aceptar el certificado, elija Rechazar transferencia.

Aceptar o rechazar una transferencia de certificado (CLI)

Para completar este procedimiento, necesitará el certificado *certificateId* de transferencia que desee aceptar o rechazar.

Realice este procedimiento desde la cuenta receptora a la que se transfirió el certificado.

Para aceptar o rechazar un certificado que se ha transferido a su Cuenta de AWS

1. Para aceptar el certificado, use el comando accept-certificate-transfer.

   ```
   aws iot accept-certificate-transfer --certificate-id certificateId
   ```

2. Para rechazar el certificado, use el comando reject-certificate-transfer.

   ```
   aws iot reject-certificate-transfer --certificate-id certificateId
   ```

Cancelar una transferencia de certificado

Puede cancelar la transferencia de un certificado antes de que se haya aceptado mediante la consola de AWS IoT o la AWS CLI.

Comenzar una transferencia de certificado (consola)

Para completar este procedimiento, necesitará el ID de la transferencia de certificado que quiera cancelar.

Realice este procedimiento desde la cuenta con la que inició la transferencia del certificado.

Para cancelar una transferencia de certificado

1. Inicie sesión en la consola AWS de administración y abra la AWS IoT consola.
2. En el panel de navegación de la izquierda, elija Secure (Seguridad) y, a continuación, elija Certificates (Certificados).

   Elija el certificado con el estado Transferido cuya transferencia desee cancelar y abra su menú de opciones.
3. En el menú de opciones del certificado, elija la opción Revocar transferencia para cancelar la transferencia del certificado.

> **Important**
>
> Tenga cuidado de no confundir la opción Revocar transferencia con la opción Revocar. La opción Revocar transferencia cancela la transferencia del certificado, mientras que la opción Revocar hace que el certificado quede inutilizable de forma irreversible para AWS IoT.

Cancelar una transferencia de certificado (CLI)

Para completar este procedimiento, necesitará el certificado *certificateId* de transferencia que desee cancelar.

Realice este procedimiento desde la cuenta con la que inició la transferencia del certificado.

Utilice el comando cancel-certificate-transfer para cancelar la transferencia del certificado.

```
aws iot cancel-certificate-transfer --certificate-id certificateId
```

Validación personalizada de certificados de cliente

AWS IoT Core admite la validación personalizada de certificados de cliente para los certificados de cliente X.509, lo que mejora la administración de la autenticación de los clientes. Este método de validación de certificados también se conoce como comprobaciones de certificados previas a la autenticación, en las que se evalúan los certificados de los clientes en función de sus propios criterios (definidos en una función de Lambda) y se revocan los certificados de los clientes o el certificado de la entidad de certificación que los firma para impedir que los clientes se conecten a AWS IoT Core. Por ejemplo, puede crear sus propias comprobaciones de revocación de certificados para validar el estado de los certificados con las autoridades de validación que admiten los puntos de conexión Protocolo de verificación de certificados en línea (OCSP) o Listas de revocación de certificados (CRL) e impedir las conexiones de los clientes con certificados revocados. Los criterios utilizados para evaluar los certificados de los clientes se definen en una función de Lambda (también conocida como Lambda de autenticación previa). Debe utilizar los puntos de conexión establecidos en las configuraciones de dominio y el tipo de autenticación debe ser el certificado X.509. Además, los clientes deben proporcionar la extensión de indicación del nombre del servidor (SNI) al conectarse a. AWS IoT Core

> **Note**
>
> Esta función no se admite en las AWS GovCloud (US) regiones.

El proceso de validación personalizada de certificados de cliente implica realizar los siguientes pasos.

- Paso 1: registre sus certificados de cliente X.509 en AWS IoT Core
- Paso 2: creación de una función de Lambda
- Paso 3: AWS IoT Autorizar la invocación de la función Lambda
- Paso 4: establecimiento de una configuración de autenticación para un dominio

Paso 1: registre sus certificados de cliente X.509 en AWS IoT Core

Si aún no lo ha hecho, registre y active sus certificados de cliente X.509 con. AWS IoT Core De no ser así, vaya al siguiente paso.

Para registrar y activar sus certificados de cliente con ellos AWS IoT Core, siga estos pasos:

1. Si crea certificados de cliente directamente con AWS IoT. Estos certificados de cliente se registrarán automáticamente en AWS IoT Core.
2. Si crea sus propios certificados de cliente, siga estas instrucciones para registrarlos AWS IoT Core.
3. Siga estas instrucciones para activar sus certificados de cliente.

Paso 2: creación de una función de Lambda

Debe crear una función de Lambda que verifique certificados y que se llame cada vez que un cliente intente conectarse al punto de conexión configurado. Al crear esta función de Lambda, siga las instrucciones generales de Creación de su primera función de Lambda. Además, asegúrese de que la función de Lambda cumpla con los formatos de solicitud y respuesta esperados de la siguiente manera:

Ejemplo de evento de la función de Lambda

```
{
 "connectionMetadata": {
```

```
  "id": "string"
 },
 "principalId": "string",
 "serverName": "string",
 "clientCertificateChain": [
  "string",
  "string"
 ]
}
```

`connectionMetadata`

Metadatos o información adicional relacionada con la conexión del cliente a AWS IoT Core.

`principalId`

Es el identificador de la entidad principal asociado al cliente en la conexión TLS.

`serverName`

Es la cadena de nombre de host Indicación del nombre del servidor (SNI). AWS IoT Core requiere que los dispositivos envíen la extensión SNI al protocolo de seguridad de la capa de transporte (TLS) y proporcionen la dirección completa del punto de conexión en el campo `host_name`.

`clientCertificateChain`

Matriz de cadenas que representa la cadena de certificados X.509 del cliente.

Ejemplo de respuesta de la función de Lambda

```
{
 "isAuthenticated": "boolean"
}
```

`isAuthenticated`

Es un valor booleano que indica si la solicitud se ha autenticado.

> **Note**
>
> En la respuesta de Lambda, `isAuthenticated` debe ser `true` para continuar con la autenticación y autorización. De lo contrario, se puede desactivar el certificado de cliente

de IoT y bloquear la autenticación personalizada con certificados de cliente X.509 para una mayor autenticación y autorización.

Paso 3: AWS IoT Autorizar la invocación de la función Lambda

Tras crear la función Lambda, debe conceder permiso AWS IoT para invocarla mediante el comando CLI add-permission. Tenga en cuenta que esta función de Lambda se invocará en cada intento de conexión al punto de conexión configurado. Para obtener más información, consulte AWS IoT Autorizar la invocación de la función Lambda.

Paso 4: establecimiento de una configuración de autenticación para un dominio

En la siguiente sección se describe cómo establecer una configuración de autenticación para un dominio personalizado mediante la AWS CLI.

Establecimiento de la configuración del certificado de cliente para un dominio (CLI)

Si no tiene ninguna configuración de dominio, utilice el comando de la CLI create-domain-configuration para crear una. Si ya tiene una configuración de dominio, utilice el comando de la CLI update-domain-configuration para actualizar la configuración del certificado de cliente de un dominio. Debe añadir el ARN de la función de Lambda que ha creado en el paso anterior.

```
aws iot create-domain-configuration \
    --domain-configuration-name domainConfigurationName \
    --authentication-type AWS_X509|CUSTOM_AUTH_X509 \
    --application-protocol SECURE_MQTT|HTTPS \
    --client-certificate-config 'clientCertificateCallbackArn":"arn:aws:lambda:us-
east-2:123456789012:function:my-function:1"}'
```

```
aws iot update-domain-configuration \
    --domain-configuration-name domainConfigurationName \
    --authentication-type AWS_X509|CUSTOM_AUTH_X509 \
    --application-protocol SECURE_MQTT|HTTPS \
    --client-certificate-config '{"clientCertificateCallbackArn":"arn:aws:lambda:us-
east-2:123456789012:function:my-function:1"}'
```

`domain-configuration-name`

Es el nombre de la configuración del dominio.

`authentication-type`

Es el tipo de autenticación de la configuración del dominio. Para obtener más información, consulte Choosing an authentication type.

`application-protocol`

Es el protocolo de aplicación con el que los dispositivos se comunican con AWS IoT Core. Para obtener más información, consulte Choosing an application protocol.

`client-certificate-config`

Objeto que especifica la configuración de autenticación de cliente de un dominio.

`clientCertificateCallbackArn`

El nombre de recurso de Amazon (ARN) de la función Lambda que se AWS IoT invoca en la capa TLS cuando se establece una nueva conexión. Para personalizar la autenticación del cliente y realizar una validación de certificados de cliente personalizada, debe añadir el ARN de la función de Lambda que creó en el paso anterior.

Para obtener más información, consulte CreateDomainConfigurationy consulte la referencia UpdateDomainConfigurationde la API.AWS IoT Para obtener más información sobre la configuración de dominios, consulte Domain configurations.

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 a interfaces AWS IoT HTTP mediante el AWS SDK y AWS CLI.

Las funciones de IAM también te permiten acceder AWS IoT a otros AWS recursos de tu cuenta en tu nombre. Por ejemplo, si desea que un dispositivo publique su estado en una tabla de DynamoDB, las funciones de IAM AWS IoT le permiten interactuar con Amazon DynamoDB. Para obtener más información, consulte Roles de IAM.

En el caso de las conexiones de un intermediario de mensajes a través de HTTP, AWS IoT autentica a los usuarios, grupos y roles mediante el proceso de firma de la versión 4 de Signature. Para obtener más información, consulte Firmar solicitudes AWS de API.

Al usar la versión 4 de AWS Signature con AWS IoT, los clientes deben admitir lo siguiente 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

Para obtener información, consulte Gestión de identidad y acceso para AWS IoT.

Identidades de Amazon Cognito

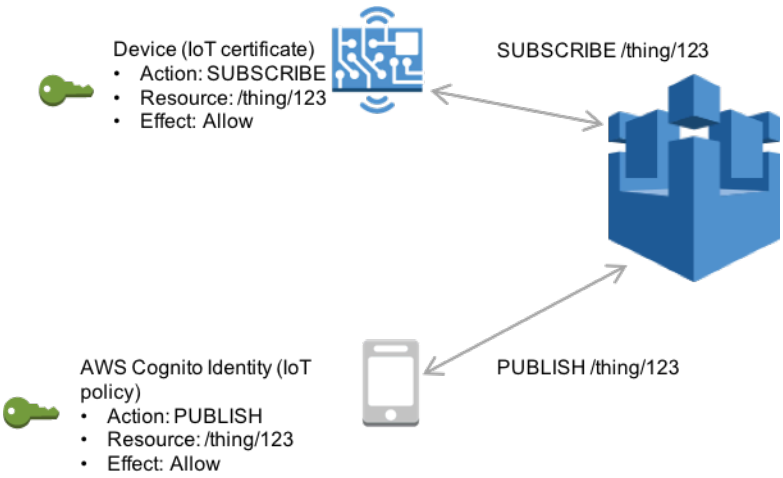
Amazon Cognito Identity le permite crear AWS credenciales temporales con privilegios limitados para utilizarlas en aplicaciones móviles y web. Cuando se utiliza una identidad de Amazon Cognito, se crean grupos de identidades que generan identidades únicas para los usuarios y se autentican con proveedores de identidades como Iniciar sesión con Amazon, Facebook y Google. También puede usar identidades de Amazon Cognito con sus propias identidades autenticadas por el desarrollador. Para obtener más información, consulte Amazon Cognito Identity.

Para usar una identidad de Amazon Cognito, se debe definir un grupo de identidades de Amazon Cognito asociado a un rol de IAM. La función de IAM está asociada a una política de IAM que otorga permisos a las identidades de su grupo de identidades para acceder a AWS recursos como los servicios de llamadas. AWS

Amazon Cognito Identity crea identidades no autenticadas y autenticadas. Las identidades no autenticadas se utilizan para los usuarios invitados de una aplicación móvil o web que desean usar la aplicación sin iniciar sesión. Los usuarios no autenticados solo reciben los permisos especificados en la política de IAM asociada al grupo de identidades.

Cuando utilice identidades autenticadas, además de la política de IAM adjunta al grupo de identidades, debe adjuntar una AWS IoT política a una identidad de Amazon Cognito. Para adjuntar una AWS IoT política, utilice la AttachPolicyAPI y conceda permisos a un usuario individual de su aplicación. AWS IoT Puedes usar la AWS IoT política para asignar permisos específicos a clientes específicos y sus dispositivos.

Los usuarios autenticados y no autenticados son tipos de identidad diferentes. Si no adjuntas una AWS IoT política a Amazon Cognito Identity, un usuario autenticado no podrá obtener la autorización AWS IoT y no tendrá acceso a AWS IoT los recursos ni a las acciones. Para obtener más información sobre la creación de políticas para las identidades de Amazon Cognito, consulte Ejemplos de política de publicación/suscripción y Autorización con identidades de Amazon Cognito.



Autenticación y autorización personalizada

AWS IoT Core le permite definir autorizadores personalizados para que pueda gestionar la autenticación y la autorización de sus propios clientes. Esto resulta útil cuando necesita utilizar mecanismos de autenticación distintos de los que son compatibles de AWS IoT Core forma nativa. (Para obtener más información sobre los mecanismos compatibles de forma nativa, consulte the section called “Autenticación del cliente”).

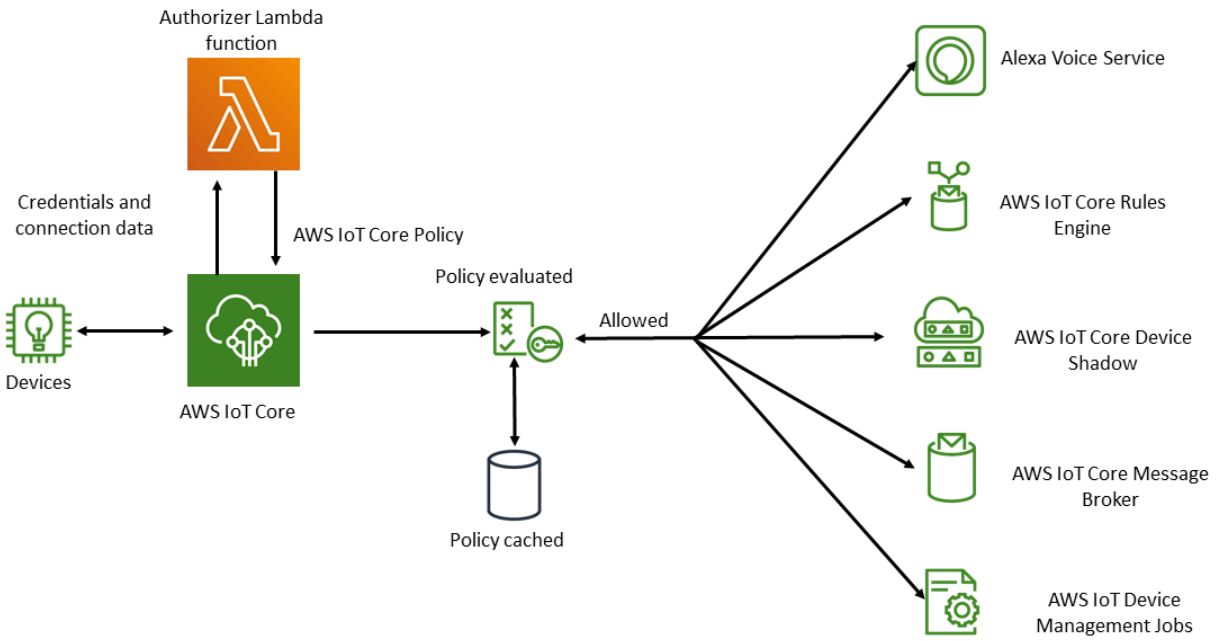
Por ejemplo, si va a migrar los dispositivos existentes sobre el terreno AWS IoT Core y estos dispositivos utilizan un token de portador personalizado o un nombre de usuario y una contraseña de MQTT para autenticarse, puede migrarlos a ellos AWS IoT Core sin tener que proporcionarles nuevas identidades. Puede utilizar la autenticación personalizada con cualquiera de los protocolos de comunicación compatibles. AWS IoT Core Para obtener más información acerca de los protocolos compatibles con AWS IoT Core , consulte the section called “Protocolos de comunicación de dispositivos”.

Temas

- Comprender el flujo de trabajo de autenticación personalizada
- Creación y administración de autorizadores personalizados (CLI)
- Autenticación personalizada con certificados de cliente X.509
- Conectarse a AWS IoT Core mediante una autenticación personalizada
- Resolución de problemas de sus autorizadores

Comprender el flujo de trabajo de autenticación personalizada

La autenticación personalizada le permite definir cómo autenticar y autorizar a los clientes mediante recursos de autorizador. Cada autorizador contiene una referencia a una función de Lambda administrada por el cliente, una clave pública opcional para validar las credenciales del dispositivo e información de configuración adicional. El siguiente diagrama ilustra el flujo de trabajo de autorización para la autenticación personalizada en AWS IoT Core.



AWS IoT Core flujo de trabajo personalizado de autenticación y autorización

En la siguiente lista se explica cada paso del flujo de trabajo de autenticación y autorización personalizada.

1. Un dispositivo se conecta al punto final de AWS IoT Core datos de un cliente mediante uno de los dispositivos compatiblesthe section called “Protocolos de comunicación de dispositivos”. El dispositivo pasa las credenciales a los campos de cabecera o a los parámetros de consulta de la solicitud (para los protocolos HTTP Publish o MQTT a través de WebSockets protocolos), o al campo de nombre de usuario y contraseña del mensaje MQTT CONNECT (para MQTT y MQTT a través de protocolos). WebSockets
2. AWS IoT Core comprueba una de estas dos condiciones:

- La solicitud entrante especifica un autorizador.
- El punto final de AWS IoT Core datos que recibe la solicitud tiene un autorizador predeterminado configurado para ello.

Si AWS IoT Core encuentra un autorizador de alguna de estas formas, AWS IoT Core activa la función Lambda asociada al autorizador.

3. (Opcional) Si ha activado la firma por token, AWS IoT Core valida la firma de la solicitud mediante la clave pública almacenada en el autorizador antes de activar la función Lambda. Si se produce un error en la validación, AWS IoT Core detiene la solicitud sin invocar la función de Lambda.
4. La función de Lambda recibe las credenciales y los metadatos de conexión de la solicitud y toma una decisión de autenticación.
5. La función Lambda devuelve los resultados de la decisión de autenticación y un documento de AWS IoT Core política que especifica qué acciones están permitidas en la conexión. La función Lambda también devuelve información que especifica la frecuencia con la que se AWS IoT Core revalidan las credenciales de la solicitud mediante la invocación de la función Lambda.
6. AWS IoT Core evalúa la actividad de la conexión con la política que ha recibido de la función Lambda.
7. Después de establecer la conexión e invocar inicialmente al autorizador personalizado de Lambda, la siguiente invocación se puede retrasar hasta cinco minutos en las conexiones inactivas sin ninguna operación de MQTT. A continuación, las invocaciones posteriores seguirán el intervalo de actualización de su autorizador de Lambda personalizado. Este enfoque puede evitar invocaciones excesivas que podrían superar el límite de simultaneidad de Lambda de su empresa. Cuenta de AWS

Consideraciones de escalado

Como una función de Lambda gestiona la autenticación y la autorización de su autorizador, está sujeta a los límites de precios y servicios de Lambda, como la tasa de ejecución simultánea. Para obtener más información, consulte Precios de Lambda. Puede administrar la carga de la función de Lambda ajustando los parámetros `refreshAfterInSeconds` y `disconnectAfterInSeconds` de la respuesta de la función de Lambda. Para obtener más información acerca del contenido de la respuesta de una función de Lambda, consulte the section called “Definición de la función de Lambda”.

> **Note**
>
> Si deja habilitada la firma, puede evitar que clientes no reconocidos activen su Lambda de forma excesiva. Tenga esto en cuenta antes de deshabilitar el inicio de sesión en su autorizador.

> **Note**
>
> El límite de tiempo de espera de la función de Lambda para el autorizador personalizado es de 5 segundos.

Creación y administración de autorizadores personalizados (CLI)

AWS IoT Core implementa esquemas de autenticación y autorización personalizados mediante autorizadores personalizados. Un autorizador personalizado es un AWS IoT Core recurso que le brinda la flexibilidad de definir e implementar las reglas y políticas en función de sus requisitos específicos. Para crear un autorizador personalizado con step-by-step instrucciones, consulte el tutorial: Creación de un autorizador personalizado para. AWS IoT Core

Cada autorizador está formado por los siguientes componentes:

- Nombre: cadena única definida por el usuario que identifica al autorizador.
- ARN de la función de Lambda: el nombre de recurso de Amazon (ARN) de la función de Lambda que implementa la lógica de autorización y autenticación.
- Nombre de la clave del token: el nombre de la clave que se utiliza para extraer el token de los encabezados HTTP, los parámetros de consulta o el nombre de usuario de MQTT CONNECT con el fin de realizar la validación de la firma. Este valor es obligatorio si la firma está habilitada en el autorizador.
- Indicador de deshabilitación de firma (opcional): valor booleano que especifica si se debe deshabilitar el requisito de firma en las credenciales. Esto resulta útil en situaciones en las que no tiene sentido firmar las credenciales, como los esquemas de autenticación que utilizan el nombre de usuario y la contraseña de MQTT. El valor predeterminado es `false`, por lo que la firma está habilitada de forma predeterminada.

- Clave pública de firma del token: la clave pública que AWS IoT Core utiliza para validar la firma del token. Su longitud mínima es de 2048 bits. Este valor es obligatorio si la firma está habilitada en el autorizador.

Lambda cobra por el número de veces que se ejecuta la función de Lambda y por el tiempo que tarda en ejecutarse el código de la función. Para obtener más información acerca de los precios de Lambda, consulte Precios de Lambda. Para obtener más información acerca de la creación de funciones de Lambda, consulte la Guía para desarrolladores de Lambda.

> **Note**
>
> Si deja habilitada la firma, puede evitar que clientes no reconocidos activen su Lambda de forma excesiva. Tenga esto en cuenta antes de deshabilitar el inicio de sesión en su autorizador.

> **Note**
>
> El límite de tiempo de espera de la función de Lambda para el autorizador personalizado es de 5 segundos.

En este capítulo:

- Definición de la función de Lambda
- Creación de un autorizador
- Autorizar la invocación AWS IoT de la función Lambda
- Probar los autorizadores
- Administración de puntos de conexión

Definición de la función de Lambda

Cuando AWS IoT Core invoca al autorizador, activa la Lambda asociada al autorizador con un evento que contiene el siguiente objeto JSON. El objeto JSON de ejemplo contiene todos los campos posibles. No se incluye ningún campo que no sea relevante para la solicitud de conexión.

```
{
```

```
    "token" :"aToken",
    "signatureVerified": Boolean, // Indicates whether the device gateway has validated
 the signature.
    "protocols": ["tls", "http", "mqtt"], // Indicates which protocols to expect for
 the request.
    "protocolData": {
        "tls" : {
            "serverName": "serverName" // The server name indication (SNI) host_name
 string.
        },
        "http": {
            "headers": {
                "#{name}": "#{value}"
            },
            "queryString": "?#{name}=#{value}"
        },
        "mqtt": {
            "username": "myUserName",
            "password": "myPassword", // A base64-encoded string.
            "clientId": "myClientId" // Included in the event only when the device
 sends the value.
        }
    },
    "connectionMetadata": {
        "id": UUID // The connection ID. You can use this for logging.
    },
}
```

La función de Lambda debe usar esta información para autenticar la conexión entrante y decidir qué acciones se permiten en la conexión. La función debe enviar una respuesta que contenga los siguientes valores.

- `isAuthenticated`: un valor booleano que indica si la solicitud se ha autenticado.
- `principalId`: una cadena alfanumérica que actúa como identificador del token enviado por la solicitud de autorización personalizada. El valor debe ser una cadena alfanumérica entre 1 y 128 caracteres, y debe coincidir con este patrón de expresión regular (regex): `([a-zA-Z0-9]){1,128}`. No se permite el uso de caracteres especiales que no sean alfanuméricos con el in. `principalId` AWS IoT Core Consulte la documentación de otros AWS servicios si se permiten caracteres especiales no alfanuméricos para el. `principalId`
- `policyDocuments`: Una lista de documentos de políticas con formato JSON Para obtener más información sobre la creación de AWS IoT Core políticas, consulte. AWS IoT Core the section

called “AWS IoT Core políticas” El número máximo de documentos de políticas es de 10. Cada documento de política puede contener un máximo de 2048 caracteres.

- `disconnectAfterInSeconds`: un entero que especifica la duración máxima (en segundos) de la conexión a la puerta de enlace de AWS IoT Core . El valor mínimo es de 300 segundos y el máximo es de 86 400 segundos. El valor predeterminado es 86 400.

> **Note**
>
> El valor de `disconnectAfterInSeconds` (devuelto por la función de Lambda) se determina cuando se establece la conexión. Este valor no se puede modificar durante las siguientes invocaciones de Lambda de actualización de políticas.

- `refreshAfterInSeconds`: un entero que especifica el intervalo entre las actualizaciones de la política. Cuando pasa este intervalo, AWS IoT Core invoca la función de Lambda para permitir la actualización de las políticas. El valor mínimo es de 300 segundos y el máximo es de 86 400 segundos.

El siguiente objeto JSON contiene un ejemplo de una respuesta que la función de Lambda puede enviar.

```
{ "isAuthenticated":true, //A Boolean that determines whether
client can connect. "principalId": "xxxxxxxx",  //A string that
identifies the connection in logs. "disconnectAfterInSeconds": 86400,
  "refreshAfterInSeconds": 300,   "policyDocuments": [       {
 "Version": "2012-10-17",         "Statement": [           {
    "Action": "iot:Publish",              "Effect": "Allow",
      "Resource": "arn:aws:iot:us-east-1:<your_aws_account_id>:topic/
customauthtesting"            }         ]       }     ] }
```

El `policyDocument` valor debe contener un documento de política válido AWS IoT Core . Para obtener más información sobre AWS IoT Core las políticas, consultethe section called “AWS IoT Core políticas”. En MQTT sobre TLS y MQTT sobre WebSockets conexiones, almacena en AWS IoT Core caché esta política durante el intervalo especificado en el valor del campo. `refreshAfterInSeconds` En el caso de las conexiones HTTP, se invoca la función de Lambda para cada solicitud de autorización, a menos que su dispositivo utilice conexiones HTTP persistentes (también denominadas HTTP keep-alive o reutilización de conexiones HTTP). Puede optar por habilitar el almacenamiento en caché al configurar el autorizador. Durante este intervalo, AWS IoT

Core autoriza las acciones en una conexión establecida en contra de esta política en caché sin volver a activar la función Lambda. Si se producen errores durante la autenticación personalizada, AWS IoT Core finaliza la conexión. AWS IoT Core también termina la conexión si ha estado abierta durante más tiempo que el valor especificado en el `disconnectAfterInSeconds` parámetro.

A continuación, se JavaScript incluye un ejemplo de función Lambda de Node.js que busca una contraseña en el mensaje de MQTT Connect con un valor `test` de y devuelve una política que concede permiso AWS IoT Core para conectarse con un cliente `myClientName` denominado y publicar en un tema que contenga el mismo nombre de cliente. Si no encuentra la contraseña esperada, devuelve una política que deniega esas dos acciones.

```
// A simple Lambda function for an authorizer. It demonstrates
// how to parse an MQTT password and generate a response.

exports.handler = function(event, context, callback) {
    var uname = event.protocolData.mqtt.username;
    var pwd = event.protocolData.mqtt.password;
    var buff = new Buffer(pwd, 'base64');
    var passwd = buff.toString('ascii');
    switch (passwd) {
        case 'test':
            callback(null, generateAuthResponse(passwd, 'Allow'));
            break;
        default:
            callback(null, generateAuthResponse(passwd, 'Deny'));
    }
};

// Helper function to generate the authorization response.
var generateAuthResponse = function(token, effect) {
    var authResponse = {};
    authResponse.isAuthenticated = true;
    authResponse.principalId = 'TEST123';

    var policyDocument = {};
    policyDocument.Version = '2012-10-17       ';
    policyDocument.Statement = [];
    var publishStatement = {};
    var connectStatement = {};
    connectStatement.Action = ["iot:Connect"];
    connectStatement.Effect = effect;
```

```
    connectStatement.Resource = ["arn:aws:iot:us-east-1:123456789012:client/
myClientName"];
    publishStatement.Action = ["iot:Publish"];
    publishStatement.Effect = effect;
    publishStatement.Resource = ["arn:aws:iot:us-east-1:123456789012:topic/telemetry/
myClientName"];
    policyDocument.Statement[0] = connectStatement;
    policyDocument.Statement[1] = publishStatement;
    authResponse.policyDocuments = [policyDocument];
    authResponse.disconnectAfterInSeconds = 3600;
    authResponse.refreshAfterInSeconds = 300;

    return authResponse;
}
```

La función de Lambda anterior devuelve el siguiente JSON cuando recibe la contraseña esperada de `test` en el mensaje MQTT Connect. Los valores de las propiedades `password` y `principalId` serán los valores del mensaje MQTT Connect.

```
{
  "password": "password",
  "isAuthenticated": true,
  "principalId": "principalId",
  "policyDocuments": [
    {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Action": "iot:Connect",
          "Effect": "Allow",
          "Resource": "*"
        },
        {
          "Action": "iot:Publish",
          "Effect": "Allow",
          "Resource": "arn:aws:iot:region:accountId:topic/telemetry/${iot:ClientId}"
        },
        {
          "Action": "iot:Subscribe",
          "Effect": "Allow",
          "Resource": "arn:aws:iot:region:accountId:topicfilter/telemetry/
${iot:ClientId}"
        },
```

```
        {
          "Action": "iot:Receive",
          "Effect": "Allow",
          "Resource": "arn:aws:iot:region:accountId:topic/telemetry/${iot:ClientId}"
        }
      ]
    }
  ],
  "disconnectAfterInSeconds": 3600,
  "refreshAfterInSeconds": 300
}
```

Creación de un autorizador

Puede crear un autorizador mediante la API. CreateAuthorizer El siguiente ejemplo describe el comando.

```
aws iot create-authorizer
--authorizer-name MyAuthorizer
--authorizer-function-arn arn:aws:lambda:us-
west-2:<account_id>:function:MyAuthorizerFunction  //The ARN of the Lambda function.
[--token-key-name MyAuthorizerToken //The key used to extract the token from headers.
[--token-signing-public-keys FirstKey=
 "-----BEGIN PUBLIC KEY-----
  [...insert your public key here...]
  -----END PUBLIC KEY-----"
[--status ACTIVE]
[--tags <value>]
[--signing-disabled | --no-signing-disabled]
```

Puede utilizar el parámetro `signing-disabled` para deshabilitar la validación de firmas en cada invocación de su autorizador. Le recomendamos encarecidamente que no deshabilite la firma a menos que sea necesario. La validación de firmas lo protege contra las invocaciones excesivas de su función de Lambda desde dispositivos desconocidos. No se puede actualizar el estado `signing-disabled` de un autorizador una vez creado. Para cambiar este comportamiento, debe crear otro autorizador personalizado con un valor de parámetro `signing-disabled` diferente.

Los valores de los parámetros `tokenKeyName` y `tokenSigningPublicKeys` son opcionales si se deshabilita la firma. Son valores obligatorios si la firma está habilitada.

Tras crear la función Lambda y el autorizador personalizado, debe conceder explícitamente al AWS IoT Core servicio permiso para invocar la función en su nombre. Puede hacerlo con el siguiente comando.

> **Note**
>
> Es posible que el punto de conexión predeterminado de IoT no admita el uso de autorizadores personalizados con funciones de Lambda. En su lugar, puede usar las configuraciones de dominio para definir un nuevo punto de conexión y, a continuación, especificarlo para el autorizador personalizado.

```
aws lambda add-permission --function-name <lambda_function_name>
--principal iot.amazonaws.com --source-arn <authorizer_arn>
--statement-id Id-123 --action "lambda:InvokeFunction"
```

Autorizar la invocación AWS IoT de la función Lambda

En esta sección, concederá el permiso del recurso de autorizador personalizado que acaba de crear para ejecutar la función de Lambda. Para conceder el permiso, puede utilizar el comando de la CLI add-permission.

Conceda permiso a su función Lambda mediante el AWS CLI

1. Después de insertar sus valores, introduzca el siguiente comando. Tenga en cuenta que el valor `statement-id` debe ser único. Reemplace *`Id-1234`* por el valor exacto que tiene; de lo contrario, podría producirse un error `ResourceConflictException`.

   ```
   aws lambda add-permission  \
   --function-name "custom-auth-function" \
   --principal "iot.amazonaws.com" \
   --action "lambda:InvokeFunction" \
   --statement-id "Id-1234" \
   --source-arn authorizerArn
   ```

2. Si el comando tiene éxito, devuelve una declaración de permiso, como la de este ejemplo. Puede continuar con la siguiente sección para probar el autorizador personalizado.

   ```
   {
   ```

```
    "Statement": "{\"Sid\":\"Id-1234\",\"Effect\":\"Allow\",\"Principal
\":{\"Service\":\"iot.amazonaws.com\"},\"Action\":\"lambda:InvokeFunction
\",\"Resource\":\"arn:aws:lambda:Region:57EXAMPLE833:function:custom-
auth-function\",\"Condition\":{\"ArnLike\":{\"AWS:SourceArn\":
\"arn:aws:lambda:Region:57EXAMPLE833:function:custom-auth-function\"}}}"
}
```

Si el comando no tiene éxito, devuelve un error, como en este ejemplo. Tendrá que revisar y corregir el error antes de continuar.

```
An error occurred (AccessDeniedException) when calling the AddPermission operation:
 User: arn:aws:iam::57EXAMPLE833:user/EXAMPLE-1 is not authorized to perform:
 lambda:AddPer
mission on resource: arn:aws:lambda:Region:57EXAMPLE833:function:custom-auth-
function
```

Probar los autorizadores

Puede usar la TestInvokeAuthorizerAPI para probar los valores de invocación y retorno de su autorizador. Esta API te permite especificar los metadatos del protocolo y probar la validación de la firma en tu autorizador.

En las siguientes pestañas se muestra cómo utilizarla AWS CLI para probar el autorizador.

Unix-like

```
aws iot test-invoke-authorizer --authorizer-name NAME_OF_AUTHORIZER \
--token TOKEN_VALUE --token-signature TOKEN_SIGNATURE
```

Windows CMD

```
aws iot test-invoke-authorizer --authorizer-name NAME_OF_AUTHORIZER ^
--token TOKEN_VALUE --token-signature TOKEN_SIGNATURE
```

Windows PowerShell

```
aws iot test-invoke-authorizer --authorizer-name NAME_OF_AUTHORIZER `
--token TOKEN_VALUE --token-signature TOKEN_SIGNATURE
```

El valor del parámetro `token-signature` es el token firmado. Para obtener información sobre cómo obtener este valor, consulte the section called “Firmar el token”.

Si el autorizador utiliza un nombre de usuario y una contraseña, puede transmitir esta información mediante el parámetro `--mqtt-context`. Las siguientes pestañas muestran cómo usar la API `TestInvokeAuthorizer` para enviar un objeto JSON que contenga un nombre de usuario, una contraseña y un nombre de cliente a su autorizador personalizado.

Unix-like

```
aws iot test-invoke-authorizer --authorizer-name NAME_OF_AUTHORIZER  \
--mqtt-context '{"username": "USER_NAME", "password": "dGVzdA==",
 "clientId":"CLIENT_NAME"}'
```

Windows CMD

```
aws iot test-invoke-authorizer --authorizer-name NAME_OF_AUTHORIZER  ^
--mqtt-context '{"username": "USER_NAME", "password": "dGVzdA==",
 "clientId":"CLIENT_NAME"}'
```

Windows PowerShell

```
aws iot test-invoke-authorizer --authorizer-name NAME_OF_AUTHORIZER  `
--mqtt-context '{"username": "USER_NAME", "password": "dGVzdA==",
 "clientId":"CLIENT_NAME"}'
```

Los datos de usuario deben estar codificados en base64. El siguiente ejemplo muestra cómo codificar una contraseña en un entorno similar a Unix.

```
echo -n PASSWORD | base64
```

Administración de puntos de conexión

Puede administrar sus autorizadores mediante lo siguiente. APIs

- ListAuthorizers: Muestra todos los autorizadores de tu cuenta.
- DescribeAuthorizer: Muestra las propiedades del autorizador especificado. Estos valores incluyen la fecha de creación, la fecha de la última modificación y otros atributos.

- SetDefaultAuthorizer: Especifica el autorizador predeterminado para los puntos finales de AWS IoT Core datos. AWS IoT Core utiliza este autorizador si un dispositivo no transfiere AWS IoT Core las credenciales y no especifica un autorizador. Para obtener más información sobre el uso de AWS IoT Core credenciales, consulte. the section called “Autenticación del cliente”
- UpdateAuthorizer: cambia el estado, el nombre de la clave del token o las claves públicas del autorizador especificado.
- DeleteAuthorizer: elimina el autorizador especificado.

> **Note**
>
> No se puede actualizar el requisito de firma de un autorizador. Esto significa que no es posible deshabilitar el inicio de sesión en un autorizador existente que lo requiera. Tampoco se puede requerir el inicio de sesión en un autorizador existente que no lo requiera.

Autenticación personalizada con certificados de cliente X.509

Al conectar dispositivos a AWS IoT Core, tiene varios tipos de autenticación disponibles. Puede usar certificados de cliente X.509, que se pueden usar para autenticar las conexiones de clientes y dispositivos, o definir autorizadores personalizados para administrar su propia lógica de autenticación y autorización de clientes. En este tema se explica cómo utilizar la autenticación personalizada con los certificados de cliente X.509.

El uso de la autenticación personalizada con certificados X.509 puede resultar útil si ya ha autenticado sus dispositivos con certificados X.509 y desea realizar validaciones adicionales y autorizaciones personalizadas. Por ejemplo, si almacena los datos de sus dispositivos, como sus números de serie, en el certificado de cliente X.509, una vez AWS IoT Core autenticado el certificado de cliente X.509, puede utilizar un autorizador personalizado para identificar dispositivos específicos en función de la información almacenada en el campo del certificado. CommonName El uso de la autenticación personalizada con los certificados X.509 puede mejorar la gestión de la seguridad de los dispositivos al conectarlos AWS IoT Core y proporciona más flexibilidad a la hora de gestionar la lógica de autenticación y autorización. AWS IoT Core admite la autenticación personalizada con certificados X.509 mediante el certificado X.509 y el tipo de autenticación con autorizador personalizado, que funciona tanto con el protocolo MQTT como con el protocolo HTTPS. Para obtener más información sobre los tipos de autenticación y los protocolos de aplicación que admiten los puntos de conexión de los dispositivos de AWS IoT Core , consulte Protocolos de comunicación de dispositivos.

> **Note**
>
> Las regiones no admiten la autenticación personalizada con certificados de cliente X.509. AWS GovCloud (US)

> **Important**
>
> Debe usar un punto de conexión creado con configuraciones de dominio. Además, los clientes deben proporcionar la extensión de indicación del nombre del servidor (SNI) al conectarse a. AWS IoT Core

El proceso para autenticar los dispositivos mediante la autenticación personalizada con certificados de cliente X.509 consta de los siguientes pasos.

- Paso 1: registre sus certificados de cliente X.509 en AWS IoT Core
- Paso 2: creación de una función de Lambda
- Paso 3: creación de un autorizador personalizado
- Paso 4: definición del tipo de autenticación y el protocolo de aplicación en una configuración de dominio

Paso 1: registre sus certificados de cliente X.509 en AWS IoT Core

Si aún no lo ha hecho, registre y active sus certificados de cliente X.509 con. AWS IoT Core De no ser así, vaya al siguiente paso.

Para registrar y activar sus certificados de cliente con ellos AWS IoT Core, siga estos pasos:

1. Si crea certificados de cliente directamente con AWS IoT. Estos certificados de cliente se registrarán automáticamente en AWS IoT Core.
2. Si crea sus propios certificados de cliente, siga estas instrucciones para registrarlos AWS IoT Core.
3. Siga estas instrucciones para activar sus certificados de cliente.

Paso 2: creación de una función de Lambda

AWS IoT Core utiliza autorizadores personalizados para implementar esquemas de autenticación y autorización personalizados. Un autorizador personalizado está asociado a una función de Lambda que determina si un dispositivo está autenticado y qué operaciones puede realizar el dispositivo. Cuando un dispositivo se conecta a AWS IoT Core, AWS IoT Core recupera los detalles del autorizador, incluidos el nombre del autorizador y la función Lambda asociada, e invoca la función Lambda. La función de Lambda recibe un evento que contiene un objeto JSON con los datos del certificado de cliente X.509 del dispositivo. La función de Lambda utiliza este objeto JSON del evento para evaluar la solicitud de autenticación, decidir las acciones que se van a realizar y enviar una respuesta.

Ejemplo de evento de la función de Lambda

El siguiente objeto JSON de ejemplo contiene todos los campos posibles que se pueden incluir. El objeto JSON real solo contendrá los campos relevantes para la solicitud de conexión específica.

```
{
 "token": "aToken",
 "signatureVerified": true,
 "protocols": [
  "tls",
  "mqtt"
 ],
 "protocolData": {
  "tls": {
   "serverName": "serverName",
   "x509CertificatePem": "x509CertificatePem",
   "principalId": "principalId"
  },
  "mqtt": {
   "clientId": "myClientId",
                    "username": "myUserName",
                    "password": "myPassword"
  }
 },
 "connectionMetadata": {
  "id": "UUID"
 }
}
```

`signatureVerified`

Es un valor booleano que indica si la firma del token configurada en el autorizador se verifica o no antes de invocar la función de Lambda del autorizador. Si el autorizador está configurado para desactivar la firma de token, este campo será falso.

`protocols`

Es una matriz que contiene los protocolos esperados para la solicitud.

`protocolData`

Es un objeto que contiene información de los protocolos utilizados en la conexión. Proporciona detalles específicos del protocolo que pueden ser útiles para la autenticación, la autorización y mucho más.

`tls`: este objeto contiene información relacionada con el protocolo TLS (seguridad de la capa de transporte).

- `serverName`: es la cadena de nombre de host Indicación del nombre del servidor (SNI). AWS IoT Core requiere que los dispositivos envíen la extensión SNI al protocolo de seguridad de la capa de transporte (TLS) y proporcionen la dirección completa del punto de conexión en el campo `host_name`.
- `x509CertificatePem`: es el certificado X.509 en formato PEM, que se utiliza para la autenticación de cliente en la conexión TLS.
- `principalId`: es el identificador de la entidad principal asociado al cliente en la conexión TLS.

`mqtt`: este objeto incluye información relacionada con el protocolo MQTT.

- `clientId`: solo es necesario incluir una cadena en caso de que el dispositivo envíe este valor.
- `username`: es el nombre de usuario proporcionado en el paquete MQTT Connect.
- `password`: es la contraseña proporcionada en el paquete MQTT Connect.

`connectionMetadata`

Son los metadatos de la conexión.

`id`: es el identificador de conexión, que puede utilizar para registrarse y solucionar problemas.

> **Note**
>
> En este objeto JSON del evento, `x509CertificatePem` y `principalId` son dos campos nuevos en la solicitud. El valor de `principalId` es el mismo que el valor de `certificateId`. Para obtener más información, consulte Certificate.

Ejemplo de respuesta de la función de Lambda

La función de Lambda debe usar información del objeto JSON del evento para autenticar la conexión entrante y decidir qué acciones se permiten en la conexión.

El siguiente objeto JSON contiene un ejemplo de una respuesta que puede enviar la función de Lambda.

```
{
 "isAuthenticated": true,
 "principalId": "xxxxxxxx",
 "disconnectAfterInSeconds": 86400,
 "refreshAfterInSeconds": 300,
 "policyDocuments": [
  {
   "Version": "2012-10-17",
   "Statement": [
    {
     "Effect": "Allow",
     "Action": "iot:Publish",
     "Resource": "arn:aws:iot:us-east-1:123456789012:topic/customauthtesting"
    }
   ]
  }
 ]
}
```

En este ejemplo, esta función debe enviar una respuesta que contenga los siguientes valores.

`isAuthenticated`

Es un valor booleano que indica si la solicitud se ha autenticado.

`principalId`

Es una cadena alfanumérica que actúa como identificador del token enviado por la solicitud de autorización personalizada. El valor debe ser una cadena alfanumérica con un mínimo de 1 y un máximo de 128 caracteres. Identifica la conexión en los registros. El valor de `principalId` debe ser el mismo que el valor de `principalId` en el objeto JSON del evento (es decir, el CertificateID del certificado X.509).

`policyDocuments`

Una lista de documentos de políticas con formato JSON. AWS IoT Core El valor es opcional y admite variables de política de objetos y variables de política de certificados. El número máximo de documentos de políticas es de 10. Cada documento de política puede contener un máximo de 2048 caracteres. Si tiene varias políticas asociadas a su certificado de cliente y a la función de Lambda, el permiso es una recopilación de todas las políticas. Para obtener más información sobre la creación de AWS IoT Core políticas, consulte Políticas.

`disconnectAfterInSeconds`

Un entero que especifica la duración máxima (en segundos) de la conexión a la AWS IoT Core puerta de enlace. El valor mínimo es 300 segundos y el máximo, 86 400 segundos. `disconnectAfterInSeconds` es para toda la vida útil de una conexión y no se actualiza al actualizar la política de forma consecutiva.

`refreshAfterInSeconds`

Es un entero que especifica el intervalo entre las actualizaciones de la política. Cuando pasa este intervalo, AWS IoT Core invoca la función Lambda para permitir la actualización de las políticas. El valor mínimo es de 300 segundos y el máximo es de 86 400 segundos.

Ejemplo de función de Lambda

A continuación, se muestra una función de Lambda Node.js de ejemplo. La función examina el certificado X.509 del cliente y extrae la información relevante, como el número de serie, la huella digital y el nombre del sujeto. Si la información extraída coincide con los valores esperados, el cliente podrá conectarse. Este mecanismo garantiza que solo los clientes autorizados con certificados válidos puedan establecer una conexión.

```
const crypto = require('crypto');

exports.handler = async (event) => {
```

```
    // Extract the certificate PEM from the event
    const certPem = event.protocolData.tls.x509CertificatePem;

    // Parse the certificate using Node's crypto module
    const cert = new crypto.X509Certificate(certPem);

    var effect = "Deny";
    // Allow permissions only for a particular certificate serial, fingerprint, and
 subject
    if (cert.serialNumber === "7F8D2E4B9C1A5036DE8F7C4B2A91E5D80463BC9A1257" // This is
 a random serial
        && cert.fingerprint ===
 "F2:9A:C4:1D:B5:E7:08:3F:6B:D0:4E:92:A7:C1:5B:8D:16:0F:E3:7A" // This is a random
 fingerprint
        && cert.subject === "allow.example.com") {
      effect = "Allow";
    }

    return generateAuthResponse(event.protocolData.tls.principalId, effect);
};


// Helper function to generate the authorization response.
function generateAuthResponse(principalId, effect) {
    const authResponse = {
        isAuthenticated: true,
        principalId,
        disconnectAfterInSeconds: 3600,
        refreshAfterInSeconds: 300,
        policyDocuments: [
          {
            Version: "2012-10-17",
            Statement: [
              {
                Action: ["iot:Connect"],
                Effect: effect,
                Resource: [
                  "arn:aws:iot:us-east-1:123456789012:client/myClientName"
                ]
              },
              {
                Action: ["iot:Publish"],
                Effect: effect,
                Resource: [
```

```
                "arn:aws:iot:us-east-1:123456789012:topic/telemetry/myClientName"
              ]
            },
            {
              Action: ["iot:Subscribe"],
              Effect: effect,
              Resource: [
                 "arn:aws:iot:us-east-1:123456789012:topicfilter/telemetry/
myClientName"
              ]
            },
            {
              Action: ["iot:Receive"],
              Effect: effect,
              Resource: [
                 "arn:aws:iot:us-east-1:123456789012:topic/telemetry/myClientName"
              ]
            }
          ]
        }
      ]
    };

  return authResponse;
}
```

La función de Lambda anterior devuelve el siguiente JSON cuando recibe un certificado con el número de serie, la huella digital y el sujeto esperados. El valor de `x509CertificatePem` será el certificado de cliente proporcionado en el protocolo de enlace TLS. Para obtener más información, consulte Defining your Lambda function.

```
{
 "isAuthenticated": true,
 "principalId": "principalId in the event JSON object",
 "policyDocuments": [
  {
   "Version": "2012-10-17",
   "Statement": [
    {
     "Action": "iot:Connect",
     "Effect": "Allow",
     "Resource": "arn:aws:iot:us-east-1:123456789012:client/myClientName"
    },
```

```
    {
     "Action": "iot:Publish",
     "Effect": "Allow",
     "Resource": "arn:aws:iot:us-east-1:123456789012:topic/telemetry/myClientName"
    },
    {
     "Action": "iot:Subscribe",
     "Effect": "Allow",
     "Resource": "arn:aws:iot:us-east-1:123456789012:topicfilter/telemetry/
myClientName"
    },
    {
     "Action": "iot:Receive",
     "Effect": "Allow",
     "Resource": "arn:aws:iot:us-east-1:123456789012:topic/telemetry/myClientName"
    }
   ]
  }
 ],
 "disconnectAfterInSeconds": 3600,
 "refreshAfterInSeconds": 300
}
```

Paso 3: creación de un autorizador personalizado

Tras definir la función de Lambda, cree un autorizador personalizado para administrar su propia lógica de autenticación y autorización de clientes. Puede seguir las instrucciones detalladas en el Step 3: Create a customer authorizer resource and its authorization. Para obtener más información, consulte Creating an authorizer.

En el proceso de creación del autorizador personalizado, debe conceder el permiso de AWS IoT para invocar la función de Lambda una vez creada. Para obtener instrucciones detalladas, consulte AWS IoT Autorizar la invocación de la función Lambda.

Paso 4: definición del tipo de autenticación y el protocolo de aplicación en una configuración de dominio

Para autenticar los dispositivos mediante la autenticación personalizada con certificados de cliente X.509, debe establecer el tipo de autenticación y el protocolo de aplicación en una configuración de dominio y debe enviar la extensión SNI. El valor de `authenticationType` debe ser `CUSTOM_AUTH_X509` y el valor de `applicationProtocol` puede ser `SECURE_MQTT` o `HTTPS`.

Definición del tipo de autenticación y el protocolo de aplicación en una configuración de dominio (CLI)

Si no tiene una configuración de dominio, utilice el comando create-domain-configuration para crear una. El valor de `authenticationType` debe ser `CUSTOM_AUTH_X509` y el valor de `applicationProtocol` puede ser `SECURE_MQTT` o `HTTPS`.

```
aws iot create-domain-configuration \
    --domain-configuration-name domainConfigurationName \
    --authentication-type CUSTOM_AUTH_X509 \
    --application-protocol SECURE_MQTT \
    --authorizer-config '{
        "defaultAuthorizerName": my-custom-authorizer
    }'
```

Si ya tiene una configuración de dominio, utilice el comando update-domain-configuration para actualizar `authenticationType` y `applicationProtocol`, si es necesario. Tenga en cuenta que no puede cambiar el tipo o el protocolo de autenticación en el punto de conexión predeterminado (`iot:Data-ATS`).

```
aws iot update-domain-configuration \
    --domain-configuration-name domainConfigurationName \
    --authentication-type CUSTOM_AUTH_X509 \
    --application-protocol SECURE_MQTT \
    --authorizer-config '{
        "defaultAuthorizerName": my-custom-authorizer
    }'
```

`domain-configuration-name`

Es el nombre de la configuración del dominio.

`authentication-type`

Es el tipo de autenticación de la configuración del dominio. Para obtener más información, consulte Choosing an authentication type.

`application-protocol`

Es el protocolo de aplicación con el que los dispositivos se comunican con AWS IoT Core. Para obtener más información, consulte Choosing an application protocol.

`--authorizer-config`

Es un objeto que especifica la configuración del autorizador en una configuración de dominio.

`defaultAuthorizerName`

Es el nombre del autorizador para una configuración de dominio.

Para obtener más información, consulte CreateDomainConfigurationy UpdateDomainConfigurationdesde la referencia de la AWS IoT API. Para obtener más información sobre la configuración de dominios, consulte Domain configurations.

Conectarse a AWS IoT Core mediante una autenticación personalizada

Los dispositivos se pueden conectar AWS IoT Core mediante una autenticación personalizada con cualquier protocolo AWS IoT Core compatible con la mensajería del dispositivo. Para obtener más información acerca de los protocolos de comunicación compatibles, consulte the section called "Protocolos de comunicación de dispositivos". Los datos de conexión que se transfieren a la función de Lambda de su autorizador dependen del protocolo que utilice. Para obtener más información acerca de cómo crear la función de Lambda de su autorizador, consulte the section called "Definición de la función de Lambda". En las secciones siguientes se explica cómo conectarse para autenticarse mediante cada protocolo compatible.

HTTPS

Los dispositivos a los que se envían datos AWS IoT Core mediante la API de publicación HTTP pueden transferir las credenciales a través de los encabezados de las solicitudes o de los parámetros de consulta en sus solicitudes HTTP POST. Los dispositivos pueden especificar un autorizador para invocarlo mediante el encabezado `x-amz-customauthorizer-name` o el parámetro de consulta. Si tiene habilitada la firma por token en su autorizador, debe pasar *`token-key-name`* y `x-amz-customauthorizer-signature` en los encabezados de las solicitudes o en los parámetros de consulta. Tenga en cuenta que el *`token-signature`* valor debe estar codificado en una URL cuando se utilice JavaScript desde el navegador.

> **Note**
>
> El autorizador del cliente para el protocolo HTTPS solo admite operaciones de publicación. Para obtener más información acerca del protocolo HTTP, consulte the section called "Protocolos de comunicación de dispositivos".

Los siguientes ejemplos de solicitudes muestran cómo se transfieren estos parámetros tanto en los encabezados de las solicitudes como en los parámetros de consulta.

```
//Passing credentials via headers
POST /topics/topic?qos=qos HTTP/1.1
Host: your-endpoint
x-amz-customauthorizer-signature: token-signature
token-key-name: token-value
x-amz-customauthorizer-name: authorizer-name

//Passing credentials via query parameters
POST /topics/topic?qos=qos&x-amz-customauthorizer-signature=token-signature&token-key-
name=token-value HTTP/1.1
```

MQTT

Los dispositivos que se conectan AWS IoT Core mediante una conexión MQTT pueden pasar las credenciales a través de los `password` campos `username` y de los mensajes MQTT. De forma opcional, el valor de `username` también puede contener una cadena de consulta que transfiere valores adicionales (como un token, una firma y el nombre del autorizador) a su autorizador. Puede utilizar esta cadena de consulta si desea utilizar un esquema de autenticación basado en símbolos en lugar de valores `username` y `password`.

> **Note**
>
> Los datos del campo de contraseña están codificados en base64 mediante. AWS IoT Core Debe descodificarlos con la función de Lambda.

El siguiente ejemplo contiene una cadena `username` con parámetros adicionales que especifican un token y una firma.

```
username?x-amz-customauthorizer-name=authorizer-name&x-amz-customauthorizer-
signature=token-signature&token-key-name=token-value
```

Para invocar un autorizador, los dispositivos que se conecten AWS IoT Core mediante MQTT y la autenticación personalizada deben conectarse al puerto 443. También deben pasar la extensión TLS de negociación de protocolos de capa de aplicación (ALPN) con un valor de `mqtt` y la extensión de indicación del nombre del servidor (SNI) con el nombre de host de su terminal de datos. AWS IoT Core Para evitar posibles errores, el valor de `x-amz-customauthorizer-signature` debe estar codificado como URL. También recomendamos encarecidamente que los valores de `x-amz-customauthorizer-name` y `token-key-name` estén codificados como URL. Para obtener más

información acerca de estos valores, consultethe section called “Protocolos de comunicación de dispositivos”. V2 SDK de dispositivos, SDK para móviles y cliente de dispositivo de AWS IoT puede configurar estas dos extensiones.

MQTT ha terminado WebSockets

Los dispositivos que se conectan AWS IoT Core mediante MQTT Over WebSockets pueden transferir las credenciales de una de las dos maneras siguientes.

- Mediante los encabezados de las solicitudes o los parámetros de consulta de la solicitud HTTP UPGRADE para establecer la WebSockets conexión.
- A través de los campos `username` y `password` del mensaje MQTT CONNECT.

Si pasa las credenciales a través del mensaje de conexión de MQTT, se requieren las extensiones TLS ALPN y SNI. Para obtener más información, consulte the section called “MQTT”. El siguiente ejemplo ilustra cómo pasar credenciales a través de la solicitud de actualización de HTTP.

```
GET /mqtt HTTP/1.1
Host: your-endpoint
Upgrade: WebSocket
Connection: Upgrade
x-amz-customauthorizer-signature: token-signature
token-key-name: token-value
sec-WebSocket-Key: any random base64 value
sec-websocket-protocol: mqtt
sec-WebSocket-Version: websocket version
```

Firmar el token

Debe firmar el token con la clave privada del par de claves pública-privada que utilizó en la llamada a `create-authorizer`. Los siguientes ejemplos muestran cómo crear la firma simbólica mediante un comando similar a Unix y. JavaScript Utilizan el algoritmo hash SHA-256 para codificar la firma.

Command line

```
echo -n TOKEN_VALUE | openssl dgst -sha256 -sign PEM encoded RSA private key |
 openssl base64
```

JavaScript

```
const crypto = require('crypto')

const key = "PEM encoded RSA private key"

const k = crypto.createPrivateKey(key)
let sign = crypto.createSign('SHA256')
sign.write(t)
sign.end()
const s = sign.sign(k, 'base64')
```

Resolución de problemas de sus autorizadores

En este tema se explican los problemas más comunes que pueden causar problemas en los flujos de trabajo de autenticación personalizada y los pasos para resolverlos. Para solucionar los problemas de forma más eficaz, habilite CloudWatch los registros AWS IoT Core y establezca el nivel de registro en DEBUG. Puede habilitar CloudWatch los registros en la AWS IoT Core consola () https://console.aws.amazon.com/iot/. Para obtener más información acerca de cómo habilitar y configurar registros de AWS IoT Core, consulte the section called “Configure el AWS IoT registro”.

> **Note**
>
> Si deja el nivel de registro en DEBUG durante períodos prolongados, CloudWatch podría almacenar grandes cantidades de datos de registro. Esto puede aumentar sus CloudWatch cargos. Considere la posibilidad de utilizar el registro basado en recursos para aumentar la verbosidad solo para los dispositivos de un grupo de objetos concreto. Para obtener más información sobre el registro basado en recursos, consulte the section called “Configure el AWS IoT registro”. Además, cuando haya terminado de solucionar problemas, reduzca el nivel de registro a un nivel menos detallado.

Antes de empezar a solucionar problemas, consulte the section called “Comprender el flujo de trabajo de autenticación personalizada” para obtener una visión general del proceso de autenticación personalizada. Esto le ayudará a entender dónde buscar el origen del problema.

En este tema se analizan las dos áreas siguientes para que las investigue.

- Cuestiones relacionadas con la función de Lambda de su autorizador.
- Problemas relacionados con el dispositivo.

Comprobar si hay problemas con la función de Lambda de su autorizador

Lleve a cabo los siguientes pasos para asegurarse de que los intentos de conexión de sus dispositivos estén invocando la función de Lambda.

1. Compruebe qué función de Lambda está asociada a su autorizador.

   Puede hacerlo llamando a la DescribeAuthorizerAPI o haciendo clic en el autorizador deseado en la sección Segura de la AWS IoT Core consola.
2. Compruebe las métricas de invocación de la función de Lambda. Para ello, siga estos pasos.

   a. Abra la AWS Lambda consola (https://console.aws.amazon.com/lambda/) y seleccione la función asociada a su autorizador.

   b. Seleccione la pestaña Monitorizar y consulte las métricas correspondientes al período de tiempo correspondiente a su problema.
3. Si no ve ninguna invocación, compruebe que AWS IoT Core tiene permiso para invocar la función Lambda. Si ve invocaciones, vaya directamente al siguiente paso. Lleve a cabo los siguientes pasos para comprobar que la función de Lambda tiene los permisos necesarios.

   a. Seleccione la pestaña Permisos para su función en la consola. AWS Lambda

   b. Busque la sección Política basada en recursos en la parte inferior de la página. Si la función de Lambda tiene los permisos necesarios, la política tiene el aspecto que se muestra en el siguiente ejemplo.

```
{
  "Version":"2012-10-17",
  "Id": "default",
  "Statement": [
    {
      "Sid": "Id123",
      "Effect": "Allow",
      "Principal": {
        "Service": "iot.amazonaws.com"
```

```
      },
      "Action": "lambda:InvokeFunction",
      "Resource": "arn:aws:lambda:us-
east-1:111111111111:function:FunctionName",
      "Condition": {
        "ArnLike": {
          "AWS:SourceArn": "arn:aws:iot:us-east-1:111111111111:authorizer/
AuthorizerName"
        },
        "StringEquals": {
          "AWS:SourceAccount": "111111111111"
        }
      }
    }
  ]
}
```

c. Esta política concede al AWS IoT Core director el `InvokeFunction` permiso para desempeñar sus funciones. Si no lo ves, tendrás que añadirlo mediante la AddPermissionAPI. El siguiente ejemplo le muestra cómo hacerlo utilizando la AWS CLI.

```
aws lambda add-permission --function-name FunctionName --principal
 iot.amazonaws.com --source-arn AuthorizerARn --statement-id Id-123 --action
 "lambda:InvokeFunction"
```

4. Si ve invocaciones, compruebe que no haya errores. Un error podría indicar que la función Lambda no gestiona correctamente el evento de conexión que se AWS IoT Core le envía.

   Para obtener información sobre cómo gestionar el evento en la función de Lambda, consulte the section called “Definición de la función de Lambda”. Puedes usar la función de prueba de la AWS Lambda consola (https://console.aws.amazon.com/lambda/) para codificar los valores de prueba de la función y asegurarte de que la función gestiona los eventos correctamente.

5. Si ve invocaciones sin errores, pero sus dispositivos no pueden conectarse (ni publicar, suscribirse ni recibir mensajes), el problema podría deberse a que la política que devuelve la función de Lambda no concede permisos para las acciones que los dispositivos están intentando realizar. Lleve a cabo los siguientes pasos para determinar si hay algún problema con la política que devuelve la función.

   a. Utilice una consulta de Amazon CloudWatch Logs Insights para escanear los registros durante un período breve y comprobar si hay errores. El siguiente ejemplo de consulta ordena los eventos por marca de tiempo y busca errores.

```
display clientId, eventType, status, @timestamp | sort @timestamp desc | filter
 status = "Failure"
```

b. Actualice la función Lambda para registrar los datos a los que regresa AWS IoT Core y el evento que activa la función. Puede usar estos registros para inspeccionar la política que crea la función.

6. Si ve invocaciones sin errores, pero sus dispositivos no pueden conectarse (ni publicar, suscribirse ni recibir mensajes), otro motivo puede ser que la función de Lambda supere el límite de tiempo de espera. El límite de tiempo de espera de la función de Lambda para el autorizador personalizado es de 5 segundos. Puede comprobar la duración de la función en CloudWatch registros o métricas.

Investigar problemas con los dispositivos

Si no encuentra problemas al invocar la función de Lambda o con la política que devuelve la función, busque problemas con los intentos de conexión de sus dispositivos. Las solicitudes de conexión mal formadas pueden provocar que AWS IoT Core no se active el autorizador. Se pueden producir problemas de conexión tanto en la capa TLS como en la de aplicación.

Posibles problemas con la capa TLS:

- Los clientes deben incluir un encabezado de nombre de host (HTTP, MQTT over WebSockets) o la extensión TLS con indicación del nombre del servidor (HTTP, MQTT over WebSockets, MQTT) en todas las solicitudes de autenticación personalizadas. En ambos casos, el valor transferido debe coincidir con uno de los extremos de datos de su cuenta. AWS IoT Core Estos son los puntos de conexión que se devuelven al ejecutar los siguientes comandos de la CLI.
  - `aws iot describe-endpoint --endpoint-type iot:Data-ATS`
  - `aws iot describe-endpoint --endpoint-type iot:Data`(para puntos VeriSign finales antiguos)
- Los dispositivos que utilizan una autenticación personalizada para las conexiones MQTT también deben pasar la extensión TLS de negociación de protocolo de capa de aplicación (ALPN) con un valor de `mqtt`.
- La autenticación personalizada actualmente solo está disponible en el puerto 443.

Posibles problemas en la capa de aplicación:

- Si la firma está habilitada (el campo `signingDisabled` es falso en su autorizador), busque los siguientes problemas de firma.
  - Asegúrese de pasar la firma simbólica en el encabezado `x-amz-customauthorizer-signature` o en un parámetro de cadena de consulta.
  - Asegúrese de que el servicio no firme un valor distinto del token.
  - Asegúrese de pasar el token al encabezado o parámetro de consulta que especificó en el campo `token-key-name` de su autorizador.
- Asegúrese de que el nombre del autorizador que pasa en el encabezado `x-amz-customauthorizer-name` o la cadena de consulta es válido o de que ha definido un autorizador predeterminado para su cuenta.

Autorización

Autorización es el proceso de concesión de permisos a una identidad autenticada. Usted concede permisos de AWS IoT Core uso AWS IoT Core y políticas de IAM. En este tema se abordan las políticas de AWS IoT Core . Para obtener más información sobre las políticas de IAM, consulte Gestión de identidad y acceso para AWS IoT y ¿Cómo AWS IoT funciona con IAM.

AWS IoT Core las políticas determinan lo que puede hacer una identidad autenticada. Los dispositivos y las aplicaciones móviles, web y de escritorio utilizan identidades autenticadas. Una identidad autenticada puede incluso ser un usuario que escribe comandos AWS IoT Core CLI. Una identidad solo puede ejecutar AWS IoT Core operaciones si tiene una política que le conceda permiso para realizar esas operaciones.

Tanto AWS IoT Core las políticas como las políticas de IAM se utilizan AWS IoT Core para controlar las operaciones que puede realizar una identidad (también denominada principal). El tipo de política que utilice depende del tipo de identidad con la que se autentique. AWS IoT Core

AWS IoT Core las operaciones 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 y recibir datos AWS IoT Core.

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

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.

AWS IoT Core Tipos de API y políticas del plano de datos

Protocolo y mecanismo de autenticación	SDK	Tipo de identidad	Tipo de política		
MQTT a través de TLS/TCP, autenticación mutua TLS (puerto 8883 o 443)[‡])	AWS IoT SDK de dispositivo	Certificados X.509	AWS IoT Core política		
MQTT sobre HTTPS/ WebSocket, autenticación AWS SigV4 (puerto 443)	AWS SDK móvil	Identidad sin autenticar de Amazon Cognito	IAM y políticas AWS IoT Core		
		Entidad autenticada de Amazon Cognito	Política de IAM		
		Identidad federada o de IAM	Política de IAM		
HTTPS, autenticación AWS Signature versión 4 (puerto 443)	AWS CLI	Identidad de Amazon Cognito, de IAM o federada	Política de IAM		

Protocolo y mecanismo de autenticación	SDK	Tipo de identidad	Tipo de política		
HTTPS, autenticación mutua TLS (puerto 8443)	Sin compatibilidad de SDK	Certificados X.509	AWS IoT Core política		
HTTPS sobre autenticación personalizada (Puerto 443)	AWS IoT SDK de dispositivo	Autorizador personalizado	Política de autorizador personalizada		

AWS IoT Core Tipos de API y políticas del plano de control

Protocolo y mecanismo de autenticación	SDK	Tipo de identidad	Tipo de política		
Autenticación de la versión 4 de la AWS firma HTTPS (puerto 443)	AWS CLI	Identidad de Amazon Cognito	Política de IAM		
		Identidad federada o de IAM	Política de IAM		

AWS IoT Core las políticas se adjuntan a los certificados X.509, a las identidades de Amazon Cognito o a los grupos de cosas. Las políticas de IAM están asociadas a un usuario, grupo o rol de IAM. Si utiliza la AWS IoT consola o la AWS IoT Core CLI para adjuntar la política (a un certificado, Amazon Cognito Identity o un grupo de cosas), utilizará una AWS IoT Core política. De lo contrario, utilizará una política de IAM. AWS IoT Core las políticas asociadas a un grupo de cosas se aplican

a cualquier cosa dentro de ese grupo de cosas. Para que la AWS IoT Core política surta efecto, el nombre de la cosa `clientId` y el de la cosa deben coincidir.

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 Core. Por ejemplo, pensemos en un dispositivo al que se conecta AWS IoT Core 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 la CLI en una línea de comandos. Al usar una política, puede permitir o denegar el acceso al usuario a cualquier comando o AWS IoT Core recurso. También puede controlar el acceso de una aplicación a los recursos de AWS IoT Core .

Los cambios realizados en una política pueden tardar unos minutos en hacerse efectivos debido a la forma en que se almacenan en caché en AWS IoT los documentos de la política. Es decir, es posible que el acceso a un recurso al que se haya concedido acceso recientemente tarde unos minutos y que se pueda acceder a un recurso durante varios minutos después de que se haya revocado su acceso.

AWS formación y certificación

Para obtener información sobre la autorización AWS IoT Core, realice el curso Profundiza en la AWS IoT Core autenticación y la autorización en el sitio web de AWS formación y certificación.

AWS IoT Core políticas

AWS IoT Core las políticas son documentos JSON. Siguen las mismas convenciones que las políticas de IAM. AWS IoT Core admite políticas con nombre específico, por lo que muchas identidades pueden hacer referencia al mismo documento de política. Las políticas con nombre cuentan con varias versiones para facilitar su restauración.

AWS IoT Core las políticas permiten controlar el acceso al plano AWS IoT Core de datos. El plano de datos de AWS IoT Core se compone de operaciones que le permiten conectarse con el agente de mensajes de AWS IoT Core , enviar y recibir mensajes MQTT y obtener o actualizar la sombra de dispositivo de un objeto.

Una AWS IoT Core política es un documento JSON que contiene una o más declaraciones de políticas. Cada instrucción contiene:

- `Effect`, que especifica si se permite o se deniega la acción.
- `Action`, que especifica la acción que la política permite o deniega.

- `Resource`, que especifica los recursos en los que se permite o se deniega la acción.

Los cambios realizados en una política pueden tardar entre 6 y 8 minutos en hacerse efectivos debido a la forma en que se almacenan en AWS IoT caché los documentos de la política. Es decir, es posible que el acceso a un recurso al que se haya concedido acceso recientemente tarde unos minutos y que se pueda acceder a un recurso durante varios minutos después de que se haya revocado su acceso.

AWS IoT Core las políticas se pueden adjuntar a los certificados X.509, a las identidades de Amazon Cognito y a los grupos de cosas. Las políticas asociadas a un grupo de objetos se aplican a cualquier objeto dentro de ese grupo. Para que la política surta efecto, `clientId` y el nombre de la cosa deben coincidir. Las políticas de AWS IoT Core siguen la misma lógica de evaluación de políticas que las políticas de IAM. De forma predeterminada, todas las políticas se deniegan implícitamente. Un permiso explícito en cualquier política basada en identidades o recursos anula el comportamiento predeterminado. Una denegación explícita en cualquier política invalida cualquier permiso concedido. Para obtener más información, consulte Lógica de evaluación de políticas en la Guía del usuario de AWS Identity and Access Management .

Temas

- AWS IoT Core acciones políticas
- AWS IoT Core recursos de acción
- AWS IoT Core variables de política
- Prevención de la sustitución confusa entre servicios
- AWS IoT Core ejemplos de políticas
- Autorización con identidades de Amazon Cognito

AWS IoT Core acciones políticas

AWS IoT Core define las siguientes acciones de política:

Acciones de política de MQTT

`iot:Connect`

Representa el permiso para conectarse al agente de AWS IoT Core mensajes. 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. Después de que el segundo cliente se conecta, el agente cierra la conexión existente. Utilice el permiso `iot:Connect` para garantizar que solo puedan conectarse los clientes autorizados que utilizan un ID de cliente específico.

`iot:DeleteConnection`

Representa el permiso para desconectarse de un cliente de MQTT conectado desde AWS IoT Core. El permiso `iot:DeleteConnection` se comprueba cada vez que se presenta una solicitud para desconectar un cliente de forma ineludible. Al desconectar un cliente, AWS IoT Core cierra la conexión de red del cliente y, si lo desea, limpia el estado de la sesión.

`iot:GetRetainedMessage`

Representa el permiso para obtener el contenido de un único mensaje retenido. Los mensajes retenidos son los mensajes que se publicaron con el indicador RETAIN establecido y almacenados por AWS IoT Core. Para obtener permiso y obtener una lista de todos los mensajes retenidos de la cuenta, consulte `iot:ListRetainedMessages`.

`iot:ListRetainedMessages`

Representa el permiso para recuperar información resumida sobre los mensajes retenidos de la cuenta, pero no el contenido de los mensajes. Los mensajes retenidos son los mensajes que se publicaron con el indicador RETAIN establecido y almacenados por AWS IoT Core. El ARN del recurso especificado para esta acción debe ser `*`. Para tener permiso y obtener el contenido de un único mensaje retenido, consulte `iot:GetRetainedMessage`.

`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. Puede utilizarlo para permitir a los clientes publicar en determinados patrones de tema.

> **Note**
>
> Para conceder el permiso `iot:Publish`, también debe otorgar el permiso `iot:Connect`.

`iot:Receive`

Representa el permiso para recibir un mensaje de AWS IoT Core. El permiso `iot:Receive` se confirma cada vez que se entrega un mensaje a un cliente. Dado que este permiso se comprueba

en cada entrega, puede utilizarlo para revocar permisos a clientes que están suscritos en ese momento a un tema.

`iot:RetainPublish`

Representa el permiso para publicar un mensaje MQTT con el indicador RETAIN activado.

> **Note**
>
> Para conceder el permiso `iot:RetainPublish`, también debe otorgar el permiso `iot:Publish`.

`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. Utilícelo para permitir a los clientes suscribirse a temas que coinciden con patrones de tema específicos.

> **Note**
>
> Para conceder el permiso `iot:Subscribe`, también debe otorgar el permiso `iot:Connect`.

Acciones de política de sombra de dispositivo

`iot:DeleteThingShadow`

Representa el permiso para eliminar la sombra de dispositivo de un objeto. El permiso `iot:DeleteThingShadow` se comprueba cada vez que se presenta una solicitud para eliminar el contenido de la sombra de dispositivo de un objeto.

`iot:GetThingShadow`

Representa el permiso para recuperar la sombra de dispositivo de un objeto. El permiso `iot:GetThingShadow` se comprueba cada vez que se presenta una solicitud para eliminar el contenido de la sombra de dispositivo de un objeto.

`iot:ListNamedShadowsForThing`

Representa el permiso para enumerar las sombras de un objeto. El permiso `iot:ListNamedShadowsForThing` se comprueba cada vez que se presenta una solicitud para enumerar las sombras de un objeto.

`iot:UpdateThingShadow`

Representa el permiso para actualizar la sombra de un dispositivo. El permiso `iot:UpdateThingShadow` se comprueba cada vez que se presenta una solicitud para actualizar el contenido de la sombra de dispositivo.

> **Note**
>
> Las acciones de política de ejecución de trabajo se aplican únicamente al punto de conexión HTTP TLS. Si utiliza el punto de conexión MQTT, debe utilizar las acciones de política MQTT definidas en este tema.
> Para ver un ejemplo de una política de ejecución de tareas que lo demuestre, compruebe the section called “Ejemplo de políticas de trabajos básica” si funciona con el protocolo MQTT.

Acciones AWS IoT Core políticas de Job Executions

`iotjobsdata:DescribeJobExecution`

Representa el permiso para recuperar una ejecución de trabajo para un objeto determinado. El permiso `iotjobsdata:DescribeJobExecution` se comprueba cada vez que se presenta una solicitud para obtener la ejecución de un trabajo.

`iotjobsdata:GetPendingJobExecutions`

Representa el permiso para recuperar la lista de trabajos que no están en un estado final para un objeto. El permiso `iotjobsdata:GetPendingJobExecutions` se comprueba cada vez que se presenta una solicitud para recuperar la lista.

`iotjobsdata:UpdateJobExecution`

Representa el permiso para actualizar una ejecución de trabajo. El permiso `iotjobsdata:UpdateJobExecution` se comprueba cada vez que se presenta una solicitud para actualizar el estado de una ejecución de trabajo.

`iotjobsdata:StartNextPendingJobExecution`

Representa el permiso para obtener e iniciar la próxima ejecución de trabajo pendiente para un objeto, es decir, para actualizar una ejecución de trabajo con un estado QUEUED a IN_PROGRESS. El permiso `iotjobsdata:StartNextPendingJobExecution` se comprueba cada vez que se presenta una solicitud para iniciar la siguiente ejecución de trabajo pendiente.

AWS IoT Core Acción política sobre el proveedor de credenciales

`iot:AssumeRoleWithCertificate`

Representa el permiso para llamar al proveedor de AWS IoT Core credenciales para que asuma una función de IAM con la autenticación basada en certificados. El `iot:AssumeRoleWithCertificate` permiso se comprueba cada vez que se solicita al proveedor de AWS IoT Core credenciales que asuma un rol.

AWS IoT Core recursos de acción

Para especificar un recurso para una acción AWS IoT Core política, utilice el nombre de recurso de Amazon (ARN) del recurso. Todos los recursos ARNs siguen el siguiente formato:

```
arn:partition: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. Los ejemplos de ARN son para el ID de cuenta `123456789012`, en la partición `aws` y específicos de la región `us-east-1`. Para obtener más información sobre los formatos ARNs, consulte Amazon Resource Names (ARNs) en la Guía del AWS Identity and Access Management usuario.

Action	Tipo de recurso	Nombre del recurso	Ejemplo de ARN
`iot:Connect`	`client`	El ID de cliente del cliente	`arn:aws:iot:us-east-1:123456789012:client/myClientId`

Action	Tipo de recurso	Nombre del recurso	Ejemplo de ARN
iot:DeleteConnection	client	El ID de cliente del cliente	arn:aws:iot:us-east-1:123456789012:client/myClientId
iot:DeleteThingShadow	thing	El nombre de la cosa y el nombre de la sombra, si corresponde	arn:aws:iot:us-east-1:123456789012:thing/thingOne arn:aws:iot:us-east-1:123456789012:thing/thingOne/shadowOne
iotjobsdata:DescribeJobExecution	thing	El nombre de la cosa	arn:aws:iot:us-east-1:123456789012:thing/thingOne
iotjobsdata:GetPendingJobExecutions	thing	El nombre de la cosa	arn:aws:iot:us-east-1:123456789012:thing/thingOne
iot:GetRetainedMessage	topic	Un tema de mensaje retenido	arn:aws:iot:us-east-1:123456789012:topic/myTopicName
iot:GetThingShadow	thing	El nombre de la cosa y el nombre de la sombra, si corresponde	arn:aws:iot:us-east-1:123456789012:thing/thingOne arn:aws:iot:us-east-1:123456789012:thing/thingOne/shadowOne
iot:ListNamedShadowsForThing	Todos	Todos	*

Action	Tipo de recurso	Nombre del recurso	Ejemplo de ARN
iot:ListRetainedMessages	Todos	Todos	*
iot:Publish	topic	Una cadena de tema	arn:aws:iot:us-east-1:123456789012:topic/myTopicName
iot:Receive	topic	Una cadena de tema	arn:aws:iot:us-east-1:123456789012:topic/myTopicName
iot:RetainPublish	topic	Un tema para publicar con el conjunto de indicadores RETAIN	arn:aws:iot:us-east-1:123456789012:topic/myTopicName
iotjobsdata:StartNextPendingJobExecution	thing	El nombre de la cosa	arn:aws:iot:us-east-1:123456789012:thing/thingOne
iot:Subscribe	topicfilter	Una cadena de filtro de temas	arn:aws:iot:us-east-1:123456789012:topicfilter/myTopicFilter
iotjobsdata:UpdateJobExecution	thing	El nombre de la cosa	arn:aws:iot:us-east-1:123456789012:thing/thingOne

Action	Tipo de recurso	Nombre del recurso	Ejemplo de ARN
`iot:UpdateThingShadow`	`thing`	El nombre de la cosa y el nombre de la sombra, si corresponde	`arn:aws:iot:us-east-1:123456789012:thing/thingOne arn:aws:iot:us-east-1:123456789012:thing/thingOne/shadowOne`
`iot:AssumeRoleWithCertificate`	`rolealias`	El alias de rol que apunta al ARN de rol	`arn:aws:iot:us-east-1:123456789012:rolealias/CredentialProviderRole_alias`

AWS IoT Core variables de política

AWS IoT Core define las variables de política que se pueden utilizar en AWS IoT Core las políticas del `Condition` bloque `Resource` o. Cuando se evalúa una política, las variables de política se sustituyen por valores reales. Por ejemplo, si un dispositivo está conectado al agente de AWS IoT Core mensajes con un ID de cliente de 100-234-3456, la variable de política se sustituye en el documento de `iot:ClientId` política por 100-234-3456.

AWS IoT Core las políticas pueden utilizar caracteres comodín y seguir una convención similar a la de las políticas de IAM. La inserción de un * (asterisco) en la cadena se puede tratar como un comodín, ya que coincide con cualquier carácter. Por ejemplo, se puede utilizar * para describir varios nombres de temas de MQTT en el atributo `Resource` de una política. Los caracteres + y # se tratan como cadenas literales en una política. Para ver un ejemplo de política que muestra cómo usar caracteres comodín, consulte Uso de caracteres comodín en MQTT y en las políticas AWS IoT Core.

También puede utilizar variables políticas predefinidas con valores fijos para representar caracteres que de otro modo tendrían un significado especial. Estos caracteres especiales incluyen `${*}`, `${?}` y `${$}`. Para obtener más información sobre las variables de política y los caracteres especiales, consulte Elementos de la política de IAM: variables y etiquetas y Creación de una condición con varias claves o valores.

Temas

- Variables de política básicas AWS IoT Core
- Variables de la política de objeto
- Variables de AWS IoT Core política de certificados X.509

Variables de política básicas AWS IoT Core

AWS IoT Core define las siguientes variables de política básicas:

- `aws:SourceIp`: la dirección IP del cliente conectado al agente de AWS IoT Core mensajes.
- `iot:ClientId`: es el ID de cliente que se utiliza para conectarse con el agente de mensajes de AWS IoT Core .
- `iot:DomainName`: el nombre de dominio del cliente al que está conectado AWS IoT Core.

Ejemplos

- Ejemplos de ClientId y variables de política SourceIp
- Ejemplos de la variable de política iot:DomainName

Ejemplos de **`ClientId`** y variables de política **`SourceIp`**

La siguiente AWS IoT Core política muestra una política que utiliza variables de política. `aws:SourceIp`se puede usar en el elemento Condición de tu política para permitir que los directores realicen solicitudes a la API solo dentro de un rango de direcciones específico. Para ver ejemplos, consulte Autorizar a los usuarios y los servicios en la nube a usar Jobs AWS IoT.

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

```
    "Effect": "Allow",
    "Action": [
     "iot:Publish"
    ],
    "Resource": [
     "arn:aws:iot:us-east-1:123456789012:topic/my/topic/${iot:ClientId}"
    ],
    "Condition": {
     "IpAddress": {
      "aws:SourceIp": "123.45.167.89"
     }
    }
   }
  ]
}
```

En estos ejemplos, `${iot:ClientId}` se sustituye por el ID del cliente conectado al agente de AWS IoT Core mensajes cuando se evalúa la política. Cuando utiliza variables de la política como `${iot:ClientId}`, puede abrir de forma accidental 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/my/${iot:ClientId}/topic"
 ]
}
```

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 `my/+/topic`. Para protegerse contra estas brechas de seguridad, utilice la acción `iot:Connect` política para controlar qué cliente IDs se puede conectar. Por ejemplo, esta política permite conectarse 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/clientid"
   ]
  }
 ]
}
```

> **Note**
>
> No se recomienda utilizar la variable de política `${iot:ClientId}` con `Connect`. No se comprueba el valor de `ClientId`, por lo que un asociador con un identificador de cliente diferente puede superar la validación pero provocar la desconexión. Como cualquier `ClientId` está permitido, si se establece un ID de cliente aleatorio, se pueden omitir las políticas de los grupos de objetos.

Ejemplos de la variable de política **`iot:DomainName`**

Puede agregar la variable de política `iot:DomainName` para restringir los dominios que se pueden usar. Agregar la variable de política `iot:DomainName` permite que los dispositivos se conecten solo a determinados puntos de conexión configurados.

La siguiente política permite que los dispositivos se conecten al dominio especificado.

```
{
 "Version":"2012-10-17",
 "Statement": {
  "Sid": "AllowConnectionsToSpecifiedDomain",
  "Effect": "Allow",
  "Action": [
   "iot:Connect"
  ],
  "Resource": "arn:aws:iot:us-east-1:123456789012:client/clientid",
  "Condition": {
   "StringEquals": {
```

```
    "iot:DomainName": "d1234567890abcdefghij-ats.iot.us-east-1.amazonaws.com"
   }
  }
 }
}
```

La siguiente política impide que los dispositivos se conecten al dominio especificado.

```
{
 "Version":"2012-10-17",
 "Statement": {
  "Sid": "DenyConnectionsToSpecifiedDomain",
  "Effect": "Deny",
  "Action": [
   "iot:Connect"
  ],
  "Resource": "arn:aws:iot:us-east-1:123456789012:client/clientid",
  "Condition": {
   "StringEquals": {
    "iot:DomainName": "d1234567890abcdefghij-ats.iot.us-east-1.amazonaws.com"
   }
  }
 }
}
```

Para obtener más información acerca de los operadores de condición de las políticas, consulte Elementos de la política de JSON de IAM: operadores de condición. Para obtener más información sobre las configuraciones de dominios, consulte What is a domain configuration?.

Variables de la política de objeto

Las variables de política de cosas permiten escribir AWS IoT Core políticas que concedan o denieguen permisos en función de las propiedades de las cosas, como los nombres, los tipos y los valores de los atributos de las cosas. Puede usar variables de política de cosas para aplicar la misma política y controlar muchos AWS IoT Core dispositivos. Para obtener más información acerca del aprovisionamiento de dispositivos, consulte Aprovisionamiento de dispositivos.

Si utiliza una asociación de objetos no exclusiva, se puede asociar el mismo certificado a varios objetos. Para mantener una asociación clara y evitar posibles conflictos, debe hacer coincidir el ID de

cliente con el nombre del objeto. En este caso, el nombre de la cosa se obtiene del ID de cliente del `Connect` mensaje MQTT que se envía cuando una cosa se conecta a AWS IoT Core ella.

Tenga en cuenta lo siguiente cuando utilice variables en las políticas de AWS IoT Core .

- Utilice la AttachThingPrincipalAPI para adjuntar certificados o principios (identidades autenticadas de Amazon Cognito) a un objeto.
- Si existe una asociación de objetos no exclusiva, al reemplazar los nombres de objeto con variables de política de objeto, el valor de `clientId` en el mensaje de conexión de MQTT o la conexión TLS deben coincidir exactamente con el nombre del objeto.

Las siguientes variables de política de objeto están disponibles:

- `iot:Connection.Thing.ThingName`

  Esto se traduce en el nombre del elemento del AWS IoT Core registro para el que se está evaluando la política. AWS IoT Core usa el certificado que presenta el dispositivo cuando se autentica para determinar qué se debe usar para verificar la conexión. Esta variable de política solo está disponible cuando un dispositivo se conecta a través de MQTT o MQTT a través del protocolo. WebSocket
- `iot:Connection.Thing.ThingTypeName`

  Esta variable se resuelve en el tipo de objeto asociado al objeto para el que se evalúa la política. El ID de cliente de la WebSocket conexión MQTT/ debe ser el mismo que el nombre del elemento. Esta variable de política solo está disponible cuando se conecta a través de MQTT o MQTT a través del 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 está disponible como una variable de política: `iot:Connection.Thing.Attributes[`*`attributeName`*`]` donde *`attributeName`* está el nombre del atributo. El ID de cliente de la MQTT/WebSocket conexión debe ser el mismo que el nombre de la cosa. Esta variable de política solo está disponible cuando se conecta a través de MQTT o MQTT a través del WebSocket protocolo.
- `iot:Connection.Thing.IsAttached`

  `iot:Connection.Thing.IsAttached: ["true"]`obliga a que solo los dispositivos que estén registrados AWS IoT y conectados a la red principal puedan acceder a los permisos

incluidos en la política. Puede utilizar esta variable para impedir que un dispositivo se conecte AWS IoT Core si presenta un certificado que no esté adjunto a un elemento de IoT en el AWS IoT Core registro. Esta variable tiene valores que `false` indican que el elemento de conexión está adjunto al certificado `true` o a la identidad de Amazon Cognito en el registro mediante la API. AttachThingPrincipal El nombre de la cosa se toma como identificador de cliente.

Si el ID de cliente coincide con el nombre de objeto o si asocia el certificado a un objeto de forma exclusiva, el uso de variables de política en la definición de la política puede simplificar la administración de políticas. En lugar de crear políticas individuales para cada elemento de IoT, puede definir una única política mediante las variables de política del objeto. Esta política se puede aplicar a todos los dispositivos de forma dinámica. A continuación, se muestra un ejemplo de política que muestra cómo funciona. Para obtener más información, consulte ???.

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Condition": {
    "StringLike": {
     "iot:ClientId": "*${iot:Connection.Thing.Attributes[envType]}"
    }
   },
   "Effect": "Allow",
   "Action": "iot:Connect",
   "Resource": "arn:aws:iot:us-east-1:123456789012:client/*"
  }
 ]
}
```

Este ejemplo de política permite que las cosas se conecten AWS IoT Core si su ID de cliente termina con el valor de su atributo. `envType` Solo se permitirá la conexión a los objetos con un patrón de ID de cliente que coincide.

Variables de AWS IoT Core política de certificados X.509

Las variables de política de certificados X.509 ayudan a escribir AWS IoT Core políticas. Estas políticas conceden permisos en función de los atributos del certificado X.509. En las siguientes secciones se describe cómo se pueden utilizar estas variables de política de certificado.

> **Important**
>
> Si el certificado X.509 no incluye un atributo de certificado concreto, pero la variable de política de certificados correspondiente se utiliza en el documento de política, la evaluación de la política podría provocar un comportamiento inesperado.

CertificateId

En la RegisterCertificateAPI, `certificateId` aparece en el cuerpo de la respuesta. Para obtener información sobre su certificado, utilice el formulario `certificateId` in DescribeCertificate.

Atributos de emisor

Las siguientes variables AWS IoT Core de política permiten permitir o denegar permisos, en función de los atributos del 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 sujeto

Las siguientes variables AWS IoT Core de política permiten conceder o denegar permisos, en función de los atributos del sujeto del 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` podría contener una lista de nombres de países, pero cuando se evalúa en una política, `iot:Certificate.Subject.Country` se reemplaza por el nombre del primer país.

Puede solicitar un valor de atributo específico distinto del primero mediante un índice de base uno. 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 índice 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.

Atributos de nombre alternativo del emisor

Las siguientes variables AWS IoT Core de política 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 sujeto

Las siguientes variables AWS IoT Core de política permiten conceder o denegar permisos, en función de los atributos del nombre alternativo del sujeto 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 utilizarlas `iot:Certificate.SerialNumber` para permitir o denegar el acceso a AWS IoT Core los recursos, 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.

Uso de variables de política de certificado X.509

En este tema se explica cómo utilizar las variables de política de certificados. Las variables de política de certificado X.509 son esenciales para crear políticas de AWS IoT Core que concedan permisos basados en los atributos de certificado X.509. Si el certificado X.509 no incluye un atributo de certificado concreto, pero la variable de política de certificados correspondiente se utiliza en el documento de política, la evaluación de la política podría provocar un comportamiento inesperado. Esto se debe a que la variable de política que falta no se evalúa en la instrucción de la política.

En este tema:

- Ejemplo de certificado X.509
- Uso de los atributos del emisor del certificado como variables de política de certificados
- Uso de los atributos del sujeto del certificado como variables de política de certificados

- Uso de los atributos del nombre alternativo del emisor del certificado como variables de la política de certificados
- Uso de los atributos del nombre alternativo del sujeto del certificado como variables de la política de certificados
- Uso de otro atributo del certificado como variable de política de certificados
- Limitaciones aplicables a las variables de política de certificado X.509
- Ejemplos de políticas que utilizan variables de política de certificados

Ejemplo de certificado X.509

Un certificado X.509 típico puede tener el siguiente aspecto. Este certificado de ejemplo incluye los atributos del certificado. Durante la evaluación de las políticas de AWS IoT Core , los siguientes atributos del certificado se rellenarán como variables de la política de certificado: `Serial Number`, `Issuer`, `Subject`, `X509v3 Issuer Alternative Name` y `X509v3 Subject Alternative Name`.

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            92:12:85:cb:b7:a5:e0:86
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=US, O=IoT Devices, OU=SmartHome, ST=WA, CN=IoT Devices Primary CA,
    GN=Primary CA1/initials=XY/dnQualifier=Example corp,
    SN=SmartHome/ title=CA1/pseudonym=Primary_CA/generationQualifier=2/serialNumber=987

        Validity
            Not Before: Mar 26 03:25:40 2024 GMT
            Not After : Apr 28 03:25:40 2025 GMT
        Subject: C=US, O=IoT Devices, OU=LightBulb, ST=NY, CN=LightBulb Device Cert,
    GN=Bulb/initials=ZZ/dnQualifier=Bulb001,
    SN=Multi Color/title=RGB/pseudonym=RGB Device/generationQualifier=4/
serialNumber=123
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (2048 bit)
                Modulus:
                    << REDACTED >>
                Exponent: 65537 (0x10001)
        X509v3 extensions:
```

```
            X509v3 Basic Constraints:
                CA:FALSE
            X509v3 Key Usage:
                Digital Signature, Non Repudiation, Key Encipherment
            X509v3 Subject Alternative Name:
                DNS:example.com, IP Address:1.2.3.4, URI:ResourceIdentifier001,
email:device1@example.com, DirName:/C=US/O=IoT/OU=SmartHome/CN=LightBulbCert
            X509v3 Issuer Alternative Name:
                DNS:issuer.com, IP Address:5.6.7.8, URI:PrimarySignerCA,
email:primary@issuer.com, DirName:/C=US/O=Issuer/OU=IoT Devices/CN=Primary Issuer CA
   Signature Algorithm: sha256WithRSAEncryption
        << REDACTED >>
```

Uso de los atributos del emisor del certificado como variables de política de certificados

La siguiente tabla proporciona detalles sobre cómo se rellenarán los atributos del emisor del certificado en una AWS IoT Core política.

Atributos del emisor que se deben rellenar en una política

Atributos del emisor de certificados	Variables de política de certificados
• C=US • O=IoT Devices • OU= SmartHome • ST=WA • CN=IoT Devices Primary CA • GN=Primario CA1 • initials=XY • dnQualifier=Example corp • SN= SmartHome • título= CA1 • pseudonym=Primary_CA • generationQualifier=2 • serialNumber=987	• `iot:Certificate.Issuer.Country = US` • `iot:Certificate.Issuer.Organization = IoT Devices` • `iot:Certificate.Issuer.OrganizationalUnit = SmartHome` • `iot:Certificate.Issuer.State = WA` • `iot:Certificate.Issuer.CommonName = IoT Devices Primary CA` • `iot:Certificate.Issuer.GivenName = Primary CA1` • `iot:Certificate.Issuer.initials = XY` • `iot:Certificate.Issuer.Distinguished NameQualifier = Example corp` • `iot:Certificate.Issuer.Surname = SmartHome` • `iot:Certificate.Issuer.Title = CA1`

Atributos del emisor de certificados	Variables de política de certificados
	• `iot:Certificate.Issuer.Pseudonym = Primary_CA` • `iot:Certificate.Issuer.GenerationQualifier = 2` • `iot:Certificate.Issuer.SerialNumber = 987`

Uso de los atributos del sujeto del certificado como variables de política de certificados

La siguiente tabla proporciona detalles sobre cómo se rellenarán los atributos del sujeto del certificado en una AWS IoT Core política.

Atributos del sujeto que se van a rellenar en una política

Atributos del sujeto del certificado	Variables de política de certificados
• C=US • O=IoT Devices • ST=NY • CN= Certificado de LightBulb dispositivo • GN=Bulb • initials=ZZ • dnQualifier=Bulb001 • SN=Multi Color • title=RGB • pseudonym=RGB Device • generationQualifier=4 • serialNumber=123	• `iot:Certificate.Subject.Country = US` • `iot:Certificate.Subject.Organization = IoT Devices` • `iot:Certificate.Subject.State = NY` • `iot:Certificate.Subject.CommonName = LightBulb Device Cert` • `iot:Certificate.Subject.GivenName = Bulb` • `iot:Certificate.Subject.initials = ZZ` • `iot:Certificate.Subject.DistinguishedNameQualifier = Bulb001` • `iot:Certificate.Subject.Surname = Multi Color` • `iot:Certificate.Subject.Title = RGB` • `iot:Certificate.Subject.Pseudonym = RGB Device`

Atributos del sujeto del certificado	Variables de política de certificados
	• `iot:Certificate.Subject.GenerationQualifier = 4` • `iot:Certificate.Subject.SerialNumber = 123`

Uso de los atributos del nombre alternativo del emisor del certificado como variables de la política de certificados

En la siguiente tabla se proporcionan detalles sobre cómo se rellenan los atributos del nombre alternativo del emisor de certificados en una política de AWS IoT Core .

Atributos del nombre alternativo del emisor que se deben rellenar en una política

Nombre alternativo del emisor de X509v3	Atributo de una política
• DNS:issuer.com • IP Address:5.6.7.8 • URI: CA PrimarySigner • email:primary@issuer.com • DirName: /C= US/O=Issuer/OU=IoT Devices/CN = CA del emisor principal	• `iot:Certificate.Issuer.AlternativeName.DNSName = issuer.com` • `iot:Certificate.Issuer.AlternativeName.IPAddress = 5.6.7.8` • `iot:Certificate.Issuer.AlternativeName.UniformResourceIdentifier = PrimarySignerCA` • `iot:Certificate.Issuer.AlternativeName.RFC822Name = primary@issuer.com` • `iot:Certificate.Issuer.AlternativeName.DirectoryName = cn=Primary Issuer CA,ou=IoT Devices,o=Issuer,c=US`

Uso de los atributos del nombre alternativo del sujeto del certificado como variables de la política de certificados

En la siguiente tabla se proporcionan detalles sobre cómo se rellenan los atributos del nombre alternativo del sujeto de certificados en una política de AWS IoT Core .

Atributos del nombre alternativo del sujeto que se deben rellenar en una política

Nombre alternativo del sujeto de X509v3	Atributo de una política
• DNS:example.com • IP Address:1.2.3.4 • URIResourceIdentifier: 001 • email:device1@example.com • DirName: /C= = US/O=IoT/OU=SmartHome/CN LightBulbCert	• `iot:Certificate.Subject.AlternativeName.DNSName = example.com` • `iot:Certificate.Subject.AlternativeName.IPAddress = 1.2.3.4` • `iot:Certificate.Subject.AlternativeName.UniformResourceIdentifier = ResourceIdentifier001` • `iot:Certificate.Subject.AlternativeName.RFC822Name = device1@example.com` • `iot:Certificate.Subject.AlternativeName.DirectoryName = cn=LightBulbCert,ou=SmartHome,o=IoT,c=US`

Uso de otro atributo del certificado como variable de política de certificados

La siguiente tabla proporciona detalles sobre cómo se rellenarán los demás atributos del certificado en una AWS IoT Core política.

Otros atributos que se deben rellenar en una política

Otro atributo de certificado	Variable de política de certificado
`Serial Number: 92:12:85:cb:b7:a5:e0:86`	`iot:Certificate.SerialNumber = 105256223 89124227206`

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:

Variables de política que faltan

Si el certificado X.509 no incluye un atributo de certificado concreto, pero la variable de política de certificados correspondiente se utiliza en el documento de política, la evaluación de la política podría provocar un comportamiento inesperado. Esto se debe a que la variable de política que falta no se evalúa en la instrucción de la política.

SerialNumber Formato de certificado

AWS IoT Core trata el número de serie del certificado como la representación en cadena de un entero decimal. Por ejemplo, si una política solo permite conexiones con un identificador de cliente que coincida con el número de serie del certificado, el identificador de cliente debe ser el número de serie en formato decimal.

Caracteres comodín

Si los atributos de certificado contienen caracteres de certificado, la variable de política no se sustituirá por el valor de atributo del certificado. Esto dejará el texto `${policy-variable}` en el documento de política. Esto puede producir un error de autorización. Se pueden utilizar los siguientes caracteres comodín: *, $, +, ? y #.

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 del certificado. Esto dejará el texto `${policy-variable}` en el documento de política. Esto puede producir un error de autorización.

Caracteres especiales

Cualquier carácter especial, como `,`, '', \, +, =, <, > y `;` debe tener el prefijo de una barra invertida (\) cuando se utiliza en una variable de política. Por ejemplo, `Amazon Web Services O=Amazon.com Inc. L=Seattle ST=Washington C=US` se convierte en `Amazon Web Service O\=Amazon.com Inc. L\=Seattle ST\=Washington C\=US`.

Ejemplos de políticas que utilizan variables de política de certificados

El siguiente documento de política permite las conexiones con un ID de cliente que coincida con el número de serie del certificado y la publicación en el tema que coincida con el patrón `${iot:Certificate.Subject.Organization}/device-stats/${iot:ClientId}/*`.

> **Important**
>
> Si el certificado X.509 no incluye un atributo de certificado concreto, pero la variable de política de certificados correspondiente se utiliza en el documento de política, la evaluación de la política podría provocar un comportamiento inesperado. Esto se debe a que la variable de política que falta no se evalúa en la instrucción de la política. Por ejemplo, si asocia el siguiente documento de política a un certificado que no contiene el atributo `iot:Certificate.Subject.Organization`, las variables de la política de certificado `iot:Certificate.Subject.Organization` no se rellenarán durante la evaluación de la política.

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/${iot:Certificate.SerialNumber}"
   ]
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Publish"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topic/
${iot:Certificate.Subject.Organization}/device-stats/${iot:ClientId}/*"
   ]
  }
```

```
 ]
}
```

También puede usar el operador de condición Null para asegurarse de que las variables de política de certificados utilizadas en una política se rellenen durante la evaluación de la política. El siguiente documento de política permite `iot:Connect` con certificados solo cuando están presentes los atributos número de serie del certificado y nombre común del sujeto del certificado.

Todas las variables de la política de certificados tienen valores de cadena, por lo que se admiten todos los operadores de condición de cadena.

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/*"
   ],
   "Condition": {
    "Null": {
     "iot:Certificate.SerialNumber": "false",
     "iot:Certificate.Subject.CommonName": "false"
    }
   }
  }
 ]
}
```

Prevención de la sustitución confusa entre servicios

El problema de la sustitución confusa es un problema de seguridad en el que una entidad que no tiene permiso para realizar una acción puede obligar a una entidad con más privilegios a realizar la acción. En AWS, la suplantación de identidad entre servicios puede provocar el confuso problema de un diputado. La suplantación entre servicios puede producirse cuando un servicio (el servicio que lleva a cabo las llamadas) llama a otro servicio (el servicio al que se llama). El servicio que lleva a

cabo las llamadas se puede manipular para utilizar sus permisos a fin de actuar en función de los recursos de otro cliente de una manera en la que no debe tener permiso para acceder. Para evitarlo, AWS proporciona herramientas que le ayudan a proteger sus datos para todos los servicios con entidades principales de servicio a las que se les ha dado acceso a los recursos de su cuenta.

Para limitar los permisos que AWS IoT otorga otro servicio al recurso, se recomienda utilizar las claves de contexto de condición `aws:SourceAccount`global `aws:SourceArn`y las claves contextuales globales en las políticas de recursos. Si se utilizan ambas claves contextuales de condición global, el valor `aws:SourceAccount` y la cuenta del valor `aws:SourceArn` deben utilizar el mismo ID de cuenta cuando se utilicen en la misma declaración de política.

La forma más eficaz de protegerse contra el problema de la sustitución confusa es utilizar la clave de contexto de condición global de `aws:SourceArn` con el nombre de recurso de Amazon (ARN) completo del recurso. Para AWS IoT ello, `aws:SourceArn` debe cumplir con el formato: `arn:`*`aws`*`:iot:`*`region`*`:`*`account-id`*`:`*`resource-type/resource-id`* para permisos específicos del recurso o`arn:`*`aws`*`:iot:`*`region`*`:`*`account-id`*`:*`. El identificador del recurso puede ser el nombre o el identificador del recurso permitido o una declaración comodín del recurso permitido. IDs Asegúrese de que *`region`* coincida con su AWS IoT región y con su ID de *`account-id`* cuenta de cliente.

El siguiente ejemplo muestra cómo evitar el confuso problema de los diputados mediante el uso de las claves de contexto `aws:SourceArn` y condición `aws:SourceAccount` global de la política de confianza de AWS IoT roles. Para obtener más ejemplos, consulte Ejemplos detallados de prevención del suplente confuso.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"iot.amazonaws.com"
         },
         "Action":"sts:AssumeRole",
         "Condition":{
            "StringEquals":{
               "aws:SourceAccount":"123456789012"
        },
            "ArnLike":{
               "aws:SourceArn":"arn:aws:iot:us-east-1:123456789012:*"
```

```
        }
         }
      }
   ]
}
```

> **Note**
>
> Si recibe errores de denegación de acceso, puede deberse a que la integración del servicio con AWS Security Token Service (STS) no admita las claves de contexto `aws:SourceArn` y `aws:SourceAccount`.

Ejemplos detallados de prevención del suplente confuso

En esta sección se proporcionan ejemplos detallados de cómo evitar el confuso problema de los diputados mediante el uso de las claves de contexto aws:SourceArn y condición aws:SourceAccount global de la política de confianza de AWS IoT roles.

- Aprovisionamiento de flotas
- JITP
- Proveedor de credenciales

Aprovisionamiento de flotas

Puede configurar el aprovisionamiento de flotas con un recurso de plantilla de aprovisionamiento. Cuando una plantilla de aprovisionamiento hace referencia a un rol de aprovisionamiento, la política de confianza de ese rol puede incluir las claves de condición `aws:SourceArn` y `aws:SourceAccount`. Estas claves limitan los recursos para los que la configuración puede invocar la solicitud `sts:AssumeRole`.

El rol con la siguiente política de confianza solo lo puede asumir la entidad principal de IoT (`iot.amazonaws.com`) para la plantilla de aprovisionamiento especificada en el `SourceArn`.

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

```
         "Effect":"Allow",
         "Principal":{
            "Service":"iot.amazonaws.com"
         },
         "Action":"sts:AssumeRole",
         "Condition":{
            "StringEquals":{
               "aws:SourceAccount":"123456789012"
        },
            "ArnLike":{
               "aws:SourceArn":"arn:aws:iot:us-
east-1:123456789012:provisioningtemplate/example_template"
         }
          }
      }
   ]
}
```

JITP

En el just-in-time aprovisionamiento (JITP), puede utilizar la plantilla de aprovisionamiento como un recurso independiente de la CA o definir el cuerpo de la plantilla y la función como parte de la configuración del certificado de la CA. El valor de la política de confianza `aws:SourceArn` en el AWS IoT rol depende de cómo se defina la plantilla de aprovisionamiento.

Definición de la plantilla de aprovisionamiento como un recurso independiente

Si define la plantilla de aprovisionamiento como un recurso independiente, el valor de `aws:SourceArn` puede ser "`arn:aws:iot:`*`region`*`:`*`account-id`*`:provisioningtemplate/`*`example_template`*"". Puede usar el mismo ejemplo de política en Aprovisionamiento de flotas.

Definición de la plantilla de aprovisionamiento en un certificado de CA

Si define la plantilla de aprovisionamiento en un recurso de certificado de CA, el valor de `aws:SourceArn` puede ser "`arn:aws:iot:`*`region`*`:`*`account-id`*`:cacert/`*`cert_id`*" o "`arn:aws:iot:`*`region`*`:`*`account-id`*`:cacert/`*`*`". Puede usar un comodín si desconoce el identificador del recurso (como el ID de un certificado de CA) en el momento de la creación.

El rol con la siguiente política de confianza solo lo puede asumir la entidad principal de IoT (`iot.amazonaws.com`) para el certificado de CA especificado en el `SourceArn`.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"iot.amazonaws.com"
         },
         "Action":"sts:AssumeRole",
         "Condition":{
            "StringEquals":{
               "aws:SourceAccount":"123456789012"
        },
            "ArnLike":{
               "aws:SourceArn":"arn:aws:iot:us-
east-1:123456789012:cacert/
8ecde6884f3d87b1125ba31ac3fcb13d7016de7f57cc904fe1cb97c6ae98196e"
        }
         }
      }
   ]
}
```

Al crear un certificado de CA, puede hacer referencia a un rol de aprovisionamiento en la configuración de registro. La política de confianza del rol de aprovisionamiento puede utilizar el `aws:SourceArn` para restringir los recursos de los que se puede asumir el rol. Sin embargo, durante la CACertificate llamada de registro inicial para registrar el certificado de CA, no tendría que especificar el ARN del certificado de CA en la `aws:SourceArn` condición.

Para solucionar este problema, es decir, para especificar la política de confianza de la función de aprovisionamiento para el certificado de CA específico con el que está registrado AWS IoT Core, puede hacer lo siguiente:

- En primer lugar, llama a Register CACertificate sin proporcionar el `RegistrationConfig` parámetro.
- Una vez registrado el certificado de CA AWS IoT Core, llame a Update CACertificate en él.

En la CACertificate llamada de actualización, proporcione una `RegistrationConfig` que incluya la política de confianza de la función de aprovisionamiento `aws:SourceArn` establecida en el ARN del certificado de CA recién registrado.

Proveedor de credenciales

Para el proveedor de AWS IoT Core credenciales, utilice el mismo Cuenta de AWS que utiliza para crear el alias del rol y especifique una sentencia que coincida con el ARN del recurso del tipo de recurso rolealias en. `aws:SourceAccount` `aws:SourceArn` Al crear un rol de IAM para usarlo con el proveedor de AWS IoT Core credenciales, debe incluir en la `aws:SourceArn` condición de cualquier alias ARNs de rol que pueda necesitar asumir el rol, autorizando así la solicitud de servicios cruzados. `sts:AssumeRole`

El rol con la siguiente política de confianza solo lo puede asumir la entidad principal del proveedor de credenciales de AWS IoT Core (`credentials.iot.amazonaws.com`) para el roleAlias especificado en el `SourceArn`. Si una entidad principal intenta obtener las credenciales de un alias de rol distinto del especificado en la condición `aws:SourceArn`, se denegará la solicitud, incluso si ese otro alias de rol hace referencia al mismo rol de IAM.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "credentials.iot.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnLike": {
          "aws:SourceArn": "arn:aws:iot:us-
east-1:123456789012:rolealias/example_rolealias"
        }
      }
    }
  ]
```

```
}
```

AWS IoT Core ejemplos de políticas

Los ejemplos de políticas de esta sección ilustran los documentos de políticas que se utilizan para completar tareas comunes en AWS IoT Core. Puede utilizarlos como ejemplos para empezar a crear las políticas para sus soluciones.

Los ejemplos de esta sección utilizan estos elementos de política:

- the section called “AWS IoT Core acciones políticas”
- the section called “AWS IoT Core recursos de acción”
- the section called “Ejemplos de políticas basadas en identidades”
- the section called “Variables de política básicas AWS IoT Core”
- the section called “Variables de AWS IoT Core política de certificados X.509”

Ejemplos de políticas de esta sección:

- Ejemplos de política de conexión
- Ejemplos de política de publicación/suscripción
- Ejemplos de políticas de conexión y publicación
- Ejemplos de políticas de mensajes retenidos
- Ejemplos de políticas de certificado
- Ejemplos de políticas de objeto
- Ejemplo de políticas de trabajos básica

Ejemplos de política de conexión

La siguiente política deniega el permiso al cliente IDs `client1` y `client2` al que conectarse AWS IoT Core, al tiempo que permite que los dispositivos se conecten mediante un ID de cliente. El ID de cliente coincide con el nombre de un elemento registrado en el AWS IoT Core registro y adjunto al principal que se utiliza para la conexión:

> **Note**
>
> En el caso de los dispositivos registrados, recomendamos que utilice variables de política de objetos para las acciones de `Connect` y que los asocie a la entidad principal que se utiliza para la conexión.

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Deny",
   "Action": [
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/client1",
    "arn:aws:iot:us-east-1:123456789012:client/client2"
   ]
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/${iot:Connection.Thing.ThingName}"
   ],
   "Condition": {
    "Bool": {
     "iot:Connection.Thing.IsAttached": "true"
    }
   }
  }
 ]
}
```

La siguiente política otorga permiso para conectarse AWS IoT Core con el ID de cliente`client1`. Este ejemplo de política es para dispositivos no registrados.

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/client1"
   ]
  }
 ]
}
```

Ejemplos de políticas de sesiones persistentes de MQTT

`connectAttributes` le permite especificar qué atributos quiere usar en su mensaje de conexión en sus políticas de IAM, como `PersistentConnect` y `LastWill`. Para obtener más información, consulte Uso de connectAttributes.

La siguiente política permite conectarse con la característica `PersistentConnect`:

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": "arn:aws:iot:us-east-1:123456789012:client/client1",
   "Condition": {
    "ForAllValues:StringEquals": {
     "iot:ConnectAttributes": [
      "PersistentConnect"
     ]
    }
   }
  }
```

```
 ]
}
```

La siguiente política no permite `PersistentConnect`, pero se permiten otras características:

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": "arn:aws:iot:us-east-1:123456789012:client/client1",
   "Condition": {
    "ForAllValues:StringNotEquals": {
     "iot:ConnectAttributes": [
      "PersistentConnect"
     ]
    }
   }
  }
 ]
}
```

La política anterior también puede expresarse utilizando `StringEquals`, se permite cualquier otra característica, incluidas las nuevas:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": "arn:aws:iot:us-east-1:123456789012:client/client1"
        },
        {
```

```
            "Effect": "Deny",
            "Action": [
                "iot:Connect"
            ],
            "Resource": "*",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "iot:ConnectAttributes": [
                        "PersistentConnect"
                    ]
            }
        }
        }
    ]
}
```

La siguiente política permite la conexión mediante `PersistentConnect` y `LastWill`, pero no se permite ninguna otra característica nueva:

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": "arn:aws:iot:us-east-1:123456789012:client/client1",
   "Condition": {
    "ForAllValues:StringEquals": {
     "iot:ConnectAttributes": [
      "PersistentConnect",
      "LastWill"
     ]
    }
   }
  }
 ]
}
```

La siguiente política permite la conexión limpia por parte de clientes con o sin `LastWill`, no se permitirá ninguna otra característica:

```
{
    "Version":"2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "iot:Connect"
        ],
        "Resource": "arn:aws:iot:us-east-1:123456789012:client/client1",
        "Condition": {
            "StringEquals": {
                "iot:ConnectAttributes": "LastWill"
        }
        }
    }]
}
```

La siguiente política solo permite la conexión mediante las características predeterminadas:

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": "arn:aws:iot:us-east-1:123456789012:client/client1",
   "Condition": {
    "ForAllValues:StringEquals": {
     "iot:ConnectAttributes": []
    }
   }
  }
 ]
}
```

La siguiente política permite conectarse únicamente con `PersistentConnect`, se permite cualquier característica nueva siempre que la conexión utilice `PersistentConnect`:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": "arn:aws:iot:us-east-1:123456789012:client/client1",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "iot:ConnectAttributes": [
                        "PersistentConnect"
                    ]
            }
        }
        }
    ]
}
```

La siguiente política establece que la conexión debe tener uso de `PersistentConnect` y `LastWill`, no se permite ninguna característica nueva:

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": "arn:aws:iot:us-east-1:123456789012:client/client1",
   "Condition": {
    "ForAllValues:StringEquals": {
     "iot:ConnectAttributes": [
      "PersistentConnect",
      "LastWill"
```

```
    ]
   }
  }
 },
 {
  "Effect": "Deny",
  "Action": [
   "iot:Connect"
  ],
  "Resource": "*",
  "Condition": {
   "ForAllValues:StringEquals": {
    "iot:ConnectAttributes": [
     "PersistentConnect"
    ]
   }
  }
 },
 {
  "Effect": "Deny",
  "Action": [
   "iot:Connect"
  ],
  "Resource": "*",
  "Condition": {
   "ForAllValues:StringEquals": {
    "iot:ConnectAttributes": [
     "LastWill"
    ]
   }
  }
 },
 {
  "Effect": "Deny",
  "Action": [
   "iot:Connect"
  ],
  "Resource": "*",
  "Condition": {
   "ForAllValues:StringEquals": {
    "iot:ConnectAttributes": []
   }
  }
 }
```

```
 ]
}
```

La siguiente política no debe tener `PersistentConnect` pero puede tener `LastWill`, no se permite ninguna otra característica nueva:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "iot:Connect"
            ],
            "Resource": "*",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "iot:ConnectAttributes": [
                        "PersistentConnect"
                    ]
            }
        }
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": "arn:aws:iot:us-east-1:123456789012:client/client1",
            "Condition": {
                "ForAllValues:StringEquals": {
                    "iot:ConnectAttributes": [
                        "LastWill"
                    ]
            }
        }
        }
    ]
}
```

La siguiente política solo permite que los clientes que tengan un `LastWill` con tema "`my/lastwill/topicName`", se permite cualquier característica siempre que utilice el tema `LastWill`:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": "arn:aws:iot:us-east-1:123456789012:client/client1",
            "Condition": {
                "ArnEquals": {
                "iot:LastWillTopic": "arn:aws:iot:us-east-1:123456789012:topic/my/
lastwill/topicName"
            }
        }
        }
    ]
}
```

La siguiente política permite conectarse únicamente con un `LastWillTopic` específico, se permite cualquier característica nueva siempre que la conexión utilice `LastWillTopic`:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": "arn:aws:iot:us-east-1:123456789012:client/client1",
            "Condition": {
                "ArnEquals": {
                "iot:LastWillTopic": "arn:aws:iot:us-east-1:123456789012:topic/my/
lastwill/topicName"
            }
        }
```

```
        },
        {
            "Effect": "Deny",
            "Action": [
                "iot:Connect"
            ],
            "Resource": "*",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "iot:ConnectAttributes": [
                        "PersistentConnect"
                    ]
            }
        }
        }
    ]
}
```

Ejemplos de política de publicación/suscripción

La política que utilices depende de la forma a la que te conectes AWS IoT Core. Puedes conectarte AWS IoT Core mediante un cliente MQTT, HTTP o WebSocket. Al conectarse con un cliente MQTT, se autenticará con un certificado X.509. Cuando se conecta a través de HTTP o el WebSocket protocolo, se autentica con Signature Version 4 y Amazon Cognito.

> **Note**
>
> En el caso de los dispositivos registrados, recomendamos que utilice variables de política de objetos para las acciones de `Connect` y que los asocie a la entidad principal que se utiliza para la conexión.

En esta sección:

- Uso de caracteres comodín en MQTT y en las políticas AWS IoT Core
- Políticas para publicar, suscribirse y recibir mensajes sobre temas to/from específicos
- Políticas para publicar, suscribirse y recibir mensajes y temas con un prefijo específico to/from
- Políticas para publicar, suscribirse y recibir mensajes: temas específicos de cada dispositivo to/from

- Políticas para publicar, suscribirse y recibir mensajes, to/from temas con el atributo «thing» en el nombre del tema
- Políticas para denegar la publicación de mensajes en subtemas del nombre de un tema
- Políticas para denegar la recepción de mensajes de subtemas del nombre de un tema
- Políticas para suscribirse a temas con caracteres comodín de MQTT
- Políticas para HTTP y clientes WebSocket

Uso de caracteres comodín en MQTT y en las políticas AWS IoT Core

MQTT y AWS IoT Core las políticas tienen caracteres comodín diferentes y debe elegirlos tras considerarlos detenidamente. En MQTT, los caracteres comodín se # utilizan en + los filtros de temas de MQTT para suscribirse a varios nombres de temas. AWS IoT Core las políticas utilizan * y ? como caracteres comodín y siguen las convenciones de las políticas de IAM. En un documento de política el * representa cualquier combinación de caracteres y un signo de interrogación ? representa cualquier carácter único. En los documentos de política, los caracteres comodín MQTT + y # se consideran caracteres sin ningún significado especial. Para describir varios nombres de temas y filtros de temas en el atributo `resource` de una política, utilice los caracteres comodín * y ? en lugar de los caracteres comodín de MQTT.

Al elegir caracteres comodín para usarlos en un documento de política, tenga en cuenta que el carácter * no se limita a un único nivel de tema. El carácter + se limita a un único nivel de tema en un filtro de temas de MQTT. Para limitar una especificación de comodín a un único nivel de filtro de temas de MQTT, considere la posibilidad de utilizar varios caracteres ?. Para obtener más información sobre el uso de caracteres comodín en un recurso de políticas y más ejemplos de sus coincidencias, consulte Uso de caracteres comodín en un recurso. ARNs

En la siguiente tabla se muestran los distintos caracteres comodín que se utilizan en MQTT y en las políticas de AWS IoT Core de los clientes de MQTT.

Carácter comodín	¿Es un carácter comodín de MQTT?	Ejemplo en MQTT	¿Es un carácter AWS IoT Core comodín de política	Ejemplo de AWS IoT Core políticas para clientes de MQTT
#	Sí	`some/#`	No	N/A
+	Sí	`some/+/topic`	No	N/A

*	No	N/A	Sí	`topicfilter/some/*/topic`
				
				`topicfilter/some/sensor*/topic`

?	No	N/A	Sí	`topic/some/?????/topic`
				
				`topicfilter/some/sensor???/topic`

Políticas para publicar, suscribirse y recibir mensajes sobre temas to/from específicos

A continuación se muestran ejemplos de dispositivos registrados y no registrados para publicar, suscribir y recibir mensajes to/from del tema denominado «some_specific_topic». Los ejemplos también destacan que `Publish` y `Receive` utilizan “topic” como recurso y `Subscribe` utiliza “topicfilter” como recurso.

Registered devices

Para los dispositivos registrados en el AWS IoT Core registro, la siguiente política permite que los dispositivos se conecten con un ClientiD que coincida con el nombre de un elemento del registro. También proporciona permisos `Publish`, `Subscribe` y `Receive` para el tema denominado “some_specific_topic”.

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

```
"Statement": [
 {
  "Effect": "Allow",
  "Action": [
   "iot:Connect"
  ],
  "Resource": [
   "arn:aws:iot:us-east-1:123456789012:client/${iot:Connection.Thing.ThingName}"
  ],
  "Condition": {
   "Bool": {
    "iot:Connection.Thing.IsAttached": "true"
   }
  }
 },
 {
  "Effect": "Allow",
  "Action": [
   "iot:Publish"
  ],
  "Resource": [
   "arn:aws:iot:us-east-1:123456789012:topic/some_specific_topic"
  ]
 },
 {
  "Effect": "Allow",
  "Action": [
   "iot:Subscribe"
  ],
  "Resource": [
   "arn:aws:iot:us-east-1:123456789012:topicfilter/some_specific_topic"
  ]
 },
 {
  "Effect": "Allow",
  "Action": [
   "iot:Receive"
  ],
  "Resource": [
   "arn:aws:iot:us-east-1:123456789012:topic/some_specific_topic"
  ]
 }
]
```

```
}
```

Unregistered devices

En el caso de los dispositivos que no están registrados en el AWS IoT Core registro, la siguiente política permite que los dispositivos se conecten mediante el ClientID1, el ClientID2 o el ClientID3. También proporciona permisos `Publish`, `Subscribe` y `Receive` para el tema denominado “some_specific_topic”.

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

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

Políticas para publicar, suscribirse y recibir mensajes y temas con un prefijo específico to/from

A continuación se muestran ejemplos de dispositivos registrados y no registrados para publicar, suscribir y recibir mensajes con to/from temas con el prefijo «topic_prefix».

> **Note**
>
> Observe el uso del carácter comodín * en este ejemplo. Si bien el carácter comodín * es útil para permitir el uso de varios nombres de temas en una sola instrucción, puede tener consecuencias imprevistas, ya que proporciona a los dispositivos más privilegios de los necesarios. Por lo tanto, le recomendamos que utilice únicamente el carácter comodín * después de considerarlo detenidamente.

Registered devices

Para los dispositivos registrados en el AWS IoT Core registro, la siguiente política permite que los dispositivos se conecten con un ClientiD que coincida con el nombre de un elemento del registro. También proporciona permisos `Publish`, `Subscribe` y `Receive` para los temas con el prefijo «topic_prefix».

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

```
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/${iot:Connection.Thing.ThingName}"
   ],
   "Condition": {
    "Bool": {
     "iot:Connection.Thing.IsAttached": "true"
    }
   }
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Publish",
    "iot:Receive"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topic/topic_prefix*"
   ]
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Subscribe"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topicfilter/topic_prefix*"
   ]
  }
 ]
}
```

Unregistered devices

En el caso de los dispositivos que no están registrados en el AWS IoT Core registro, la siguiente política permite que los dispositivos se conecten mediante el ClientID1, el ClientID2 o el ClientID3. También proporciona permisos `Publish`, `Subscribe` y `Receive` para los temas con el prefijo «topic_prefix».

```
{
```

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

Políticas para publicar, suscribirse y recibir mensajes: temas específicos de cada dispositivo to/from

A continuación, se muestran ejemplos de dispositivos registrados y no registrados para publicar, suscribir y recibir mensajes to/from sobre temas específicos del dispositivo en cuestión.

Registered devices

Para los dispositivos registrados en el AWS IoT Core registro, la siguiente política permite que los dispositivos se conecten con un ClientiD que coincida con el nombre de un elemento del registro. Da permiso para publicar en un tema de objeto específico (`sensor/device/${iot:Connection.Thing.ThingName}`) y también para suscribirse y recibir información del tema de objeto específico (`command/device/${iot:Connection.Thing.ThingName}`). Si el nombre del elemento en el registro es «cosa1", el dispositivo podrá publicar en el tema" 1". sensor/device/thing1". The device will also be able to subscribe to and receive from the topic "command/device/thing

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/${iot:Connection.Thing.ThingName}"
   ],
   "Condition": {
    "Bool": {
     "iot:Connection.Thing.IsAttached": "true"
    }
   }
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Publish"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topic/sensor/device/
${iot:Connection.Thing.ThingName}"
   ]
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Subscribe"
```

```
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topicfilter/command/device/
${iot:Connection.Thing.ThingName}"
   ]
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Receive"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topic/command/device/
${iot:Connection.Thing.ThingName}"
   ]
  }
 ]
}
```

Unregistered devices

En el caso de los dispositivos que no están registrados en el AWS IoT Core registro, la siguiente política permite que los dispositivos se conecten mediante el ClientID1, el ClientID2 o el ClientID3. Da permiso para publicar en un tema de cliente específico (`sensor/device/${iot:ClientId}`) y también para suscribirse y recibir información del tema de cliente específico (`command/device/${iot:ClientId}`). Si el dispositivo se conecta con ClientID como ClientID1, podrá publicar en el tema "1". sensor/device/clientId El dispositivo también podrá suscribirse y recibir información del tema `device/clientId1/command`.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "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"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/sensor/device/
${iot:Connection.Thing.ThingName}"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/command/device/
${iot:Connection.Thing.ThingName}"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/command/device/
${iot:Connection.Thing.ThingName}"
            ]
        }
    ]
}
```

Políticas para publicar, suscribirse y recibir mensajes, to/from temas con el atributo «thing» en el nombre del tema

A continuación se muestra un ejemplo de dispositivos registrados para publicar, suscribir y recibir mensajes, to/from temas cuyos nombres incluyen atributos de cosas.

> **Note**
>
> Los atributos de objeto solo existen para los dispositivos registrados en el registro de AWS IoT Core . No existe un ejemplo correspondiente a los dispositivos no registrados.

Registered devices

Para los dispositivos registrados en el AWS IoT Core registro, la siguiente política permite que los dispositivos se conecten con un ClientiD que coincida con el nombre de un elemento del registro. Da permiso para publicar en el tema (`sensor/${iot:Connection.Thing.Attributes[version]}`) y suscribirse y recibir información del tema (`command/${iot:Connection.Thing.Attributes[location]}`) si el nombre del tema incluye atributos de objeto. Si el nombre del objeto en el registro es «`version=v1y»location=Seattle`, el dispositivo podrá publicar en el tema "»sensor/v1", and subscribe to and receive from the topic "command/Seattle.

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

```
    "arn:aws:iot:us-east-1:123456789012:topic/sensor/
${iot:Connection.Thing.Attributes[version]}"
   ]
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Subscribe"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topicfilter/command/
${iot:Connection.Thing.Attributes[location]}"
   ]
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Receive"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topic/command/
${iot:Connection.Thing.Attributes[location]}"
   ]
  }
 ]
}
```

Unregistered devices

Como los atributos de objetos solo existen para los dispositivos registrados en el registro de AWS IoT Core , no existe un ejemplo correspondiente para los dispositivos no registrados.

Políticas para denegar la publicación de mensajes en subtemas del nombre de un tema

A continuación, se muestran ejemplos de dispositivos registrados y no registrados para publicar mensajes en todos los temas, excepto en algunos subtemas.

Registered devices

Para los dispositivos registrados en el AWS IoT Core registro, la siguiente política permite que los dispositivos se conecten con un ClientiD que coincida con el nombre de un elemento del

registro. Da permiso para publicar en todos los temas con el prefijo «departament/», pero no en el subtema «department/admins».

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/${iot:Connection.Thing.ThingName}"
   ],
   "Condition": {
    "Bool": {
     "iot:Connection.Thing.IsAttached": "true"
    }
   }
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Publish"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topic/department/*"
   ]
  },
  {
   "Effect": "Deny",
   "Action": [
    "iot:Publish"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topic/department/admins"
   ]
  }
 ]
}
```

Unregistered devices

En el caso de los dispositivos que no están registrados en el AWS IoT Core registro, la siguiente política permite que los dispositivos se conecten mediante el ClientID1, el ClientID2 o el ClientID3. Da permiso para publicar en todos los temas con el prefijo «departament/», pero no en el subtema «department/admins».

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

Políticas para denegar la recepción de mensajes de subtemas del nombre de un tema

A continuación se muestran ejemplos para que dispositivos registrados y no registrados se suscriban y reciban mensajes de temas con prefijos específicos, excepto determinados subtemas.

Registered devices

Para los dispositivos registrados en el AWS IoT Core registro, la siguiente política permite que los dispositivos se conecten con un ClientiD que coincida con el nombre de un elemento del registro. La política permite a los dispositivos suscribirse a cualquier tema con el prefijo «topic_prefix». Al utilizarla `NotResource` en la declaración for`iot:Receive`, permitimos que el dispositivo reciba mensajes de todos los temas a los que esté suscrito, excepto los temas con el prefijo «topic_». prefix/restricted". For example, with this policy, devices can subscribe to "topic_prefix/topic1" and even "topic_prefix/restricted", however, they will only receive messages from the topic "topic_prefix/topic1" and no messages from the topic "topic_prefix/restricted

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/${iot:Connection.Thing.ThingName}"
   ],
   "Condition": {
    "Bool": {
     "iot:Connection.Thing.IsAttached": "true"
    }
   }
  },
  {
   "Effect": "Allow",
   "Action": "iot:Subscribe",
   "Resource": "arn:aws:iot:us-east-1:123456789012:topicfilter/topic_prefix/*"
  },
  {
   "Effect": "Allow",
   "Action": "iot:Receive",
```

```
   "NotResource": "arn:aws:iot:us-east-1:123456789012:topic/topic_prefix/
restricted/*"
  }
 ]
}
```

Unregistered devices

En el caso de los dispositivos que no estén registrados en el AWS IoT Core registro, la siguiente política permite que los dispositivos se conecten mediante el ClientID1, el ClientID2 o el ClientID3. La política permite a los dispositivos suscribirse a cualquier tema con el prefijo «topic_prefix». Al utilizarla `NotResource` en la declaración for`iot:Receive`, permitimos que el dispositivo reciba mensajes de todos los temas a los que esté suscrito, excepto los temas con el prefijo «topic_». prefix/restricted". For example, with this policy, devices can subscribe to "topic_prefix/topic1" and even "topic_prefix/restricted". However, they will only receive messages from the topic "topic_prefix/topic1" and no messages from the topic "topic_prefix/restricted

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

```
            "NotResource": "arn:aws:iot:us-east-1:123456789012:topic/
topic_prefix/restricted/*"
        }
    ]
}
```

Políticas para suscribirse a temas con caracteres comodín de MQTT

Los caracteres comodín + y # de MQTT se tratan como cadenas literales, pero no se tratan como caracteres comodín cuando se utilizan en políticas. AWS IoT Core En MQTT, + y # se tratan como caracteres comodín solo cuando se suscribe a un filtro de tema, pero se tratan como cadenas literales en todos los demás contextos. Le recomendamos que utilice estos caracteres comodín MQTT únicamente como parte de AWS IoT Core las políticas tras considerarlos detenidamente.

A continuación se muestran ejemplos de elementos registrados y no registrados que utilizan caracteres comodín de MQTT en las políticas. AWS IoT Core Estos caracteres comodín se tratan como cadenas literales.

Registered devices

Para los dispositivos registrados en el AWS IoT Core registro, la siguiente política permite que los dispositivos se conecten con un ClientiD que coincida con el nombre de un elemento del registro. La política permite que los dispositivos se suscriban a los temas «department/+/employees» y «location/#». Como los términos + y # se consideran cadenas literales en AWS IoT Core las políticas, los dispositivos pueden suscribirse al tema «departamento/+/empleados», pero no también al tema «». department/engineering/employees". Similarly, devices can subscribe to the topic "location/#" but not to the topic "location/Seattle". However, once the device subscribes to the topic "department/+/employees", the policy will allow them to receive messages from the topic "department/engineering/employees". Similarly, once the device subscribes to the topic "location/#", they will receive messages from the topic "location/Seattle

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

```
    "Resource": [
     "arn:aws:iot:us-east-1:123456789012:client/${iot:Connection.Thing.ThingName}"
    ],
    "Condition": {
     "Bool": {
      "iot:Connection.Thing.IsAttached": "true"
     }
    }
   },
   {
    "Effect": "Allow",
    "Action": "iot:Subscribe",
    "Resource": "arn:aws:iot:us-east-1:123456789012:topicfilter/department/+/
employees"
   },
   {
    "Effect": "Allow",
    "Action": "iot:Subscribe",
    "Resource": "arn:aws:iot:us-east-1:123456789012:topicfilter/location/#"
   },
   {
    "Effect": "Allow",
    "Action": "iot:Receive",
    "Resource": "arn:aws:iot:us-east-1:123456789012:topic/*"
   }
 ]
}
```

Unregistered devices

En el caso de los dispositivos que no están registrados en el AWS IoT Core registro, la siguiente política permite que los dispositivos se conecten mediante el ClientID1, el ClientID2 o el ClientID3. La política permite que los dispositivos se suscriban a los temas de «department/+/employees» y «location/#». Como los términos + y # se consideran cadenas literales en AWS IoT Core las políticas, los dispositivos pueden suscribirse al tema «departamento/+/empleados», pero no también al tema «». department/engineering/employees". Similarly, devices can subscribe to the topic "location/#" but not "location/Seattle". However, once the device subscribes to the topic "department/+/employees", the policy will allow them to receive messages from the topic "department/engineering/employees". Similarly, once the device subscribes to the topic "location/#", they will receive messages from the topic "location/Seattle

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

Políticas para HTTP y clientes WebSocket

Cuando se conecta a través de HTTP o el WebSocket protocolo, se autentica con Signature Version 4 y Amazon Cognito. Las identidades de Amazon Cognito pueden ser autenticadas o no autenticadas. Las identidades autenticadas pertenecen a los usuarios que se han autenticado mediante un proveedor de identidad compatible. Las identidades no autenticadas suelen pertenecer a usuarios invitados que no se autentiquen con un proveedor de identidades. Amazon Cognito

proporciona un identificador y AWS credenciales únicos para admitir identidades no autenticadas. Para obtener más información, consulte the section called “Autorización con identidades de Amazon Cognito”.

Para las siguientes operaciones, AWS IoT Core utiliza AWS IoT Core políticas asociadas a las identidades de Amazon Cognito a través de la `AttachPolicy` API. Esto reduce los permisos asociados al grupo de identidades de Amazon Cognito con identidades autenticadas.

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

Esto significa que una identidad de Amazon Cognito necesita el permiso de la política de rol de IAM y la política de AWS IoT Core . La política de roles de IAM se adjunta al grupo y la AWS IoT Core política a Amazon Cognito Identity a través AWS IoT Core `AttachPolicy` de la API.

Los usuarios autenticados y no autenticados son tipos de identidad diferentes. Si no adjuntas una AWS IoT política a Amazon Cognito Identity, un usuario autenticado no podrá obtener la autorización AWS IoT y no tendrá acceso a AWS IoT los recursos ni a las acciones.

> **Note**
>
> Para otras AWS IoT Core operaciones o para identidades no autenticadas, AWS IoT Core no limita los permisos asociados a la función 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.

HTTP

Para permitir que identidades de Amazon Cognito sin autenticar publiquen mensajes sobre HTTP en un tema específico de la identidad de Amazon Cognito, asocie la política de IAM siguiente al 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/${cognito-
identity.amazonaws.com:sub}"
            ]
        }
    ]
}
```

Para permitir la autenticación de usuarios, asocie la política anterior al rol del grupo de Amazon Cognito Identity y a Amazon Cognito Identity mediante la API. AWS IoT Core AttachPolicy

> **Note**
>
> Al autorizar las identidades de Amazon Cognito AWS IoT Core , tiene en cuenta ambas políticas y concede los privilegios mínimos especificados. Solo se permite una acción si ambas políticas permiten la acción solicitada. Si una de ellas no permite una acción, esa acción no se autoriza.

MQTT

Para permitir que las identidades de Amazon Cognito no autenticadas publiquen mensajes de MQTT WebSocket sobre un tema específico de Amazon Cognito Identity en su cuenta, adjunte la siguiente política de IAM al rol del grupo de Amazon Cognito Identity:

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

```
                "iot:Publish"
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/${cognito-
identity.amazonaws.com:sub}"]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:client/${cognito-
identity.amazonaws.com:sub}"]
        }
    ]
}
```

Para permitir la autenticación de usuarios, asocie la política anterior al rol del grupo de Amazon Cognito Identity y a Amazon Cognito Identity mediante la API. AWS IoT Core AttachPolicy

> **Note**
>
> Al autorizar las identidades de Amazon Cognito AWS IoT Core , tiene en cuenta ambas y concede los privilegios mínimos especificados. Solo se permite una acción si ambas políticas permiten la acción solicitada. Si una de ellas no permite una acción, esa acción no se autoriza.

Ejemplos de políticas de conexión y publicación

En el caso de los dispositivos registrados como elementos del AWS IoT Core registro, la siguiente política concede permiso para conectarse AWS IoT Core con un ID de cliente que coincida con el nombre de la cosa y restringe el dispositivo a la publicación en un tema de MQTT específico para el ID de cliente o el nombre de la cosa. Para que la conexión se realice correctamente, el nombre de la cosa debe estar registrado en el AWS IoT Core registro y autenticarse con una identidad o un principal asociado a la cosa:

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

```
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish"
      ],
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:topic/${iot:Connection.Thing.ThingName}"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect"
      ],
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"
      ]
    }
  ]
}
```

En el caso de los dispositivos que no están registrados como cosas en el AWS IoT Core registro, la siguiente política concede permiso para conectarse AWS IoT Core con un ID de cliente `client1` y restringe el dispositivo a la publicación en un tema de MQTT específico del ID de cliente:

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

```
        "iot:Connect"
      ],
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:client/client1"
      ]
    }
  ]
}
```

Ejemplos de políticas de mensajes retenidos

El uso de mensajes retenidos requiere políticas específicas. Los mensajes retenidos son mensajes MQTT publicados con el indicador RETAIN establecido y almacenados por. AWS IoT Core En esta sección se presentan ejemplos de políticas que permiten el uso común de los mensajes retenidos.

En esta sección:

- Política para conectar y publicar mensajes retenidos
- Política para conectar y publicar mensajes retenidos Will
- Política para enumerar y obtener mensajes retenidos

Política para conectar y publicar mensajes retenidos

Para que un dispositivo publique los mensajes retenidos, debe poder conectarse, publicar (cualquier mensaje MQTT) y publicar los mensajes retenidos en MQTT. La siguiente política concede estos permisos para el tema: `device/sample/configuration` al cliente **`device1`**. Para ver otro ejemplo que concede permiso para conectarse, consulte the section called “Ejemplos de políticas de conexión y publicación”.

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/device1"
   ]
```

```
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Publish",
    "iot:RetainPublish"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topic/device/sample/configuration"
   ]
  }
 ]
}
```

Política para conectar y publicar mensajes retenidos Will

Los clientes pueden configurar un mensaje que AWS IoT Core se publicará cuando el cliente se desconecte inesperadamente. MQTT llama a este tipo de mensaje un mensaje Will. Un cliente debe tener una condición adicional agregada a su permiso de conexión para poder incluirlo.

El siguiente documento de política otorga a todos los clientes permiso para conectarse y publicar un mensaje Will, identificado por su tema, `will`, que AWS IoT Core también conservará.

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/device1"
   ],
   "Condition": {
    "ForAllValues:StringEquals": {
     "iot:ConnectAttributes": [
      "LastWill"
     ]
    }
   }
```

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

Política para enumerar y obtener mensajes retenidos

Los servicios y las aplicaciones pueden acceder a los mensajes retenidos sin necesidad de utilizar un cliente MQTT llamando a `ListRetainedMessages` y `GetRetainedMessage`. Los servicios y las aplicaciones que invocan estas acciones deben autorizarse mediante una política como la del ejemplo siguiente.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:ListRetainedMessages"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/device1"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:GetRetainedMessage"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/foo"
            ]
```

```
        }
    ]
}
```

Ejemplos de políticas de certificado

En el caso de los dispositivos registrados en el AWS IoT Core registro, la siguiente política otorga permiso para conectarse AWS IoT Core con un ID de cliente que coincida con el nombre de una cosa y para publicar en un tema cuyo nombre sea igual al `certificateId` del certificado que el dispositivo utilizó para autenticarse:

```
{
    "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": ["arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"]
        }
    ]
}
```

En el caso de los dispositivos que no están registrados en el AWS IoT Core registro, la siguiente política otorga permiso para conectarse AWS IoT Core con el cliente IDs `client1client2`,, `client3` y publicar en un tema cuyo nombre sea igual al `certificateId` del certificado que el dispositivo utilizó para autenticarse:

```
{
    "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": [
                "arn:aws:iot:us-east-1:123456789012:client/client1",
                "arn:aws:iot:us-east-1:123456789012:client/client2",
                "arn:aws:iot:us-east-1:123456789012:client/client3"
            ]
        }
    ]
}
```

Para los dispositivos registrados en el AWS IoT Core registro, la siguiente política otorga permiso para conectarse AWS IoT Core con un ID de cliente que coincida con el nombre de la cosa y para publicar en un tema cuyo nombre sea igual al `CommonName` campo del asunto del certificado que el dispositivo utilizó para autenticarse:

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

```
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"]
        }
    ]
}
```

> **Note**
>
> En este ejemplo, el nombre común del sujeto del certificado se utiliza como identificador de tema, partiendo del supuesto de que el nombre común del sujeto es único para cada certificado registrado. Si los certificados se comparten entre varios dispositivos, el nombre común del asunto es el mismo para todos los dispositivos que comparten este certificado, lo que permite publicar privilegios en el mismo tema desde varios dispositivos (no se recomienda).

Para los dispositivos que no están registrados en el AWS IoT Core registro, la siguiente política otorga permiso para conectarse AWS IoT Core con el cliente IDs `client1client2`,, `client3` y publicar en un tema cuyo nombre sea igual al `CommonName` campo del asunto del certificado que el dispositivo utilizó para autenticarse:

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

```
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/client1",
                "arn:aws:iot:us-east-1:123456789012:client/client2",
                "arn:aws:iot:us-east-1:123456789012:client/client3"
            ]
        }
    ]
}
```

> **Note**
>
> En este ejemplo, el nombre común del sujeto del certificado se utiliza como identificador de tema, partiendo del supuesto de que el nombre común del sujeto es único para cada certificado registrado. Si los certificados se comparten entre varios dispositivos, el nombre común del asunto es el mismo para todos los dispositivos que comparten este certificado, lo que permite publicar privilegios en el mismo tema desde varios dispositivos (no se recomienda).

Para los dispositivos registrados en el AWS IoT Core registro, la siguiente política otorga permiso para conectarse AWS IoT Core con un ID de cliente que coincida con el nombre de la cosa y para publicar en un tema cuyo nombre lleve el prefijo `admin/` cuando el certificado utilizado para autenticar el dispositivo tenga el `Subject.CommonName.2` campo establecido en`Administrator`:

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

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

Para los dispositivos que no están registrados en el AWS IoT Core registro, la siguiente política otorga permiso para conectarse AWS IoT Core con el cliente IDs `client1` `client3` y publicar en un tema cuyo nombre lleve el prefijo `admin/` cuando el certificado utilizado para autenticar el dispositivo tenga el `Subject.CommonName.2` campo configurado en`Administrator`: `client2`

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/client1",
                "arn:aws:iot:us-east-1:123456789012:client/client2",
                "arn:aws:iot:us-east-1:123456789012:client/client3"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/admin/*"],
            "Condition": {
```

```
                "StringEquals": {
                    "iot:Certificate.Subject.CommonName.2": "Administrator"
            }
        }
        }
    ]
}
```

En el caso de los dispositivos registrados en el AWS IoT Core registro, la siguiente política permite que un dispositivo utilice su nombre para publicar sobre un tema específico, lo que `admin/` consiste en indicar `ThingName` cuándo el certificado utilizado para autenticar el dispositivo tiene alguno de sus `Subject.CommonName` campos configurado en: `Administrator`

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

En el caso de los dispositivos que no estén registrados en el AWS IoT Core registro IDs `client1client2`, la siguiente política otorga permiso para conectarse AWS IoT Core con el cliente `client3` y publicar en el tema `admin` cuando el certificado utilizado para autenticar el dispositivo tenga alguno de sus `Subject.CommonName` campos configurado en: `Administrator`

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/client1",
                "arn:aws:iot:us-east-1:123456789012:client/client2",
                "arn:aws:iot:us-east-1:123456789012:client/client3"
            ]
        },
        {
            "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 que un dispositivo se conecte si el certificado con el que se autenticó AWS IoT Core está adjunto al elemento para el que se está evaluando la política:

```
{
```

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

La siguiente política permite que un dispositivo publique si el certificado está asociado a un objeto con un tipo de objeto en particular y si el objeto tiene un atributo de `attributeName` con valor `attributeValue`. Para obtener más información acerca de las variables de políticas, consulte Variables de política de objeto.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish"
      ],
      "Resource": "arn:aws:iot:us-east-1:123456789012:topic/device/stats",
      "Condition": {
        "StringEquals": {
          "iot:Connection.Thing.Attributes[attributeName]": "attributeValue",
          "iot:Connection.Thing.ThingTypeName": "Thing_Type_Name"
        },
        "Bool": {
          "iot:Connection.Thing.IsAttached": "true"
        }
      }
    }
  ]
```

```
}
```

La siguiente política permite que un dispositivo publique en un tema que comience con un atributo de ese objeto. Si el certificado del dispositivo no está asociado al objeto, esta variable no se resolverá y generará un error de acceso denegado. Para obtener más información acerca de las variables de políticas, consulte Variables de política de objeto.

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

Ejemplo de políticas de trabajos básica

En este ejemplo se muestran las declaraciones de políticas necesarias para que un objetivo de trabajo que sea un único dispositivo pueda recibir una solicitud de trabajo y comunicar el estado de ejecución del trabajo con AWS IoT.

*us-west-2:57EXAMPLE833*Sustitúyalo por el tuyo Región de AWS, dos puntos (:)) y tu Cuenta de AWS número de 12 dígitos y, a continuación, *uniqueThingName* sustitúyelo por el nombre del recurso que representa el dispositivo. AWS IoT

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

```
            ],
            "Resource": [
                "arn:aws:iot:us-west-2:123456789012:client/uniqueThingName"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-west-2:123456789012:topic/test/dc/pubtopic",
                "arn:aws:iot:us-west-2:123456789012:topic/$aws/events/job/*",
                "arn:aws:iot:us-west-2:123456789012:topic/$aws/events/jobExecution/
*",
                "arn:aws:iot:us-west-2:123456789012:topic/$aws/
things/uniqueThingName/jobs/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-west-2:123456789012:topicfilter/test/dc/subtopic",
                "arn:aws:iot:us-west-2:123456789012:topicfilter/$aws/events/
jobExecution/*",
                "arn:aws:iot:us-west-2:123456789012:topicfilter/$aws/
things/uniqueThingName/jobs/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:us-west-2:123456789012:topic/test/dc/subtopic",
                "arn:aws:iot:us-west-2:123456789012:topic/$aws/
things/uniqueThingName/jobs/*"
            ]
        },
        {
```

```
            "Effect": "Allow",
            "Action": [
                "iotjobsdata:DescribeJobExecution",
                "iotjobsdata:GetPendingJobExecutions",
                "iotjobsdata:StartNextPendingJobExecution",
                "iotjobsdata:UpdateJobExecution"
            ],
            "Resource": [
                "arn:aws:iot:us-west-2:123456789012:topic/$aws/
things/uniqueThingName"
            ]
        }
    ]
}
```

Autorización con identidades de Amazon Cognito

Hay dos tipos de identidades de Amazon Cognito: autenticadas y sin autenticar. Si la aplicación admite identidades de Amazon Cognito no autenticadas, no se realiza ninguna autenticación, por lo que no se sabe quién es el usuario.

Identidades no autenticadas: en el caso de las identidades de Amazon Cognito no autenticadas, se conceden permisos asociando un rol de IAM a un grupo de identidades no autenticadas. Le recomendamos que solo conceda acceso a aquellos recursos que desee que estén disponibles para usuarios desconocidos.

> **Important**
>
> Para los usuarios de Amazon Cognito no autenticados que se conecten AWS IoT Core a, le recomendamos que dé acceso a recursos muy limitados en las políticas de IAM.

Identidades autenticadas: para las identidades autenticadas de Amazon Cognito, debe especificar los permisos en dos lugares.

- Asocie una política de IAM al grupo autenticado de identidades de Amazon Cognito y
- Adjunte una AWS IoT Core política a Amazon Cognito Identity (usuario autenticado).

Ejemplos de políticas para usuarios de Amazon Cognito autenticados y no autenticados que se conectan a AWS IoT Core

El siguiente ejemplo muestra los permisos tanto en la política de IAM como en la política de IoT de una identidad de Amazon Cognito. El usuario autenticado quiere publicar en un tema específico de un dispositivo (por ejemplo). device/DEVICE_ID/status

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

El siguiente ejemplo muestra los permisos en una política de IAM de un rol no autenticado de Amazon Cognito. El usuario no autenticado desea publicar en temas no específicos de un dispositivo que no requieren autenticación.

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

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

GitHub ejemplos

Los siguientes ejemplos de aplicaciones web GitHub muestran cómo incorporar políticas adjuntas a los usuarios autenticados en el proceso de registro y autenticación de los usuarios.

- Uso de la aplicación web MQTT publish/subscribe React y AWS AmplifySDK de AWS IoT Device para JavaScript
- Aplicación web MQTT publish/subscribe React que utiliza AWS AmplifySDK de AWS IoT Device para JavaScript, la y una función Lambda

Amplify es un conjunto de herramientas y servicios que le ayuda a crear aplicaciones web y móviles que se integran con AWS los servicios. Para obtener más información acerca de Amplify, consulte la Documentación de Amplify Framework.

En los dos ejemplos se realizan los siguientes pasos.

1. Cuando un usuario se registra para obtener una cuenta, la aplicación crea una identidad y un grupo de usuarios de Amazon Cognito.
2. Cuando un usuario se autentica, la aplicación crea y asocia una política a la identidad. Esto otorga al usuario permisos de publicación y suscripción.

3. El usuario puede utilizar la aplicación para publicar temas de MQTT y suscribirse a ellos.

El primer ejemplo utiliza la operación de la API `AttachPolicy` directamente dentro de la operación de autenticación. El siguiente ejemplo muestra cómo implementar esta llamada a la API dentro de una aplicación web de React que usa Amplify y el SDK de AWS IoT Device para JavaScript.

```
function attachPolicy(id, policyName) {
    var Iot = new AWS.Iot({region: AWSConfiguration.region, apiVersion:
 AWSConfiguration.apiVersion, endpoint: AWSConfiguration.endpoint});
    var params = {policyName: policyName, target: id};

    console.log("Attach IoT Policy: " + policyName + " with cognito identity id: " +
 id);
    Iot.attachPolicy(params, function(err, data) {
         if (err) {
               if (err.code !== 'ResourceAlreadyExistsException') {
                  console.log(err);
               }
          }
         else  {
            console.log("Successfully attached policy with the identity", data);
         }
     });
}
```

Este código aparece en el AuthDisplayarchivo.js.

El segundo ejemplo implementa la operación de la API `AttachPolicy` en una función de Lambda. El siguiente ejemplo muestra cómo Lambda utiliza esta llamada a la API.

```
iot.attachPolicy(params, function(err, data) {
     if (err) {
           if (err.code !== 'ResourceAlreadyExistsException') {
              console.log(err);
              res.json({error: err, url: req.url, body: req.body});
           }
      }
     else  {
        console.log(data);
```

```
        res.json({success: 'Create and attach policy call succeed!', url: req.url,
body: req.body});
    }
});
```

Este código aparece dentro de la función `iot.GetPolicy` en el archivo app.js.

> **Note**
>
> Cuando llama a la función con AWS las credenciales que obtiene a través de los grupos de identidades de Amazon Cognito, el objeto de contexto de la función Lambda contiene un valor para. `context.cognito_identity_id` Para obtener más información, consulte lo siguiente.
>
> - AWS Lambda objeto de contexto en Node.js
> - AWS Lambda objeto de contexto en Python
> - AWS Lambda objeto de contexto en Ruby
> - AWS Lambda objeto de contexto en Java
> - AWS Lambda objeto de contexto en Go
> - AWS Lambda objeto de contexto en C#
> - AWS Lambda objeto de contexto en PowerShell

Autorizar llamadas directas a los AWS servicios mediante el proveedor de AWS IoT Core credenciales

Los dispositivos pueden usar certificados X.509 para conectarse AWS IoT Core mediante protocolos de autenticación mutua TLS. Otros AWS servicios no admiten la autenticación basada en certificados, pero se pueden llamar a ellos mediante AWS credenciales en formato AWS Signature Version 4. El algoritmo Signature de la versión 4 normalmente requiere que la persona que llama tenga un identificador de clave de acceso y una clave de acceso secreta. AWS IoT Core tiene un proveedor de credenciales que le permite utilizar el certificado X.509 integrado como identidad única del dispositivo para AWS autenticar las solicitudes. Así se elimina la necesidad de almacenar un ID de clave de acceso y una clave de acceso secreta en el dispositivo.

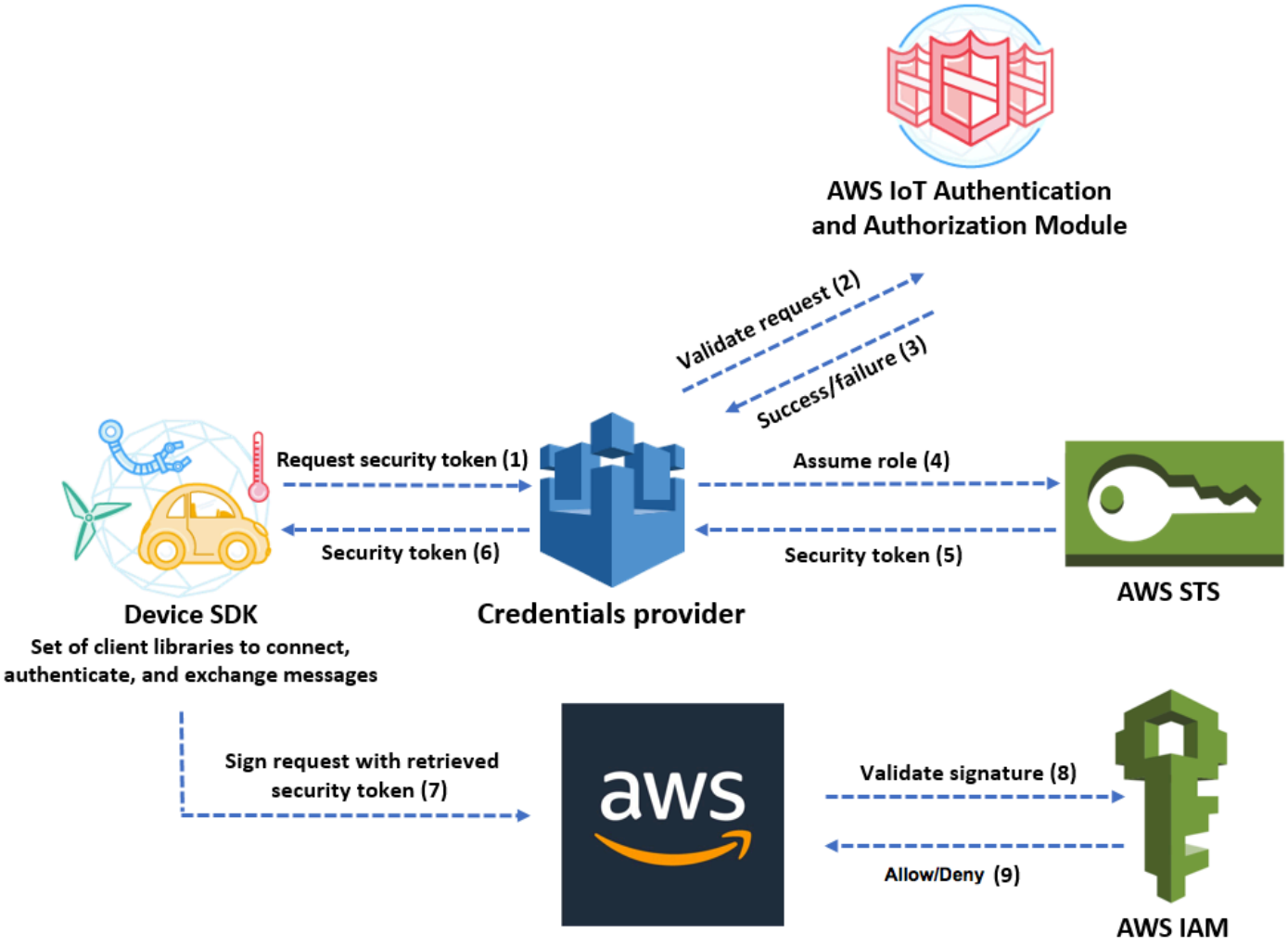
El proveedor de credenciales autentica a un intermediario mediante un certificado X.509 y emite un token de seguridad temporal con privilegios limitados. El token se puede usar para firmar y autenticar cualquier solicitud. AWS Esta forma de autenticar sus AWS solicitudes requiere que cree y configure un rol AWS Identity and Access Management (de IAM) y que adjunte las políticas de IAM adecuadas al rol para que el proveedor de credenciales pueda asumir el rol en su nombre. Para obtener más información sobre AWS IoT Core e IAM, consulte Gestión de identidad y acceso para AWS IoT.

AWS IoT requiere que los dispositivos envíen la extensión de indicación del nombre del servidor (SNI) al protocolo Transport Layer Security (TLS) y proporcionen la dirección completa del punto final en el campo. `host_name` El campo `host_name` debe contener el punto de conexión al que está llamando y debe ser:

- El `endpointAddress` devuelto por `aws iot describe-endpoint --endpoint-type iot:CredentialProvider`.

Las conexiones que intenten realizar los dispositivos sin el valor de `host_name` correcto fallarán.

En el siguiente diagrama se ilustra el flujo de trabajo del proveedor de credenciales.



1. El AWS IoT Core dispositivo realiza una solicitud HTTPS al proveedor de credenciales para obtener un token de seguridad. La solicitud incluye el certificado X.509 del dispositivo para autenticación.
2. El proveedor de credenciales reenvía la solicitud al módulo de AWS IoT Core autenticación y autorización para validar el certificado y comprobar que el dispositivo tiene permiso para solicitar el token de seguridad.
3. Si el certificado es válido y tiene permiso para solicitar un token de seguridad, el módulo de AWS IoT Core autenticación y autorización se devuelve correctamente. De lo contrario, envía una excepción al dispositivo.
4. Después de validar correctamente el certificado, el proveedor de credenciales invoca a AWS Security Token Service (AWS STS) para asumir el rol de IAM que creó para el mismo.
5. AWS STS devuelve un token de seguridad temporal con privilegios limitados al proveedor de credenciales.

6. El proveedor de credenciales devuelve el token de seguridad al dispositivo.
7. El dispositivo usa el token de seguridad para firmar una AWS solicitud con la versión 4 de AWS Signature.
8. El servicio solicitado invoca a IAM para validar la firma y autorizar la solicitud frente a políticas de acceso asociadas al rol de IAM que creó para el proveedor de credenciales.
9. Si IAM valida la firma correctamente y autoriza la solicitud, la solicitud se realiza correctamente. De lo contrario, IAM envía una excepción.

En la siguiente sección se describe cómo utilizar un certificado para obtener un token de seguridad. Se ha escrito suponiendo que ya ha registrado un dispositivo y creado y activado su propio certificado para el mismo.

Cómo utilizar un certificado para obtener un token de seguridad

1. Configure el rol de IAM que el proveedor de credenciales asume en nombre de su dispositivo. Asocie la siguiente política de confianza al rol.

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

   Para cada AWS servicio al que desee llamar, adjunte una política de acceso al rol. El proveedor de credenciales admite las siguientes variables de políticas:

   - `credentials-iot:ThingName`
   - `credentials-iot:ThingTypeName`
   - `credentials-iot:AwsCertificateId`

   Cuando el dispositivo proporciona el nombre de objeto en su solicitud a un servicio de AWS , el proveedor de credenciales agrega `credentials-iot:ThingName` y `credentials-iot:ThingTypeName` como variables de contexto para el token de seguridad. El proveedor

de credenciales proporciona `credentials-iot:AwsCertificateId` como una variable de contexto incluso si el dispositivo no proporciona el nombre de la cosa en la solicitud. Transfiera el nombre de la cosa como valor del encabezado de solicitud HTTP `x-amzn-iot-thingname`.

Estas tres variables solo funcionan para las políticas de IAM, no para las políticas de AWS IoT Core .

2. Asegúrese de que el usuario que realiza el siguiente paso (la creación de un alias de rol) tiene permiso para transferir a AWS IoT Core el rol que se acaba de crear. La siguiente política otorga ambos `iam:GetRole` `iam:PassRole` permisos a un AWS usuario. El permiso `iam:GetRole` permite al usuario obtener información acerca del rol que acaba de crear. El `iam:PassRole` permiso permite al usuario transferir la función a otro AWS servicio.

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "iam:GetRole",
            "iam:PassRole"
        ],
        "Resource": "arn:aws:iam::123456789012:role/your role name"
    }
}
```

3. Cree un alias de AWS IoT Core rol. El dispositivo que va a realizar llamadas directas a AWS los servicios debe saber qué función ARN utilizar al conectarse. AWS IoT Core La codificación de forma rígida de un ARN de rol no es una buena solución, ya que requiere actualizar el dispositivo cada vez que el ARN del rol cambia. Una solución mejor consiste en utilizar la API `CreateRoleAlias` para crear un alias de rol que apunte al ARN del rol. Si el ARN del rol cambia, solo tiene que actualizar el alias de rol. No es necesario realizar ningún cambio en el dispositivo. Esta API adopta los siguiente parámetros:

`roleAlias`

Obligatorio. Una cadena arbitraria que identifica el alias del rol. Sirve como clave principal en el modelo de datos del alias de rol. Contiene 1-128 caracteres y debe incluir únicamente caracteres alfanuméricos y los símbolos =, @ y -. Se permiten los caracteres alfabéticos en mayúsculas y minúsculas. Los nombres de alias de roles distinguen entre mayúsculas y minúsculas.

`roleArn`

Obligatorio. El ARN del rol al que hace referencia el alias del rol.

`credentialDurationSeconds`

Opcional. El tiempo (en segundos) que la credencial es válida. El valor mínimo es de 900 segundos (15 minutos). El valor máximo es de 43 200 segundos (12 horas). El valor predeterminado es de 3600 segundos (1 hora).

> **Important**
>
> El proveedor de AWS IoT Core credenciales puede emitir una credencial con una duración máxima de 43.200 segundos (12 horas). Hacer que la credencial sea válida durante un máximo de 12 horas puede ayudar a reducir el número de llamadas al proveedor de credenciales al almacenar la credencial en caché durante más tiempo. El valor `credentialDurationSeconds` debe ser menor o igual que la duración máxima de la sesión del rol de IAM a la que hace referencia el alias del rol. Para obtener más información, consulte Modificación de la duración máxima de sesión (AWS API) de un rol en la Guía del usuario de AWS Identity and Access Management.

Para obtener más información acerca esta API, consulte CreateRoleAlias.

4. Asocie una política al certificado de dispositivo. La política asociada al certificado del dispositivo debe conceder permiso al dispositivo para asumir el rol. Para ello, puede conceder permiso para la acción `iot:AssumeRoleWithCertificate` al alias de rol, como en el ejemplo siguiente.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iot:AssumeRoleWithCertificate",
            "Resource": "arn:aws:iot:us-east-1:123456789012:rolealias/your role
 alias"
        }
    ]
}
```

5. Realice una solicitud HTTPS al proveedor de credenciales para obtener un token de seguridad. Proporcione la información siguiente:

   - Certificado: dado que se trata de una solicitud HTTP a través de autenticación mutua de TLS, debe proporcionar el certificado y la clave privada a su cliente al realizar la solicitud. Utilice el mismo certificado y la misma clave privada con los que registró el certificado AWS IoT Core.

     Para asegurarse de que el dispositivo se está comunicando con AWS IoT Core (y no con un servicio que se hace pasar por él), consulte Autenticación de servidor, siga los enlaces para descargar los certificados de CA correspondientes y, a continuación, cópielos en su dispositivo.
   - RoleAlias: el nombre del alias de rol que creó para el proveedor de credenciales. Los nombres de alias de rol distinguen mayúsculas de minúsculas y deben coincidir con el alias de rol creado en AWS IoT Core.
   - ThingName: El nombre de la cosa que creaste cuando la registraste AWS IoT Core . Esto se transfiere como valor del encabezado HTTP `x-amzn-iot-thingname`. Este valor solo es obligatorio si se utilizan los atributos de las cosas como variables de política en nuestras AWS IoT Core políticas de IAM.

     > **Note**
     >
     > Los ThingNamedatos que proporcione `x-amzn-iot-thingname` deben coincidir con el nombre del recurso AWS IoT Thing asignado a un certificado. Si no coincide, se devuelve un error 403.

   Ejecute el siguiente comando AWS CLI para obtener el punto final del proveedor de credenciales para su Cuenta de AWS. Para obtener más información acerca esta API, consulte DescribeEndpoint. Para ver los puntos de conexión activados para FIPS, consulte Puntos de conexión de proveedores de credenciales de AWS IoT Core.

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

   El siguiente objeto de JSON es la salida de ejemplo del comando describe-endpoint. Contiene el `endpointAddress` que utiliza para solicitar un token de seguridad.

   ```
   {
   ```

```
    "endpointAddress": "your_aws_account_specific_prefix.credentials.iot.your
 region.amazonaws.com"
}
```

Utilice el punto de conexión para realizar una solicitud HTTPS al proveedor de credenciales para devolver un token de seguridad. El ejemplo de comando siguiente utiliza `curl`, pero puede utilizar cualquier cliente HTTP.

> **Note**
>
> El nombre de RoleAlias distingue entre mayúsculas y minúsculas y debe coincidir con el alias del rol creado en. AWS IoT

```
curl --cert your certificate --key your private key -H "x-amzn-iot-thingname: your
 thing name" --cacert AmazonRootCA1.pem https://your endpoint /role-aliases/your
 role alias/credentials
```

Este comando devuelve un objeto de token de seguridad objeto que contiene un `accessKeyId`, una `secretAccessKey`, un `sessionToken` y un vencimiento. El siguiente objeto de JSON es la salida de ejemplo del comando `curl`.

```
   {"credentials":{"accessKeyId":"access key","secretAccessKey":"secret access
 key","sessionToken":"session token","expiration":"2018-01-18T09:18:06Z"}}
```

A continuación, puede usar los `sessionToken` valores `accessKeyIdsecretAccessKey`, y para firmar las solicitudes de los AWS servicios. Para ver una end-to-end demostración, consulte Cómo eliminar la necesidad de AWS credenciales codificadas en los dispositivos mediante el uso de la entrada del blog sobre el proveedor de AWS IoT credenciales en el blog de AWS seguridad.

Acceso entre cuentas con IAM

AWS IoT Core le permite permitir a un director publicar o suscribirse a un tema que esté definido en un tema que Cuenta de AWS no sea propiedad del director. 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 administrada por el cliente tal como se describe en Creación de políticas de IAM, tal y como haría para otros usuarios y certificados en su cuenta de Cuenta de AWS.

En el caso de los dispositivos registrados en el AWS IoT Core registro, la siguiente política otorga permiso a los dispositivos que se conecten AWS IoT Core mediante un ID de cliente que coincida con el nombre del dispositivo y a que publiquen en el `my/topic/`*`thing-name`* lugar donde *`thing-name`* está el nombre del dispositivo:

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

En el caso de los dispositivos que no estén registrados en el AWS IoT Core registro, la siguiente política permite que un dispositivo utilice el nombre de la cosa `client1` registrado en el AWS IoT Core registro de su cuenta (123456789012) para conectarse a AWS IoT Core un tema específico del ID de cliente cuyo nombre lleve el prefijo: `my/topic/`

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/client1"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/my/topic/${iot:ClientId}"
            ]
        }
    ]
}
```

A continuación, siga los pasos que se indican en Creación de un rol para delegar permisos a un usuario de IAM. Especifique el ID de la cuenta de 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::567890123456:user/MyUser"
        },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Protección de datos en AWS IoT Core

El modelo de responsabilidad AWS compartida modelo se aplica a la protección de datos en AWS IoT Core. Como se describe en este modelo, AWS es responsable de proteger la infraestructura global que ejecuta todos los Nube de AWS. Eres responsable de mantener el control sobre el contenido alojado en esta infraestructura. También eres responsable de las tareas de administración y configuración de seguridad para los Servicios de AWS que utiliza. Para obtener más información sobre la privacidad de los datos, consulte las Preguntas frecuentes sobre la privacidad de datos. Para obtener información sobre la protección de datos en Europa, consulte la publicación de blog sobre el Modelo de responsabilidad compartida de AWS y GDPR en el Blog de seguridad de AWS .

Con fines de protección de datos, le recomendamos que proteja Cuenta de AWS las credenciales y configure los usuarios individuales con AWS IAM Identity Center o AWS Identity and Access Management (IAM). De esta manera, solo se otorgan a cada usuario los permisos necesarios para cumplir sus obligaciones laborales. También recomendamos proteger sus datos de la siguiente manera:

- Utiliza la autenticación multifactor (MFA) en cada cuenta.
- Se utiliza SSL/TLS para comunicarse con AWS los recursos. Exigimos TLS 1.2 y recomendamos TLS 1.3.
- Configure la API y el registro de actividad de los usuarios con AWS CloudTrail. Para obtener información sobre el uso de CloudTrail senderos para capturar AWS actividades, consulte Cómo trabajar con CloudTrail senderos en la Guía del AWS CloudTrail usuario.
- Utilice soluciones de AWS cifrado, junto con todos los controles de seguridad predeterminados Servicios de AWS.
- Utiliza servicios de seguridad administrados avanzados, como Amazon Macie, que lo ayuden a detectar y proteger la información confidencial almacenada en Amazon S3.

- Si necesita módulos criptográficos validados por FIPS 140-3 para acceder a AWS través de una interfaz de línea de comandos o una API, utilice un punto final FIPS. Para obtener más información sobre los puntos de conexión de FIPS disponibles, consulte Estándar de procesamiento de la información federal (FIPS) 140-3.

Se recomienda encarecidamente no introducir nunca información confidencial o sensible, como por ejemplo, direcciones de correo electrónico de clientes, en etiquetas o campos de formato libre, tales como el campo Nombre. Esto incluye cuando trabaja con AWS IoT o Servicios de AWS utiliza la consola, la API o. AWS CLI AWS SDKs Cualquier dato que introduzca en etiquetas o campos de formato libre utilizados para los nombres se pueden emplear para los registros de facturación o diagnóstico. Si proporciona una URL a un servidor externo, recomendamos encarecidamente que no incluya información de credenciales en la URL a fin de validar la solicitud para ese servidor.

Para obtener más información sobre la protección de datos, consulte la entrada de blog relativa al modelo de responsabilidad compartida de AWS y GDPR en el blog de seguridad de AWS .

AWS IoT los dispositivos recopilan datos, los manipulan y, a continuación, los envían a otro servicio web. Es posible que decida almacenar algunos datos en su dispositivo durante un breve período de tiempo. Usted es responsable de proporcionar cualquier protección de datos sobre dichos datos en reposo. Cuando el dispositivo envía datos a AWS IoT, lo hace a través de una conexión TLS, como se explica más adelante en esta sección. AWS IoT los dispositivos pueden enviar datos a cualquier AWS servicio. Para obtener más información sobre la seguridad de los datos de cada servicio, consulta la documentación de ese servicio. AWS IoT se puede configurar para escribir registros en los CloudWatch registros y registrar las llamadas a la AWS IoT API en AWS CloudTrail. Para obtener más información sobre la seguridad de los datos de estos servicios, consulte Autenticación y control de acceso para Amazon CloudWatch y Cifrado de archivos de CloudTrail registro con claves AWS KMS gestionadas.

Cifrado de datos en AWS IoT

De forma predeterminada, todos los AWS IoT datos en tránsito y en reposo están cifrados. Los datos en tránsito se cifran mediante TLS y los datos en reposo se cifran mediante claves AWS propias. AWS IoT admite la administración por el cliente AWS KMS keys (claves KMS) desde el Servicio de administración de AWS claves (AWS KMS). Sin embargo, Device Advisor y AWS IoT Wireless utilizan únicamente y Clave propiedad de AWS para cifrar los datos de los clientes.

Seguridad del transporte en AWS IoT Core

TLS (Transport Layer Security, Seguridad de la capa de transporte) es un protocolo criptográfico diseñado para una comunicación segura a través de una red informática. La puerta de enlace de AWS IoT Core dispositivos requiere que los clientes cifren todas las comunicaciones en tránsito mediante TLS para las conexiones de los dispositivos a la puerta de enlace. El TLS se utiliza para garantizar la confidencialidad de los protocolos de aplicación (MQTT, HTTP y) compatibles. WebSocket AWS IoT Core TLS se admite y está disponible en una serie de lenguajes de programación y sistemas operativos. AWS El servicio específico AWS cifra los datos que contiene. Para obtener más información sobre el cifrado de datos en otros AWS servicios, consulta la documentación de seguridad de ese servicio.

Contenido

- Protocolos TLS
- Políticas de seguridad
- Notas importantes para la seguridad del transporte en AWS IoT Core
- Seguridad de transporte para LoRa dispositivos inalámbricos WAN

Protocolos TLS

AWS IoT Core admite las siguientes versiones del protocolo TLS:

- TLS 1.3
- TLS 1.2

Con AWS IoT Core, puede configurar los ajustes de TLS (para TLS 1.2 y TLS 1.3) en las configuraciones de dominio. Para obtener más información, consulte ???.

Políticas de seguridad

Una política de seguridad es una combinación de protocolos TLS y sus cifrados que determina qué protocolos y cifrados se admiten durante las negociaciones de TLS entre un cliente y un servidor. Puede configurar sus dispositivos para que utilicen políticas de seguridad predefinidas en función de sus necesidades. Tenga en cuenta que AWS IoT Core no admite políticas de seguridad personalizadas.

Al conectarlos a ellos, puedes elegir una de las políticas de seguridad predefinidas para tus dispositivos AWS IoT Core. Los nombres de las políticas de seguridad predefinidas más recientes AWS IoT Core incluyen información sobre la versión según el año y el mes en que se publicaron. La política de seguridad predefinida que se utiliza de forma predeterminada es `IoTSecurityPolicy_TLS13_1_2_2022_10`. Para especificar una política de seguridad, puede utilizar la AWS IoT consola o la AWS CLI. Para obtener más información, consulte ???.

En la siguiente tabla se describen las políticas de seguridad predefinidas más recientes compatibles con AWS IoT Core . Se ha eliminado `IotSecurityPolicy_` de los nombres de política en la fila de encabezado para que quepan.

Política de seguridad	TLS13_1_3_2022_10	TLS13_1_2_2022_10	TLS12_1_2_2022_10	TLS12_1_0_2016_01*		TLS12_1_0_2015_01*	
Puerto TCP	443/8443/8883	443/8443/8883	443/8443/8883	443	8443/8883	443	8443/8883
Protocolos TLS							
TLS 1.2		✓	✓	✓	✓	✓	✓
TLS 1.3	✓	✓					
Cifrados TLS							
TLS_AES_128_GCM_SHA256	✓	✓					
TLS_AES_256_GCM_SHA384	✓	✓					
TLS_CHACHA20_POLY1305_SHA256	✓	✓					

Política de seguridad	TLS13_1_3_2022_10	TLS13_1_2_2022_10	TLS12_1_2_2022_10	TLS12_1_0_2016_01*		TLS12_1_0_2015_01*	
ECDHE-RSA- -GCM-AES128 SHA256		✓	✓	✓	✓	✓	✓
ECDHE-RSA-AES128 -SHA256		✓	✓	✓	✓	✓	✓
ECDHE-RSA-AES128 -SHA		✓	✓	✓	✓	✓	✓
ECDHE-RSA-AES256 -GCM-SHA384		✓	✓	✓	✓	✓	✓
ECDHE-RSA-AES256 -SHA384		✓	✓	✓	✓	✓	✓
ECDHE-RSA-AES256 -SHA		✓	✓	✓	✓	✓	✓

Política de seguridad	TLS13_1_3_2022_10	TLS13_1_2_2022_10	TLS12_1_2_2022_10	TLS12_1_0_2016_01*		TLS12_1_0_2015_01*	
AES128-GCM-SHA256		✓	✓	✓	✓	✓	✓
AES128-SHA256		✓	✓	✓		✓	✓
AES128-SHA		✓	✓	✓	✓	✓	✓
AES256-GCM-SHA384		✓	✓	✓	✓	✓	✓
AES256-SHA256		✓	✓	✓	✓	✓	✓
AES256-SHA		✓	✓	✓	✓	✓	✓
DHE-RSA--SHA AES256						✓	✓
ECDHE-ECDSA-AES128-GCM-SHA256		✓	✓	✓	✓	✓	✓
ECDHE-ECDSA-- AES128 SHA256		✓	✓	✓	✓	✓	✓

Política de seguridad	TLS13_1_3_2022_10	TLS13_1_2_2022_10	TLS12_1_2_2022_10	TLS12_1_0_2016_01*		TLS12_1_0_2015_01*	
ECDHE-ECDSA--SHA AES128		✓	✓	✓	✓	✓	✓
ECDHE-ECDSA--GCM-AES256 SHA384		✓	✓	✓	✓	✓	✓
ECDHE-ECDSA-- AES256 SHA384		✓	✓	✓	✓	✓	✓
ECDHE-ECDSA--SHA AES256		✓	✓	✓	✓	✓	✓

Note

`TLS12_1_0_2016_01`solo está disponible en las siguientes direcciones Regiones de AWS: ap-east-1, ap-northeast-2, ap-south-1, ap-southeast-2, ca-central-1, cn-north-1, cn-northwest-1, eu-northwest-1, eu-north-1, eu-west-2, eu-west-3, me-south-1 us-east-1, us-east-1, us-east-2, -1, -2, us-west-1. us-gov-west us-gov-west
`TLS12_1_0_2015_01`solo está disponible en las siguientes direcciones Regiones de AWS: ap-northeast-1, ap-southeast-1, eu-central-1, eu-west-1, us-east-1, us-west-2.

Notas importantes para la seguridad del transporte en AWS IoT Core

En el caso de los dispositivos que se conectan AWS IoT Core mediante MQTT, TLS cifra la conexión entre los dispositivos y el intermediario y AWS IoT Core utiliza la autenticación del cliente TLS para identificar los dispositivos. Para obtener más información, consulta Autenticación del cliente. En el caso de los dispositivos que se conectan AWS IoT Core mediante HTTP, el TLS cifra la conexión entre los dispositivos y el intermediario, y la autenticación se delega a la versión 4 de Signature. AWS Para obtener más información, consulte Firma de solicitudes con Signature Version 4 en la Referencia general de AWS .

Al conectar dispositivos a AWS IoT Core, no es obligatorio enviar la extensión de indicación del nombre del servidor (SNI), pero se recomienda encarecidamente. Para utilizar características como el registro multicuenta, los dominios personalizados, los puntos de conexión de VPC y las políticas de TLS configuradas, debe utilizar la extensión SNI y proporcionar la dirección completa del punto de conexión en el campo `host_name`. El campo `host_name` debe contener el punto de conexión al que está llamando. El punto de conexión tiene que ser uno de los siguientes:

- El valor de `endpointAddress` devuelto por `aws iot describe-endpoint --endpoint-type iot:Data-ATS`
- El valor de `domainName` devuelto por `aws iot describe-domain-configuration --domain-configuration-name "domain_configuration_name"`

Las conexiones que intenten realizar dispositivos con un `host_name` valor incorrecto o no válido fallarán. AWS IoT Core registrará los errores CloudWatch para el tipo de autenticación de autenticación personalizada.

AWS IoT Core no admite la extensión SessionTicket TLS.

Seguridad de transporte para LoRa dispositivos inalámbricos WAN

LoRaLos dispositivos WAN siguen las prácticas de seguridad descritas en LoRaWAN™ SECURITY: un documento técnico preparado para la LoRa Alianza™ por Gemalto, Actility y Semtech.

Para obtener más información sobre la seguridad del transporte con dispositivos LoRa WAN, consulte Seguridad de los datos y el transporte de la LoRa WAN.

Cifrado de datos en AWS IoT

La protección de datos se refiere a la protección de los datos mientras están en tránsito (a medida que viajan y vienen AWS IoT Core) y en reposo (mientras están almacenados en dispositivos u otros AWS servicios). Todos los datos enviados a AWS IoT Core se envían a través de una conexión TLS mediante MQTT, HTTPS y WebSocket los protocolos, lo que los hace seguros de forma predeterminada mientras están en tránsito. AWS IoT Core recopila datos de los dispositivos y, a continuación, los envía a otros AWS servicios para su posterior procesamiento. Para obtener más información acerca del cifrado de datos en otros servicios de AWS , consulte la documentación de seguridad de ese servicio. Para obtener más información, consulte Cifrado de datos en reposo.

FreeRTOS proporciona una biblioteca PKCS #11 que resume el almacenamiento de claves, el acceso a objetos criptográficos y la administración de sesiones. Es su responsabilidad utilizar esta biblioteca para cifrar los datos en reposo en sus dispositivos. Para obtener más información, consulte Biblioteca de estándar de criptografía de clave pública (PKCS) 11 de FreeRTOS.

Cifrado de datos en reposo en AWS IoT Core

De forma predeterminada, todos los AWS IoT Core datos en reposo se cifran mediante claves AWS propias. AWS IoT Core también admite claves simétricas administradas por el cliente desde AWS Key Management Service (AWS KMS). Con las claves gestionadas por el cliente, puedes crear, poseer y gestionar las AWS KMS claves de tu AWS cuenta. AWS IoT Core utilizará sus claves de KMS para cifrar sus datos en reposo. Tiene pleno control sobre estas claves de KMS, incluida la creación y el mantenimiento de sus políticas de claves. También puede configurar las políticas de IAM para los roles que acceden AWS KMS a fin de controlar los permisos de estas claves.

AWS claves propias

AWS las claves propias son un conjunto de claves de KMS que un AWS servicio posee y administra para su uso en varias AWS cuentas. AWS los servicios pueden usar claves AWS propias para proteger sus datos. De forma predeterminada, AWS IoT Core cifra los datos en reposo mediante claves AWS propias. Estas claves las administra el servicio. No puedes ver, administrar ni usar las claves AWS propias. Sin embargo, no es necesario realizar ninguna acción para proteger estas claves.

Para obtener más información sobre las claves AWS propias, consulta las claves AWS propias en la Guía para AWS Key Management Service desarrolladores.

Claves administradas por el cliente

Las claves administradas por el cliente son claves de KMS de su AWS cuenta que usted crea, posee y administra. Tiene pleno control sobre estas claves de AWS KMS , incluida la creación y el mantenimiento de sus políticas de claves. También puede configurar las políticas de IAM para los roles que acceden AWS KMS a fin de controlar los permisos de estas claves. Puede configurarlo AWS IoT Core para usar claves KMS administradas por el cliente para cifrar sus datos.

Para obtener más información acerca de las claves administradas por el cliente, consulte Claves administradas por el cliente en la Guía para desarrolladores de AWS Key Management Service .

Para habilitar la entrada de claves administradas por el cliente AWS IoT Core, sigue estos pasos:

Temas

- Paso 1: Crear una clave administrada por el cliente
- Paso 2: Cree una función de IAM para conceder AWS IoT Core permisos de uso de la clave KMS
- Paso 3: Opte por las claves gestionadas por el cliente en AWS IoT Core
- Paso 4: Se requieren permisos adicionales para las operaciones AWS IoT Core del plano de control
- Paso 5: administración de claves
- Paso 6: supervisión del estado de las claves

Paso 1: Crear una clave administrada por el cliente

Puede crear una clave simétrica gestionada por el cliente mediante la AWS KMS consola o los comandos AWS KMS CLI. La opción `keySpec` debe ser `SYMMETRIC_DEFAULT` y la opción `keyUsage` debe ser `ENCRYPT_DECRYPT`.

> **Note**
>
> AWS IoT Core solo admite AWS KMS claves con especificaciones `SYMMETRIC_DEFAULT` clave y uso de claves para `ENCRYPT_DECRYPT` claves administradas por el cliente.

El siguiente es un ejemplo de AWS CLI comando para crear una clave KMS que se pueda usar con las claves administradas AWS IoT Core por el cliente.

```
aws kms create-key --key-spec SYMMETRIC_DEFAULT --key-usage ENCRYPT_DECRYPT --region
 us-west-2
```

A continuación, se muestra un ejemplo del resultado del comando.

```
{
    "KeyMetadata": {
        "AWSAccountId": "111122223333",
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "Arn": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "CreationDate": "2024-09-19T11:45:23.982000-07:00",
        "Enabled": true,
        "Description": "",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "KeyState": "Enabled",
        "Origin": "AWS_KMS",
        "KeyManager": "CUSTOMER",
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "KeySpec": "SYMMETRIC_DEFAULT",
        "EncryptionAlgorithms": [
            "SYMMETRIC_DEFAULT"
        ],
        "MultiRegion": false
    }
}
```

Para obtener más información, consulte Creación de una clave simétrica administrada por el cliente en la Guía para desarrolladores de AWS Key Management Service .

Política de claves

Al crear una clave administrada por el cliente, puede especificar una política de claves. Las políticas de clave controlan el acceso a la clave administrada por el cliente. Cada clave administrada por el cliente debe tener exactamente una política de clave, que contiene instrucciones que determinan quién puede usar la clave y cómo puede utilizarla. Para obtener más información, consulte Políticas de claves en la Guía para desarrolladores de AWS Key Management Service .

AWS IoT Core utiliza una función de IAM en su cuenta para acceder a la clave gestionada por el cliente. Si utiliza una política de claves personalizadas, asegúrese de que el rol de IAM creado en esta clave tenga los siguientes permisos:

- kms:DescribeKey
- kms:Decrypt
- kms:Encrypt
- kms:GenerateDataKeyWithoutPlaintext
- kms:ReEncryptTo
- kms:ReEncryptFrom

Paso 2: Cree una función de IAM para conceder AWS IoT Core permisos de uso de la clave KMS

Si quieres AWS IoT Core usar la clave de KMS que has creado para cifrar tus datos en reposo, también tienes que crear un rol de IAM en tu cuenta, que AWS IoT Core puedes asumir que accederá a la clave de KMS.

El rol debe tener la siguiente política de confianza para poder AWS IoT Core asumirlo.

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Principal": {
            "Service": "iot.amazonaws.com"
        },
        "Action": "sts:AssumeRole",
        "Condition": {
            "StringEquals": {
                "aws:SourceAccount": "111122223333"
            },
            "ArnLike": {
                "aws:SourceArn": "arn:aws:iot:us-west-2:111122223333:*"
            }
        }
    }
}
```

Asegúrese de que las políticas de IAM asociadas al rol de IAM tengan los siguientes permisos en la clave de KMS:

- kms:DescribeKey
- kms:Decrypt

- `kms:Encrypt`
- `kms:GenerateDataKeyWithoutPlaintext`
- `kms:ReEncryptTo`
- `kms:ReEncryptFrom`

A continuación, se muestra un ejemplo de política de IAM con los permisos necesarios para las claves administradas por el cliente.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowIoTToAccessKMSResource",
            "Effect": "Allow",
            "Action": [
                "kms:DescribeKey",
                "kms:Decrypt",
                "kms:Encrypt",
                "kms:ReEncryptTo",
                "kms:ReEncryptFrom",
                "kms:GenerateDataKeyWithoutPlaintext"
            ],
            "Resource": [
                "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
            ],
            "Condition": {
                "StringEquals": {
                    "kms:EncryptionContext:aws-crypto-ec:vendor": "iot.amazonaws.com"
                }
            }
        }
    ]
}
```

Para obtener más información, consulte Creación de un rol para delegar permisos a un usuario de IAM en la Guía del usuario de AWS Identity and Access Management .

Paso 3: Opte por las claves gestionadas por el cliente en AWS IoT Core

Tras completar todos los pasos anteriores, ejecute el comando de la CLI `update-encryption-configuration` para habilitar el uso de las claves administradas por el cliente en AWS IoT Core. Al optar por las claves administradas por el cliente, todos AWS IoT Core los recursos de su AWS cuenta se cifrarán con la AWS KMS clave especificada.

1. Para habilitar el AWS IoT Core uso de claves administradas por el cliente AWS CLI, ejecute el comando `update-encryption-configuration` CLI.

```
aws iot update-encryption-configuration --encryption-type "CUSTOMER_MANAGED_KMS_KEY"
 \
--kms-access-role-arn "arn:aws:iam::111122223333:role/myrole" \
--kms-key-arn "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab" --region us-west-2
```

2. Para verificar el AWS IoT Core uso de las claves administradas por el cliente AWS CLI, ejecute el comando `describe-encryption-configuration` CLI:

```
aws iot describe-encryption-configuration --region us-west-2
```

   Si ha activado las claves gestionadas por el cliente AWS IoT Core, el resultado puede tener el siguiente aspecto:

```
{
    "encryptionType": "CUSTOMER_MANAGED_KMS_KEY",
    "kmsKeyArn": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "kmsAccessRoleArn": "arn:aws:iam::111122223333:role/myrole",
    "configurationDetails": {
        "configurationStatus": "HEALTHY"
    },
    "lastModifiedDate": "2024-09-26T22:01:02.365000-07:00"
}
```

   El campo `lastModifiedDate` indica la fecha en la que se actualizó por última vez la configuración del cifrado.

   Si no ha habilitado las claves administradas por el cliente, el resultado puede tener el siguiente aspecto:

```
{
    "encryptionType": "AWS_OWNED_KMS_KEY",
    "lastModifiedDate": "2024-09-26T22:01:02.365000-07:00"
}
```

Paso 4: Se requieren permisos adicionales para las operaciones AWS IoT Core del plano de control

Tras optar por las claves administradas por el cliente, todos los AWS IoT Core recursos que pertenecen a su AWS cuenta se cifran con la clave KMS proporcionada. Todas las operaciones del plano de control ahora requieren que la persona que llama tenga `kms:Decrypt` permisos sobre la clave KMS, además de los permisos necesarios para la operación específica en el AWS IoT Core recurso. Si la persona que llama no tiene permiso `kms:Decrypt` y realiza una llamada a la API que requiere el cifrado o el descifrado de los datos (por ejemplo, `GetPolicy`), recibirá una `UnauthorizedException`.

Por ejemplo, al llamar a `GetPolicy`, necesita tanto el permiso `iot:GetPolicy` como el permiso `kms:Decrypt` para la clave de KMS administrada por el cliente para que la llamada a la API se realice correctamente.

> **Note**
>
> Al actualizar los usuarios o funciones de IAM para conceder AWS KMS permisos sobre la clave utilizada en la configuración de cifrado, asegúrese de que la política de claves de KMS también conceda los permisos necesarios a los respectivos usuarios o funciones de IAM.

AWS KMS permisos para **`UpdateEncryptionConfiguration`**

La llamada a la `UpdateEncryptionConfiguration` API necesita los siguientes AWS KMS permisos en la clave de KMS para poder optar por las claves administradas por el cliente o modificar la configuración de la clave:

- `kms:DescribeKey`
- `kms:Decrypt`
- `kms:Encrypt`
- `kms:GenerateDataKeyWithoutPlaintext`
- `kms:ReEncryptTo`

- `kms:ReEncryptFrom`

AWS KMS permisos para todos los demás planos de control APIs

La mayoría de los planos de control APIs requieren `kms:Decrypt` permisos cuando las claves administradas por el cliente están habilitadas. Sin embargo, algunos APIs no requieren estos permisos adicionales:

APIs que no requieren AWS KMS permisos

Los `List*` y `Delete*` APIs no entran en este cubo. Los clientes siempre pueden invocar cualquier API del plano de control `List*` o `Delete*` y esas llamadas a la API se realizarán correctamente incluso si la persona que llama no tiene el permiso `kms:Decrypt`. Estas llamadas a la API se realizarán correctamente incluso si la clave gestionada por el cliente no está en buen estado `List*` y `Delete*` APIs no se realiza ningún descifrado.

- Lista* APIs: todas las operaciones de listado (por ejemplo,,`ListThings`,`ListPolicies`) `ListCertificates`
- Eliminar* APIs: todas las operaciones de eliminación (por ejemplo,,,`DeleteThing`) `DeletePolicy` `DeleteCertificate`

Paso 5: administración de claves

AWS IoT Core comprueba periódicamente la configuración de claves gestionadas por el cliente para garantizar que las operaciones de cifrado y descifrado no se vean afectadas. Estas comprobaciones de estado se realizan una vez por minuto y verifican AWS IoT Core la capacidad de los usuarios para acceder a la clave y utilizar tanto la AWS KMS clave como la función de IAM asociada para las operaciones de cifrado y descifrado.

HEALTHY

AWS IoT Core puede acceder correctamente a la AWS KMS clave a través de la función de IAM especificada y realizar operaciones. encryption/decryption Todos los componentes funcionan correctamente.

UNHEALTHY

AWS IoT Core no puede acceder a la AWS KMS clave ni usarla. Esto impide nuevas operaciones de cifrado y puede afectar a la funcionalidad del servicio. El campo `errorCode` indica si el problema está relacionado con la clave o con el rol de IAM.

Acciones de los clientes que pueden afectar al estado de la clave

Varias acciones de los clientes pueden provocar que el estado de la clave cambie de `HEALTHY` a `UNHEALTHY`:

Acciones relacionadas con las claves

- Eliminar una AWS KMS clave: cuando programas la eliminación de una clave, está en un `Pending deletion` estado y no se puede usar
- Deshabilitar una AWS KMS clave: al deshabilitar una clave KMS, ya no se puede usar para operaciones de cifrado o descifrado
- Programación de la eliminación de la clave: la clave queda inutilizable cuando se completa la eliminación.
- Modificación de la política de claves: eliminar los permisos de acceso necesarios AWS IoT Core
- Cambiar los permisos de uso de las claves: restringir AWS KMS las acciones necesarias

Acciones relacionadas con los roles de IAM

- Eliminar la función de IAM: no AWS IoT Core se puede asumir la función para acceder a la clave
- Modificación de los permisos de los roles: se eliminan AWS KMS los permisos necesarios de la política de roles
- Cambiar la política de confianza: impedir que el AWS IoT Core servicio asuma el rol
- Añadir condiciones restrictivas: condiciones que AWS IoT Core impiden el uso del rol

Acciones de cuentas

- Cambios de acceso a las claves entre cuentas: modificación de los permisos para las claves en otras cuentas.
- Políticas de control de servicios (SCPs): políticas a nivel de la organización que restringen el acceso AWS KMS
- Políticas de IAM de cuenta: políticas que anulan el acceso a la clave o entran en conflicto con dicho acceso.

> **Important**
>
> Cualquier cambio en las AWS KMS claves, las funciones de IAM o las políticas que utilice AWS IoT Core debe probarse primero en los entornos de desarrollo. Supervise de cerca el

estado de salud clave después de realizar cualquier cambio para garantizar que AWS IoT Core la funcionalidad no se vea afectada.

Actualización de la configuración del cifrado

Actualice la configuración de cifrado AWS IoT Core para cambiar de una clave administrada por el cliente a otra, o entre claves AWS propias y claves administradas por el cliente.

Para cambiar la configuración a otra clave administrada por el cliente:

1. Cree una nueva clave administrada por el cliente siguiendo los pasos que se indican en the section called “Paso 1: Crear una clave administrada por el cliente”.
2. Actualice la política de roles de IAM para incluir los permisos tanto para las claves antiguas como para las nuevas durante el periodo de actualización.
3. Actualice la configuración del cifrado para utilizar la nueva clave:

```
aws iot update-encryption-configuration --encryption-type "CUSTOMER_MANAGED_KMS_KEY"
 \
--kms-access-role-arn "arn:aws:iam::111122223333:role/myrole" \
--kms-key-arn "arn:aws:kms:us-west-2:111122223333:key/new-key-id"
```

Para cambiar la configuración de claves administradas por el cliente a claves AWS propias:

```
aws iot update-encryption-configuration --encryption-type "AWS_OWNED_KMS_KEY"
```

> **Note**
>
> Al actualizar la configuración del cifrado para las nuevas claves administradas por el cliente, asegúrese de que tanto las claves antiguas como las nuevas permanezcan accesibles para que la operación se lleve a cabo correctamente.

Escenarios de error más comunes e impactos

En la siguiente tabla se describen los escenarios de error más comunes cuando se eliminan o desactivan las claves:

Escenario	Impacto inmediato	Consecuencias a largo plazo
Clave deshabilitada	Todas las encryption/decryption operaciones nuevas fallan inmediatamente	Interrupción del servicio hasta que se vuelva a habilitar o sustituir la clave
Clave programada para su eliminación	El estado de la clave cambia a pendiente de eliminación y todas encryption/decryption las operaciones fallarán	Error automático del servicio cuando se completa la eliminación
Clave eliminada permanentemente	Error inmediato y permanente de todas las operaciones	Pérdida permanente de datos e incapacidad para recuperar los datos cifrados
Política de claves modificada incorrectamente	AWS IoT Core pierde los permisos de acceso a la clave	Se produce un error en el servicio hasta que se corrija la política
Rol de IAM eliminado	AWS IoT Core no puede asumir el rol de acceso a la clave	Error al completar el servicio de cifrado
El rol de IAM se ha modificado incorrectamente	AWS IoT Core no puede asumir el rol o usar el rol para acceder a la clave	Se produce un error en el servicio hasta que se corrija el rol de IAM

Prevención y prácticas recomendadas

Para evitar la eliminación o desactivación accidental de las claves y minimizar el riesgo de errores en el servicio:

Implementación de políticas de ciclo de vida de claves

- Establezca procedimientos claros para la creación, rotación y retirada de claves. Documente qué claves utilizan qué AWS IoT Core recursos y mantenga un inventario de las claves activas.

Uso de políticas de IAM para restringir la eliminación de claves

Cree políticas de IAM que impidan que los usuarios no autorizados eliminen o deshabiliten las claves de cifrado críticas. Utilice condiciones que requieran una aprobación adicional para las operaciones de eliminación de claves.

Habilite el CloudTrail registro

Supervise todas las operaciones AWS KMS clave CloudTrail para detectar actividades de administración de claves no autorizadas o accidentales. Configure alertas para la eliminación de claves, la deshabilitación o los cambios en las políticas.

Ejecución de pruebas en los procedimientos de sustitución de claves

Realice pruebas periódicamente en los procedimientos de sustitución de claves en entornos que no sean de producción para asegurarse de que puede recuperarse rápidamente de los errores relacionados con las claves.

Mantenimiento de copias de seguridad de las claves

Si bien no puede exportar el material AWS KMS clave, mantenga registros detallados de las claves ARNs, las políticas y las AWS IoT Core configuraciones asociadas para facilitar el reemplazo rápido de las claves si es necesario.

Supervisión del estado de las claves

Supervise continuamente la métrica `CMK.Health` y configure alertas automatizadas para detectar los cambios en el estado de las claves. Implemente respuestas automatizadas para abordar rápidamente los problemas relacionados con las claves.

> **Important**
>
> Realice pruebas siempre en los procedimientos de actualización de claves en los entornos de desarrollo antes de implementarlos en producción. Tenga un plan de reversión documentado y asegúrese de que los procedimientos de sustitución de claves se puedan ejecutar rápidamente en caso de emergencia.

Paso 6: supervisión del estado de las claves

Como parte de las comprobaciones periódicas AWS IoT Core , se emiten CloudWatch métricas y registros para proporcionar visibilidad del estado de la configuración de claves gestionada por el cliente

AWS IoT Core emite la `CMK.Health` métrica al CloudWatch menos una vez cada minuto. La métrica proporciona información sobre el estado de las claves administradas por el cliente que se utilizan AWS IoT Core para cifrar y descifrar los datos.

La métrica `CMK.Health` puede tener los siguientes valores:

- El valor es`1`: AWS IoT Core es capaz de utilizar las claves de cifrado correctamente para cifrar y descifrar sus datos.
- El valor AWS IoT Core es`0`: no puede utilizar las claves de cifrado para cifrar y descifrar sus datos.

AWS IoT Core también emite registros AWS IoT V2 cuando cambia el estado de las claves de cifrado. Estos registros proporcionan detalles adicionales sobre la actualización del estado. Para ver estos registros, debe habilitar los registros de la AWS IoT versión 2. Los registros de `HEALTHY` se emiten en el nivel `INFO` y los registros de `UNHEALTHY` se emiten en el nivel `ERROR`. Para obtener más información sobre los niveles de registro, consulte Niveles de registro.

Los siguientes ejemplos son entradas de CloudWatch registro emitidas AWS IoT Core para indicar la actualización del estado de las claves gestionadas por el cliente.

Para supervisar los cambios de estado de la clave y responder a dichos cambios:

1. Configure CloudWatch las alarmas para la `CMK.Health` métrica:

```
aws cloudwatch put-metric-alarm --region us-west-2 \
  --alarm-name "IoTCore-CMK-Health-Alert" \
  --alarm-description "Alert when IoT Core CMK health is unhealthy" \
  --metric-name "CMK.Health" \
  --namespace "AWS/IoT" \
  --statistic "Minimum" \
  --period 300 \
  --evaluation-periods 1 \
  --threshold 1 \
  --comparison-operator "LessThanThreshold" \
  --alarm-actions "arn:aws:sns:us-west-2:111122223333:iot-alerts"
```

2. Habilite AWS IoT el registro de la versión 2 para capturar eventos detallados de cambios en el estado de salud con códigos y mensajes de error.
3. Compruebe el estado de la configuración para solucionar problemas:

```
aws iot describe-encryption-configuration --region us-west-2
```

4. Investigue el estado UNHEALTHY examinando el campo `errorCode`:
   - `KMS_KEY_VALIDATION_ERROR`— Problema con la AWS KMS clave (deshabilitada, eliminada o problemas de política)
   - `ROLE_VALIDATION_ERROR`: problema con el rol de IAM (eliminado, problemas de política o problemas de confianza)

De UNHEALTHY a HEALTHY

Cuando el estado de las claves de cifrado se actualice de `UNHEALTHY` a`HEALTHY`, AWS IoT Core emitirá un mensaje de AWS IoT registro de la versión 2 con el siguiente formato.

```
{
    "timestamp": "2017-08-10 15:37:23.476",
    "logLevel": "INFO",
    "traceId": "8421693b-f4f0-4e4a-9235-0cff8bab897d",
    "accountId": "111122223333",
    "status": "SUCCESS",
    "cmkStatus": "HEALTHY",
    "kmsKeyArn": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "kmsAccessRoleArn": "arn:aws:iam::111122223333:role/myrole",
    "eventType": "CmkHealthCheck"
}
```

De HEALTHY a UNHEALTHY

Cuando el estado de las claves de cifrado se actualice de `HEALTHY` a`UNHEALTHY`, AWS IoT Core emitirá un mensaje de AWS IoT registro de la versión 2 con el siguiente formato.

```
{
    "timestamp": "2017-08-10 15:37:23.476",
    "logLevel": "ERROR",
    "traceId": "8421693b-f4f0-4e4a-9235-0cff8bab897d",
    "accountId": "111122223333",
```

```
    "status": "FAILURE",
    "cmkStatus": "UNHEALTHY",
    "errorCode": "KMS_KEY_VALIDATION_ERROR / ROLE_VALIDATION_ERROR",
    "errorMessage": "Error message on why there was a failure",
    "kmsKeyArn": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "kmsAccessRoleArn": "arn:aws:iam::111122223333:role/myrole",
    "eventType": "CmkHealthCheck"
}
```

> **Warning**
>
> Cuando la clave está en buen estado`UNHEALTHY`, AWS IoT Core las operaciones fallan inmediatamente. Si esto ocurre, revise las configuraciones de la clave, los permisos de los roles de IAM y las políticas. Supervise la métrica `CMK.Health` para ver si hay cambios en el estado. Si se siguen produciendo un error en las operaciones después de revisar las configuraciones, póngase en contacto con el administrador de la cuenta o con el Centro de AWS Support para obtener asistencia adicional.

AWS CloudTrail eventos

También puede supervisar el uso AWS IoT Core de la clave KMS para las operaciones de cifrado y descifrado. AWS IoT Core realizará `DescribeKey` `DecryptReEncrypt`, y `GenerateDataKeyWithoutPlaintext` operaciones en su clave KMS para cifrar o descifrar los datos pertenecientes a su AWS cuenta almacenada en reposo.

Hay CloudTrail eventos para`DescribeKey`, `DecryptReEncrypt`, y. `GenerateDataKeyWithoutPlaintext` Estos eventos supervisan AWS KMS las operaciones solicitadas AWS IoT Core para acceder a los datos cifrados por la clave gestionada por el cliente.

`Decrypt`Ejemplo de

```
{
    "eventVersion": "1.09",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
        "accountId": "111122223333",
        "accessKeyId": "**********************",
```

```
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
                "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
                "accountId": "111122223333",
                "userName": "*****"
            },
            "attributes": {
                "creationDate": "2024-09-16T20:23:39Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "iot.amazonaws.com"
    },
    "eventTime": "2024-09-16T20:32:48Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "iot.amazonaws.com",
    "userAgent": "iot.amazonaws.com",
    "requestParameters": {
        "encryptionContext": {
            "kms-arn": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
            "aws-crypto-ec:vendor": "iot.amazonaws.com",
            "branch-key-id": "111122223333",
            "type": "branch:ACTIVE"
        },
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT",
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "responseElements": null,
    "requestID": "1afb6d98-8388-455d-8b48-e62c9e0cf7f4",
    "eventID": "b59a5f16-0d98-46d8-a590-0e040a48b39b",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
```

```
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

Gestión de identidad y acceso para AWS IoT

AWS Identity and Access Management (IAM) es una herramienta Servicio de AWS que ayuda al administrador a controlar de forma segura el acceso a AWS los recursos. Los administradores de IAM controlan quién puede autenticarse (iniciar sesión) y quién puede autorizarse (tener permisos) para usar los recursos. AWS IoT La IAM es una Servicio de AWS opción que puede utilizar sin coste adicional.

Temas

- Público
- Autenticación con identidades de IAM
- Administración del acceso con políticas
- ¿Cómo AWS IoT funciona con IAM
- AWS IoT ejemplos de políticas basadas en la identidad
- AWS políticas gestionadas para AWS IoT
- Solución de problemas AWS IoT de identidad y acceso

Público

La forma de usar AWS Identity and Access Management (IAM) varía según la función que desempeñes:

- Usuario del servicio: solicite permisos al administrador si no puede acceder a las características (consulte Solución de problemas AWS IoT de identidad y acceso).
- Administrador del servicio: determine el acceso de los usuarios y envíe las solicitudes de permiso (consulte ¿Cómo AWS IoT funciona con IAM).
- Administrador de IAM: escriba políticas para administrar el acceso (consulte AWS IoT ejemplos de políticas basadas en la identidad).

Autenticación con identidades de IAM

AWS IoT Las identidades pueden ser certificados de dispositivo (X.509), identidades de Amazon Cognito o usuarios o grupos de IAM. En este tema se analizan únicamente identidades de IAM. Para obtener más información sobre las demás identidades AWS IoT compatibles, consulte. Autenticación del cliente

La autenticación es la forma de iniciar sesión AWS con sus credenciales de identidad. Debe autenticarse como usuario de Usuario raíz de la cuenta de AWS IAM o asumir una función de IAM.

Puede iniciar sesión como una identidad federada con las credenciales de una fuente de identidad, como AWS IAM Identity Center (IAM Identity Center), la autenticación de inicio de sesión único o las credenciales. Google/Facebook Para obtener más información sobre el inicio de sesión, consulte Cómo iniciar sesión en la Cuenta de AWS en la Guía del usuario de AWS Sign-In .

Para el acceso programático, AWS proporciona un SDK y una CLI para firmar criptográficamente las solicitudes. Para obtener más información, consulte AWS Signature Version 4 para solicitudes de API en la Guía del usuario de IAM.

Cuenta de AWS usuario root

Al crear un Cuenta de AWS, se comienza con una identidad de inicio de sesión denominada usuario Cuenta de AWS raíz que tiene acceso completo a todos Servicios de AWS los recursos. Se recomiendaencarecidamente que no utilice el usuario raíz para las tareas diarias. Para ver la lista completa de las tareas que requieren credenciales de usuario raíz, consulte Tareas que requieren credenciales de usuario raíz en la Guía del usuario de IAM.

Usuarios y grupos de IAM

Un usuario de IAM es una identidad con permisos específicos para una sola persona o aplicación. Recomendamos el uso de credenciales temporales en lugar de usuarios de IAM con credenciales de larga duración. Para obtener más información, consulte Exigir a los usuarios humanos que utilicen la federación con un proveedor de identidad para acceder AWS mediante credenciales temporales en la Guía del usuario de IAM.

Un grupo de IAM especifica un conjunto de usuarios de IAM y facilita la administración de los permisos para grupos grandes de usuarios. Para obtener más información, consulte Casos de uso para usuarios de IAM en la Guía del usuario de IAM.

Roles de IAM

Un Rol de IAM es una identidad con permisos específicos que proporciona credenciales temporales. Puede asumir un rol cambiando de un rol de usuario a uno de IAM (consola) o llamando a una AWS CLI operación de AWS API. Para obtener más información, consulte Métodos para asumir un rol en la Guía del usuario de IAM.

Los roles de IAM son útiles para el acceso de usuario federado, los permisos de usuario de IAM temporales, el acceso entre cuentas, el acceso entre servicios y las aplicaciones que se ejecutan en Amazon EC2. Para obtener más información, consulte Acceso a recursos entre cuentas en IAM en la Guía del usuario de IAM.

Administración del acceso con políticas

El acceso se controla AWS creando políticas y adjuntándolas a AWS identidades o recursos. Una política define los permisos cuando están asociados a una identidad o un recurso. AWS evalúa estas políticas cuando un director hace una solicitud. La mayoría de las políticas se almacenan AWS como documentos JSON. Para obtener más información sobre los documentos de políticas de JSON, consulte Información general de políticas de JSON en la Guía del usuario de IAM.

Mediante las políticas, los administradores especifican quién tiene acceso a qué, definiendo qué entidad principal puede realizar acciones sobre qué recursos y en qué condiciones.

De forma predeterminada, los usuarios y los roles no tienen permisos. Un administrador de IAM crea políticas de IAM y las agrega a roles, que los usuarios pueden asumir posteriormente. Las políticas de IAM definen permisos independientemente del método que se utilice para realizar la operación.

Políticas basadas en identidades

Las políticas basadas en identidad son documentos de política de permisos JSON que asocia a una identidad (usuario, grupo o rol). Estas políticas controlan qué acciones pueden realizar las identidades, en qué recursos y en qué condiciones. Para obtener más información sobre cómo crear una política basada en la identidad, consulte Definición de permisos de IAM personalizados con políticas administradas por el cliente en la Guía del usuario de IAM.

Las políticas basadas en identidad pueden ser políticas insertadas (incrustadas directamente en una sola identidad) o políticas administradas (políticas independientes asociadas a varias identidades). Para obtener información sobre cómo elegir entre políticas administradas e insertadas, consulte Selección entre políticas administradas y políticas insertadas en la Guía del usuario de IAM.

Políticas basadas en recursos

Las políticas basadas en recursos son documentos de políticas JSON que se asocian a un recurso. Los ejemplos incluyen las Políticas de confianza de roles de IAM y las Políticas de bucket de Amazon S3. En los servicios que admiten políticas basadas en recursos, los administradores de servicios pueden utilizarlos para controlar el acceso a un recurso específico. Debe especificar una entidad principal en una política basada en recursos.

Las políticas basadas en recursos son políticas insertadas que se encuentran en ese servicio. No puedes usar políticas AWS gestionadas de IAM en una política basada en recursos.

Listas de control de acceso () ACLs

Las listas de control de acceso (ACLs) controlan qué responsables (miembros de la cuenta, usuarios o roles) tienen permisos para acceder a un recurso. ACLs son similares a las políticas basadas en recursos, aunque no utilizan el formato de documento de políticas JSON.

Amazon S3 y Amazon VPC son ejemplos de servicios compatibles. AWS WAF ACLs Para obtener más información ACLs, consulte la descripción general de la lista de control de acceso (ACL) en la Guía para desarrolladores de Amazon Simple Storage Service.

Otros tipos de políticas

AWS admite tipos de políticas adicionales que pueden establecer los permisos máximos otorgados por los tipos de políticas más comunes:

- Límites de permisos: establecen los permisos máximos que una política basada en identidad puede conceder a una entidad de IAM. Para obtener más información, consulte Límites de permisos para las entidades de IAM en la Guía del usuario de IAM.
- Políticas de control de servicios (SCPs): especifican los permisos máximos para una organización o unidad organizativa en AWS Organizations. Para obtener más información, consulte Políticas de control de servicios en la Guía del usuario de AWS Organizations .
- Políticas de control de recursos (RCPs): establece los permisos máximos disponibles para los recursos de tus cuentas. Para obtener más información, consulte Políticas de control de recursos (RCPs) en la Guía del AWS Organizations usuario.
- Políticas de sesión: políticas avanzadas que se pasan como parámetro cuando se crea una sesión temporal para un rol o un usuario federado. Para obtener más información, consulte Políticas de sesión en la Guía del usuario de IAM.

Varios tipos de políticas

Cuando se aplican varios tipos de políticas a una solicitud, los permisos resultantes son más complicados de entender. Para saber cómo se AWS determina si se debe permitir una solicitud cuando se trata de varios tipos de políticas, consulte la lógica de evaluación de políticas en la Guía del usuario de IAM.

¿Cómo AWS IoT funciona con IAM

Antes de utilizar IAM para gestionar el acceso AWS IoT, debe saber qué funciones de IAM están disponibles para su uso. AWS IoTPara obtener una visión general de cómo funcionan con IAM AWS IoT y otros AWS servicios, consulte AWS Servicios que funcionan con IAM en la Guía del usuario de IAM.

Temas

- AWS IoT políticas basadas en la identidad
- AWS IoT políticas basadas en recursos
- Autorización basada en etiquetas AWS IoT
- AWS IoT Funciones de IAM

AWS IoT políticas basadas en la identidad

Con las políticas basadas en identidad de IAM, puede especificar las acciones permitidas o denegadas y los recursos, además de las condiciones en las que se permiten o deniegan las acciones. AWS IoT admite acciones, recursos y claves de condiciones específicos. Para obtener información sobre todos los elementos que utiliza en una política JSON, consulte Referencia de los elementos de las políticas JSON de IAM en la Guía del usuario de IAM.

Acciones

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

El elemento `Action` de una política JSON describe las acciones que puede utilizar para conceder o denegar el acceso en una política. Incluya acciones en una política para conceder permisos y así llevar a cabo la operación asociada.

En la siguiente tabla se enumeran las acciones de IAM IoT, la AWS IoT API asociada y el recurso que manipula la acción.

Acciones de políticas	AWS IoT API	Recursos
IoT: AcceptCertificateTransfer	AcceptCertificateTransfer	`arn:aws:iot:` *region*`:`*account-id* `:cert/`*cert-id* **Note** La Cuenta de AWS especificada en el ARN debe ser la cuenta a la que se transfiere el certificado.
IoT: AddThingToThingGroup	AddThingToThingGroup	`arn:aws:iot:` *region*`:`*account-id* `:thinggroup/` *thing-group-name* `arn:aws:iot:` *region*`:`*account-id* `:thing/`*thing-name*
IoT: AssociateTargetsWithJob	AssociateTargetsWithJob	none
IoT: AttachPolicy	AttachPolicy	`arn:aws:iot:` *region*`:`*account-id* `:thinggroup/` *thing-group-name* o `arn:aws:iot:` *region*`:`*account-id* `:cert/`*cert-id*
IoT: AttachPrincipalPolicy	AttachPrincipalPolicy	`arn:aws:iot:` *region*`:`*account-id* `:cert/`*cert-id*
IoT: AttachSecurityProfile	AttachSecurityProfile	`arn:aws:iot:` *region*`:`*account-id* `:securityprofile/` *security-profile-name* `arn:aws:iot:` *region*`:`*account-id* `:dimension/` *dimension-name*
IoT: AttachThingPrincipal	AttachThingPrincipal	`arn:aws:iot:` *region*`:`*account-id* `:cert/`*cert-id*

Acciones de políticas	AWS IoT API	Recursos
IoT: CancelCertificateTransfer	CancelCertificateTransfer	`arn:aws:iot:` *`region`*`:`*`account-id`* `:cert/`*`cert-id`* **Note** La Cuenta de AWS especificada en el ARN debe ser la cuenta a la que se transfiere el certificado.
IoT: CancelJob	CancelJob	`arn:aws:iot:` *`region`*`:`*`account-id`* `:job/`*`job-id`*
IoT: CancelJobExecution	CancelJobExecution	`arn:aws:iot:` *`region`*`:`*`account-id`* `:job/`*`job-id`* `arn:aws:iot:` *`region`*`:`*`account-id`* `:thing/`*`thing-name`*
IoT: ClearDefaultAuthorizer	ClearDefaultAuthorizer	Ninguno
IoT: CreateAuthorizer	CreateAuthorizer	`arn:aws:iot:` *`region`*`:`*`account-id`* `:authorizer/` *`authorizer-function-name`*
IoT: CreateCertificateFromCsr	CreateCertificateFromCsr	*
IoT: CreateDimension	CreateDimension	`arn:aws:iot:` *`region`*`:`*`account-id`* `:dimension/` *`dimension-name`*

Acciones de políticas	AWS IoT API	Recursos
IoT: CreateJob	CreateJob	arn:aws:iot: *region*:*account-id* :job/*job-id* arn:aws:iot: *region*:*account-id* :thinggroup/ *thing-group-name* arn:aws:iot: *region*:*account-id* :thing/*thing-name* arn:aws:iot: *region*:*account-id* :jobtemplate/ *job-template-id*
IoT: CreateJobTemplate	CreateJobTemplate	arn:aws:iot: *region*:*account-id* :job/*job-id* arn:aws:iot: *region*:*account-id* :jobtemplate/ *job-template-id*
IoT: CreateKeysAndCertificate	CreateKeysAndCertificate	*
IoT: CreatePolicy	CreatePolicy	arn:aws:iot: *region*:*account-id* :policy/*policy-name*
IoT: CreatePolicyVersion	CreatePolicyVersion	arn:aws:iot: *region*:*account-id* :policy/*policy-name* **Note** Debe ser una AWS IoT política, no una política de IAM.
IoT: CreateRoleAlias	CreateRoleAlias	(parámetro: roleAlias) arn:aws:iot: *region*:*account-id* :rolealias/ *role-alias-name*

Acciones de políticas	AWS IoT API	Recursos
IoT: CreateSecurityProfile	CreateSecurityProfile	arn:aws:iot: *region*:*account-id* :securityprofile/ *security-profile-name* arn:aws:iot: *region*:*account-id* :dimension/ *dimension-name*
IoT: CreateThing	CreateThing	arn:aws:iot: *region*:*account-id* :thing/*thing-name*
IoT: CreateThingGroup	CreateThingGroup	arn:aws:iot: *region*:*account-id* :thinggroup/ *thing-group-name* para el grupo que se está crenado y para el grupo principal, si se utiliza
IoT: CreateThingType	CreateThingType	arn:aws:iot: *region*:*account-id* :thingtype/ *thing-type-name*
IoT: CreateTopicRule	CreateTopicRule	arn:aws:iot: *region*:*account-id* :rule/*rule-name*
IoT: DeleteAuthorizer	DeleteAuthorizer	arn:aws:iot: *region*:*account-id* :authorizer/ *authorizer-name*
IoT: eliminar CACertificate	Eliminar CACertificate	arn:aws:iot: *region*:*account-id* :cacert/*cert-id*
IoT: DeleteCertificate	DeleteCertificate	arn:aws:iot: *region*:*account-id* :cert/*cert-id*
IoT: DeleteDimension	DeleteDimension	arn:aws:iot: *region*:*account-id* :dimension/ *dimension-name*
IoT: DeleteJob	DeleteJob	arn:aws:iot: *region*:*account-id* :job/*job-id*
IoT: DeleteJobTemplate	DeleteJobTemplate	arn:aws:iot: *region*:*account-id* :job/*job-template-id*

Acciones de políticas	AWS IoT API	Recursos
IoT: DeleteJob Execution	DeleteJob Execution	arn:aws:iot: *region*:*account-id* :job/*job-id* arn:aws:iot: *region*:*account-id* :thing/*thing-name*
IoT: DeletePolicy	DeletePolicy	arn:aws:iot: *region*:*account-id* :policy/*policy-name*
IoT: DeletePolicyVersion	DeletePolicyVersion	arn:aws:iot: *region*:*account-id* :policy/*policy-name*
IoT: DeleteRegistrationCode	DeleteRegistrationCode	*
IoT: DeleteRoleAlias	DeleteRoleAlias	arn:aws:iot: *region*:*account-id* :rolealias/ *role-alias-name*
IoT: DeleteSecurityProfile	DeleteSecurityProfile	arn:aws:iot: *region*:*account-id* :securityprofile/ *security-profile-name* arn:aws:iot: *region*:*account-id* :dimension/ *dimension-name*
IoT: DeleteThing	DeleteThing	arn:aws:iot: *region*:*account-id* :thing/*thing-name*
IoT: DeleteThingGroup	DeleteThingGroup	arn:aws:iot: *region*:*account-id* :thinggroup/ *thing-group-name*
IoT: DeleteThingType	DeleteThingType	arn:aws:iot: *region*:*account-id* :thingtype/ *thing-type-name*
IoT: DeleteTopicRule	DeleteTopicRule	arn:aws:iot: *region*:*account-id* :rule/*rule-name*
IoT: eliminar V2 LoggingLevel	Eliminar V2 LoggingLevel	arn:aws:iot: *region*:*account-id* :thinggroup/ *thing-group-name*

Acciones de políticas	AWS IoT API	Recursos
IoT: Deprecate ThingType	Deprecate ThingType	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thingtype/` *`thing-type-name`*
IoT: DescribeAuthorizer	DescribeAuthorizer	`arn:aws:iot:` *`region`*`:`*`account-id`* `:authorizer/` *`authorizer-function-name`* (parámetro: authorizerName) none
IoT: describe CACertificate	Describa CACertificate	`arn:aws:iot:` *`region`*`:`*`account-id`* `:cacert/`*`cert-id`*
IoT: DescribeCertificate	DescribeCertificate	`arn:aws:iot:` *`region`*`:`*`account-id`* `:cert/`*`cert-id`*
IoT: DescribeDefaultAuthorizer	DescribeDefaultAuthorizer	Ninguno
IoT: DescribeEndpoint	DescribeEndpoint	*
IoT: DescribeEventConfigurations	DescribeEventConfigurations	none
IoT: DescribeIndex	DescribeIndex	`arn:aws:iot:` *`region`*`:`*`account-id`* `:index/`*`index-name`*
IoT: DescribeJob	DescribeJob	`arn:aws:iot:` *`region`*`:`*`account-id`* `:job/`*`job-id`*
IoT: DescribeJobExecution	DescribeJobExecution	Ninguno
IoT: DescribeJobTemplate	DescribeJobTemplate	`arn:aws:iot:` *`region`*`:`*`account-id`* `:job/`*`job-template-id`*

Acciones de políticas	AWS IoT API	Recursos
IoT: DescribeRoleAlias	DescribeRoleAlias	arn:aws:iot: *region*:*account-id* :rolealias/ *role-alias-name*
IoT: DescribeThing	DescribeThing	arn:aws:iot: *region*:*account-id* :thing/*thing-name*
IoT: DescribeThingGroup	DescribeThingGroup	arn:aws:iot: *region*:*account-id* :thinggroup/ *thing-group-name*
IoT: DescribeThingRegistrationTask	DescribeThingRegistrationTask	Ninguno
IoT: DescribeThingType	DescribeThingType	arn:aws:iot: *region*:*account-id* :thingtype/ *thing-type-name*
IoT: DetachPolicy	DetachPolicy	arn:aws:iot: *region*:*account-id* :cert/*cert-id* o arn:aws:iot: *region*:*account-id* :thinggroup/ *thing-group-name*
IoT: DetachPrincipalPolicy	DetachPrincipalPolicy	arn:aws:iot: *region*:*account-id* :cert/*cert-id*
IoT: DetachSecurityProfile	DetachSecurityProfile	arn:aws:iot: *region*:*account-id* :securityprofile/ *security-profile-name* arn:aws:iot: *region*:*account-id* :dimension/ *dimension-name*
IoT: DetachThingPrincipal	DetachThingPrincipal	arn:aws:iot: *region*:*account-id* :cert/*cert-id*

Acciones de políticas	AWS IoT API	Recursos
IoT: DisableTopicRule	DisableTopicRule	arn:aws:iot: *region*:*account-id* :rule/*rule-name*
IoT: EnableTopicRule	EnableTopicRule	arn:aws:iot: *region*:*account-id* :rule/*rule-name*
IoT: GetEffectivePolicies	GetEffectivePolicies	arn:aws:iot: *region*:*account-id* :cert/*cert-id*
IoT: GetIndexingConfiguration	GetIndexingConfiguration	Ninguno
IoT: GetJobDocument	GetJobDocument	arn:aws:iot: *region*:*account-id* :job/*job-id*
IoT: GetLoggingOptions	GetLoggingOptions	*
IoT: GetPolicy	GetPolicy	arn:aws:iot: *region*:*account-id* :policy/*policy-name*
IoT: GetPolicyVersion	GetPolicyVersion	arn:aws:iot: *region*:*account-id* :policy/*policy-name*
IoT: GetRegistrationCode	GetRegistrationCode	*
IoT: GetTopicRule	GetTopicRule	arn:aws:iot: *region*:*account-id* :rule/*rule-name*
IoT: ListAttachedPolicies	ListAttachedPolicies	arn:aws:iot: *region*:*account-id* :thinggroup/ *thing-group-name* o arn:aws:iot: *region*:*account-id* :cert/*cert-id*

Acciones de políticas	AWS IoT API	Recursos
IoT: ListAuthorizers	ListAuthorizers	Ninguno
IoT: lista CACertificates	Lista CACertificates	*
IoT: ListCertificates	ListCertificates	*
IoT: ListCertificatesBy CA	ListCertificatesByCA	*
IoT: ListIndices	ListIndices	Ninguno
IoT: ListJobExecutionsForJob	ListJobExecutionsForJob	Ninguno
IoT: ListJobExecutionsForThing	ListJobExecutionsForThing	Ninguno
IoT: ListJobs	ListJobs	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thinggroup/` *`thing-group-name`* si se utiliza thingGroupName el parámetro
IoT: ListJobTemplates	ListJobs	Ninguno
IoT: ListOutgoingCertificates	ListOutgoingCertificates	*
IoT: ListPolicies	ListPolicies	*
IoT: ListPolicyPrincipals	ListPolicyPrincipals	*

Acciones de políticas	AWS IoT API	Recursos
IoT: ListPolicyVersions	ListPolicyVersions	arn:aws:iot: *region*:*account-id* :policy/*policy-name*
IoT: ListPrincipalPolicies	ListPrincipalPolicies	arn:aws:iot: *region*:*account-id* :cert/*cert-id*
IoT: ListPrincipalThings	ListPrincipalThings	arn:aws:iot: *region*:*account-id* :cert/*cert-id*
IoT: ListRoleAliases	ListRoleAliases	Ninguno
IoT: ListTargetsForPolicy	ListTargetsForPolicy	arn:aws:iot: *region*:*account-id* :policy/*policy-name*
IoT: ListThingGroups	ListThingGroups	Ninguno
IoT: ListThingGroupsForThing	ListThingGroupsForThing	arn:aws:iot: *region*:*account-id* :thing/*thing-name*
IoT: ListThingPrincipals	ListThingPrincipals	arn:aws:iot: *region*:*account-id* :thing/*thing-name*
IoT: ListThingRegistrationTaskReports	ListThingRegistrationTaskReports	Ninguno
IoT: ListThingRegistrationTasks	ListThingRegistrationTasks	Ninguno
IoT: ListThingTypes	ListThingTypes	*
IoT: ListThings	ListThings	*

Acciones de políticas	AWS IoT API	Recursos
IoT: ListThingsInThingGroup	ListThingsInThingGroup	arn:aws:iot: *region*:*account-id* :thinggroup/ *thing-group-name*
IoT: ListTopicRules	ListTopicRules	*
IoT: lista V2 LoggingLevels	Lista V2 LoggingLevels	Ninguno
IoT: registroCACertificate	RegistrarseCACertificate	*
IoT: RegisterCertificate	RegisterCertificate	*
IoT: RegisterThing	RegisterThing	Ninguno
IoT: RejectCertificateTransfer	RejectCertificateTransfer	arn:aws:iot: *region*:*account-id* :cert/*certificate-id*
IoT: RemoveThingFromThingGroup	RemoveThingFromThingGroup	arn:aws:iot: *region*:*account-id* :thinggroup/ *thing-group-name* arn:aws:iot: *region*:*account-id* :thing/*thing-name*
IoT: ReplaceTopicRule	ReplaceTopicRule	arn:aws:iot: *region*:*account-id* :rule/*rule-name*
IoT: SearchIndex	SearchIndex	arn:aws:iot: *region*:*account-id* :index/*index-id*
IoT: SetDefaultAuthorizer	SetDefaultAuthorizer	arn:aws:iot: *region*:*account-id* :authorizer/ *authorizer-function-name*

Acciones de políticas	AWS IoT API	Recursos
IoT: SetDefaultPolicyVersion	SetDefaultPolicyVersion	arn:aws:iot: *region*:*account-id* :policy/*policy-name*
IoT: SetLoggingOptions	SetLoggingOptions	arn:aws:iot: *region*:*account-id* :role/*role-name*
IoT: SETv2LoggingLevel	Set V2LoggingLevel	arn:aws:iot: *region*:*account-id* :thinggroup/ *thing-group-name*
IoT: SetV2LoggingOptions	Set V2LoggingOptions	arn:aws:iot: *region*:*account-id* :role/*role-name*
IoT: StartThingRegistrationTask	StartThingRegistrationTask	Ninguno
IoT: StopThingRegistrationTask	StopThingRegistrationTask	Ninguno
IoT: TestAuthorization	TestAuthorization	arn:aws:iot: *region*:*account-id* :cert/*cert-id*
IoT: TestInvokeAuthorizer	TestInvokeAuthorizer	Ninguno
IoT: TransferCertificate	TransferCertificate	arn:aws:iot: *region*:*account-id* :cert/*cert-id*
IoT: UpdateAuthorizer	UpdateAuthorizer	arn:aws:iot: *region*:*account-id* :authorizerfunction/ *authorizer-function-name*
IoT: actualización CACertificate	Actualización CACertificate	arn:aws:iot: *region*:*account-id* :cacert/*cert-id*

Acciones de políticas	AWS IoT API	Recursos
IoT: UpdateCertificate	UpdateCertificate	arn:aws:iot: *region*:*account-id* :cert/*cert-id*
IoT: UpdateDimension	UpdateDimension	arn:aws:iot: *region*:*account-id* :dimension/ *dimension-name*
IoT: UpdateEventConfigurations	UpdateEventConfigurations	Ninguno
IoT: UpdateIndexingConfiguration	UpdateIndexingConfiguration	Ninguno
IoT: UpdateRoleAlias	UpdateRoleAlias	arn:aws:iot: *region*:*account-id* :rolealias/ *role-alias-name*
IoT: UpdateSecurityProfile	UpdateSecurityProfile	arn:aws:iot: *region*:*account-id* :securityprofile/ *security-profile-name* arn:aws:iot: *region*:*account-id* :dimension/ *dimension-name*
IoT: UpdateThing	UpdateThing	arn:aws:iot: *region*:*account-id* :thing/*thing-name*
IoT: UpdateThingGroup	UpdateThingGroup	arn:aws:iot: *region*:*account-id* :thinggroup/ *thing-group-name*
IoT: UpdateThingGroupsForThing	UpdateThingGroupsForThing	arn:aws:iot: *region*:*account-id* :thing/*thing-name* arn:aws:iot: *region*:*account-id* :thinggroup/ *thing-group-name*

Las acciones políticas AWS IoT utilizan el siguiente prefijo antes de la acción:`iot:`. Por ejemplo, para conceder permiso a alguien para que muestre todas las cosas de IoT registradas en su

Cuenta de AWS `ListThings` API, debes incluir la `iot:ListThings` acción en su política. Las declaraciones de política deben incluir un `NotAction` elemento `Action` o. AWS IoT define su propio conjunto de acciones que describen las tareas que puede realizar con este servicio.

Para especificar varias acciones en una única instrucción, sepárelas con comas del siguiente modo:

```
"Action": [
      "ec2:action1",
      "ec2:action2"
```

Puede utilizar caracteres comodín para especificar varias acciones (*). Por ejemplo, para especificar todas las acciones que comiencen con la palabra `Describe`, incluya la siguiente acción:

```
"Action": "iot:Describe*"
```

Para ver una lista de AWS IoT acciones, consulte las acciones definidas por AWS IoT en la Guía del usuario de IAM.

Acciones de Trusted Advisor

En la siguiente tabla se enumeran las acciones de IAM IoT Device Advisor, la API de AWS IoT Device Advisor asociada y el recurso que manipula la acción.

Acciones de políticas	AWS IoT API	Recursos
iotdeviceadvisor: CreateSui teDefinition	CreateSui teDefinition	Ninguno
asesor de dispositivos de IoT: DeleteSui teDefinition	DeleteSui teDefinition	`arn:aws:iotdeviceadvisor:` *`region`*`:`*`account-id`* `:suitedefinition/` *`suite-definition-id`*
asesor de dispositivos de IoT: GetSuiteD efinition	GetSuiteD efinition	`arn:aws:iotdeviceadvisor:` *`region`*`:`*`account-id`* `:suitedefinition/` *`suite-definition-id`*

Acciones de políticas	AWS IoT API	Recursos
asesor de dispositivos de IoT: GetSuiteRun	GetSuiteRun	arn:aws:iotdeviceadvisor: *region*:*account-id* :suitedefinition/ *suite-run-id*
asesor de dispositivos de IoT: GetSuiteRunReport	GetSuiteRunReport	arn:aws:iotdeviceadvisor: *region*:*account-id* :suiterun/ *suite-definition-id* /*suite-run-id*
asesor de dispositivos de IoT: ListSuiteDefinitions	ListSuiteDefinitions	Ninguno
asesor de dispositivos de IoT: ListSuiteRuns	ListSuiteRuns	arn:aws:iotdeviceadvisor: *region*:*account-id* :suitedefinition/ *suite-definition-id*
asesor de dispositivos de IoT: ListTagsForResource	ListTagsForResource	arn:aws:iotdeviceadvisor: *region*:*account-id* :suitedefinition/ *suite-definition-id* arn:aws:iotdeviceadvisor: *region*:*account-id* :suiterun/suite-definition-id/ *suite-run-id*
asesor de dispositivos de IoT: StartSuiteRun	StartSuiteRun	arn:aws:iotdeviceadvisor: *region*:*account-id* :suitedefinition/ *suite-definition-id*

Acciones de políticas	AWS IoT API	Recursos
asesor de dispositivos de IoT: TagResource	TagResource	arn:aws:iotdeviceadvisor: *region*:*account-id* :suitedefinition/ *suite-definition-id* arn:aws:iotdeviceadvisor: *region*:*account-id* :suiterun/suite-definition-id/ *suite-run-id*
asesor de dispositivos de IoT: UntagResource	UntagResource	arn:aws:iotdeviceadvisor: *region*:*account-id* :suitedefinition/ *suite-definition-id* arn:aws:iotdeviceadvisor: *region*:*account-id* :suiterun/suite-definition-id/ *suite-run-id*
asesor de dispositivos de IoT: UpdateSuiteDefinition	UpdateSuiteDefinition	arn:aws:iotdeviceadvisor: *region*:*account-id* :suitedefinition/ *suite-definition-id*
asesor de dispositivos de IoT: StopSuiteRun	StopSuiteRun	arn:aws:iotdeviceadvisor: *region*:*account-id* :suiterun/suite-definition-id/ *suite-run-id*

Las acciones políticas en AWS IoT Device Advisor utilizan el siguiente prefijo antes de la acción:. `iotdeviceadvisor:` Por ejemplo, para conceder a alguien el permiso de enumerar todas las definiciones de conjuntos de aplicaciones Cuenta de AWS registradas en ella en la ListSuiteDefinitions API, debes incluir la `iotdeviceadvisor:ListSuiteDefinitions` acción en su política.

Recursos

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

El elemento `Resource` de la política JSON especifica el objeto u objetos a los que se aplica la acción. Como práctica recomendada, especifique un recurso utilizando el Nombre de recurso de Amazon (ARN). En el caso de las acciones que no admiten permisos por recurso, utilice un carácter comodín (*) para indicar que la instrucción se aplica a todos los recursos.

```
"Resource": "*"
```

AWS IoT recursos

Acciones de políticas	AWS IoT API	Recursos
IoT: AcceptCertificateTransfer	AcceptCertificateTransfer	`arn:aws:iot:` *`region`*`:`*`account-id`* `:cert/`*`cert-id`* **Note** La Cuenta de AWS especificada en el ARN debe ser la cuenta a la que se transfiere el certificado.
IoT: AddThingToThingGroup	AddThingToThingGroup	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thinggroup/` *`thing-group-name`* `arn:aws:iot:` *`region`*`:`*`account-id`* `:thing/`*`thing-name`*
IoT: AssociateTargetsWithJob	AssociateTargetsWithJob	Ninguno
IoT: AttachPolicy	AttachPolicy	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thinggroup/` *`thing-group-name`* o `arn:aws:iot:` *`region`*`:`*`account-id`* `:cert/`*`cert-id`*
IoT: AttachPrincipalPolicy	AttachPrincipalPolicy	`arn:aws:iot:` *`region`*`:`*`account-id`* `:cert/`*`cert-id`*

Acciones de políticas	AWS IoT API	Recursos
IoT: AttachThingPrincipal	AttachThingPrincipal	`arn:aws:iot:` *`region`*`:`*`account-id`* `:cert/`*`cert-id`*
IoT: CancelCertificateTransfer	CancelCertificateTransfer	`arn:aws:iot:` *`region`*`:`*`account-id`* `:cert/`*`cert-id`* **Note** La Cuenta de AWS especificada en el ARN debe ser la cuenta a la que se transfiere el certificado.
IoT: CancelJob	CancelJob	`arn:aws:iot:` *`region`*`:`*`account-id`* `:job/`*`job-id`*
IoT: CancelJobExecution	CancelJobExecution	`arn:aws:iot:` *`region`*`:`*`account-id`* `:job/`*`job-id`* `arn:aws:iot:` *`region`*`:`*`account-id`* `:thing/`*`thing-name`*
IoT: ClearDefaultAuthorizer	ClearDefaultAuthorizer	Ninguno
IoT: CreateAuthorizer	CreateAuthorizer	`arn:aws:iot:` *`region`*`:`*`account-id`* `:authorizer/` *`authorizer-function-name`*
IoT: CreateCertificateFromCsr	CreateCertificateFromCsr	*

Acciones de políticas	AWS IoT API	Recursos
IoT: CreateJob	CreateJob	arn:aws:iot: *region*:*account-id* :job/*job-id* arn:aws:iot: *region*:*account-id* :thinggroup/ *thing-group-name* arn:aws:iot: *region*:*account-id* :thing/*thing-name* arn:aws:iot: *region*:*account-id* :jobtemplate/ *job-template-id*
IoT: CreateJobTemplate	CreateJobTemplate	arn:aws:iot: *region*:*account-id* :job/*job-id* arn:aws:iot: *region*:*account-id* :jobtemplate/ *job-template-id*
IoT: CreateKeysAndCertificate	CreateKeysAndCertificate	*
IoT: CreatePolicy	CreatePolicy	arn:aws:iot: *region*:*account-id* :policy/*policy-name*
CreatePolicyVersion	IoT: CreatePolicyVersion	arn:aws:iot: *region*:*account-id* :policy/*policy-name* **Note** Debe ser una AWS IoT política, no una política de IAM.
IoT: CreateRoleAlias	CreateRoleAlias	(parámetro: roleAlias) arn:aws:iot: *region*:*account-id* :rolealias/ *role-alias-name*

Acciones de políticas	AWS IoT API	Recursos
IoT: CreateThing	CreateThing	arn:aws:iot: *region*:*account-id* :thing/*thing-name*
IoT: CreateThingGroup	CreateThingGroup	arn:aws:iot: *region*:*account-id* :thinggroup/ *thing-group-name* para el grupo que se está crenado y para el grupo principal, si se utiliza
IoT: CreateThingType	CreateThingType	arn:aws:iot: *region*:*account-id* :thingtype/ *thing-type-name*
IoT: CreateTopicRule	CreateTopicRule	arn:aws:iot: *region*:*account-id* :rule/*rule-name*
IoT: DeleteAuthorizer	DeleteAuthorizer	arn:aws:iot: *region*:*account-id* :authorizer/ *authorizer-name*
IoT: eliminar CACertificate	Eliminar CACertificate	arn:aws:iot: *region*:*account-id* :cacert/*cert-id*
IoT: DeleteCertificate	DeleteCertificate	arn:aws:iot: *region*:*account-id* :cert/*cert-id*
IoT: DeleteJob	DeleteJob	arn:aws:iot: *region*:*account-id* :job/*job-id*
IoT: DeleteJobExecution	DeleteJobExecution	arn:aws:iot: *region*:*account-id* :job/*job-id* arn:aws:iot: *region*:*account-id* :thing/*thing-name*
IoT: DeleteJobTemplate	DeleteJobTemplate	arn:aws:iot: *region*:*account-id* :jobtemplate/ *job-template-id*
IoT: DeletePolicy	DeletePolicy	arn:aws:iot: *region*:*account-id* :policy/*policy-name*

Acciones de políticas	AWS IoT API	Recursos
IoT: DeletePolicyVersion	DeletePolicyVersion	arn:aws:iot: *region*:*account-id* :policy/*policy-name*
IoT: DeleteRegistrationCode	DeleteRegistrationCode	*
IoT: DeleteRoleAlias	DeleteRoleAlias	arn:aws:iot: *region*:*account-id* :rolealias/ *role-alias-name*
IoT: DeleteThing	DeleteThing	arn:aws:iot: *region*:*account-id* :thing/*thing-name*
IoT: DeleteThingGroup	DeleteThingGroup	arn:aws:iot: *region*:*account-id* :thinggroup/ *thing-group-name*
IoT: DeleteThingType	DeleteThingType	arn:aws:iot: *region*:*account-id* :thingtype/ *thing-type-name*
IoT: DeleteTopicRule	DeleteTopicRule	arn:aws:iot: *region*:*account-id* :rule/*rule-name*
IoT: eliminar V2 LoggingLevel	Eliminar V2 LoggingLevel	arn:aws:iot: *region*:*account-id* :thinggroup/ *thing-group-name*
IoT: DeprecateThingType	DeprecateThingType	arn:aws:iot: *region*:*account-id* :thingtype/ *thing-type-name*
IoT: DescribeAuthorizer	DescribeAuthorizer	arn:aws:iot: *region*:*account-id* :authorizer/ *authorizer-function-name* (parámetro: authorizerName) none
IoT: describeCACertificate	DescribaCACertificate	arn:aws:iot: *region*:*account-id* :cacert/*cert-id*

Acciones de políticas	AWS IoT API	Recursos
IoT: DescribeCertificate	DescribeCertificate	arn:aws:iot: *region*:*account-id* :cert/*cert-id*
IoT: DescribeDefaultAuthorizer	DescribeDefaultAuthorizer	Ninguno
IoT: DescribeEndpoint	DescribeEndpoint	*
IoT: DescribeEventConfigurations	DescribeEventConfigurations	none
IoT: DescribeIndex	DescribeIndex	arn:aws:iot: *region*:*account-id* :index/*index-name*
IoT: DescribeJob	DescribeJob	arn:aws:iot: *region*:*account-id* :job/*job-id*
IoT: DescribeJobExecution	DescribeJobExecution	Ninguno
IoT: DescribeJobTemplate	DescribeJobTemplate	arn:aws:iot: *region*:*account-id* :jobtemplate/ *job-template-id*
IoT: DescribeRoleAlias	DescribeRoleAlias	arn:aws:iot: *region*:*account-id* :rolealias/ *role-alias-name*
IoT: DescribeThing	DescribeThing	arn:aws:iot: *region*:*account-id* :thing/*thing-name*
IoT: DescribeThingGroup	DescribeThingGroup	arn:aws:iot: *region*:*account-id* :thinggroup/ *thing-group-name*
IoT: DescribeThingRegistrationTask	DescribeThingRegistrationTask	Ninguno

Acciones de políticas	AWS IoT API	Recursos
IoT: DescribeThingType	DescribeThingType	arn:aws:iot: *region*:*account-id* :thingtype/ *thing-type-name*
IoT: DetachPolicy	DetachPolicy	arn:aws:iot: *region*:*account-id* :cert/*certificate-id* o arn:aws:iot: *region*:*account-id* :thinggroup/ *thing-group-name*
IoT: DetachPrincipalPolicy	DetachPrincipalPolicy	arn:aws:iot: *region*:*account-id* :cert/*certificate-id*
IoT: DetachThingPrincipal	DetachThingPrincipal	arn:aws:iot: *region*:*account-id* :cert/*certificate-id*
IoT: DisableTopicRule	DisableTopicRule	arn:aws:iot: *region*:*account-id* :rule/*rule-name*
IoT: EnableTopicRule	EnableTopicRule	arn:aws:iot: *region*:*account-id* :rule/*rule-name*
IoT: GetEffectivePolicies	GetEffectivePolicies	arn:aws:iot: *region*:*account-id* :cert/*certificate-id*
IoT: GetIndexingConfiguration	GetIndexingConfiguration	Ninguno
IoT: GetJobDocument	GetJobDocument	arn:aws:iot: *region*:*account-id* :job/*job-id*
IoT: GetLoggingOptions	GetLoggingOptions	*
IoT: GetPolicy	GetPolicy	arn:aws:iot: *region*:*account-id* :policy/*policy-name*

Acciones de políticas	AWS IoT API	Recursos
IoT: GetPolicy Version	GetPolicyVersion	arn:aws:iot: *region*:*account-id* :policy/*policy-name*
IoT: GetRegistrationCode	GetRegistrationCode	*
IoT: GetTopicRule	GetTopicRule	arn:aws:iot: *region*:*account-id* :rule/*rule-name*
IoT: ListAttachedPolicies	ListAttachedPolicies	arn:aws:iot: *region*:*account-id* :thinggroup/ *thing-group-name* o arn:aws:iot: *region*:*account-id* :cert/*cert-id*
IoT: ListAuthorizers	ListAuthorizers	Ninguno
IoT: lista CACertificates	Lista CACertificates	*
IoT: ListCertificates	ListCertificates	*
IoT: ListCertificatesBy CA	ListCertificatesByCA	*
IoT: ListIndices	ListIndices	Ninguno
IoT: ListJobExecutionsForJob	ListJobExecutionsForJob	Ninguno
IoT: ListJobExecutionsForThing	ListJobExecutionsForThing	Ninguno

Acciones de políticas	AWS IoT API	Recursos
IoT: ListJobs	ListJobs	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thinggroup/`*`thing-group-name`* si se utiliza thingGroupName el parámetro
IoT: ListJobTemplates	ListJobTemplates	Ninguno
IoT: ListOutgoingCertificates	ListOutgoingCertificates	*
IoT: ListPolicies	ListPolicies	*
IoT: ListPolicyPrincipals	ListPolicyPrincipals	`arn:aws:iot:` *`region`*`:`*`account-id`* `:policy/`*`policy-name`*
IoT: ListPolicyVersions	ListPolicyVersions	`arn:aws:iot:` *`region`*`:`*`account-id`* `:policy/`*`policy-name`*
IoT: ListPrincipalPolicies	ListPrincipalPolicies	`arn:aws:iot:` *`region`*`:`*`account-id`* `:cert/`*`cert-id`*
IoT: ListPrincipalThings	ListPrincipalThings	`arn:aws:iot:` *`region`*`:`*`account-id`* `:cert/`*`cert-id`*
IoT: ListRoleAliases	ListRoleAliases	Ninguno
IoT: ListTargetsForPolicy	ListTargetsForPolicy	`arn:aws:iot:` *`region`*`:`*`account-id`* `:policy/`*`policy-name`*
IoT: ListThingGroups	ListThingGroups	Ninguno
IoT: ListThingGroupsForThing	ListThingGroupsForThing	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thing/`*`thing-name`*

Acciones de políticas	AWS IoT API	Recursos
IoT: ListThing Principals	ListThing Principals	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thing/`*`thing-name`*
IoT: ListThing RegistrationTaskReports	ListThing RegistrationTaskReports	Ninguno
IoT: ListThing RegistrationTasks	ListThing RegistrationTasks	Ninguno
IoT: ListThing Types	ListThingTypes	*
IoT: ListThings	ListThings	*
IoT: ListThingsInThingGroup	ListThingsInThingGroup	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thinggroup/` *`thing-group-name`*
IoT: ListTopicRules	ListTopicRules	*
IoT: lista V2LoggingLevels	Lista V2LoggingLevels	Ninguno
IoT: registro CACertificate	RegistrarseCACertificate	*
IoT: RegisterCertificate	RegisterCertificate	*
IoT: RegisterThing	RegisterThing	Ninguno
IoT: RejectCertificateTransfer	RejectCertificateTransfer	`arn:aws:iot:` *`region`*`:`*`account-id`* `:cert/`*`cert-id`*

Acciones de políticas	AWS IoT API	Recursos
IoT: RemoveThingFromThingGroup	RemoveThingFromThingGroup	arn:aws:iot: *region*:*account-id* :thinggroup/ *thing-group-name* arn:aws:iot: *region*:*account-id* :thing/*thing-name*
IoT: ReplaceTopicRule	ReplaceTopicRule	arn:aws:iot: *region*:*account-id* :rule/*rule-name*
IoT: SearchIndex	SearchIndex	arn:aws:iot: *region*:*account-id* :index/*index-id*
IoT: SetDefaultAuthorizer	SetDefaultAuthorizer	arn:aws:iot: *region*:*account-id* :authorizer/ *authorizer-function-name*
IoT: SetDefaultPolicyVersion	SetDefaultPolicyVersion	arn:aws:iot: *region*:*account-id* :policy/*policy-name*
IoT: SetLoggingOptions	SetLoggingOptions	*
IoT: SETv2LoggingLevel	Set V2LoggingLevel	*
IoT: SetV2LoggingOptions	Set V2LoggingOptions	*
IoT: StartThingRegistrationTask	StartThingRegistrationTask	Ninguno
IoT: StopThingRegistrationTask	StopThingRegistrationTask	Ninguno
IoT: TestAuthorization	TestAuthorization	arn:aws:iot: *region*:*account-id* :cert/*cert-id*

Acciones de políticas	AWS IoT API	Recursos
IoT: TestInvokeAuthorizer	TestInvokeAuthorizer	Ninguno
IoT: TransferCertificate	TransferCertificate	arn:aws:iot: *region*:*account-id* :cert/*cert-id*
IoT: UpdateAuthorizer	UpdateAuthorizer	arn:aws:iot: *region*:*account-id* :authorizerfunction/ *authorizer-function-name*
IoT: actualización CACertificate	Actualización CACertificate	arn:aws:iot: *region*:*account-id* :cacert/*cert-id*
IoT: UpdateCertificate	UpdateCertificate	arn:aws:iot: *region*:*account-id* :cert/*cert-id*
IoT: UpdateEventConfigurations	UpdateEventConfigurations	Ninguno
IoT: UpdateIndexingConfiguration	UpdateIndexingConfiguration	Ninguno
IoT: UpdateRoleAlias	UpdateRoleAlias	arn:aws:iot: *region*:*account-id* :rolealias/ *role-alias-name*
IoT: UpdateThing	UpdateThing	arn:aws:iot: *region*:*account-id* :thing/*thing-name*
IoT: UpdateThingGroup	UpdateThingGroup	arn:aws:iot: *region*:*account-id* :thinggroup/ *thing-group-name*
IoT: UpdateThingGroupsForThing	UpdateThingGroupsForThing	arn:aws:iot: *region*:*account-id* :thing/*thing-name*

Para obtener más información sobre el formato de ARNs, consulte Amazon Resource Names (ARNs) y AWS Service Namespaces.

Algunas AWS IoT acciones, como las de creación de recursos, no se pueden realizar en un recurso específico. En dichos casos, debe utilizar el carácter comodín (*).

```
"Resource": "*"
```

Para ver una lista de los tipos de AWS IoT recursos y sus correspondientes ARNs, consulte los recursos definidos por AWS IoT en la Guía del usuario de IAM. Para obtener información sobre las acciones con las que puede especificar el ARN de cada recurso, consulte Acciones definidas por AWS IoT.

Recursos de Device Advisor

Para definir las restricciones a nivel de recursos para las políticas de IAM de AWS IoT Device Advisor, utilice los siguientes formatos de ARN de recursos para las definiciones y ejecuciones de conjuntos.

Formato de ARN de recurso de definición de conjunto

```
arn:aws:iotdeviceadvisor:region:account-id:suitedefinition/suite-definition-id
```

Formato de ARN de recurso de ejecución de conjunto

```
arn:aws:iotdeviceadvisor:region:account-id:suiterun/suite-definition-id/suite-run-id
```

Claves de condición

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

El elemento `Condition` especifica cuándo se ejecutan las instrucciones en función de criterios definidos. Puede crear expresiones condicionales que utilizan operadores de condición, tales como igual o menor que, para que la condición de la política coincida con los valores de la solicitud. Para ver todas las claves de condición AWS globales, consulte las claves de contexto de condición AWS globales en la Guía del usuario de IAM.

AWS IoT define su propio conjunto de claves de condición y también admite el uso de algunas claves de condición globales. Para ver todas las claves de condición AWS globales, consulte las claves de contexto de condición AWS globales en la Guía del usuario de IAM.

AWS IoT claves de condición

AWS IoT claves de condición	Descripción	Tipo
`aws:RequestTag/${tag-key}`	Una clave de etiqueta que está presente en la solicitud que el usuario realiza a AWS IoT.	Cadena
`aws:ResourceTag/${tag-key}`	El componente clave de etiqueta de una etiqueta adjunta a un AWS IoT recurso.	Cadena
`aws:TagKeys`	La lista de todos los nombres de clave de etiqueta asociados con el recurso de la solicitud.	Cadena

Para ver una lista de claves de AWS IoT condición, consulte las claves de condición AWS IoT en la Guía del usuario de IAM. Para saber con qué acciones y recursos puede utilizar una clave de condición, consulte Acciones definidas por AWS IoT.

Ejemplos

Para ver ejemplos de políticas AWS IoT basadas en la identidad, consulte. AWS IoT ejemplos de políticas basadas en la identidad

AWS IoT políticas basadas en recursos

Las políticas basadas en recursos son documentos de políticas de JSON que especifican qué acciones puede realizar un director específico en el AWS IoT recurso y en qué condiciones.

AWS IoT no admite las políticas de IAM basadas en recursos. Sin embargo, apoya AWS IoT las políticas basadas en los recursos. Para obtener más información, consulte AWS IoT Core políticas.

Autorización basada en etiquetas AWS IoT

Puede adjuntar etiquetas a AWS IoT los recursos o pasarles etiquetas en una solicitud AWS IoT. Para controlar el acceso en función de etiquetas, debe proporcionar información de las etiquetas en el elemento de condición de una política utilizando las claves de condición `iot:ResourceTag/key-name`, `aws:RequestTag/key-name` o `aws:TagKeys`. Para obtener más información, consulte Uso de etiquetas con políticas de IAM. Para obtener más información sobre el etiquetado de AWS IoT recursos, consulteEtiquetar sus recursos AWS IoT.

Para consultar un ejemplo de política basada en la identidad para limitar el acceso a un recurso en función de las etiquetas de ese recurso, consulte Visualización de AWS IoT los recursos en función de las etiquetas.

AWS IoT Funciones de IAM

Un rol de IAM es una entidad dentro de usted Cuenta de AWS que tiene permisos específicos.

Utilizar credenciales temporales con AWS IoT

Puede utilizar credenciales temporales para iniciar sesión con federación, asumir un rol de IAM o asumir un rol de acceso entre cuentas. Las credenciales de seguridad temporales se obtienen llamando a operaciones de AWS STS API como AssumeRoleo GetFederationToken.

AWS IoT admite el uso de credenciales temporales.

Roles vinculados a servicios

Los roles vinculados a un servicio permiten a AWS los servicios acceder a los recursos de otros servicios para completar una acción en tu nombre. Los roles vinculados a servicios aparecen en la cuenta de IAM y son propiedad del servicio. Un administrador de IAM puede ver, pero no editar, los permisos de los roles vinculados a servicios.

AWS IoT no admite los roles vinculados al servicio.

Roles de servicio

Esta característica permite que un servicio asuma un rol de servicio en su nombre. Este rol permite que el servicio obtenga acceso a los recursos de otros servicios para completar una acción en su nombre. Los roles de servicio aparecen en su cuenta de IAM y son propiedad de la cuenta. Esto significa que un administrador de IAM puede cambiar los permisos de este rol. Sin embargo, hacerlo podría deteriorar la funcionalidad del servicio.

AWS IoT ejemplos de políticas basadas en la identidad

De forma predeterminada, los usuarios y los roles de IAM no tienen permiso para crear, ver ni modificar recursos de AWS IoT . Tampoco pueden realizar tareas con la API Consola de administración de AWS AWS CLI, o AWS . Un administrador de IAM debe crear políticas de IAM que concedan permisos a los usuarios y a los roles para realizar operaciones de la API concretas en los recursos especificados que necesiten. El administrador debe asociar esas políticas a los usuarios o grupos que necesiten esos permisos.

Para obtener información acerca de cómo crear una política basada en identidad de IAM con estos documentos de políticas JSON de ejemplo, consulte Creación de políticas en la pestaña JSON en la Guía del usuario de IAM.

Temas

- Prácticas recomendadas relativas a políticas
- Uso de la consola AWS IoT
- Permitir a los usuarios consultar sus propios permisos
- Visualización de AWS IoT los recursos en función de las etiquetas
- Visualización de los recursos de AWS IoT Device Advisor en función de las etiquetas

Prácticas recomendadas relativas a políticas

Las políticas basadas en la identidad determinan si alguien puede crear AWS IoT recursos de tu cuenta, acceder a ellos o eliminarlos. Estas acciones pueden generar costos adicionales para su Cuenta de AWS. Siga estas directrices y recomendaciones al crear o editar políticas basadas en identidades:

- Comience con las políticas AWS administradas y avance hacia los permisos con privilegios mínimos: para empezar a conceder permisos a sus usuarios y cargas de trabajo, utilice las políticas AWS administradas que otorgan permisos para muchos casos de uso comunes. Están

disponibles en su. Cuenta de AWS Le recomendamos que reduzca aún más los permisos definiendo políticas administradas por el AWS cliente que sean específicas para sus casos de uso. Con el fin de obtener más información, consulte las políticas administradas por AWS o las políticas administradas por AWS para funciones de tarea en la Guía de usuario de IAM.

- Aplique permisos de privilegio mínimo: cuando establezca permisos con políticas de IAM, conceda solo los permisos necesarios para realizar una tarea. Para ello, debe definir las acciones que se pueden llevar a cabo en determinados recursos en condiciones específicas, también conocidos como permisos de privilegios mínimos. Con el fin de obtener más información sobre el uso de IAM para aplicar permisos, consulte Políticas y permisos en IAM en la Guía del usuario de IAM.
- Utilice condiciones en las políticas de IAM para restringir aún más el acceso: puede agregar una condición a sus políticas para limitar el acceso a las acciones y los recursos. Por ejemplo, puede escribir una condición de políticas para especificar que todas las solicitudes deben enviarse utilizando SSL. También puedes usar condiciones para conceder el acceso a las acciones del servicio si se utilizan a través de una acción específica Servicio de AWS, por ejemplo CloudFormation. Para obtener más información, consulte Elementos de la política de JSON de IAM: Condición en la Guía del usuario de IAM.
- Utiliza el analizador de acceso de IAM para validar las políticas de IAM con el fin de garantizar la seguridad y funcionalidad de los permisos: el analizador de acceso de IAM valida políticas nuevas y existentes para que respeten el lenguaje (JSON) de las políticas de IAM y las prácticas recomendadas de IAM. El analizador de acceso de IAM proporciona más de 100 verificaciones de políticas y recomendaciones procesables para ayudar a crear políticas seguras y funcionales. Para más información, consulte Validación de políticas con el Analizador de acceso de IAM en la Guía del usuario de IAM.
- Requerir autenticación multifactor (MFA): si tiene un escenario que requiere usuarios de IAM o un usuario raíz en Cuenta de AWS su cuenta, active la MFA para mayor seguridad. Para exigir la MFA cuando se invoquen las operaciones de la API, añada condiciones de MFA a sus políticas. Para más información, consulte Acceso seguro a la API con MFA en la Guía del usuario de IAM.

Para obtener más información sobre las prácticas recomendadas de IAM, consulte Prácticas recomendadas de seguridad en IAM en la Guía del usuario de IAM.

Uso de la consola AWS IoT

Para acceder a la AWS IoT consola, debe tener un conjunto mínimo de permisos. Estos permisos deben permitirle enumerar y ver detalles sobre los AWS IoT recursos de su cuenta Cuenta de AWS. Si crea una política basada en identidades que sea más restrictiva que el mínimo de permisos

necesarios, la consola no funcionará del modo esperado para las entidades (usuarios o roles) que tengan esa política.

Para garantizar que esas entidades puedan seguir utilizando la AWS IoT consola, adjunte también la siguiente política AWS administrada a las entidades:`AWSIoTFullAccess`. Para obtener más información, consulte Adición de permisos a un usuario en la Guía del usuario de IAM.

No es necesario que concedas permisos mínimos de consola a los usuarios que solo realizan llamadas a la API AWS CLI o a la AWS API. En su lugar, permite acceso únicamente a las acciones que coincidan con la operación de API que intenta realizar.

Permitir a los usuarios consultar sus propios permisos

En este ejemplo, se muestra cómo podría crear una política que permita a los usuarios de IAM ver las políticas administradas e insertadas que se asocian a la identidad de sus usuarios. Esta política incluye permisos para completar esta acción en la consola o mediante programación mediante la API AWS CLI o AWS .

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
```

```
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Visualización de AWS IoT los recursos en función de las etiquetas

Puede utilizar las condiciones de su política basada en identidad para controlar el acceso a los recursos de AWS IoT basados en etiquetas. En este ejemplo, se muestra cómo crear una política que permita visualizar un objeto. Sin embargo, los permisos solo se conceden si la etiqueta de la cosa `Owner` tiene el valor del nombre de usuario de dicho usuario. Esta política también proporciona los permisos necesarios para llevar a cabo esta acción en la consola.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ListBillingGroupsInConsole",
            "Effect": "Allow",
            "Action": "iot:ListBillingGroups",
            "Resource": "*"
        },
        {
            "Sid": "ViewBillingGroupsIfOwner",
            "Effect": "Allow",
           "Action": "iot:DescribeBillingGroup",
            "Resource": "arn:aws:iot:*:*:billinggroup/*",
            "Condition": {
                "StringEquals": {"aws:ResourceTag/Owner": "${aws:username}"}
        }
        }
    ]
}
```

También puede asociar esta política al usuario de IAM en su cuenta. Si un usuario llamado `richard-roe` intenta ver un grupo AWS IoT de facturación, el grupo de facturación debe estar

etiquetado `Owner=richard-roe` o`owner=richard-roe`. De lo contrario, se le deniega el acceso. La clave de la etiqueta de condición `Owner` coincide con los nombres de las claves de condición `Owner` y `owner` porque no distinguen entre mayúsculas y minúsculas. Para obtener más información, consulte Elementos de la política de JSON de IAM: Condición en la Guía del usuario de IAM.

Visualización de los recursos de AWS IoT Device Advisor en función de las etiquetas

Puede utilizar las condiciones de su política basada en la identidad para controlar el acceso a los recursos de AWS IoT Device Advisor basados en etiquetas. El siguiente ejemplo muestra cómo crear una política que permita ver una definición de conjunto determinada. Sin embargo, el permiso solo se otorga si la etiqueta de definición de conjunto tiene `SuiteType` ajustado con el valor `MQTT`. Esta política también proporciona los permisos necesarios para llevar a cabo esta acción en la consola.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ViewSuiteDefinition",
            "Effect": "Allow",
            "Action": "iotdeviceadvisor:GetSuiteDefinition",
            "Resource": "arn:aws:iotdeviceadvisor:*:*:suitedefinition/*",
            "Condition": {
                "StringEquals": {"aws:ResourceTag/SuiteType": "MQTT"}
        }
        }
    ]
}
```

AWS políticas gestionadas para AWS IoT

Para añadir permisos a usuarios, grupos y roles, es más fácil usar políticas AWS administradas que escribirlas tú mismo. Se necesita tiempo y experiencia para crear políticas administradas por el cliente de IAM que proporcionen a su equipo solo los permisos necesarios. Para empezar rápidamente, puedes usar nuestras políticas AWS gestionadas. Estas políticas cubren casos de uso comunes y están disponibles en su Cuenta de AWS. Para obtener más información sobre las

políticas AWS administradas, consulte las políticas AWS administradas en la Guía del usuario de IAM.

AWS los servicios mantienen y AWS actualizan las políticas gestionadas. No puede cambiar los permisos en las políticas AWS gestionadas. En ocasiones, los servicios agregan permisos adicionales a una política administrada de AWS para admitir características nuevas. Este tipo de actualización afecta a todas las identidades (usuarios, grupos y roles) donde se asocia la política. Es más probable que los servicios actualicen una política administrada de AWS cuando se lanza una nueva característica o cuando se ponen a disposición nuevas operaciones. Los servicios no eliminan los permisos de una política AWS administrada, por lo que las actualizaciones de la política no afectarán a los permisos existentes.

Además, AWS admite políticas administradas para funciones laborales que abarcan varios servicios. Por ejemplo, la política ReadOnlyAccess AWS gestionada proporciona acceso de solo lectura a todos los AWS servicios y recursos. Cuando un servicio lanza una nueva característica, AWS agrega permisos de solo lectura para las operaciones y los recursos nuevos. Para obtener una lista y descripciones de las políticas de funciones de trabajo, consulte Políticas administradas de AWS para funciones de trabajo en la Guía del usuario de IAM.

> **Note**
>
> AWS IoT funciona tanto con las políticas de IAM como con AWS IoT las de IAM. En este tema se analizan únicamente las políticas de IAM, que definen una acción de política para las operaciones de las API del plano de control y del plano de datos. Véase también AWS IoT Core políticas.

AWS política gestionada: Acceso AWSIo TConfig

Puede asociar la política `AWSIoTConfigAccess` a las identidades de IAM.

Esta política concede los permisos de identidad asociados que permiten el acceso a todas las operaciones de configuración de AWS IoT . Esta política puede afectar al procesamiento y al almacenamiento de datos. Para ver esta política en Consola de administración de AWS, consulte AWSIoTConfigAcceso.

Detalles de los permisos

Esta política incluye los siguientes permisos.

- `iot`— Recuperar AWS IoT datos y realizar acciones de configuración de IoT.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:AcceptCertificateTransfer",
                "iot:AddThingToThingGroup",
                "iot:AssociateTargetsWithJob",
                "iot:AttachPolicy",
                "iot:AttachPrincipalPolicy",
                "iot:AttachThingPrincipal",
                "iot:CancelCertificateTransfer",
                "iot:CancelJob",
                "iot:CancelJobExecution",
                "iot:ClearDefaultAuthorizer",
                "iot:CreateAuthorizer",
                "iot:CreateCertificateFromCsr",
                "iot:CreateJob",
                "iot:CreateKeysAndCertificate",
                "iot:CreateOTAUpdate",
                "iot:CreatePolicy",
                "iot:CreatePolicyVersion",
                "iot:CreateRoleAlias",
                "iot:CreateStream",
                "iot:CreateThing",
                "iot:CreateThingGroup",
                "iot:CreateThingType",
                "iot:CreateTopicRule",
                "iot:DeleteAuthorizer",
                "iot:DeleteCACertificate",
                "iot:DeleteCertificate",
                "iot:DeleteJob",
```

```
                "iot:DeleteJobExecution",
                "iot:DeleteOTAUpdate",
                "iot:DeletePolicy",
                "iot:DeletePolicyVersion",
                "iot:DeleteRegistrationCode",
                "iot:DeleteRoleAlias",
                "iot:DeleteStream",
                "iot:DeleteThing",
                "iot:DeleteThingGroup",
                "iot:DeleteThingType",
                "iot:DeleteTopicRule",
                "iot:DeleteV2LoggingLevel",
                "iot:DeprecateThingType",
                "iot:DescribeAuthorizer",
                "iot:DescribeCACertificate",
                "iot:DescribeCertificate",
                "iot:DescribeDefaultAuthorizer",
                "iot:DescribeEndpoint",
                "iot:DescribeEventConfigurations",
                "iot:DescribeIndex",
                "iot:DescribeJob",
                "iot:DescribeJobExecution",
                "iot:DescribeRoleAlias",
                "iot:DescribeStream",
                "iot:DescribeThing",
                "iot:DescribeThingGroup",
                "iot:DescribeThingRegistrationTask",
                "iot:DescribeThingType",
                "iot:DetachPolicy",
                "iot:DetachPrincipalPolicy",
                "iot:DetachThingPrincipal",
                "iot:DisableTopicRule",
                "iot:EnableTopicRule",
                "iot:GetEffectivePolicies",
                "iot:GetIndexingConfiguration",
                "iot:GetJobDocument",
                "iot:GetLoggingOptions",
                "iot:GetOTAUpdate",
                "iot:GetPolicy",
                "iot:GetPolicyVersion",
                "iot:GetRegistrationCode",
                "iot:GetTopicRule",
                "iot:GetV2LoggingOptions",
                "iot:ListAttachedPolicies",
```

```
                "iot:ListAuthorizers",
                "iot:ListCACertificates",
                "iot:ListCertificates",
                "iot:ListCertificatesByCA",
                "iot:ListIndices",
                "iot:ListJobExecutionsForJob",
                "iot:ListJobExecutionsForThing",
                "iot:ListJobs",
                "iot:ListOTAUpdates",
                "iot:ListOutgoingCertificates",
                "iot:ListPolicies",
                "iot:ListPolicyPrincipals",
                "iot:ListPolicyVersions",
                "iot:ListPrincipalPolicies",
                "iot:ListPrincipalThings",
                "iot:ListRoleAliases",
                "iot:ListStreams",
                "iot:ListTargetsForPolicy",
                "iot:ListThingGroups",
                "iot:ListThingGroupsForThing",
                "iot:ListThingPrincipals",
                "iot:ListThingRegistrationTaskReports",
                "iot:ListThingRegistrationTasks",
                "iot:ListThings",
                "iot:ListThingsInThingGroup",
                "iot:ListThingTypes",
                "iot:ListTopicRules",
                "iot:ListV2LoggingLevels",
                "iot:RegisterCACertificate",
                "iot:RegisterCertificate",
                "iot:RegisterThing",
                "iot:RejectCertificateTransfer",
                "iot:RemoveThingFromThingGroup",
                "iot:ReplaceTopicRule",
                "iot:SearchIndex",
                "iot:SetDefaultAuthorizer",
                "iot:SetDefaultPolicyVersion",
                "iot:SetLoggingOptions",
                "iot:SetV2LoggingLevel",
                "iot:SetV2LoggingOptions",
                "iot:StartThingRegistrationTask",
                "iot:StopThingRegistrationTask",
                "iot:TestAuthorization",
                "iot:TestInvokeAuthorizer",
```

```
                "iot:TransferCertificate",
                "iot:UpdateAuthorizer",
                "iot:UpdateCACertificate",
                "iot:UpdateCertificate",
                "iot:UpdateEventConfigurations",
                "iot:UpdateIndexingConfiguration",
                "iot:UpdateRoleAlias",
                "iot:UpdateStream",
                "iot:UpdateThing",
                "iot:UpdateThingGroup",
                "iot:UpdateThingGroupsForThing",
                "iot:UpdateAccountAuditConfiguration",
                "iot:DescribeAccountAuditConfiguration",
                "iot:DeleteAccountAuditConfiguration",
                "iot:StartOnDemandAuditTask",
                "iot:CancelAuditTask",
                "iot:DescribeAuditTask",
                "iot:ListAuditTasks",
                "iot:CreateScheduledAudit",
                "iot:UpdateScheduledAudit",
                "iot:DeleteScheduledAudit",
                "iot:DescribeScheduledAudit",
                "iot:ListScheduledAudits",
                "iot:ListAuditFindings",
                "iot:CreateSecurityProfile",
                "iot:DescribeSecurityProfile",
                "iot:UpdateSecurityProfile",
                "iot:DeleteSecurityProfile",
                "iot:AttachSecurityProfile",
                "iot:DetachSecurityProfile",
                "iot:ListSecurityProfiles",
                "iot:ListSecurityProfilesForTarget",
                "iot:ListTargetsForSecurityProfile",
                "iot:ListActiveViolations",
                "iot:ListViolationEvents",
                "iot:ValidateSecurityProfileBehaviors"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS política gestionada: AWSIo TConfig ReadOnlyAccess

Puede asociar la política `AWSIoTConfigReadOnlyAccess` a las identidades de IAM.

Esta política concede los permisos de identidad asociados que permiten el acceso de solo lectura a todas las operaciones de configuración de AWS IoT . Para ver esta política en Consola de administración de AWS, consulte AWSIoTConfigReadOnlyAccess.

Detalles de los permisos

Esta política incluye los siguientes permisos.

- `iot`: realice operaciones de solo lectura de las acciones de configuración de IoT.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:DescribeAuthorizer",
                "iot:DescribeCACertificate",
                "iot:DescribeCertificate",
                "iot:DescribeDefaultAuthorizer",
                "iot:DescribeEndpoint",
                "iot:DescribeEventConfigurations",
                "iot:DescribeIndex",
                "iot:DescribeJob",
                "iot:DescribeJobExecution",
                "iot:DescribeRoleAlias",
                "iot:DescribeStream",
                "iot:DescribeThing",
                "iot:DescribeThingGroup",
                "iot:DescribeThingRegistrationTask",
                "iot:DescribeThingType",
                "iot:GetEffectivePolicies",
                "iot:GetIndexingConfiguration",
```

```
                "iot:GetJobDocument",
                "iot:GetLoggingOptions",
                "iot:GetOTAUpdate",
                "iot:GetPolicy",
                "iot:GetPolicyVersion",
                "iot:GetRegistrationCode",
                "iot:GetTopicRule",
                "iot:GetV2LoggingOptions",
                "iot:ListAttachedPolicies",
                "iot:ListAuthorizers",
                "iot:ListCACertificates",
                "iot:ListCertificates",
                "iot:ListCertificatesByCA",
                "iot:ListIndices",
                "iot:ListJobExecutionsForJob",
                "iot:ListJobExecutionsForThing",
                "iot:ListJobs",
                "iot:ListOTAUpdates",
                "iot:ListOutgoingCertificates",
                "iot:ListPolicies",
                "iot:ListPolicyPrincipals",
                "iot:ListPolicyVersions",
                "iot:ListPrincipalPolicies",
                "iot:ListPrincipalThings",
                "iot:ListRoleAliases",
                "iot:ListStreams",
                "iot:ListTargetsForPolicy",
                "iot:ListThingGroups",
                "iot:ListThingGroupsForThing",
                "iot:ListThingPrincipals",
                "iot:ListThingRegistrationTaskReports",
                "iot:ListThingRegistrationTasks",
                "iot:ListThings",
                "iot:ListThingsInThingGroup",
                "iot:ListThingTypes",
                "iot:ListTopicRules",
                "iot:ListV2LoggingLevels",
                "iot:SearchIndex",
                "iot:TestAuthorization",
                "iot:TestInvokeAuthorizer",
                "iot:DescribeAccountAuditConfiguration",
                "iot:DescribeAuditTask",
                "iot:ListAuditTasks",
                "iot:DescribeScheduledAudit",
```

```
                "iot:ListScheduledAudits",
                "iot:ListAuditFindings",
                "iot:DescribeSecurityProfile",
                "iot:ListSecurityProfiles",
                "iot:ListSecurityProfilesForTarget",
                "iot:ListTargetsForSecurityProfile",
                "iot:ListActiveViolations",
                "iot:ListViolationEvents",
                "iot:ValidateSecurityProfileBehaviors"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS política gestionada: AWSIo TData Acceso

Puede asociar la política `AWSIoTDataAccess` a las identidades de IAM.

Esta política concede a los permisos de identidad asociados que permiten el acceso a todas las operaciones de AWS IoT datos. Las operaciones de datos envían datos sobre protocolos MQTT y HTTP. Para ver esta política en la Consola de administración de AWS, consulte `AWSIoTDataAccess`.

Detalles de los permisos

Esta política incluye los siguientes permisos.

- `iot`— Recupere AWS IoT datos y permita el acceso total a las acciones AWS IoT de mensajería.

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

```
            "Effect": "Allow",
            "Action": [
                "iot:Connect",
                "iot:Publish",
                "iot:Subscribe",
                "iot:Receive",
                "iot:GetThingShadow",
                "iot:UpdateThingShadow",
                "iot:DeleteThingShadow",
                "iot:ListNamedShadowsForThing"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS política gestionada: AWSIo TFull Acceso

Puede asociar la política `AWSIoTFullAccess` a las identidades de IAM.

Esta política concede los permisos de identidad asociados que permiten el acceso a todas las operaciones de configuración y mensajería de AWS IoT . Para ver esta política en Consola de administración de AWS, consulte `AWSIoTFullAccess`.

Detalles de los permisos

Esta política incluye los siguientes permisos.

- `iot`— Recupere AWS IoT datos y permita el acceso total a las acciones de AWS IoT configuración y mensajería.
- `iotjobsdata`— Recupere los datos de AWS IoT Jobs y permita el acceso total a las operaciones de la API del plano de datos de AWS IoT Jobs.

```
{
```

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

AWS política gestionada: AWSIo TLogging

Puede asociar la política `AWSIoTLogging` a las identidades de IAM.

Esta política concede los permisos de identidad asociados que permiten el acceso para crear grupos de Amazon CloudWatch Logs y transmitir registros a los grupos. Esta política está asociada a su función de CloudWatch registro. Para ver esta política en Consola de administración de AWS, consulte `AWSIoTLogging`.

Detalles de los permisos

Esta política incluye los siguientes permisos.

- `logs`— Recuperar CloudWatch registros. También permite crear grupos de CloudWatch registros y transmitir los registros a los grupos.

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

```
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:PutMetricFilter",
                "logs:PutRetentionPolicy",
                "logs:GetLogEvents",
                "logs:DeleteLogStream"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

AWS política gestionada: AWSIo TOTAUpdate

Puede asociar la política `AWSIoTOTAUpdate` a las identidades de IAM.

Esta política concede los permisos de identidad asociados que permiten acceder a los AWS IoT trabajos de creación y firma de AWS IoT código y a describir los trabajos de los firmantes de AWS código. Para ver esta política en Consola de administración de AWS, consulte`AWSIoTOTAUpdate`.

Detalles de los permisos

Esta política incluye los siguientes permisos.

- `iot`— Crear AWS IoT trabajos y trabajos de firma de código.
- `signer`— Realizar la creación de trabajos de firmante de AWS código.

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

```
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "iot:CreateJob",
            "signer:DescribeSigningJob"
        ],
        "Resource": "*"
    }
}
```

AWS política gestionada: acciones AWSIo TRule

Puede asociar la política `AWSIoTRuleActions` a las identidades de IAM.

Esta política concede a los permisos de identidad asociados que permiten el acceso a todas las Servicio de AWS acciones de las AWS IoT reglas. Para ver esta política en Consola de administración de AWS, consulte `AWSIoTRuleActions`.

Detalles de los permisos

Esta política incluye los siguientes permisos.

- `iot`: realice acciones para publicar los mensajes de acción de las reglas.
- `dynamodb`: inserte un mensaje en una tabla de DynamoDB o divida un mensaje en varias columnas de una tabla de DynamoDB.
- `s3`: almacene un objeto a un bucket de Amazon S3.
- `kinesis`: envíe un mensaje a un objeto de flujo de Amazon Kinesis.
- `firehose`: Inserte un registro en un objeto de flujo de Firehose.
- `cloudwatch`- Cambie el estado CloudWatch de la alarma o envíe los datos del mensaje a la CloudWatch métrica.
- `sns`: realice la operación para publicar una notificación utilizando Amazon SNS. Esta operación se centra en temas relacionados con las AWS IoT redes sociales.

- `sqs`: inserte un mensaje para agregarlo a la cola SQS.
- `es`- Enviar un mensaje al OpenSearch servicio de servicio.

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "dynamodb:PutItem",
            "kinesis:PutRecord",
            "iot:Publish",
            "s3:PutObject",
            "sns:Publish",
            "sqs:SendMessage*",
            "cloudwatch:SetAlarmState",
            "cloudwatch:PutMetricData",
            "es:ESHttpPut",
            "firehose:PutRecord"
        ],
        "Resource": "*"
    }
}
```

AWS política gestionada: AWSIo TThings registro

Puede asociar la política `AWSIoTThingsRegistration` a las identidades de IAM.

Esta política otorga los permisos de identidad asociados que permiten acceder y registrar objetos de forma masiva mediante la API `StartThingRegistrationTask`. Esta política puede afectar al procesamiento y al almacenamiento de datos. Para ver esta política en Consola de administración de AWS, consulte `AWSIoTThingsRegistration`.

Detalles de los permisos

Esta política incluye los siguientes permisos.

- `iot`: realice acciones para crear objetos y asociar políticas y certificados al registrarse de forma masiva.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:AddThingToThingGroup",
                "iot:AttachPolicy",
                "iot:AttachPrincipalPolicy",
                "iot:AttachThingPrincipal",
                "iot:CreateCertificateFromCsr",
                "iot:CreatePolicy",
                "iot:CreateThing",
                "iot:DescribeCertificate",
                "iot:DescribeThing",
                "iot:DescribeThingGroup",
                "iot:DescribeThingType",
                "iot:DetachPolicy",
                "iot:DetachThingPrincipal",
                "iot:GetPolicy",
                "iot:ListAttachedPolicies",
                "iot:ListPolicyPrincipals",
                "iot:ListPrincipalPolicies",
                "iot:ListPrincipalThings",
                "iot:ListTargetsForPolicy",
                "iot:ListThingGroupsForThing",
                "iot:ListThingPrincipals",
                "iot:RegisterCertificate",
                "iot:RegisterThing",
                "iot:RemoveThingFromThingGroup",
                "iot:UpdateCertificate",
                "iot:UpdateThing",
                "iot:UpdateThingGroupsForThing",
                "iot:AddThingToBillingGroup",
                "iot:DescribeBillingGroup",
                "iot:RemoveThingFromBillingGroup"
            ],
```

```
            "Resource": [
                "*"
            ]
        }
    ]
}
```

AWS IoT actualizaciones de las políticas AWS gestionadas

Consulte los detalles sobre las actualizaciones de las políticas AWS administradas AWS IoT desde que este servicio comenzó a rastrear estos cambios. Para recibir alertas automáticas sobre los cambios en esta página, suscríbase a la fuente RSS de la página del historial del AWS IoT documento.

Cambio	Descripción	Fecha
AWSIoTFullAcceso: actualización a una política existente	AWS IoT se agregaron nuevos permisos para permitir a los usuarios acceder a las operaciones de la API del plano de datos de AWS IoT Jobs mediante el protocolo HTTP. Un nuevo prefijo de política de IAM proporciona un control de acceso más detallado para acceder a los puntos finales del plano de datos de AWS IoT Jobs. `iotjobsdata:` Para las operaciones de la API del plano de control, se seguirá utilizando el prefijo `iot:`. Para obtener más información, consulte AWS	11 de mayo de 2022

Cambio	Descripción	Fecha
	IoT Core políticas para el protocolo HTTPS.	
AWS IoT comenzó a rastrear los cambios	AWS IoT comenzó a realizar un seguimiento de los cambios de sus políticas AWS gestionadas.	11 de mayo de 2022

Solución de problemas AWS IoT de identidad y acceso

Utilice la siguiente información como ayuda para diagnosticar y solucionar los problemas habituales que pueden surgir al trabajar con un AWS IoT IAM.

Temas

- No estoy autorizado a realizar ninguna acción en AWS IoT
- No estoy autorizado a realizar iam: PassRole
- Quiero permitir que personas ajenas a mí accedan Cuenta de AWS a mis AWS IoT recursos

No estoy autorizado a realizar ninguna acción en AWS IoT

Si recibe un error que indica que no tiene autorización para realizar una acción, las políticas se deben actualizar para permitirle realizar la acción.

En el siguiente ejemplo, el error se produce cuando el usuario de IAM, `mateojackson`, intenta utilizar la consola para consultar los detalles acerca de un recurso de objeto, pero no tiene los permisos `iot:DescribeThing`.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 iot:DescribeThing on resource: MyIoTThing
```

En este caso, la política del usuario de `mateojackson` debe actualizarse para permitir el acceso al recurso mediante la acción `iot:DescribeThing`.

Si necesita ayuda, póngase en contacto con su AWS administrador. El administrador es la persona que le proporcionó las credenciales de inicio de sesión.

Uso de AWS IoT Device Advisor

Si utiliza AWS IoT Device Advisor, el siguiente ejemplo de error se produce cuando el usuario `mateojackson` intenta utilizar la consola para ver los detalles sobre la definición de una suite pero no tiene los `iotdeviceadvisor:GetSuiteDefinition` permisos necesarios.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 iotdeviceadvisor:GetSuiteDefinition on resource: MySuiteDefinition
```

En este caso, la política del usuario de `mateojackson` debe actualizarse para permitir el acceso al recurso *`MySuiteDefinition`* mediante la acción `iotdeviceadvisor:GetSuiteDefinition`.

No estoy autorizado a realizar iam: PassRole

Si recibe un error que indica que no tiene autorización para realizar la acción `iam:PassRole`, las políticas deben actualizarse a fin de permitirle pasar un rol a AWS IoT.

Algunas Servicios de AWS permiten transferir una función existente a ese servicio en lugar de crear una nueva función de servicio o una función vinculada a un servicio. Para ello, debe tener permisos para transferir la función al servicio.

En el siguiente ejemplo, el error se produce cuando un usuario de IAM denominado `marymajor` intenta utilizar la consola para realizar una acción en AWS IoT. Sin embargo, la acción requiere que el servicio cuente con permisos que otorguen un rol de servicio. Mary no tiene permisos para transferir el rol al servicio.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

En este caso, las políticas de Mary se deben actualizar para permitirle realizar la acción `iam:PassRole`.

Si necesita ayuda, póngase en contacto con su AWS administrador. El administrador es la persona que le proporcionó las credenciales de inicio de sesión.

Quiero permitir que personas ajenas a mí accedan Cuenta de AWS a mis AWS IoT recursos

Se puede crear un rol que los usuarios de otras cuentas o las personas externas a la organización puedan utilizar para acceder a sus recursos. Se puede especificar una persona de confianza para que asuma el rol. En el caso de los servicios que respaldan políticas basadas en recursos o listas de control de acceso (ACLs), puedes usar esas políticas para permitir que las personas accedan a tus recursos.

Para obtener más información, consulte lo siguiente:

- Para saber si AWS IoT es compatible con estas funciones, consulte. ¿Cómo AWS IoT funciona con IAM
- Para obtener información sobre cómo proporcionar acceso a los recursos de su Cuentas de AWS propiedad, consulte Proporcionar acceso a un usuario de IAM en otro usuario de su propiedad Cuenta de AWS en la Guía del usuario de IAM.
- Para obtener información sobre cómo proporcionar acceso a tus recursos a terceros Cuentas de AWS, consulta Cómo proporcionar acceso a recursos que Cuentas de AWS son propiedad de terceros en la Guía del usuario de IAM.
- Para obtener información sobre cómo proporcionar acceso mediante una federación de identidades, consulte Proporcionar acceso a usuarios autenticados externamente (identidad federada) en la Guía del usuario de IAM.
- Para conocer sobre la diferencia entre las políticas basadas en roles y en recursos para el acceso entre cuentas, consulte Acceso a recursos entre cuentas en IAM en la Guía del usuario de IAM.

Registro y supervisión

La supervisión es una parte importante del mantenimiento de la fiabilidad, la disponibilidad y el rendimiento de AWS IoT sus AWS soluciones. Debe recopilar los datos de supervisión de todas las partes de la AWS solución para poder depurar con mayor facilidad una falla multipunto, en caso de que se produzca alguna. Para obtener información sobre los procedimientos de registro y monitorización , consulte Monitorización AWS IoT

Herramientas de monitorización

AWS proporciona herramientas que puede utilizar para supervisar AWS IoT. Puede configurar algunas de estas herramientas para que realicen la monitorización por usted. Algunas de las

herramientas requieren intervención manual. Le recomendamos que automatice las tareas de supervisión en la medida de lo posible.

Herramientas de monitorización automatizadas

Puede utilizar las siguientes herramientas de supervisión automatizadas para observar AWS IoT e informar cuando algo va mal:

- Amazon CloudWatch Alarms: observe una sola métrica durante un período de tiempo que especifique y realice una o más acciones en función del valor de la métrica en relación con un umbral determinado durante varios períodos de tiempo. La acción es una notificación enviada a un tema del Servicio de Notificación Simple (Amazon SNS) o a una política de Amazon EC2 Auto Scaling. CloudWatch las alarmas no invocan acciones simplemente porque se encuentren en un estado determinado. El estado debe haber cambiado y debe mantenerse durante el número de periodos especificado. Para obtener más información, consulte Supervisa AWS IoT las alarmas y las métricas con Amazon CloudWatch.
- Amazon CloudWatch Logs: supervise, almacene y acceda a sus archivos de registro desde AWS CloudTrail u otras fuentes. Amazon CloudWatch Logs también le permite ver los pasos críticos que siguen los casos de prueba de AWS IoT Device Advisor, los eventos generados y los mensajes MQTT enviados desde sus dispositivos o AWS IoT Core durante la ejecución de las pruebas. Estos registros permiten depurar y tomar medidas correctivas en sus dispositivos. Para obtener más información, consulte Supervise AWS IoT mediante registros CloudWatch Para obtener más información sobre el uso de Amazon CloudWatch, consulte Supervisión de archivos de registro en la Guía del CloudWatch usuario de Amazon.
- Amazon CloudWatch Events: haga coincidir los eventos y diríjalos a una o más funciones o transmisiones de destino para realizar cambios, capturar información de estado y tomar medidas correctivas. Para obtener más información, consulta Qué es Amazon CloudWatch Events en la Guía del CloudWatch usuario de Amazon.
- AWS CloudTrail Supervisión de registros: comparta archivos de registro entre cuentas, supervise los archivos de CloudTrail registro en tiempo real enviándolos a CloudWatch Logs, cree aplicaciones de procesamiento de registros en Java y valide que sus archivos de registro no hayan cambiado después de su entrega CloudTrail. Para obtener más información, consulte Registrar las llamadas a la AWS IoT API mediante AWS CloudTrail y trabaje con archivos de CloudTrail registro en la Guía del AWS CloudTrail usuario.

Herramientas de monitorización manual

Otra parte importante del monitoreo AWS IoT consiste en monitorear manualmente los elementos que las CloudWatch alarmas no cubren. Los AWS IoT paneles de control de la consola de AWS servicio y otros proporcionan una at-a-glance vista del estado de su AWS entorno. CloudWatch Le recomendamos que compruebe también los archivos de registro. AWS IoT

- AWS IoT el panel de control muestra:
  - Certificados CA
  - Certificados
  - Políticas
  - Reglas
  - Objetos
- CloudWatch la página de inicio muestra:
  - Alarmas y estado actual.
  - Gráficos de alarmas y recursos.
  - Estado de los servicios.

  Se puede utilizar CloudWatch para hacer lo siguiente:
  - Cree paneles personalizados para monitorizar los servicios que le interesen.
  - Realizar un gráfico con los datos de las métricas para resolver problemas y descubrir tendencias.
  - Busca y examina todas las métricas AWS de tus recursos.
  - Crear y editar las alarmas de notificación de problemas.

Validación del cumplimiento de AWS IoT Core

Para saber si uno Servicio de AWS está dentro del ámbito de aplicación de programas de cumplimiento específicos, consulte Servicios de AWS Alcance por programa de cumplimiento Servicios de AWS de cumplimiento y elija el programa de cumplimiento que le interese. Para obtener información general, consulte Programas de AWS cumplimiento > Programas AWS .

Puede descargar informes de auditoría de terceros utilizando AWS Artifact. Para obtener más información, consulte Descarga de informes en AWS Artifact .

Su responsabilidad de cumplimiento al Servicios de AWS utilizarlos viene determinada por la confidencialidad de sus datos, los objetivos de cumplimiento de su empresa y las leyes y reglamentos aplicables. Para obtener más información sobre su responsabilidad de conformidad al utilizarlos Servicios de AWS, consulte AWS la documentación de seguridad.

La resiliencia en el núcleo del AWS IoT

La infraestructura AWS global se basa en Región de AWS s y zonas de disponibilidad. Región de AWS Las s proporcionan varias zonas de disponibilidad aisladas y separadas físicamente, que están conectadas mediante redes de baja latencia, alto rendimiento y alta redundancia. Con las zonas de disponibilidad, puede diseñar y utilizar aplicaciones y bases de datos que realizan una conmutación por error automática entre zonas de disponibilidad sin interrupciones. Las zonas de disponibilidad tienen una mayor disponibilidad, tolerancia a errores y escalabilidad que las infraestructuras tradicionales de centros de datos únicos o múltiples.

Para obtener más información sobre las zonas de disponibilidad y Región de AWS las zonas de disponibilidad, consulte Infraestructura global.AWS

AWS IoT Core almacena información sobre sus dispositivos en el registro de dispositivos. También almacena certificados de CA, certificados de dispositivo y datos de sombra de dispositivos. En caso de fallo del hardware o de la red, estos datos se replican automáticamente entre las zonas de disponibilidad, pero no entre las regiones.

AWS IoT Core publica los eventos de MQTT cuando se actualiza el registro del dispositivo. Puede utilizar estos mensajes para realizar una copia de seguridad de los datos del registro y guardarlos en algún lugar, como una tabla DynamoDB. Usted es responsable de guardar los certificados que AWS IoT Core cree para usted o los que cree usted mismo. Device Shadow almacena los datos de estado de tus dispositivos y se puede volver a enviar cuando un dispositivo vuelve a estar en línea. AWS IoT Device Advisor almacena información sobre la configuración del conjunto de pruebas. Estos datos se replican automáticamente en caso de fallo del hardware o de la red.

AWS IoT Core los recursos son específicos de cada región y no se replican entre ellos Regiones de AWS a menos que usted lo haga específicamente.

Para obtener información sobre las prácticas recomendadas de seguridad, consulte Mejores prácticas de seguridad en AWS IoT Core.

Uso AWS IoT Core con puntos finales de VPC de interfaz

Con AWS IoT Core, puede crear puntos finales del plano de control de IoT y puntos finales de datos de IoT dentro de su nube privada virtual (VPC) mediante puntos finales de VPC de interfaz. Los puntos finales de VPC de interfaz funcionan con una AWS tecnología que puede utilizar para acceder a los servicios en los que se ejecutan AWS mediante direcciones IP privadas. AWS PrivateLink Para obtener más información, consulte Amazon Virtual Private Cloud.

Para conectar dispositivos sobre el terreno en redes remotas, como una red corporativa, a su Amazon VPC, consulte las opciones que se muestran en la matriz de conectividad de la Network-to-Amazon VPC.

Contenido

- Creación de puntos finales de VPC para el plano de control AWS IoT Core
- Creación de puntos finales de VPC para el plano de datos AWS IoT Core
- Creación de puntos de enlace de VPC para el proveedor de credenciales AWS IoT Core
- Creación de un punto de conexión de interfaz de Amazon VPC
- Configure una zona alojada privada
- Control del acceso a AWS IoT Core más de puntos finales de VPC
- Limitaciones
- Escalar los puntos finales de VPC con AWS IoT Core
- Uso de dominios personalizados con puntos de conexión de VPC
- Disponibilidad de puntos finales de VPC para AWS IoT Core
- Uso de túneles AWS IoT Device Management seguros con puntos finales de VPC de interfaz

Creación de puntos finales de VPC para el plano de control AWS IoT Core

Puede crear un punto final de VPC para la API del plano de AWS IoT Core control para conectar sus dispositivos a AWS IoT servicios y otros AWS servicios. Para empezar con los puntos de enlace de VPC, cree un punto de enlace de VPC de interfaz y selecciónelo como servicio. AWS IoT Core AWS Si utiliza la CLI, primero llame describe-vpc-endpoint-servicespara asegurarse de que está eligiendo una zona de disponibilidad en la que AWS IoT Core esté presente en su particular Región de AWS. Por ejemplo, en us-east-1, este comando tendría el siguiente aspecto:

```
aws ec2 describe-vpc-endpoint-services --service-name com.amazonaws.us-east-1.iot.api
```

Consulte las instrucciones detalladas que aparecen a continuación para crear un punto final de interfaz de Amazon VPC para el plano de AWS IoT Core control.

Creación de puntos finales de VPC para el plano de datos AWS IoT Core

Puede crear un punto final de VPC para la API del plano de AWS IoT Core datos para conectar sus dispositivos a AWS IoT servicios y otros AWS servicios. Para empezar con los puntos de enlace de VPC, cree un punto de enlace de VPC de interfaz y selecciónelo como servicio. AWS IoT Core AWS Si utiliza la CLI, primero llame describe-vpc-endpoint-servicespara asegurarse de que está eligiendo una zona de disponibilidad en la que AWS IoT Core esté presente en su particular Región de AWS. Por ejemplo, en us-east-1, este comando tendría el siguiente aspecto:

```
aws ec2 describe-vpc-endpoint-services --service-name com.amazonaws.us-east-1.iot.data
```

> **Note**
>
> La característica VPC para crear automáticamente un registro DNS está desactivada. Para conectarse a estos puntos de conexión, debe crear manualmente un registro DNS privado. Para obtener más información sobre los registros DNS de la VPC privada, consulte DNS privado para puntos de conexión de interfaz. Para obtener más información sobre las limitaciones de la AWS IoT Core VPC, consulte. Limitaciones

Para conectar clientes MQTT a las interfaces de punto de conexión de VPC:

- Debe crear manualmente registros DNS en una zona alojada privada que esté asociada a su VPC. Para empezar, consulte Creación de una zona alojada privada.

- Dentro de su zona alojada privada, cree un registro de alias para cada IP de interfaz de red elástica para el punto de conexión de VPC. Si tiene varias interfaces de red IPs para varios puntos finales de VPC, cree registros DNS ponderados con el mismo peso en todos los registros ponderados. Estas direcciones IP están disponibles en la llamada a la DescribeNetworkInterfacesAPI cuando se filtran por el ID del punto final de la VPC en el campo de descripción.

Consulte las instrucciones detalladas que aparecen a continuación para crear un punto final de interfaz de Amazon VPC y configurar una zona alojada privada para el plano de AWS IoT Core datos.

Creación de puntos de enlace de VPC para el proveedor de credenciales AWS IoT Core

Puede crear un punto final de VPC para que el proveedor de AWS IoT Core credenciales conecte los dispositivos mediante la autenticación basada en certificados de cliente y obtenga AWS credenciales temporales en AWS formato Signature Version 4. Para empezar con los puntos de enlace de VPC para el proveedor de AWS IoT Core credenciales, ejecute el comando create-vpc-endpointCLI para crear un punto de enlace de VPC de interfaz y seleccione AWS IoT Core el proveedor de credenciales como servicio. AWS Para asegurarse de elegir una zona de disponibilidad que AWS IoT Core esté presente en la suya Región de AWS, ejecute primero el comando. describe-vpc-endpoint-services Por ejemplo, en us-east-1, este comando tendría el siguiente aspecto:

```
aws ec2 describe-vpc-endpoint-services --service-name com.amazonaws.us-
east-1.iot.credentials
```

> **Note**
>
> La característica VPC para crear automáticamente un registro DNS está desactivada. Para conectarse a estos puntos de conexión, debe crear manualmente un registro DNS privado. Para obtener más información sobre los registros DNS de la VPC privada, consulte DNS privado para puntos de conexión de interfaz. Para obtener más información sobre las limitaciones de la AWS IoT Core VPC, consulte. Limitaciones

Para conectar los clientes HTTP a las interfaces de punto de conexión de VPC:

- Debe crear manualmente registros DNS en una zona alojada privada que esté asociada a su VPC. Para empezar, consulte Creación de una zona alojada privada.

- Dentro de su zona alojada privada, cree un registro de alias para cada IP de interfaz de red elástica para el punto de conexión de VPC. Si tiene varias interfaces de red IPs para varios puntos finales de VPC, cree registros DNS ponderados con el mismo peso en todos los registros ponderados. Estas direcciones IP están disponibles en la llamada a la

DescribeNetworkInterfacesAPI cuando se filtran por el ID del punto final de la VPC en el campo de descripción.

Consulte las instrucciones detalladas que aparecen a continuación para crear un punto final de interfaz de Amazon VPC y configurar una zona alojada privada para el proveedor de AWS IoT Core credenciales.

Creación de un punto de conexión de interfaz de Amazon VPC

Puede crear un punto final de VPC de interfaz para conectarse a los AWS servicios impulsados por. AWS PrivateLink Utilice el siguiente procedimiento para crear un punto final de VPC de interfaz que se conecte al plano de AWS IoT Core datos o al proveedor de AWS IoT Core credenciales. Para obtener más información, consulte Acceder a un AWS servicio mediante un punto final de VPC de interfaz.

> **Note**
>
> Los procesos para crear un punto final de la interfaz de Amazon VPC para el plano de AWS IoT Core datos y el proveedor de AWS IoT Core credenciales son similares, pero debe realizar cambios específicos para que la conexión funcione.

Para crear un punto de conexión de VPC de interfaz utilizando la consola de puntos de conexión de VPC

1. Vaya a la consola de puntos de conexión de VPC, en Nube virtual privada en el menú de la izquierda, elija Puntos de enlace y después Crear punto de conexión.
2. En la página Crear punto de conexión, especifique la siguiente información.
   - Elija Servicio de AWS en Categoría de servicio.
   - En Nombre del servicio, busque introduciendo la palabra clave `iot`. En la lista de servicios de `iot` que se muestra, elija el punto de conexión.

     Si crea un punto final de VPC para el plano de AWS IoT Core control, elija el punto final de la API del plano de AWS IoT Core control para su. Región de AWS El punto de conexión tendrá el formato `com.amazonaws.`*`region`*`.iot.api`.

Si crea un punto de enlace de VPC para el plano de AWS IoT Core datos, elija el punto de enlace de la API del plano de AWS IoT Core datos para su región. El punto de conexión tendrá el formato `com.amazonaws.region.iot.data`.

Si crea un punto de enlace de VPC para el proveedor de AWS IoT Core credenciales, elija el punto de enlace del proveedor de AWS IoT Core credenciales para su región. El punto de conexión tendrá el formato `com.amazonaws.region.iot.credentials`.

Si crea un punto de enlace de VPC para las regiones del Estándar Federal de Procesamiento de Información (FIPS), elija el punto de enlace de la API de FIPS para usted. Región de AWS El punto de conexión tendrá el formato `com.amazonaws.region.iot-fips.api`. Esto es solo para el plano de control.

> **Note**
>
> El nombre del servicio para el plano de AWS IoT Core datos en la región de China tendrá este formato`cn.com.amazonaws.region.iot.data`. El nombre del servicio para AWS IoT Core el avión de control en la región de China tendrá ese formato`com.amazonaws.region.iot.api`.

- Para la VPC y las subredes, elija la VPC en la que desee crear el punto final y las zonas de disponibilidad (AZs) en las que desee crear la red del punto final.
- En Habilitar el nombre DNS, asegúrese de que la opción Habilitar para este punto final no esté seleccionada para el plano de AWS IoT Core datos y AWS IoT Core el proveedor de credenciales. Ni el plano de AWS IoT Core datos ni el proveedor de AWS IoT Core credenciales admiten todavía nombres DNS privados.

  En el plano AWS IoT Core de control, la opción Activar nombre DNS está seleccionada de forma predeterminada. Esto garantiza que cualquier solicitud a los puntos finales públicos del plano de AWS IoT Core control se enrute a través de los puntos finales de la VPC. Cuando está habilitada, no es necesario configurar una zona alojada privada.
- En Grupo de seguridad, elija los grupos de seguridad que deban asociarse a las interfaces de red de punto de conexión.
- Puede agregar o eliminar etiquetas si lo desea. Las etiquetas son pares de nombre-valor que se utilizan para asociar al punto de conexión.

3. Para crear su punto de conexión de VPC, elija Crear punto de conexión.

Después de crear el AWS PrivateLink punto final, en la pestaña Detalles del dispositivo, verá una lista de nombres de DNS. Puede utilizar uno de estos nombres DNS que ha creado en esta sección para configurar su zona alojada privada. Si utiliza el plano de AWS IoT Core control, no necesita configurar una zona alojada privada.

Configure una zona alojada privada

> **Note**
>
> Si utiliza el plano de AWS IoT Core control y ha seleccionado Habilitar el nombre DNS, no necesita configurar una zona alojada privada. Si lo deshabilita, debe seguir este procedimiento para configurar una zona alojada privada.

Puede utilizar uno de estos nombres de DNS que ha creado en la sección anterior para configurar su zona alojada privada.

Para el plano AWS IoT Core de datos

El nombre de DNS debe ser el nombre de configuración de su dominio o su punto de conexión de `IoT:Data-ATS`. Un nombre DNS de ejemplo puede ser: *`xxx`*`-ats.data.iot.`*`region`*`.amazonaws.com`.

Para el AWS IoT Core proveedor de credenciales

El nombre de DNS debe ser su de conexión de `iot:CredentialProvider`. Un nombre DNS de ejemplo puede ser: *`xxxx`*`.credentials.iot.`*`region`*`.amazonaws.com`.

Para el plano AWS IoT Core de control

El nombre DNS debe ser el punto final AWS IoT Core del plano de control. Un ejemplo de nombre DNS para el plano AWS IoT Core de control es *`xxxx`*`.api.iot.`*`region`*`.amazonaws.com`.

> **Note**
>
> Los procesos para configurar la zona alojada privada para el plano de AWS IoT Core datos y el proveedor de AWS IoT Core credenciales son similares, pero debe realizar cambios específicos para que la conexión funcione.

Creación de una zona alojada privada

Para crear una zona alojada privada a través de la consola de Route 53

1. Vaya a la consola Zonas alojadas de Route 53 y elija Crear zona alojada.
2. En la página Crear zona alojada, especifique la siguiente información.
   - En Nombre de dominio, introduzca la dirección del punto de conexión de su punto de conexión `iot:Data-ATS` o `iot:CredentialProvider`. El siguiente comando de la CLI AWS muestra cómo obtener el punto de conexión a través de una red pública: `aws iot describe-endpoint --endpoint-type iot:Data-ATS` o `aws iot describe-endpoint --endpoint-type iot:CredentialProvider`.

     > **Note**
     >
     > Si utiliza dominios personalizados, consulte Cómo usar dominios personalizados con puntos de conexión de VPC. El proveedor de AWS IoT Core credenciales no admite dominios personalizados.

   - En la lista Tipo, elija Zona alojada privada.
   - Si lo desea, puede agregar o eliminar etiquetas para asociarlas a su zona alojada.
3. Para crear su zona alojada privada, seleccione Crear zona alojada.

Para obtener información, consulte Crear una zona alojada privada.

Crear un registro

Una vez que haya creado una zona alojada privada, puede crear un registro que indique al DNS cómo desea que se dirija el tráfico a ese dominio.

Para crear un registro

1. En la lista de zonas alojadas que se muestra, elija la zona alojada privada que ha creado anteriormente y elija Crear registro.
2. Utilice el método del asistente para crear el registro. Si la consola presenta el método Creación rápida, elija Cambiar al asistente.
3. En Enrutamiento sencillo, elija Política de enrutamiento y, a continuación, elija Siguiente.
4. En Configurar registros, elija Definir un registro simple.

5. En la página Definir un registro simple:
   - En Nombre del registro, introduzca el punto de conexión `iot:Data-ATS` o el punto de conexión `iot:CredentialProvider`. Debe ser el mismo que el nombre de la zona alojada privada.
   - Para el tipo de registro, si solo quieres IPv4 soporte, mantén el valor como`A - Routes traffic to an IPv4 address and some AWS resources`. Si solo quieres IPv6 soporte, mantén el valor como`AAAA - Routes traffic to an IPv6 address and some AWS resources`. Si desea admitir la doble pila (ambos IPv6), cree dos registros (IPv4 A y uno `AAAA` en la zona alojada) con el mismo nombre de registro y el mismo valor/direccione el tráfico.
   - En Valor/Dirigir tráfico a, elija Alias del punto de conexión de VPC. A continuación, elija su Región y elija el punto de conexión que creó anteriormente, tal y como se describe en ??? de la lista de puntos de conexión que se muestra.
6. Elija Definir un registro simple para crear su registro.

Control del acceso a AWS IoT Core más de puntos finales de VPC

Puede restringir el acceso a los dispositivos para que solo se permita AWS IoT Core a través del punto de conexión de la VPC mediante claves de contexto de condición de la VPC. AWS IoT Core admite las siguientes claves de contexto relacionadas con la VPC:

- SourceVpc
- SourceVpce
- VPCSourceIP

> **Note**
>
> AWS IoT Core no admite políticas de puntos de conexión para puntos de conexión de VPC.

Por ejemplo, la siguiente política concede permiso para conectarse AWS IoT Core mediante un ID de cliente que coincida con el nombre de la cosa y para publicar en cualquier tema con el prefijo del nombre de la cosa, siempre que el dispositivo se conecte a un punto final de VPC con un ID de punto final de VPC concreto. Esta política denegaría los intentos de conexión a su punto de conexión de datos de IoT público.

```
{
```

```
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:SourceVpce": "vpce-1a2b3c4d"
            }
        }

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

Limitaciones

Actualmente, los puntos finales de VPC son compatibles con los puntos finales del plano de AWS IoT Core control, los puntos finales de AWS IoT Core datos y los puntosfinales de los proveedores de credenciales.AWS IoT Core Los puntos finales de VPC solo son compatibles con los puntos finales del Estándar Federal de Procesamiento de Información (FIPS) cuando se utiliza el plano de control. AWS IoT Core

Limitaciones de los puntos finales de VPC del plano de control de IoT

En esta sección se describen las limitaciones de los puntos finales de VPC del plano de control de IoT.

- Los puntos de conexión de VPC solo servirán certificados ATS.
- Los dominios personalizados no son compatibles con los puntos finales del plano de control.
- Para obtener información sobre las políticas de seguridad de FIPS, consulte las políticas de seguridad de FIPS.

Limitaciones de los puntos de conexión de VPC de datos de IoT

En esta sección se describen las limitaciones de los puntos de conexión de VPC de datos de IoT.

- Los períodos keep-alive de MQTT están limitados a 230 segundos. Los períodos keep-alive que superen esa cantidad se reducirán automáticamente a 230 segundos.
- Cada punto de conexión de VPC admite un total de 100 000 dispositivos conectados de forma simultánea. Si necesita más conexiones, consulte. Escalar los puntos finales de VPC con AWS IoT Core
- Los puntos de conexión de VPC solo entregarán certificados ATS, excepto para los dominios personalizados.
- Las políticas de punto de conexión de VPC no son compatibles.
- En el caso de los puntos finales de VPC que se crean para el plano de AWS IoT Core datos, AWS IoT Core no admite el uso de registros DNS públicos zonales o regionales.

Limitaciones de los puntos de conexión de los proveedores de credenciales

En esta sección se describen las limitaciones de los puntos de conexión de VPC de los proveedores de credenciales.

- Los puntos de conexión de VPC solo servirán certificados ATS.
- Las políticas de punto de conexión de VPC no son compatibles.
- Los dominios personalizados no son compatibles con los puntos de conexión de los proveedores de credenciales.
- En el caso de los puntos finales de VPC que se crean para el proveedor de AWS IoT Core credenciales, AWS IoT Core no admite el uso de registros DNS públicos zonales o regionales.

Escalar los puntos finales de VPC con AWS IoT Core

AWS IoT Core Los puntos finales de VPC de interfaz están limitados a 100 000 dispositivos conectados en un único punto final de interfaz. Si su caso de uso requiere más conexiones simultáneas con el agente, le recomendamos usar varios puntos de conexión de VPC y enrutar manualmente los dispositivos a través de los puntos de conexión de interfaz. Al crear registros de DNS privados para enrutar el tráfico a sus puntos de conexión de VPC, asegúrese de crear tantos registros ponderados como puntos de conexión de VPC tenga para distribuir el tráfico entre sus múltiples puntos de conexión.

Uso de dominios personalizados con puntos de conexión de VPC

Si desea utilizar dominios personalizados con puntos de conexión de VPC, debe crear sus registros de nombres de dominio personalizados en una zona alojada privada y crear registros de enrutamiento en Route53. Para obtener información, consulte Crear una zona alojada privada.

> **Note**
>
> Los dominios personalizados solo se admiten para los puntos finales de AWS IoT Core datos.

Disponibilidad de puntos finales de VPC para AWS IoT Core

AWS IoT Core Los puntos finales de VPC de interfaz están disponibles en todas las AWS IoT Core regiones compatibles. AWS IoT Core Los puntos finales de VPC de interfaz para el proveedor de AWS IoT Core credenciales no son compatibles en la región de China y. AWS GovCloud (US) Regions

Uso de túneles AWS IoT Device Management seguros con puntos finales de VPC de interfaz

AWS IoT Device Management la tunelización segura admite los puntos finales de la interfaz de la VPC. Puede usar los puntos de enlace de la VPC para mantener el tráfico entre su VPC y AWS IoT Secure Tunneling dentro de la AWS red, sin necesidad de una puerta de enlace a Internet, un dispositivo NAT, una conexión VPN o una conexión. AWS Direct Connect

Los puntos finales de VPC de interfaz cuentan con una tecnología que le permite acceder de forma privada a los servicios mediante direcciones IP privadas. AWS PrivateLink Para obtener más

información, consulte Acceder a un AWS servicio mediante un punto final de VPC de interfaz en la AWS PrivateLink Guía.

Contenido

- Requisitos previos
- Recibir notificaciones de túnel a través de puntos finales de VPC
- Creación de puntos finales de VPC para una tunelización segura
- Configuración de políticas de puntos finales de VPC en el servidor proxy
- Siguientes pasos

Requisitos previos

Antes de crear puntos de enlace de VPC para AWS IoT Secure Tunneling, compruebe que dispone de lo siguiente:

- Una AWS cuenta con los permisos necesarios para crear puntos de enlace de VPC.
- Una VPC en tu cuenta. AWS
- Comprensión de los conceptos de tunelización AWS IoT Device Management segura.
- Familiaridad con las políticas de puntos finales de VPC AWS Identity and Access Management y (IAM)

Recibir notificaciones de túnel a través de puntos finales de VPC

Para recibir notificaciones de túnel a través de un punto de enlace de VPC, sus dispositivos pueden conectarse al plano de AWS IoT Core datos a través de un punto de enlace de VPC y suscribirse al tema MQTT reservado sobre tunelización segura.

Para obtener instrucciones sobre cómo crear y configurar un punto final de VPC en el plano de AWS IoT Core datos, consulte Uso AWS IoT Core con puntos de enlace de VPC de interfaz en la Guía para desarrolladores. AWS IoT

Creación de puntos finales de VPC para una tunelización segura

Puede crear puntos finales de VPC tanto para el plano de control de tunelización segura como para el servidor proxy.

Para crear un punto final de VPC para una tunelización segura

1. Siga los pasos que se indican en Creación de un punto final de interfaz en la Guía para desarrolladores de Amazon VPC
2. En el campo Nombre del servicio, elija una de las siguientes opciones en función del tipo de punto final:

   Plano de control

   - Estándar: `com.amazonaws.<region>.iot.tunneling.api`
   - FIPS (disponible en las regiones FIPS): `com.amazonaws.<region>.iot-fips.tunneling.api`

   Servidor proxy

   - Estándar: `com.amazonaws.<region>.iot.tunneling.data`
   - FIPS (disponible en las regiones FIPS): `com.amazonaws.<region>.iot-fips.tunneling.data`

   Sustitúyalo por *`<region>`* su. Región de AWS Por ejemplo, `us-east-1`.
3. Complete los pasos restantes del proceso de creación del punto final de la VPC de acuerdo con los requisitos de su red.

Configuración de políticas de puntos finales de VPC en el servidor proxy

Además de la autorización basada en el token de acceso del cliente que se utiliza para autorizar las conexiones a los túneles, puede utilizar las políticas de punto final de la VPC para restringir aún más la forma en que los dispositivos pueden utilizar un punto final de la VPC para conectarse al servidor proxy de túnel seguro. Las políticas de puntos de enlace de la VPC siguen una sintaxis similar a la de la IAM y se configuran en el propio punto de enlace de la VPC.

Tenga en cuenta que la única acción de IAM admitida para las políticas de punto final de VPC del servidor proxy es. `iot:ConnectToTunnel`

A continuación, se muestran ejemplos de diferentes políticas de puntos finales de VPC.

Ejemplos de políticas de puntos finales de VPC para servidores proxy

Los siguientes ejemplos muestran las configuraciones de políticas de puntos finales de VPC del servidor proxy para casos de uso comunes.

Example- Política predeterminada

Esta política permite que los dispositivos de su VPC se conecten a cualquier túnel en el mismo Región de AWS lugar en el que se creó el punto final, en cualquier AWS cuenta.

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

Example- Restrinja el acceso a cuentas específicas AWS

Esta política permite que el punto final de la VPC se conecte solo a los túneles de cuentas específicas AWS .

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "iot:ConnectToTunnel",
            "Resource": [
                "arn:aws:iot:us-east-1:111122223333:tunnel/*",
                "arn:aws:iot:us-east-1:444455556666:tunnel/*"
            ]
        }
    ]
}
```

Example- Restrinja las conexiones por punto final del túnel

Puede restringir el acceso al punto final de la VPC para permitir que los dispositivos solo se conecten al extremo de origen o destino de un túnel.

Solo fuente:

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "iot:ConnectToTunnel",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iot:ClientMode": "source"
                }
            }
        }
    ]
}
```

Solo destino:

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "iot:ConnectToTunnel",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iot:ClientMode": "destination"
                }
            }
        }
    ]
}
```

Example- Restrinja el acceso en función de las etiquetas de los recursos

Esta política permite que el punto final de la VPC se conecte solo a los túneles etiquetados con un par clave-valor específico.

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "iot:ConnectToTunnel",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/Environment": "Production"
                }
            }
        }
    ]
}
```

Example- Condiciones de política combinadas

Esta política demuestra la combinación de varios elementos de política. Permite las conexiones a cualquier túnel de una AWS cuenta específica, pero solo si el túnel tiene la etiqueta `AllowConnectionsThroughPrivateLink` set en `true` y el cliente no se conecta al extremo de destino del túnel.

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "iot:ConnectToTunnel",
            "Resource": [
                "arn:aws:iot:us-east-1:111122223333:tunnel/*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/AllowConnectionsThroughPrivateLink": "true"
                }
            }
```

```
        },
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "iot:ConnectToTunnel",
            "Resource": [
                "arn:aws:iot:us-east-1:111122223333:tunnel/*"
            ],
            "Condition": {
                "StringEquals": {
                    "iot:ClientMode": "destination"
                }
            }
        }
    ]
}
```

Siguientes pasos

Después de crear y configurar los puntos de enlace de la VPC AWS IoT Secure Tunneling, tenga en cuenta lo siguiente:

- Pruebe la configuración del punto final de la VPC conectando los dispositivos a través del punto final.
- Supervise el uso de los puntos finales de la VPC mediante métricas Amazon CloudWatch .
- Revise y actualice las políticas de puntos finales de la VPC según sea necesario según sus requisitos de seguridad.

Para obtener más información sobre la tunelización AWS IoT Device Management segura, consulte. AWS IoT Secure Tunneling

Seguridad de la infraestructura en AWS IoT

Como conjunto de servicios gestionados, AWS IoT está protegido por los procedimientos de seguridad de la red AWS global que se describen en el documento técnico Amazon Web Services: Overview of Security Processes.

Utiliza las llamadas a la API AWS publicadas para acceder a AWS IoT través de la red. Los clientes deben ser compatibles con Transport Layer Security (TLS) 1.2 o una versión posterior. Los clientes

también deben ser compatibles con conjuntos de cifrado con confidencialidad directa total (PFS) tales como Ephemeral Diffie-Hellman (DHE) o Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). La mayoría de los sistemas modernos, como Java 7 y posteriores, son compatibles con estos modos. Para obtener más información, consulte Seguridad del transporte en AWS IoT Core.

Las solicitudes deben estar firmadas mediante un ID de clave de acceso y una clave de acceso secreta que esté asociada a una entidad principal de IAM. También puede utilizar AWS Security Token Service (AWS STS) para generar credenciales de seguridad temporales para firmar solicitudes.

Supervisión de la seguridad de las flotas o dispositivos de producción con Core AWS IoT

Las flotas de IoT pueden constar de un gran número de dispositivos que tienen diversas funcionalidades, son de larga duración y están distribuidos geográficamente. Estas características hacen que la configuración de la flota sea compleja y propensa a errores. Y dado que los dispositivos a menudo están limitados en potencia informática, memoria y capacidades de almacenamiento, esto limita el uso del cifrado y otras formas de seguridad en los propios dispositivos. Además, los dispositivos a menudo usan software con vulnerabilidades conocidas. Estos factores hacen que las flotas de IoT sean un objetivo atractivo para los piratas informáticos y dificultan la protección continuada de la flota de dispositivos.

AWS IoT Device Defender aborda estos desafíos proporcionando herramientas para identificar los problemas de seguridad y las desviaciones de las mejores prácticas. Puede usar AWS IoT Device Defender para analizar, auditar y monitorear los dispositivos conectados a fin de detectar comportamientos anormales y mitigar los riesgos de seguridad. AWS IoT Device Defender puede auditar las flotas de dispositivos para garantizar que cumplen las mejores prácticas de seguridad y detectar comportamientos anormales en los dispositivos. Esto permite aplicar políticas de seguridad coherentes en toda su flota de AWS IoT dispositivos y responder rápidamente cuando los dispositivos se ven comprometidos. Para obtener más información, consulte AWS IoT Device Defender.

AWS IoT Device Advisor actualiza y corrige su flota según sea necesario. AWS IoT Device Advisor actualiza los casos de prueba automáticamente. Los casos de prueba que seleccione siempre tienen la última versión. Para obtener más información, consulte Asesor de dispositivos.

Mejores prácticas de seguridad en AWS IoT Core

Esta sección contiene información sobre las mejores prácticas de seguridad para AWS IoT Core. Para obtener información sobre las reglas de seguridad de las soluciones de IoT industrial, consulte Ten security golden rules for Industrial IoT solutions (en inglés).

Protección de las conexiones MQTT en AWS IoT

AWS IoT Corees un servicio en la nube gestionado que permite que los dispositivos conectados interactúen con las aplicaciones en la nube y otros dispositivos de forma fácil y segura. AWS IoT Core es compatible con HTTP y MQTT, un protocolo de comunicación ligero diseñado específicamente para tolerar conexiones intermitentes. WebSocket Si se conecta AWS IoT mediante MQTT, cada una de sus conexiones debe estar asociada a un identificador conocido como ID de cliente. El cliente MQTT identifica de IDs forma exclusiva las conexiones MQTT. Si se establece una nueva conexión con un ID de cliente que ya se ha utilizado para otra conexión, el agente de AWS IoT mensajes descarta la conexión anterior para permitir la nueva conexión. El cliente IDs debe ser único en cada uno Cuenta de AWS de ellos Región de AWS. Esto significa que no necesitas imponer la exclusividad global del cliente IDs fuera de tu país Cuenta de AWS o entre las distintas regiones de tu Cuenta de AWS región.

El impacto y la gravedad de la interrupción de las conexiones MQTT en su flota de dispositivos depende de muchos factores. Entre ellos se incluyen:

- Tu caso de uso (por ejemplo, los datos a los que envían tus dispositivos AWS IoT, la cantidad de datos y la frecuencia con la que se envían).
- Su configuración de cliente MQTT (por ejemplo, configuración de reconexión automática, temporizaciones de interrupción asociadas y uso de sesiones persistentes de MQTT).
- Las restricciones de recursos de dispositivos.
- La causa principal de las desconexiones, su agresividad y persistencia.

Para evitar conflictos de ID de cliente y sus posibles repercusiones negativas, asegúrese de que cada dispositivo o aplicación móvil tenga una AWS IoT política de IAM que restrinja qué cliente se IDs puede utilizar para las conexiones MQTT con el AWS IoT intermediario de mensajes. Por ejemplo, puede usar una política de IAM para evitar que un dispositivo cierre por error la conexión de otro dispositivo utilizando un ID de cliente que ya está en uso. Para obtener más información, consulte Autorización.

Todos los dispositivos de su flota deben tener credenciales con privilegios que autoricen únicamente las acciones previstas, que incluyen (pero no se limitan a) las acciones de AWS IoT MQTT, como publicar mensajes o suscribirse a temas con un alcance y un contexto específicos. Las políticas de permisos específicas pueden variar en función de los casos de uso. Identifique las políticas de permisos que se ajusten mejor a sus requisitos empresariales y de seguridad.

Para simplificar la creación y la administración de políticas de permisos, puede utilizar AWS IoT Core variables de política y variables de la política de IAM. Las variables de la política se pueden colocar en una política y cuando se evalúa la política las variables se sustituyen por valores procedentes de la solicitud del dispositivo. Al usar variables de la política, puede crear una única política para la concesión de permisos a varios dispositivos. Puede identificar las variables de política relevantes para su caso de uso en función de la configuración de su AWS IoT cuenta, el mecanismo de autenticación y el protocolo de red utilizado para conectarse al agente de AWS IoT mensajes. Sin embargo, para escribir las mejores políticas de permisos, deben tenerse en cuenta los detalles concretos de su caso de uso y el modelo de amenazas.

Por ejemplo, si ha registrado sus dispositivos en el AWS IoT registro, puede utilizar variables de política de cosas en las AWS IoT políticas para conceder o denegar permisos en función de las propiedades de las cosas, como los nombres, los tipos y los valores de los atributos de las cosas. El nombre de la cosa se obtiene del ID de cliente del mensaje de conexión MQTT que se envía cuando una cosa se conecta a AWS IoT ella. Lo que las variables de política se sustituyen cuando una cosa se conecta a AWS IoT través de MQTT mediante la autenticación mutua TLS o MQTT a través del WebSocket protocolo mediante identidades autenticadas de Amazon Cognito. Puede usar la AttachThingPrincipalAPI para adjuntar certificados e identidades autenticadas de Amazon Cognito a un objeto. `iot:Connection.Thing.ThingName`es una variable de política útil para hacer cumplir las restricciones de identificación de los clientes. El siguiente ejemplo de AWS IoT política exige que se utilice el nombre de un elemento registrado como ID de cliente para las conexiones MQTT con el intermediario de AWS IoT mensajes:

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

```
  }
 ]
}
```

Si desea identificar los conflictos de ID de cliente en curso, puede habilitar y usar CloudWatch Logs for AWS IoT. Por cada conexión MQTT que el agente de AWS IoT mensajes desconecte debido a conflictos de ID de cliente, se genera un registro similar al siguiente:

```
{
    "timestamp": "2019-04-28 22:05:30.105",
    "logLevel": "ERROR",
    "traceId": "02a04a93-0b3a-b608-a27c-1ae8ebdb032a",
    "accountId": "123456789012",
    "status": "Failure",
    "eventType": "Disconnect",
    "protocol": "MQTT",
    "clientId": "clientId01",
    "principalId": "1670fcf6de55adc1930169142405c4a2493d9eb5487127cd0091ca0193a3d3f6",
    "sourceIp": "203.0.113.1",
    "sourcePort": 21335,
    "reason": "DUPLICATE_CLIENT_ID",
    "details": "A new connection was established with the same client ID"
}
```

Puede utilizar un filtro de CloudWatch registros, por ejemplo, `{$.reason= "DUPLICATE_CLIENT_ID" }` para buscar casos de conflictos de ID de cliente o para configurar los filtros de CloudWatch métricas y CloudWatch las alarmas correspondientes para una supervisión y generación de informes continuas.

Puede usar AWS IoT Device Defender para identificar políticas excesivamente permisivas AWS IoT y de IAM. AWS IoT Device Defender también proporciona una verificación de auditoría que le notifica si varios dispositivos de su flota se están conectando al agente de AWS IoT mensajes con el mismo ID de cliente.

Puede utilizar AWS IoT Device Advisor para comprobar que sus dispositivos se pueden conectar de forma fiable AWS IoT Core y seguir las prácticas recomendadas de seguridad.

Véase también

- AWS IoT Core

- Características de seguridad de AWS IoT
- AWS IoT Core variables de política
- Variables de la política de IAM
- Identidad de Amazon Cognito
- AWS IoT Device Defender
- CloudWatch Registra para AWS IoT

Mantener sincronizado el reloj del dispositivo

Es importante que la hora del dispositivo sea precisa. Los certificados X.509 tienen una fecha y una hora de caducidad. El reloj del dispositivo se utiliza para comprobar que un certificado de servidor sigue siendo válido. Si está creando dispositivos IoT comerciales, recuerde que sus productos pueden permanecer en el almacén durante largos períodos de tiempo antes de venderse. Los relojes en tiempo real pueden retrasarse durante este tiempo y las baterías pueden descargarse, por lo que no es suficiente fijar el tiempo en fábrica.

En la mayoría de los sistemas, esto significa que el software del dispositivo debe incluir un cliente NTP (Protocolo de tiempo de red). El dispositivo debe esperar hasta que se sincronice con un servidor NTP antes de intentar conectarse a AWS IoT Core. Si esto no es posible, el sistema debería proporcionar un mecanismo para que el usuario establezca la hora del dispositivo, de forma que las conexiones posteriores se realicen correctamente.

Una vez que el dispositivo se haya sincronizado con un servidor NTP, podrá abrir una conexión con AWS IoT Core. La cantidad de sesgo de reloj permitida dependerá de lo que esté tratando de hacer con la conexión.

Validar el certificado de servidor

Lo primero que hace un dispositivo para interactuar AWS IoT es abrir una conexión segura. Cuando conectes tu dispositivo a AWS IoT, asegúrate de que estás hablando con otro servidor AWS IoT y no haciéndote pasar por AWS IoT otro. Cada uno de los AWS IoT servidores está provisto de un certificado emitido para el dominio. `iot.amazonaws.com` Este certificado lo emitió una autoridad AWS IoT de certificación de confianza que verificó nuestra identidad y propiedad del dominio.

Una de las primeras cosas que se AWS IoT Core hace cuando un dispositivo se conecta es enviarle un certificado de servidor. Los dispositivos pueden comprobar que estaban esperando para

conectarse a `iot.amazonaws.com` y que el servidor al final de dicha conexión posee un certificado de una autoridad de confianza para ese dominio.

Los certificados TLS tienen formato X.509 e incluyen información diversa, como el nombre de la organización, la ubicación, el nombre de dominio y un período de validez. El período de validez se especifica como un par de valores de tiempo llamados `notBefore` y `notAfter`. Servicios como estos AWS IoT Core utilizan períodos de validez limitados (por ejemplo, un año) para sus certificados de servidor y comienzan a entregar los nuevos antes de que caduquen los antiguos.

Usar una identidad única por dispositivo

Utilice una identidad única por cliente. Los dispositivos generalmente usan certificados de cliente X.509. Las aplicaciones web y para móviles utilizan Amazon Cognito Identity. Esto le permite aplicar permisos detallados a sus dispositivos.

Por ejemplo, puede tener una aplicación que consiste en un dispositivo de teléfono móvil que recibe actualizaciones de estado de dos objetos de hogar inteligente diferentes: una bombilla y un termostato. La bombilla envía el estado de su nivel de batería y el termostato envía mensajes que informan de la temperatura.

AWS IoT autentica los dispositivos de forma individual y trata cada conexión de forma individual. Puede aplicar controles de acceso detallados a través de políticas de autorización. Puede definir una política para el termostato que le permita publicar en un espacio de tema. Puede definir una política independiente para la bombilla que le permita publicar en un espacio de tema diferente. Por último, puede definir una política para la aplicación móvil que solo le permita conectarse y suscribirse a los temas del termostato y la bombilla para recibir mensajes de estos dispositivos.

Aplique el principio de privilegios mínimos y amplíe los permisos por dispositivo tanto como sea posible. Todos los dispositivos o usuarios deben tener una AWS IoT política AWS IoT que solo les permita conectarse con un ID de cliente conocido y publicar y suscribirse a un conjunto de temas identificado y fijo.

Utilice un segundo Región de AWS como respaldo

Considere almacenar una copia de sus datos en un segundo Región de AWS como copia de seguridad. Tenga en cuenta que la AWS solución denominada Disaster Recovery for ya no AWS IoT está disponible. Si bien la GitHubbiblioteca asociada sigue siendo accesible, AWS quedó obsoleta en julio de 2023 y ya no proporciona mantenimiento ni soporte. Para implementar sus propias soluciones o explorar opciones de soporte adicionales, visite Contáctese con AWS. Si hay un

administrador AWS técnico de cuentas asociado a tu cuenta, ponte en contacto con él para obtener ayuda.

Usar aprovisionamiento justo a tiempo

Crear y aprovisionar manualmente cada dispositivo puede llevar mucho tiempo. AWS IoT proporciona una forma de definir una plantilla para aprovisionar los dispositivos cuando se conectan por primera vez. AWS IoT Para obtener más información, consulte Just-in-time aprovisionamiento.

Permisos para ejecutar pruebas de AWS IoT Device Advisor

La siguiente plantilla de políticas muestra los permisos mínimos y la entidad de IAM necesarios para ejecutar los casos de prueba de AWS IoT Device Advisor. Deberá sustituirlo por *your-device-role-arn* el rol de dispositivo Amazon Resource Name (ARN) que creó según los requisitos previos.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iot:us-east-1:123456789012:thinggroup/your-thing-group",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "iotdeviceadvisor.amazonaws.com"
            }
        }
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": [
                "execute-api:Invoke*",
                "iam:ListRoles",
                "iot:Connect",
                "iot:CreateJob",
                "iot:DeleteJob",
                "iot:DescribeCertificate",
```

```
                "iot:DescribeEndpoint",
                "iotjobsdata:DescribeJobExecution",
                "iot:DescribeJob",
                "iot:DescribeThing",
                "iotjobsdata:GetPendingJobExecutions",
                "iot:GetPolicy",
                "iot:ListAttachedPolicies",
                "iot:ListCertificates",
                "iot:ListPrincipalPolicies",
                "iot:ListThingPrincipals",
                "iot:ListThings",
                "iot:Publish",
                "iotjobsdata:StartNextPendingJobExecution",
                "iotjobsdata:UpdateJobExecution",
                "iot:UpdateThingShadow",
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:DescribeLogGroups",
                "logs:DescribeLogStreams",
                "logs:PutLogEvents",
                "logs:PutRetentionPolicy"
            ],
            "Resource": "*"
        },
        {
            "Sid": "VisualEditor2",
            "Effect": "Allow",
            "Action": "iotdeviceadvisor:*",
            "Resource": "*"
        }
    ]
}
```

Prevención de la sustitución confusa entre servicios para Device Advisor

El problema de la sustitución confusa es un problema de seguridad en el que una entidad que no tiene permiso para realizar una acción puede obligar a una entidad con más privilegios a realizar la acción. En AWS, la suplantación de identidad entre servicios puede provocar el confuso problema de un diputado. La suplantación entre servicios puede producirse cuando un servicio (el servicio que lleva a cabo las llamadas) llama a otro servicio (el servicio al que se llama). El servicio que lleva a cabo las llamadas se puedes manipular para utilizar sus permisos a fin de actuar en función de los

recursos de otro cliente de una manera en la que no debe tener permiso para acceder. Para evitarlo, AWS proporciona herramientas que le ayudan a proteger los datos de todos los servicios cuyos directores de servicio tengan acceso a los recursos de su cuenta.

Recomendamos utilizar las claves contextuales de condición global `aws:SourceArn` y `aws:SourceAccount` en las políticas de recursos para limitar los permisos que Device Advisor concede a otro servicio sobre el recurso. Si se utilizan ambas claves contextuales de condición global, el valor `aws:SourceAccount` y la cuenta del valor `aws:SourceArn` deben utilizar el mismo ID de cuenta cuando se utilicen en la misma declaración de política.

El valor de `aws:SourceArn` debe ser el ARN de su recurso de definición de conjuntos. El recurso de definición de conjuntos hace referencia al conjunto de pruebas que creó con Device Advisor.

La forma más eficaz de protegerse contra el problema de la sustitución confusa es utilizar la clave de contexto de condición global de `aws:SourceArn` con el ARN completo del recurso. Si no conoce el ARN completo del recurso o si especifica varios recursos, utilice la clave de condición de contexto global `aws:SourceArn` con comodines (*) para las partes desconocidas del ARN. Por ejemplo, `arn:aws:iotdeviceadvisor:*:`*`account-id`*`:suitedefinition/*`

El siguiente ejemplo muestra cómo puede utilizar las claves contextuales de condición global `aws:SourceArn` y `aws:SourceAccount` en Device Advisor para evitar el problema de la sustitución confusa.

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Sid": "ConfusedDeputyPreventionExamplePolicy",
        "Effect": "Allow",
        "Principal": {
            "Service": "iotdeviceadvisor.amazonaws.com"
        },
        "Action": "sts:AssumeRole",
        "Condition": {
            "ArnLike": {
                "aws:SourceArn": "arn:aws:iotdeviceadvisor:us-
east-1:123456789012:suitedefinition/ygp6rxa3tzvn"
        },
            "StringEquals": {
                "aws:SourceAccount": "123456789012"
        }
```

```
        }
    }
}
```

AWS formación y certificación

Realice el siguiente curso para aprender los conceptos clave de AWS IoT la seguridad: AWS IoT Security Primer.

Monitorización AWS IoT

La supervisión es una parte importante del mantenimiento de la confiabilidad, la disponibilidad y el rendimiento de AWS IoT sus AWS soluciones.

Le recomendamos encarecidamente que recopile datos de supervisión de todas las partes de su AWS solución para facilitar la depuración de un fallo multipunto, en caso de que se produzca. Comience por crear un plan de monitoreo que responda a las siguientes preguntas. Si no está seguro de cómo responderlas, puede continuar habilitando el registro y estableciendo las líneas base de rendimiento.

- ¿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 supervisión?
- ¿Quién debería recibir una notificación cuando surjan problemas?

El siguiente paso es habilitar el registro y establecer una línea base del AWS IoT rendimiento normal en su entorno midiendo el rendimiento en distintos momentos y bajo diferentes condiciones de carga. Mientras monitorea AWS IoT, guarde los datos de monitoreo históricos para poder compararlos con los datos de rendimiento actuales. Esto le ayudará a identificar patrones de rendimiento normales y anomalías de rendimiento, así como a idear métodos para abordarlos.

Para establecer su rendimiento de referencia AWS IoT, debe supervisar estas métricas desde el principio. Siempre puede monitorear más métricas más adelante.

- `PublishIn.Success`
- `PublishOut.Success`
- `Subscribe.Success`
- `Ping.Success`
- `Connect.Success`
- `GetThingShadow.Accepted`
- `UpdateThingShadow.Accepted`

- `DeleteThingShadow.Accepted`
- `RulesExecuted`

Los temas de esta sección pueden ayudarlo a iniciar el registro y el monitoreo de AWS IoT.

Temas

- Configure el AWS IoT registro
- Supervisa AWS IoT las alarmas y las métricas con Amazon CloudWatch
- Supervise AWS IoT mediante registros CloudWatch
- Sube registros del lado del dispositivo a Amazon CloudWatch
- Registrar las llamadas a la AWS IoT API mediante AWS CloudTrail

Configure el AWS IoT registro

Debe habilitar el registro mediante la AWS IoT consola, la CLI o la API antes de poder supervisar y registrar AWS IoT la actividad. Puede configurar el registro AWS IoT en tres niveles: nivel de cuenta, nivel de evento o nivel específico de recursos. El registro a nivel de eventos y de recursos específicos está disponible exclusivamente con el registro V2. Los clientes que utilizan el registro de la versión 1 deben realizar una migración a la versión 2 para acceder a estas funciones. Consulte los detalles.

Al considerar cómo configurar el AWS IoT registro, la configuración del registro a nivel de cuenta determina cómo se registrará la AWS IoT actividad, a menos que se especifique lo contrario. Para empezar, es posible que desee obtener registros detallados con un nivel de registro predeterminado de INFO o DEBUG. Tras revisar los registros iniciales, puede cambiar el nivel de registro predeterminado a un nivel menos detallado, como WARN o ERROR, a nivel de cuenta o evento, y establecer un nivel de registro más detallado y específico para los recursos que puedan necesitar más atención. Los niveles de registro se pueden cambiar cuando lo desee.

En este tema se describe el inicio de sesión en la nube. AWS IoT Para obtener información sobre el registro y la supervisión del lado del dispositivo, consulte Cargar registros del lado del dispositivo a. CloudWatch

Para obtener información sobre el registro y la supervisión AWS IoT Greengrass, consulte Registro y supervisión. AWS IoT Greengrass

Configurar el inicio de sesión en V2 AWS IoT

Determinar la versión de registro

La LoggingOptions API GetV2 devuelve un valor NotConfiguredException si el registro de V2 no está habilitado. Este error se produce cuando se utiliza el registro de la V1 o no se ha configurado ningún registro.

Comprender las funciones de registro de la versión 2

El registro V2 ofrece dos capacidades clave: el registro a nivel de eventos y el registro de recursos específicos. El registro a nivel de eventos permite una configuración de registro específica con niveles de registro y destinos de grupos de registros personalizables. CloudWatch El registro de recursos específicos permite filtrar los registros por grupo de cosas, IP de origen, ID de cliente o ID principal. En conjunto, estas características proporcionan un control granular y una visibilidad completa de las operaciones de IoT, lo que mejora la capacidad de búsqueda de registros y reduce los costos al eliminar la actividad de registro innecesaria.

Migración de V1 a V2

Puede migrar al registro de V2 mediante la LoggingOptions API de SetV2 mediante la AWS CLI o el SDK. Tras la migración, enruta AWS IoT automáticamente todos los registros al grupo de CloudWatch registros «AWSIotLogsV2». Importante: Si sus aplicaciones o recursos posteriores consumen información de los «AWSIotregistros», actualícelos para que utilicen la ruta del grupo de registros correspondiente.

Configuración del rol y la política de registro

Antes de habilitar el inicio de sesión AWS IoT, debe crear un rol de IAM y una política que permita escribir actividades AWS IoT de AWS IoT registro en grupos de registros en su CloudWatch nombre. También puedes generar un rol de IAM con las políticas necesarias en la sección Registros de la AWS IoT consola.

> **Note**
>
> Antes de habilitar el AWS IoT registro, asegúrese de entender los permisos de acceso a los CloudWatch registros. Los usuarios con acceso a CloudWatch los registros pueden ver la información de depuración de sus dispositivos. Para obtener más información, consulte Autenticación y control de acceso para Amazon CloudWatch Logs.

Si espera que se produzcan patrones de tráfico elevados AWS IoT Core debido a las pruebas de carga, considere la posibilidad de deshabilitar el registro de IoT para evitar la limitación. Si se detecta mucho tráfico, es posible que nuestro servicio deshabilite el inicio de sesión en su cuenta.

A continuación, se muestra cómo crear una función y una política de registro para los recursos. AWS IoT Core

Creación de un rol de registro

Para crear un rol de registro, abra la página Centro de roles de la consola de IAM y elija Crear rol.

1. En Seleccionar tipo de entidad de confianza, elija Servicio de AWS . A continuación, seleccione IoT en Caso de uso. Si no ve IoT, introduzca y busque IoT en el menú desplegable Casos de uso para otros servicios de AWS :. Seleccione Siguiente.
2. En la página Añadir permisos, verá las políticas que se asocian automáticamente al rol de servicio. Elija Siguiente.
3. En la página Nombrar, revisar y crear, introduzca un Nombre de rol y Descripción de rol para el rol y, a continuación, elija Crear rol.

Política de rol de registro

Los siguientes documentos de política proporcionan la política de roles y la política de confianza que AWS IoT permiten enviar entradas de registro CloudWatch en su nombre. Si configura el registro a nivel de eventos con un grupo de CloudWatch registros personalizado, debe actualizar la política de roles para incluir el ARN del recurso personalizado.

Si también se te permite AWS IoT Core para LoRaWAN enviar entradas de registro, verás un documento de política creado para ti en el que se registran ambas actividades.

Note

Estos documentos se crearon para usted cuando creó el rol de registro. Los documentos tienen variables y `${partition}`, `${region}` `${accountId}` , que debes reemplazar por tus valores.

- Sustituya la partición por la partición de la región.

- Sustituya la AWS región por la región que utilice. Asegúrese de usar la misma AWS región que usó para configurar la AWS CLI en su dispositivo.
- Sustituya account-id por su ID de AWS cuenta.

Política de roles:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:PutMetricFilter",
                "logs:PutRetentionPolicy",
                "iot:GetLoggingOptions",
                "iot:SetLoggingOptions",
                "iot:SetV2LoggingOptions",
                "iot:GetV2LoggingOptions",
                "iot:SetV2LoggingLevel",
                "iot:ListV2LoggingLevels",
                "iot:DeleteV2LoggingLevel"
            ],
            "Resource": [
                "arn:aws:logs:us-east-1:123456789012:log-group:AWSIotLogsV2:*"
            ]
        }
    ]
}
```

Política de confianza para registrar únicamente AWS IoT Core la actividad:

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

```
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": "iot.amazonaws.com"
        },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

AWS IoT Es posible que el registro no pueda publicar el registro en CloudWatch Logs debido a que los permisos de rol de IAM son insuficientes. Cuando esto ocurra, compruebe las métricas de CloudWatch registro para investigar y solucionar los errores.

Configure el registro en la AWS IoT (consola)

En esta sección se describe cómo configurar el AWS IoT registro mediante la AWS IoT consola. Puede configurar el registro a nivel de cuenta, de eventos y de recursos específicos.

AWS IoT Para configurar el registro:

1. Inicie sesión en la AWS IoT consola. Para obtener más información, consulte Abre la AWS IoT consola.
2. En el panel de navegación izquierdo, selecciona Registros (anteriormente, una sección en Configuración).
3. Configura el registro a nivel de cuenta: el registro a nivel de cuenta se aplica a toda tu AWS IoT flota (dispositivos o terminales), a menos que lo anule la configuración a nivel de evento o de recursos específicos.

    a. En Registro a nivel de cuenta, selecciona Administrar el registro a nivel de cuenta para realizar actualizaciones.

    b. Selecciona la casilla de verificación «Habilitar el registro» para empezar a enviar los registros. CloudWatch Si la opción «Activar registro» está desactivada, no AWS IoT se enviará ningún CloudWatch registro a los grupos de registros, independientemente de las configuraciones de registro a nivel de eventos o recursos.

    c. En Función de registro de IAM, selecciona una función existente de la lista desplegable. Puede ver los detalles del rol para inspeccionar los permisos del rol. También puede seleccionar Crear un nuevo rol para configurar un nuevo rol de IAM. La función de registro

proporciona políticas que permiten AWS IoT enviar entradas de registro CloudWatch en su nombre. Si configura el registro a nivel de eventos con un grupo de CloudWatch registros personalizado, debe actualizar la política de roles para incluir el ARN de este grupo de registros.

d. Elija el nivel de registro predeterminado que corresponda al nivel de detalle de las entradas de registro que desea que aparezcan en los registros. CloudWatch Nota: El nivel de registro «DEBUG» es el que proporciona más detalles, pero aumenta CloudWatch los costes. CloudWatch Los destinos de los grupos de registros no se pueden configurar a nivel de cuenta. Sin embargo, puede especificar grupos de registros personalizados para tipos de eventos individuales, como se describe en la siguiente sección.

e. Seleccione Actualizar registro para guardar los cambios.

4. El registro a nivel de eventos le permite capturar de forma selectiva los registros de los eventos relevantes y dirigirlos a grupos de registros específicos CloudWatch . Esto le brinda la flexibilidad de organizar los registros por casos de uso para una mejor visibilidad, compartirlos con diferentes audiencias y reducir los CloudWatch costos al habilitar los registros y establecer los niveles de registro en función de la gravedad de los eventos.

   Configure el registro a nivel de eventos: el registro a nivel de eventos captura AWS IoT eventos específicos, como los intentos de autenticación del cliente. Estos ajustes anulan el registro a nivel de cuenta.

   a. En la sección Registro a nivel de eventos, selecciona Administrar el registro a nivel de eventos para realizar actualizaciones.

   b. De forma predeterminada, los tipos de eventos heredan la configuración de registro a nivel de cuenta. Nota: Cuando se configura el registro específico de un recurso, anula la configuración a nivel de cuenta y evento.

   c. Para modificar la configuración de eventos individuales, haga clic en el valor de la fila de eventos correspondiente. Puede ajustar tanto el nivel de registro como el destino del grupo de CloudWatch registros. Al especificar un destino de grupo de CloudWatch registros personalizado, debe comprobar que la política de roles de IAM incluya los permisos para el nuevo grupo de registros. Si no se actualiza la política de roles, se AWS IoT impedirá escribir registros en el grupo de registros personalizado. Tras realizar la selección, haga clic en la marca de verificación para confirmarla. La columna «Está modificado» mostrará «Sí» para indicar los cambios pendientes.

   d. Haga clic en Actualizar registro para aplicar los cambios o seleccione Cancelar para descartarlos.

5. Configurar anulaciones específicas de un recurso: las anulaciones específicas de un recurso aplican una configuración de registro a los recursos seleccionados. Un recurso puede ser un grupo de cosas, una IP de origen, un ID de cliente o un ID principal. La configuración de registro específica del recurso anula la configuración a nivel de cuenta y a nivel de evento. Cuando está habilitada, genera registros para todos los tipos de eventos en el nivel de registro configurado para los recursos especificados. Por ejemplo, puede configurar el registro a nivel de depuración para una cosa específica y, al mismo tiempo, mantener el registro a nivel de información para todas las demás cosas.

    a. Selecciona Añadir anulaciones específicas de un recurso en la sección Anulaciones específicas de un recurso.

    b. Elija un destino de registro: grupo de cosas, IP de origen, ID de cliente o ID principal.

    c. Introduzca el valor objetivo de registro correspondiente al tipo de objetivo seleccionado.

    d. Seleccione el nivel de registro deseado en el menú desplegable de la sección Nivel de registro específico del recurso.

    e. Haga clic en Enviar para añadir la modificación o en Cancelar para descartar los cambios.

    f. Para modificar una anulación específica de un recurso existente, seleccione la casilla de verificación situada junto al recurso y haga clic en «Eliminar» para eliminar la anulación o en «Editar» para modificarla.

Después de habilitar el registro, visite Visualización AWS IoT de los registros en la CloudWatch consola para obtener más información sobre cómo ver las entradas del registro.

Configurar el inicio de sesión a nivel de cuenta y evento (AWS IoT CLI)

En esta sección se describe cómo configurar el registro global AWS IoT mediante la CLI.

Si lo desea, puede configurar el registro a nivel de eventos. El registro a nivel de eventos captura la información de registro a nivel de evento, como los eventos de autenticación y autorización o de creación de certificados. Puede personalizar tanto el nivel de registro como los destinos de los grupos de CloudWatch registros a nivel de evento. El registro a nivel de eventos funciona a un nivel más específico en comparación con el registro a nivel de cuenta y, por lo tanto, anula la configuración del registro a nivel de cuenta. Este enfoque jerárquico le permite mantener diferentes estrategias de registro para diferentes tipos de eventos en función de su importancia operativa y consideraciones de costo.

> **Note**
>
> Necesita el nombre de recurso de Amazon (ARN) del rol que desea utilizar. Si necesita crear un rol para usar en el registro, consulte Creación de un rol de registro antes de continuar. Al especificar un grupo de CloudWatch registros personalizado para cualquier tipo de evento, asegúrese de que su función de registro tenga los permisos necesarios para el grupo de registros de destino.
> La entidad principal que se utiliza para realizar la llamada a la API debe tener Transmisión de los permisos de rol para el rol de registro.

También puede realizar este procedimiento con la API mediante los métodos de la AWS API que corresponden a los comandos CLI que se muestran aquí.

Para usar la CLI para configurar el registro predeterminado para AWS IoT

1. Utilice el comando set-v2-logging-options para establecer las opciones de registro para su cuenta.

   ```
   aws iot set-v2-logging-options \
       --event-configurations event-configuration-list \
       --role-arn logging-role-arn \
       --default-log-level log-level
   ```

   donde:

   --role-arn

   El ARN del rol que otorga AWS IoT permiso para escribir en tus registros en CloudWatch Logs. Para la configuración inicial, es necesaria la configuración del Role-ARN.

   --default-log-level

   El nivel de registro que se debe usar. Los valores válidos son:`ERROR`,, `WARNINFO`, `DEBUG` o. `DISABLED` Default-log-levella configuración es necesaria para la configuración inicial.

   --no-disable-all-logs

   Un parámetro opcional que permite todos los AWS IoT registros. Utilice este parámetro para habilitar el registro cuando esté deshabilitado actualmente.

--disable-all-logs

Parámetro opcional que desactiva todos los AWS IoT registros. Utilice este parámetro para deshabilitar el registro cuando esté habilitado actualmente.

--event-configurations

Este parámetro es opcional y le permite personalizar la configuración de registro para tipos de eventos individuales:

- EventType: el tipo de evento que anula la configuración a nivel de cuenta
- LogLevel: anula la configuración a nivel de cuenta con DEBUG, INFO, ERROR, WARN o DISABLED
- LogDestination: especifique un grupo de registros personalizado para la entrega de registros CloudWatch

Puede configurar el nivel de registro y el destino del registro de forma independiente para cada tipo de evento. Si no se especifica, los eventos heredarán la configuración a nivel de cuenta

```
aws iot set-v2-logging-options \
    --event-configurations "[{\"eventType\":\"Publish-In\",\"logLevel\":\"INFO\",
\"logDestination\":\"examplePublishInLogGroup\"}]"
```

2. Utilice el comando get-v2-logging-options para obtener las opciones de registro actuales.

```
aws iot get-v2-logging-options \
    --verbose
```

donde:

--verbose

Parámetro opcional que permite recuperar todos los tipos de eventos y sus configuraciones.

Después de habilitar el registro, visite Visualización AWS IoT de los registros en la CloudWatch consola para obtener más información sobre cómo ver las entradas del registro.

> **Note**
>
> AWS IoT sigue admitiendo comandos antiguos (set-logging-optionsyget-logging-options) para configurar y obtener el registro global en su cuenta. Tenga en cuenta que, cuando se utilizan estos comandos, los registros resultantes contienen texto sin formato, en lugar de cargas JSON y que la latencia de registro normalmente es mayor. No se realizarán mejoras en la implementación de estos comandos más antiguos. Le recomendamos que utilice las versiones «v2" para configurar las opciones de registro y, cuando sea posible, que cambie las aplicaciones antiguas que utilizan las versiones anteriores.

Configurar anulaciones específicas de recursos en (CLI) AWS IoT

En esta sección se describe cómo configurar las anulaciones de recursos específicos AWS IoT mediante la CLI. Las anulaciones específicas de un recurso permiten especificar un nivel de registro para un recurso específico identificado por grupo de cosas, ID de cliente, IP de origen o ID principal. Cuando el registro específico de un recurso está habilitado, anula la configuración a nivel de cuenta y a nivel de evento. Todos los tipos de eventos generarán registros para el recurso especificado en el nivel de registro configurado, incluso si esos eventos están deshabilitados en las configuraciones a nivel de evento.

Los grupos de objetos pueden contener otros grupos de objetos para crear una relación jerárquica. Este procedimiento describe cómo configurar el registro de un solo grupo de objetos. Puede aplicar este procedimiento al grupo de objetos principal de una jerarquía para configurar el registro de todos los grupos de objetos de la jerarquía. También puede aplicar este procedimiento a un grupo de objetos secundario para anular la configuración de registro de su principal.

Un objeto puede ser miembro de un grupo de objetos. Esta pertenencia permite al objeto heredar las configuraciones, las políticas y la configuración aplicadas al grupo de objetos. Los grupos de objetos se utilizan para administrar y aplicar la configuración a varios objetos de forma colectiva, en lugar de tratar cada uno de ellos de forma individual. Cuando su ID de cliente coincida con el nombre de la cosa, AWS IoT Core asociará automáticamente la sesión del cliente con el recurso de la cosa correspondiente. Esto permite que la sesión del cliente herede las configuraciones y los ajustes aplicados a los grupos de objetos a los que pertenece el objeto, incluidos los niveles de registro. Si el ID del cliente no coincide con el nombre del objeto, puede habilitar la asociación de objeto exclusiva para establecer dicha asociación. Para obtener más información, consulte ??? .

Además de los grupos de objetos, también puede registrar los destinos, como el ID de cliente, la IP de origen y el ID de entidad principal de un dispositivo.

> **Note**
>
> Necesita el nombre de recurso de Amazon (ARN) del rol que desea utilizar. Si necesita crear un rol para usar en el registro, consulte Creación de un rol de registro antes de continuar. La entidad principal que se utiliza para realizar la llamada a la API debe tener Transmisión de los permisos de rol para el rol de registro.

También puede realizar este procedimiento con la API mediante los métodos de la AWS API que corresponden a los comandos CLI que se muestran aquí.

Para usar la CLI para configurar anulaciones específicas de recursos para AWS IoT

1. Habilite el registro a nivel de cuenta antes de configurar el registro específico de un recurso mediante el siguiente comando: aws iot set-v2-logging-options command
2. Utilice el set-v2-logging-levelcomando para configurar las anulaciones específicas de un recurso. Consulte el siguiente ejemplo para ver la configuración de grupos de cosas:

```
aws iot set-v2-logging-level \
              --log-target targetType=THING_GROUP,targetName=thing_group_name \
              --log-level log_level
```

--log-target

El tipo y nombre del recurso para el que está configurando el registro. El valor `targetType` debe ser uno de los siguientes: `THING_GROUP` | `CLIENT_ID` | `SOURCE_IP` | `PRINCIPAL_ID`. El valor del parámetro log-target puede ser texto, como se muestra en el ejemplo de comando anterior, o una cadena JSON, como en el ejemplo siguiente.

```
aws iot set-v2-logging-level \
              --log-target '{"targetType": "THING_GROUP","targetName":
 "thing_group_name"}' \
              --log-level log_level
```

--log-level

El nivel de registro utilizado cuando se generan registros para el recurso especificado. Los valores válidos son: DEBUG, INFO, ERROR, WARN y DISABLED.

3. Utilice el comando list-v2-logging-levels para enumerar los niveles de registro configurados actualmente.

```
aws iot list-v2-logging-levels
```

4. Utilice el comando delete-v2-logging-level para eliminar un nivel de registro específico de recursos, como en los ejemplos siguientes.

```
aws iot delete-v2-logging-level \
              --target-type "THING_GROUP" \
              --target-name "thing_group_name"
```

```
aws iot delete-v2-logging-level \
              --target-type=CLIENT_ID
              --target-name=ClientId1
```

--target-type

El valor `target-type` debe ser uno de los siguientes: `THING_GROUP` | `CLIENT_ID` | `SOURCE_IP` | `PRINCIPAL_ID`.

--target-name

El nombre del grupo de objetos para el que se va a quitar el nivel de registro.

Niveles de registro

Estos niveles de registro determinan los eventos que se registran y se aplican a los niveles de registro predeterminados y específicos de recursos.

ERROR

Cualquier error que provoque el fracaso de una operación.

Ejemplo: no se pudo autenticar el dispositivo debido a que el certificado expiró.

Los registros solo incluirán información de ERROR.

WARN

Todo lo que pueda llegar a producir incoherencias en el sistema, aunque no el fracaso de la operación.

Ejemplo: aproximarse al límite de velocidad de mensajes.

Los registros incluirán información de ERROR y WARN.

INFO

Información general acerca del flujo de objetos.

Ejemplo: un cliente se ha suscrito correctamente a un tema de MQTT.

Los registros incluirán información de INFO, ERROR y WARN.

DEBUG

Información que puede ser útil para depurar un problema.

Ejemplo: IoT Rules Engine detectó un mensaje publicado en el tema de reglas «rule/test» y comenzó a ejecutarse correctamente. La regla se configura con. RepublishAction

Los registros incluirán información de DEBUG, INFO, ERROR y WARN.

DISABLED

Todos los registros están desactivados.

Supervisa AWS IoT las alarmas y las métricas con Amazon CloudWatch

Puedes monitorizar el AWS IoT uso CloudWatch, que recopila y procesa datos sin procesar para AWS IoT convertirlos en métricas legibles y prácticamente 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 AWS IoT métricas se envían automáticamente CloudWatch en intervalos de un minuto. Para obtener más información, consulta ¿Qué son Amazon CloudWatch,

Amazon CloudWatch Events y Amazon CloudWatch Logs? en la Guía del CloudWatch usuario de Amazon.

Uso de AWS IoT métricas

Las métricas que publica AWS IoT proporcionan información que se 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 cosas a las que se conectan aproximadamente AWS IoT al mismo tiempo.
- Cada objeto se suscribe a un filtro de temas y, a continuación, espera una hora antes de desconectarse. Durante este periodo, 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 cosa se desconecta de AWS IoT.

Para ayudarlo a empezar, estos temas exploran algunas de las preguntas que podría tener.

- ¿Cómo se me puede enviar una notificación si mis objetos no se conectan correctamente cada día?
- ¿Cómo se me puede enviar una notificación si mis objetos no publican datos cada día?
- ¿Cómo se me puede enviar una notificación si las actualizaciones de la sombra de mi objeto se rechazan cada día?
- ¿Cómo puedo crear una CloudWatch alarma para Jobs?

Más información sobre CloudWatch alarmas y métricas

- Creación de CloudWatch alarmas para monitorizar AWS IoT
- AWS IoT métricas y dimensiones

Creación de CloudWatch alarmas para monitorizar AWS IoT

Puede crear una CloudWatch alarma que envíe un mensaje de Amazon SNS cuando la alarma cambie de estado. Una alarma vigila una métrica individual durante un periodo de tiempo que usted

especifica. Cuando el valor de la métrica supera un umbral determinado en un número de períodos de tiempo, se realizan una o varias acciones. La acción puede ser una notificación que se envía a un tema de Amazon SNS o a una política de Auto Scaling. Las alarmas activan acciones únicamente en caso de cambios de estado sostenidos. CloudWatch las alarmas no activan acciones simplemente porque se encuentren en un estado determinado; el estado debe haber cambiado y mantenido durante un número específico de períodos.

En los temas siguientes se describen algunos ejemplos del uso de alarmas de CloudWatch.

- ¿Cómo se me puede enviar una notificación si mis objetos no se conectan correctamente cada día?
- ¿Cómo se me puede enviar una notificación si mis objetos no publican datos cada día?
- ¿Cómo se me puede enviar una notificación si las actualizaciones de la sombra de mi objeto se rechazan cada día?
- ¿Cómo puedo crear una CloudWatch alarma para los trabajos?

Puede ver todas las métricas que las CloudWatch alarmas pueden monitorizarAWS IoT métricas y dimensiones.

¿Cómo se me puede enviar una notificación si mis objetos no se conectan correctamente cada día?

1. Cree un tema de Amazon SNS denominado `things-not-connecting-successfully` y registre su nombre de recurso de Amazon (ARN). Este procedimiento se referirá al ARN de su tema como *`sns-topic-arn`* .

   Para obtener más información sobre cómo crear una notificación de Amazon SNS, consulte Introducción a Amazon SNS.
2. Cree la alarma.

```
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 \
    --evaluation-periods 1 \
    --alarm-actions sns-topic-arn
```

3. Pruebe la alarma.

```
aws cloudwatch set-alarm-state --alarm-name ConnectSuccessAlarm --state-reason
 "initializing" --state-value OK
```

```
aws cloudwatch set-alarm-state --alarm-name ConnectSuccessAlarm --state-reason
 "initializing" --state-value ALARM
```

4. Compruebe que la alarma aparece en la consola de CloudWatch .

¿Cómo se me puede enviar una notificación si mis objetos no publican datos cada día?

1. Cree un tema de Amazon SNS denominado `things-not-publishing-data` y registre su nombre de recurso de Amazon (ARN). Este procedimiento se referirá al ARN de su tema como *sns-topic-arn* .

   Para obtener más información sobre cómo crear una notificación de Amazon SNS, consulte Introducción a Amazon SNS.

2. Cree la alarma.

```
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 \
    --evaluation-periods 1 \
    --alarm-actions sns-topic-arn
```

3. Pruebe la alarma.

```
aws cloudwatch set-alarm-state --alarm-name PublishInSuccessAlarm --state-reason
 "initializing" --state-value OK
```

```
aws cloudwatch set-alarm-state --alarm-name PublishInSuccessAlarm --state-reason
 "initializing" --state-value ALARM
```

4. Compruebe que la alarma aparece en la consola de CloudWatch .

¿Cómo se me puede enviar una notificación si las actualizaciones de la sombra de mi objeto se rechazan cada día?

1. Cree un tema de Amazon SNS denominado `things-shadow-updates-rejected` y registre su nombre de recurso de Amazon (ARN). Este procedimiento se referirá al ARN de su tema como *`sns-topic-arn`* .

   Para obtener más información sobre cómo crear una notificación de Amazon SNS, consulte Introducción a Amazon SNS.

2. Cree la alarma.

```
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 \
    --evaluation-periods 1 \
    --alarm-actions sns-topic-arn
```

3. Pruebe la alarma.

```
aws cloudwatch set-alarm-state --alarm-name UpdateThingShadowSuccessAlarm --state-
reason "initializing" --state-value OK
```

```
aws cloudwatch set-alarm-state --alarm-name UpdateThingShadowSuccessAlarm --state-
reason "initializing" --state-value ALARM
```

4. Compruebe que la alarma aparece en la consola de CloudWatch .

¿Cómo puedo crear una CloudWatch alarma para los trabajos?

El servicio Jobs proporciona CloudWatch métricas para que pueda supervisar sus trabajos. Puede crear CloudWatch alarmas para monitorizar cualquier tipo de alarmaMétricas de trabajos.

El siguiente comando crea una CloudWatch alarma para supervisar el número total de ejecuciones de tareas fallidas de Job *SampleOTAJob* y le notifica cuando se han producido errores en más de 20 ejecuciones de tareas. La alarma monitoriza la métrica de trabajos `FailedJobExecutionTotalCount` comprobando el valor notificado cada 300 segundos. Se activa cuando un único valor notificado es mayor que 20, lo que significa que hubo más de 20 ejecuciones de trabajo fallidas desde que se inició el trabajo. Cuando la alarma se apaga, envía una notificación al tema de Amazon SNS proporcionado.

```
aws cloudwatch put-metric-alarm \
    --alarm-name TotalFailedJobExecution-SampleOTAJob \
    --alarm-description "Alarm when total number of failed job execution exceeds the
 threshold for SampleOTAJob" \
    --namespace AWS/IoT \
    --metric-name FailedJobExecutionTotalCount \
    --dimensions Name=JobId,Value=SampleOTAJob \
    --statistic Sum \
    --threshold 20 \
    --comparison-operator GreaterThanThreshold \
    --period 300 \
    --unit Count \
    --evaluation-periods 1 \
    --alarm-actions arn:aws:sns:<AWS_REGION>:<AWS_ACCOUNT_ID>:SampleOTAJob-has-too-
many-failed-job-ececutions
```

El siguiente comando crea una CloudWatch alarma para supervisar el número de ejecuciones fallidas de Job *SampleOTAJob* en un período determinado. A continuación, lo notifica cuando más de cinco ejecuciones de un trabajo han fracasado durante ese periodo. La alarma monitoriza la métrica de trabajos `FailedJobExecutionCount` comprobando el valor notificado cada 3600 segundos. Se activa cuando un único valor notificado es mayor que 5, lo que significa que hubo más de 5

ejecuciones de trabajo fallidas en la última hora. Cuando la alarma se apaga, envía una notificación al tema de Amazon SNS proporcionado.

```
aws cloudwatch put-metric-alarm \
    --alarm-name FailedJobExecution-SampleOTAJob \
    --alarm-description "Alarm when number of failed job execution per hour exceeds the
 threshold for SampleOTAJob" \
    --namespace AWS/IoT \
    --metric-name FailedJobExecutionCount \
    --dimensions Name=JobId,Value=SampleOTAJob \
    --statistic Sum \
    --threshold 5 \
    --comparison-operator GreaterThanThreshold \
    --period 3600 \
    --unit Count \
    --evaluation-periods 1 \
    --alarm-actions arn:aws:sns:<AWS_REGION>:<AWS_ACCOUNT_ID>:SampleOTAJob-has-too-
many-failed-job-ececutions-per-hour
```

AWS IoT métricas y dimensiones

Cuando interactúas con AWS IoTél, el servicio envía métricas y dimensiones a CloudWatch cada minuto. Puedes usar AWS IoT, usar la CloudWatch consola o AWS CLI ver estas métricas.

Para ver las métricas mediante la CloudWatch consola, abra la CloudWatch consola. En el panel de navegación, elija Métricas y, a continuación, Todas las métricas. En la pestaña Examinar, busque AWS IoT para ver la lista de métricas. 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.

Para ver las métricas mediante AWS CLI, ejecute el siguiente comando.

```
aws cloudwatch list-metrics --namespace "AWS/IoT"
```

CloudWatch muestra los siguientes grupos de métricas para AWS IoT:

- AWS IoT métricas
- AWS IoT Core métricas de proveedores de credenciales
- Métricas de autenticación

- Métricas de la asociación de OCSP de certificados de servidor
- Métricas de reglas
- Métricas de acciones de reglas
- Métricas específicas de acciones HTTP
- Métricas del agente de mensajes
- Métricas de sombras de dispositivos
- Registrar métricas
- Métricas de trabajos
- Métricas de auditoría de Device Defender
- Métricas de detección de Device Defender
- Métricas de aprovisionamiento de dispositivos
- LoRamétricas de WAN
- Métricas de indexación de flotas
- Dimensiones de las métricas

AWS IoT métricas

Métrica	Description (Descripción)
`AddThingToDynamicThingGroupsFailed`	Número de eventos de error asociados a la incorporación de un objeto en un grupo de objetos dinámico. La dimensión `DynamicThingGroupName` contiene el nombre de los grupos dinámicos que no pudieron agregar objetos correctamente.
`NumLogBatchesFailedToPublishThrottled`	El lote de eventos de registro único que no se pudieron publicar debido a errores de limitación controlada.
`NumLogEventsFailedToPublishThrottled`	El número de eventos de registro en el lote que no se pudieron publicar debido a errores de limitación controlada.

AWS IoT Core métricas de proveedores de credenciales

Métrica	Description (Descripción)
`CredentialExchangeSuccess`	El número de solicitudes `AssumeRoleWithCertificate` correctas al proveedor de credenciales de AWS IoT Core .

Métricas de autenticación

> **Note**
>
> Las métricas de autenticación se muestran en la CloudWatch consola, en Protocol Metrics.

Métrica	Description (Descripción)
`Connection.AuthNError`	El número de intentos de conexión que se AWS IoT Core rechazan debido a errores de autenticación. Esta métrica solo tiene en cuenta las conexiones que envían una cadena de indicación del nombre del servidor (SNI) que coincide con un punto de conexión de su Cuenta de AWS. Esta métrica incluye los intentos de conexión desde orígenes externos, como herramientas de escaneo de internet o actividades de sondeo. La dimensión `Protocol` contiene el protocolo usado para enviar el intento de conexión.

Métricas de la asociación de OCSP de certificados de servidor

Métrica	Description (Descripción)
Recuperar OCSPStaple datos. ¿Éxito	La respuesta del OCSP se ha recibido y procesado correctamente. Esta respuesta se incluirá durante

Métrica	Description (Descripción)
	el protocolo de enlace de TLS para el dominio configurado. La dimensión `DomainConfiguratio nName` contiene el nombre del dominio configura do con la asociación de OCSP de certificados de servidor activada.

Métricas de reglas

Métrica	Description (Descripción)
`ParseError`	El número de errores de análisis JSON que se produjeron en los mensajes publicados en un tema en el que hay una regla a la escucha. La dimensión `RuleName` contiene el nombre de la regla.
`RuleExecutionThrottled`	El número de mensajes limitados por el motor de reglas por un comportamiento malintencionado o porque el número de mensajes supera el límite del motor de reglas. La dimensión `RuleName` contiene el nombre de la regla que activar.
`RuleNotFound`	No se ha podido encontrar la regla que activar. La dimensión `RuleName` contiene el nombre de la regla.
`RulesExecuted`	El número de AWS IoT reglas ejecutadas.
`TopicMatch`	El número de mensajes entrantes publicados en un tema en el que hay una regla a la escucha. La dimensión `RuleName` contiene el nombre de la regla.

Métricas de acciones de reglas

Métrica	Description (Descripción)
`Failure`	El número de llamadas a una acción de regla que produjeron un error. La dimensión `RuleName` contiene el nombre de la regla que especifica la acción. La dimensión `ActionType` contiene el tipo de acción que se invocó.
`Success`	El número de llamadas correctas a una acción de regla. La dimensión `RuleName` contiene el nombre de la regla que especifica la acción. La dimensión `ActionType` contiene el tipo de acción que se invocó.
`ErrorActionFailure`	El número de acciones de error fallidas. La dimensión `RuleName` contiene el nombre de la regla que especifica la acción. La dimensión `ActionTyp e` contiene el tipo de acción que se invocó.
`ErrorActionSuccess`	El número de acciones de error correctas. La dimensión `RuleName` contiene el nombre de la regla que especifica la acción. La dimensión `ActionTyp e` contiene el tipo de acción que se invocó.

Métricas específicas de acciones HTTP

Métrica	Description (Descripción)
`HttpCode_Other`	Se genera si el código de estado de la respuesta de la web descendente no service/application es 2xx, 4xx o 5xx.
`HttpCode_4XX`	Se genera si el código de estado de la respuesta de la web descendente service/application está entre 400 y 499.

Métrica	Description (Descripción)
`HttpCode_5XX`	Se genera si el código de estado de la respuesta de la web descendente service/application está entre 500 y 599.
`HttpInvalidUrl`	Se genera si una URL de punto de conexión, una vez reemplazadas las plantillas de sustitución, no comienza por `https://`.
`HttpRequestTimeout`	Se genera si la web descendente service/application no devuelve la respuesta dentro del límite de tiempo de espera de la solicitud. Para obtener más informaci ón, consulte Cuotas de servicio.
`HttpUnknownHost`	Se genera si la URL es válida, pero el servicio no existe o no está accesible.

Métricas del agente de mensajes

> **Note**
>
> Las métricas del agente de mensajes se muestran en la CloudWatch consola, en Protocol Metrics.

Métrica	Description (Descripción)
`Connect.AuthError`	El número de solicitudes de conexión que el agente de mensajes no pudo autorizar. La dimensión `Protocol` contiene el protocolo usado para enviar el mensaje `CONNECT`.
`Connect.ClientError`	El número de solicitudes de conexión rechazadas porque el mensaje MQTT no cumplía los requisito s definidos enAWS IoTCuotas de. La dimensión

Métrica	Description (Descripción)
	`Protocol` contiene el protocolo usado para enviar el mensaje `CONNECT`.
`Connect.ClientIDThrottle`	El número de solicitudes de conexión que se rechazaron porque el cliente superó el límite de solicitudes de conexión permitidas para un ID de cliente específico. La dimensión `Protocol` contiene el protocolo usado para enviar el mensaje `CONNECT`.
`Connect.ServerError`	El número de solicitudes de conexión que fracasaro n porque se produjo un error interno. La dimensión `Protocol` contiene el protocolo usado para enviar el mensaje `CONNECT`.
`Connect.Success`	El número de conexiones realizadas correctamente al agente de mensajes. La dimensión `Protocol` contiene el protocolo usado para enviar el mensaje `CONNECT`.
`Connect.Throttle`	Número de solicitudes de conexión que se rechazaro n porque la cuenta superó el límite permitido. La dimensión `Protocol` contiene el protocolo usado para enviar el mensaje `CONNECT`.
`Ping.Success`	El número de mensajes ping recibidos por el agente de mensajes. La dimensión `Protocol` contiene el protocolo usado para enviar el mensaje ping.
`PublishIn.AuthError`	El número de solicitudes de publicación que el agente de mensajes no pudo autorizar. La dimensión `Protocol` contiene el protocolo usado para publicar el mensaje. HTTP Publish no admite esta métrica.

Métrica	Description (Descripción)
`PublishIn.ClientError`	El número de solicitudes de publicación rechazadas por el agente de mensajes porque el mensaje no cumplía los requisitos definidos en AWS IoT Cuotas de. La dimensión `Protocol` contiene el protocolo usado para publicar el mensaje.
`PublishIn.ServerError`	El número de solicitudes de publicación que el agente de mensajes no pudo procesar porque se produjo un error interno. La dimensión `Protocol` contiene el protocolo usado para enviar el mensaje `PUBLISH`.
`PublishIn.Success`	El número de solicitudes de publicación que el agente de mensajes procesó correctamente. La dimensión `Protocol` contiene el protocolo usado para enviar el mensaje `PUBLISH`.
`PublishIn.Throttle`	El número de solicitudes de publicación que se rechazaron porque el cliente superó el límite de mensajes entrantes permitidos. La dimensión `Protocol` contiene el protocolo usado para enviar el mensaje `PUBLISH`. HTTP Publish no admite esta métrica.
`PublishOut.AuthError`	El número de solicitudes de publicación realizadas por el agente de mensajes que AWS IoT no pudo autorizar. La dimensión `Protocol` contiene el protocolo usado para enviar el mensaje `PUBLISH`.
`PublishOut.ClientError`	El número de solicitudes de publicación realizadas por el agente de mensajes que se rechazaron porque el mensaje no cumplía los requisitos definidos en AWS IoT Cuotas de. La dimensión `Protocol` contiene el protocolo usado para enviar el mensaje `PUBLISH`.

Métrica	Description (Descripción)
PublishOut.Success	El número de solicitudes de publicación realizadas correctamente por el agente de mensajes. La dimensión Protocol contiene el protocolo usado para enviar el mensaje PUBLISH.
PublishOut.Throttle	El número de solicitudes de publicación que se rechazaron porque el cliente superó el límite de mensajes salientes permitidos. La dimensión Protocol contiene el protocolo usado para enviar el mensaje PUBLISH.
PublishRetained.AuthError	El número de solicitudes de publicación con la marca RETAIN establecida que el agente de mensajes no pudo autorizar. La dimensión Protocol contiene el protocolo usado para enviar el mensaje PUBLISH.
PublishRetained.ServerError	El número de solicitudes de publicación conservadas que el agente de mensajes no pudo procesar porque se produjo un error interno. La dimensión Protocol contiene el protocolo usado para enviar el mensaje PUBLISH.
PublishRetained.Success	El número de solicitudes de publicación con la marca RETAIN establecida que el agente de mensajes procesó correctamente. La dimensión Protocol contiene el protocolo usado para enviar el mensaje PUBLISH.
PublishRetained.Throttle	El número de solicitudes de publicación con la marca RETAIN establecida que se limitaron porque el cliente superó la velocidad de mensajes entrantes permitidos. La dimensión Protocol contiene el protocolo usado para enviar el mensaje PUBLISH.

Métrica	Description (Descripción)
`Queued.Success`	El número de mensajes almacenados que el agente de mensajes procesó correctamente para los clientes que se desconectaron de su sesión persistente. Los mensajes con una QoS de 1 se almacenan mientras un cliente con una sesión persistente está desconectado.
`Queued.Throttle`	La cantidad de mensajes que no se pudieron almacenar y que se limitaron mientras los clientes con sesiones persistentes estaban desconectados. Esto ocurre cuando los clientes superan el límite de Mensajes en cola por segundo por cuenta. Los mensajes con una QoS de 1 se almacenan mientras un cliente con una sesión persistente está desconectado.
`Queued.ServerError`	El número de mensajes que no se han almacenado para una sesión persistente debido a un error interno. Cuando los clientes con una sesión persistente se desconectan, se almacenan los mensajes con una calidad de servicio (QoS) igual a 1.
`Subscribe.AuthError`	El número de solicitudes de suscripción realizadas por un cliente que no se pudieron autorizar. La dimensión `Protocol` contiene el protocolo usado para enviar el mensaje `SUBSCRIBE` .
`Subscribe.ClientError`	El número de solicitudes de suscripción que se rechazaron porque el `SUBSCRIBE` mensaje no cumplía los requisitos definidos enAWS IoTCuotas de. La dimensión `Protocol` contiene el protocolo usado para enviar el mensaje `SUBSCRIBE` .

Métrica	Description (Descripción)
`Subscribe.ServerError`	El número de solicitudes de suscripción que se rechazaron porque se produjo un error interno. La dimensión `Protocol` contiene el protocolo usado para enviar el mensaje `SUBSCRIBE` .
`Subscribe.Success`	El número de solicitudes de suscripción que el agente de mensajes procesó correctamente. La dimensión `Protocol` contiene el protocolo usado para enviar el mensaje `SUBSCRIBE` .
`Subscribe.Throttle`	Es el número de solicitudes de suscripción que se ha rechazado porque se han superado los límites de la tasa de solicitudes de suscripción permitidas para su Cuenta de AWS. Estos límites incluyen las suscripciones por segundo por cuenta, las suscripciones por cuenta y las suscripciones por conexión, que se describen en los AWS IoT Core message broker and protocol limits and quotas. La dimensión `Protocol` contiene el protocolo usado para enviar el mensaje `SUBSCRIBE` .
`Throttle.Exceeded`	Esta métrica se mostrará CloudWatch cuando un cliente MQTT limite el número de paquetes por segundo según los límites del nivel de conexión. Esta métrica no se aplica a las conexiones HTTP.
`Unsubscribe.ClientError`	El número de solicitudes de cancelación de suscripción que se rechazaron porque el `UNSUBSCRIBE` mensaje no cumplía los requisitos definidos en. AWS IoTCuotas de La dimensión `Protocol` contiene el protocolo usado para enviar el mensaje `UNSUBSCRIBE` .

Métrica	Description (Descripción)
`Unsubscribe.ServerError`	El número de solicitudes de cancelación de suscripción que se rechazaron porque se produjo un error interno. La dimensión `Protocol` contiene el protocolo usado para enviar el mensaje `UNSUBSCRIBE` .
`Unsubscribe.Success`	El número de solicitudes de cancelación de suscripción que el agente de mensajes procesó correctamente. La dimensión `Protocol` contiene el protocolo usado para enviar el mensaje `UNSUBSCRIBE` .
`Unsubscribe.Throttle`	El número de solicitudes de cancelación de suscripción que se rechazaron porque el cliente superó el límite de solicitudes de cancelación de suscripción permitidas. La dimensión `Protocol` contiene el protocolo usado para enviar el mensaje `UNSUBSCRIBE` .

Métricas de sombras de dispositivos

> **Note**
>
> Las métricas ocultas del dispositivo se muestran en la CloudWatch consola, en Protocol Metrics.

Métrica	Description (Descripción)
`DeleteThingShadow.Accepted`	El número de solicitudes `DeleteThingShadow` procesadas correctamente. La dimensión `Protocol` contiene el protocolo usado para realizar la solicitud.

Métrica	Description (Descripción)
`GetThingShadow.Accepted`	El número de solicitudes `GetThingShadow` procesadas correctamente. La dimensión `Protocol` contiene el protocolo usado para realizar la solicitud.
`ListThingShadow.Accepted`	El número de solicitudes `ListThingShadow` procesadas correctamente. La dimensión `Protocol` contiene el protocolo usado para realizar la solicitud.
`UpdateThingShadow.Accepted`	El número de solicitudes `UpdateThingShadow` procesadas correctamente. La dimensión `Protocol` contiene el protocolo usado para realizar la solicitud.

Registrar métricas

Métrica	Description (Descripción)
`CloudwatchLogs:LogGroupCreationFailed`	El número de errores en las `CreateLogGroup` solicitudes. AWS IoT Logging crea un grupo de CloudWatch registros si no existe. La `LogGroup` dimensión contiene el grupo de registros utilizado para realizar la solicitud.
`CloudwatchLogs:LogStreamCreationFailed`	El número de errores de `CreateLogStream` solicitud. AWS IoT Logging crea un flujo de CloudWatch registro para enviar eventos de registro. La `LogGroup` dimensión contiene el grupo de registros utilizado para realizar la solicitud.
`CloudwatchLogs:PutRetentionPolicyFailed`	El número de errores de `PutRetentionPolicy` solicitud. AWS IoT Logging crea un grupo de CloudWatch registros si no existe y la política de retención se establece en 30 días. La `LogGroup` dimensión contiene el grupo de registros utilizado para realizar la solicitud.

Métrica	Description (Descripción)
`CloudwatchLogs:PutLogEvents Failed`	El número de errores de `PutLogEvents` solicitud . La `LogGroup` dimensión contiene el grupo de CloudWatch registros utilizado para realizar la solicitud.

Métricas de trabajos

Métrica	Description (Descripción)
`CanceledJobExecutionCount`	El número de ejecuciones de trabajos cuyo estado ha cambiado a `CANCELED` dentro de un período de tiempo determinado por CloudWatch. (Para obtener más información sobre CloudWatch las métricas, consulta Amazon CloudWatch Metrics). La dimensión `JobId` contiene el ID del trabajo.
`CanceledJobExecutionTotalCount`	El número total de ejecuciones de trabajo cuyo estado es `CANCELED` para el trabajo especificado. La dimensión `JobId` contiene el ID del trabajo.
`ClientErrorCount`	El número de errores de cliente generados mientras se ejecuta el trabajo. La dimensión `JobId` contiene el ID del trabajo.
`FailedJobExecutionCount`	El número de ejecuciones de trabajos cuyo estado ha `FAILED` cambiado a un período de tiempo determinado por CloudWatch. (Para obtener más información sobre CloudWatch las métricas, consulta Amazon CloudWatch Metrics). La dimensión `JobId` contiene el ID del trabajo.
`FailedJobExecutionTotalCount`	El número total de ejecuciones de trabajo cuyo estado es `FAILED` para el trabajo especificado. La dimensión `JobId` contiene el ID del trabajo.

Métrica	Description (Descripción)
`InProgressJobExecutionCount`	El número de ejecuciones de trabajos cuyo estado ha `IN_PROGRESS` cambiado a un período de tiempo determinado por CloudWatch. (Para obtener más información sobre CloudWatch las métricas, consulta Amazon CloudWatch Metrics). La dimensión `JobId` contiene el ID del trabajo.
`InProgressJobExecutionTotalCount`	El número total de ejecuciones de trabajo cuyo estado es `IN_PROGRESS` para el trabajo especific ado. La dimensión `JobId` contiene el ID del trabajo.
`RejectedJobExecutionTotalCount`	El número total de ejecuciones de trabajo cuyo estado es `REJECTED` para el trabajo especificado. La dimensión `JobId` contiene el ID del trabajo.
`RemovedJobExecutionTotalCount`	El número total de ejecuciones de trabajo cuyo estado es `REMOVED` para el trabajo especificado. La dimensión `JobId` contiene el ID del trabajo.
`QueuedJobExecutionCount`	El número de ejecuciones de trabajos cuyo estado ha `QUEUED` cambiado a un período de tiempo determinado por CloudWatch. (Para obtener más información sobre CloudWatch las métricas, consulta Amazon CloudWatch Metrics). La dimensión `JobId` contiene el ID del trabajo.
`QueuedJobExecutionTotalCount`	El número total de ejecuciones de trabajo cuyo estado es `QUEUED` para el trabajo especificado. La dimensión `JobId` contiene el ID del trabajo.
`RejectedJobExecutionCount`	El número de ejecuciones de trabajos cuyo estado ha `REJECTED` cambiado a un período de tiempo determinado por CloudWatch. (Para obtener más información sobre CloudWatch las métricas, consulta Amazon CloudWatch Metrics). La dimensión `JobId` contiene el ID del trabajo.

Métrica	Description (Descripción)
`RemovedJobExecutionCount`	El número de ejecuciones de trabajos cuyo estado ha `REMOVED` cambiado a un período de tiempo determinado por CloudWatch. (Para obtener más información sobre CloudWatch las métricas, consulta Amazon CloudWatch Metrics). La dimensión `JobId` contiene el ID del trabajo.
`ServerErrorCount`	El número de errores de servidor generados mientras se ejecuta el trabajo. La dimensión `JobId` contiene el ID del trabajo.
`SuccededJobExecutionCount`	El número de ejecuciones de trabajos cuyo estado ha `SUCCESS` cambiado a un período de tiempo determinado por CloudWatch. (Para obtener más información sobre CloudWatch las métricas, consulta Amazon CloudWatch Metrics). La dimensión `JobId` contiene el ID del trabajo.
`SuccededJobExecutionTotalCount`	El número total de ejecuciones de trabajo cuyo estado es `SUCCESS` para el trabajo especificado. La dimensión `JobId` contiene el ID del trabajo.

Métricas de auditoría de Device Defender

Métrica	Description (Descripción)
`NonCompliantResources`	Número de recursos que se ha comprobado que no cumplen los requisitos de una comprobación. El sistema notifica el número de recursos no conformes en cada comprobación de cada auditoría realizada.
`ResourcesEvaluated`	Número de recursos cuya conformidad se evaluó. El sistema notifica el número de recursos que se evaluaron en cada comprobación de cada auditoría realizada.

Métrica	Description (Descripción)
`MisconfiguredDeviceDefender Notification`	Le notifica cuando la configuración de su SNS AWS IoT Device Defender está mal configurada. Dimensiones

Métricas de detección de Device Defender

Métrica	Description (Descripción)
`NumOfMetricsExported`	El número de métricas exportadas para una métrica de la nube, del dispositivo o personalizada. El sistema informa del número de métricas exportada s para la cuenta, respecto de una métrica específic a. Esta métrica solo está disponible para los clientes que utilizan la exportación de métricas.
`NumOfMetricsSkipped`	El número de métricas omitidas para una métrica de la nube, del dispositivo o personalizada. El sistema informa del número de métricas omitidas en la cuenta, respecto de una métrica específica, debido a que Device Defender Detect no tenía suficientes permisos para publicar en el tema mqtt. Esta métrica solo está disponible para los clientes que utilizan la exportación de métricas.
`NumOfMetricsExceedingSizeLi mit`	El número de métricas omitidas para la exportaci ón de una métrica de la nube, del dispositivo o personalizada debido a que el tamaño supera las restricciones de tamaño de los mensajes de MQTT. El sistema informa del número de métricas omitidas para la exportación, respecto de la cuenta y de una métrica específica, debido a que el tamaño supera las restricciones de tamaño de los mensajes de

Métrica	Description (Descripción)
	MQTT. Esta métrica solo está disponible para los clientes que utilizan la exportación de métricas.
`Violations`	El número de nuevas infracciones de los comportamientos del perfil de seguridad que se han encontrado desde la última vez que se realizó una evaluación. El sistema comunica el número de infracciones nuevas de la cuenta, de un perfil de seguridad específico y de un comportamiento concreto de un perfil de seguridad determinado.
`ViolationsCleared`	El número de infracciones de los comportamientos del perfil de seguridad que se han resuelto desde la última vez que se realizó una evaluación. El sistema comunica el número de infracciones resueltas de la cuenta, para un perfil de seguridad específico y para un comportamiento concreto de un perfil de seguridad determinado.
`ViolationsInvalidated`	El número de infracciones de los comportamientos del perfil de seguridad de las que ya no está disponible la información desde la última vez que se realizó una evaluación (debido a que el dispositivo de informe dejó de realizar informes o a que ya no se monitoriza por algún motivo). El sistema comunica el número de infracciones invalidadas de toda la cuenta, de un perfil de seguridad específico y de un comportamiento concreto de un perfil de seguridad determinado.
`MisconfiguredDeviceDefender Notification`	Le notifica cuando la configuración de SNS está mal configurada. AWS IoT Device Defender Dimensiones

Métricas de aprovisionamiento de dispositivos

AWS IoT Métricas de aprovisionamiento de flota

Métrica	Description (Descripción)
`ApproximateNumberOfThingsRegistered`	El recuento de objetos que el aprovisionamiento de flotas ha registrado. Si bien el recuento es generalmente preciso, la arquitectura distribuida de AWS IoT Core dificulta el mantenimiento de un recuento exacto de los objetos registrados. La estadística que se debe utilizar para esta métrica es: • Máximo para indicar el número total de objetos que se han registrado. Para ver un recuento de los datos registrados durante la ventana de CloudWatch agregación, consulta la `RegisterThingFailed` métrica. Dimensiones: ClaimCertificateId
`CreateKeysAndCertificateFailed`	El número de errores que se han producido por llamadas a la API de MQTT `CreateKeysAndCertificate` . La métrica se emite tanto en casos de éxito (valor = 0) como de error (valor = 1). Esta métrica se puede utilizar para realizar un seguimiento del número de certificados creados y registrados durante los CloudWatch períodos de agregación admitidos, como 5 minutos o 1 hora. Las estadísticas disponibles para esta métrica son:

Métrica	Description (Descripción)
	• Sum para informar del número de llamadas fallidas. • SampleCountpara informar del número total de llamadas correctas y fallidas.
`CreateCertificateFromCsrFailed`	El número de errores que se han producido por llamadas a la API de MQTT `CreateCertificateFromCsr` . La métrica se emite tanto en casos de éxito (valor = 0) como de error (valor = 1). Esta métrica se puede utilizar para realizar un seguimiento del número de elementos registrados durante los CloudWatch períodos de agregación admitidos, como 5 minutos o 1 hora. Las estadísticas disponibles para esta métrica son: • Sum para informar del número de llamadas fallidas. • SampleCountpara informar del número total de llamadas correctas y fallidas.

Métrica	Description (Descripción)
`RegisterThingFailed`	El número de errores que se han producido por llamadas a la API de MQTT `RegisterThing` . La métrica se emite tanto en casos de éxito (valor = 0) como de error (valor = 1). Esta métrica se puede utilizar para realizar un seguimiento del número de elementos registrados durante los CloudWatch períodos de agregación admitidos, como 5 minutos o 1 hora. Para ver el número total de objetos registrados, consulte la métrica `ApproximateNumberOfThingsRegistered` . Las estadísticas disponibles para esta métrica son: • Sum para informar del número de llamadas fallidas. • SampleCountpara informar del número total de llamadas correctas y fallidas. Dimensiones: TemplateName

Just-in-time métricas de aprovisionamiento

Métrica	Description (Descripción)
`ProvisionThing.ClientError`	El número de veces que un dispositivo no se ha podido aprovisionar debido a un error del cliente. Por ejemplo, la política especificada en la plantilla no existía.
`ProvisionThing.ServerError`	El número de veces que un dispositivo no se ha podido aprovisionar debido a un error del servidor. Los clientes pueden reintentar aprovisionar el dispositivo después de esperar y ponerse en contacto con AWS IoT si el problema persiste.

Métrica	Description (Descripción)
`ProvisionThing.Success`	El número de veces que un dispositivo se ha aprovisionado correctamente.

LoRamétricas de WAN

En la siguiente tabla se muestran las métricas de AWS IoT Core para LoRaWAN. Para obtener más información, consulte Métricas de AWS IoT Core para LoRaWAN.

AWS IoT Core para LoRaWAN métricas

Métrica	Description (Descripción)
Dispositivos o puertas de enlace activos	La cantidad de dispositivos LoRa WAN y puertas de enlace activos en su cuenta.
Recuento de mensajes de enlace ascendente	Es el número de mensajes de enlace ascendente que se envía dentro de un período de tiempo específico para todas las puertas de enlace y dispositivos activos en su Cuenta de AWS. Los mensajes de enlace ascendente son mensajes que se envían desde el dispositivo a. AWS IoT Core para LoRaWAN
Recuento de mensajes de enlace descendente	Es el número de mensajes de enlace descendente que se envía dentro de un período de tiempo específico para todas las puertas de enlace y dispositivos activos en su Cuenta de AWS. Los mensajes de enlace descendente son mensajes que se envían desde tu AWS IoT Core para LoRaWAN dispositivo.
Tasa de pérdida de mensajes	Una vez que hayas añadido el dispositivo y te hayas conectado a AWS IoT Core para LoRaWANél, el dispositivo puede iniciar un mensaje de enlace ascendente para empezar a intercambiar mensajes con la nube. Puede usar esta métrica para, a

Métrica	Description (Descripción)
	continuación, realizar un seguimiento de la tasa de mensajes de enlace ascendente que se pierden.
Métricas de uniones	Una vez que hayas agregado el dispositivo y la puerta de enlace, debes realizar un procedimiento de unión para que el dispositivo pueda enviar datos de enlace ascendente y comunicarse con ellos. AWS IoT Core para LoRaWAN Puede usar esta métrica para obtener más información sobre las métricas de unión de todos los dispositivos activos de su Cuenta de AWS.
Indicador de intensidad de señal recibida (RSSI) medio	Puede usar esta métrica para supervisar el RSSI (indicador de intensidad de señal recibida) medio dentro del período de tiempo especificado. El RSSI es una medida que indica si la señal es lo suficient emente fuerte para obtener una buena conexión inalámbrica. Este valor es negativo y debe estar más cerca de cero para disponer de una conexión sólida.
Relación señal/ruido (SNR) media	Puede usar esta métrica para monitorear la SNR (Signal-to-noise relación) promedio dentro del período de tiempo especificado. La SNR es una medida que indica si la señal recibida es suficient emente fuerte en comparación con el nivel de ruido para obtener una buena conexión inalámbrica. El valor de la SNR es positivo y debe ser mayor que cero para indicar que la potencia de la señal es más fuerte que la potencia del ruido.
Disponibilidad de la puerta de enlace	Puede usar esta métrica para obtener más informaci ón sobre la disponibilidad de esta puerta de enlace dentro de un período de tiempo específico. Esta métrica muestra el tiempo de conexión a websocket de esta puerta de enlace durante un período de tiempo específico.

Just-in-time métricas de aprovisionamiento

Métrica	Description (Descripción)
`ProvisionThing.ClientError`	El número de veces que un dispositivo no se ha podido aprovisionar debido a un error del cliente. Por ejemplo, la política especificada en la plantilla no existía.
`ProvisionThing.ServerError`	El número de veces que un dispositivo no se ha podido aprovisionar debido a un error del servidor. Los clientes pueden reintentar aprovisionar el dispositivo después de esperar y ponerse en contacto con AWS IoT si el problema persiste.
`ProvisionThing.Success`	El número de veces que un dispositivo se ha aprovisionado correctamente.

Métricas de indexación de flotas

AWS IoT métricas de indexación de flota

Métrica	Description (Descripción)
`NamedShadowCountForDynamicGroupQueryLimitExceeded`	Se procesan un máximo de 25 sombras con nombre por objeto para los términos de consulta que no son específicos del origen de datos en los grupos de objetos dinámicos. Cuando se supere este límite para un objeto, se emitirá el tipo de evento `NamedShadowCountForDynamicGroupQueryLimitExceeded` .

Dimensiones de las métricas

Las métricas utilizan el espacio de nombres y proporcionan métricas para las siguientes dimensiones.

Dimensión	Description (Descripción)
`ActionType`	El tipo de acción especificado por la regla que activó la solicitud.
`BehaviorName`	El nombre del comportamiento del perfil de seguridad de Device Defender Detect que se está monitorizando.
`ClaimCertificateId`	El `certificateId` de la reclamación utilizado para aprovisionar los dispositivos.
`CheckName`	El nombre de la comprobación de auditoría Device Defender cuyos resultados se están monitoreando.
`JobId`	El ID del trabajo cuyo progreso o conexión de mensajes success/failure se está supervisando.
`Protocol`	El protocolo utilizado para realizar la solicitud. Los valores válidos son: MQTT o HTTP
`RuleName`	El nombre de la regla activada por la solicitud.
`ScheduledAuditName`	El nombre de la auditoría programada de Device Defender cuyos resultados de comprobación se están monitoreando. Tiene el valor `OnDemand`si los resultados registrados corresponden a una auditoría que se realizó bajo demanda.
`SecurityProfileName`	El nombre del perfil de seguridad de detección de Device Defender cuyos comportamientos se están monitoreando.
`TemplateName`	El nombre de la plantilla de aprovisionamiento.

Dimensión	Description (Descripción)
`SourceArn`	Hace referencia al perfil de seguridad para la detección o el ARN de registro para la auditoría.
`RoleArn`	Hace referencia al rol que intenta asumir Device Defender.
`TopicArn`	Hace referencia al tema de SNS en el que ha intentando publicar Device Defender.
`Error`	Proporciona una breve descripción del error recibido al intentar publicar en el tema de SNS. Los valores posibles son los que se indican a continuación. • "KMSKeyNotFound«: indica que la clave KMS no existe para el tema. • "InvalidTopicName«: indica que el tema de SNS no es válido. • «KMSAccessDenegado»: indica que el rol no tiene permisos para acceder a la clave de KMS del tema. • "AuthorizationError«: indica que la función proporcionada no autoriza a Device Defender a publicar en el tema de SNS. • "SNSTopicNotFound«: indica que el tema de SNS proporcionado no existe. • "FailureToAssumeRole«: indica que la función proporcionada no autoriza a Device Defender a asumir la función. • "CrossRegionSNSTopic«: indica que el tema de SNS existe en una región diferente.

Supervise AWS IoT mediante registros CloudWatch

Cuando el AWS IoT registro está habilitado, AWS IoT envía eventos de progreso sobre cada mensaje a medida que pasa de sus dispositivos a través del agente de mensajes y el motor de reglas. En la CloudWatch consola, CloudWatch los registros aparecen en un grupo de AWSIotregistros denominado Registros.

Para obtener más información sobre CloudWatch los registros, consulte CloudWatch Registros. Para obtener información sobre AWS IoT CloudWatch los registros compatibles, consulteCloudWatch Registra las entradas de AWS IoT registro.

Visualización AWS IoT de los registros en la CloudWatch consola

> **Note**
>
> El grupo de `AWSIotLogsV2` registros no estará visible en la CloudWatch consola hasta que:
>
> - Has activado el inicio de sesión AWS IoT. Para obtener más información sobre cómo habilitar el inicio de sesión AWS IoT, consulta Configure el AWS IoT registro
> - Algunas entradas de registro se han escrito mediante AWS IoT operaciones.

Para ver AWS IoT los registros en la CloudWatch consola

1. Desplácese hasta https://console.aws.amazon.com/cloudwatch/. En el panel de navegación, seleccione Grupos de registro.
2. En el cuadro Filter (Filtro), escriba **`AWSIotLogsV2`** y, a continuación, pulse Intro.
3. Haga doble clic en el grupo de registros `AWSIotLogsV2`.
4. Seleccione Buscar todo. Aparece una lista completa de los AWS IoT registros generados para su cuenta.
5. Elija el icono de ampliar para analizar un flujo individual.

También puede escribir una consulta en el cuadro de texto Filter events (Filtrar eventos). Aquí tiene algunas consultas interesantes que probar:

- `{ $.logLevel = "INFO" }`

  Busque todos los registros que tengan un nivel de registro de `INFO`.

- `{ $.status = "Success" }`

  Busque todos los registros que tengan un estado de `Success`.
- `{ $.status = "Success" && $.eventType = "GetThingShadow" }`

  Busque todos los registros que tengan un estado de `Success` y un tipo de evento de `GetThingShadow`.

Para obtener más información sobre la creación de expresiones de filtro, consulte CloudWatch Registros de consultas.

CloudWatch Registra las entradas de AWS IoT registro

Cada componente de AWS IoT genera sus propias entradas de registro. Cada entrada de registro tiene un `eventType` que especifica la operación que provocó que se genere la entrada de registro. En esta sección se describen las entradas de registro generadas por los siguientes componentes de AWS IoT .

Temas

- Entradas de registro del agente de mensajes
- Entradas de registro de OCSP de certificados de servidor
- Entradas de registro de sombre de dispositivo
- Entradas del registro del motor de reglas
- Entradas del registro de Job
- Entradas de registro de aprovisionamiento de dispositivos
- Entradas de registro de grupo de objetos dinámicos
- Entradas de registro de indexación de flotas
- Atributos comunes CloudWatch de los registros

Entradas de registro del agente de mensajes

El agente de AWS IoT mensajes genera entradas de registro para los siguientes eventos:

Temas

- Entrada de registro Connect
- Entrada de registro Disconnect

- DeleteConnection entrada de registro
- GetRetainedMessage entrada de registro
- ListRetainedMessage entrada de registro
- Entrada de registro Publish-In
- Entrada de registro Publish-Out
- Entrada de registro en cola
- Entrada de registro de suscripción
- Entrada de registro Unsubscribe

Entrada de registro Connect

El agente de AWS IoT mensajes genera una entrada de registro con una `eventType` de `Connect` cuando se conecta un cliente MQTT.

Ejemplo de entrada de registro Connect

```
{
    "timestamp": "2017-08-10 15:37:23.476",
    "logLevel": "INFO",
    "traceId": "20b23f3f-d7f1-feae-169f-82263394fbdb",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "Connect",
    "protocol": "MQTT",
    "clientId": "abf27092886e49a8a5c1922749736453",
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
    "sourceIp": "205.251.233.181",
    "sourcePort": 13490
}
```

Además deAtributos comunes CloudWatch de los registros, las entradas de `Connect` registro contienen los siguientes atributos:

clientId

El ID del cliente que realiza la solicitud.

principalId

El ID de la entidad principal que realiza la solicitud.

protocol

El protocolo utilizado para realizar la solicitud. Los valores válidos son `MQTT` o `HTTP`.

sourceIp

La dirección IP en la que se originó la solicitud.

sourcePort

El puerto en el que se originó la solicitud.

Entrada de registro Disconnect

El agente de AWS IoT mensajes genera una entrada de registro con una `eventType` de `Disconnect` cuando un cliente MQTT se desconecta.

Ejemplo de entrada de registro Disconnect

```
{
    "timestamp": "2017-08-10 15:37:23.476",
    "logLevel": "INFO",
    "traceId": "20b23f3f-d7f1-feae-169f-82263394fbdb",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "Disconnect",
    "protocol": "MQTT",
    "clientId": "abf27092886e49a8a5c1922749736453",
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
    "sourceIp": "205.251.233.181",
    "sourcePort": 13490,
    "reason": "DUPLICATE_CLIENT_ID",
    "details": "A new connection was established with the same client ID",
    "disconnectReason": "CLIENT_INITIATED_DISCONNECT"
}
```

Además deAtributos comunes CloudWatch de los registros, las entradas de `Disconnect` registro contienen los siguientes atributos:

clientId

El ID del cliente que realiza la solicitud.

principalId

El ID de la entidad principal que realiza la solicitud.

protocol

El protocolo utilizado para realizar la solicitud. Los valores válidos son `MQTT` o `HTTP`.

sourceIp

La dirección IP en la que se originó la solicitud.

sourcePort

El puerto en el que se originó la solicitud.

razón

La razón por la que el cliente se está desconectando.

details

Una breve explicación del error.

disconnectReason

La razón por la que el cliente se está desconectando.

DeleteConnection entrada de registro

El agente de AWS IoT mensajes genera una entrada de registro con un `eventType` de `DeleteConnection` cuando se elimina una conexión de cliente MQTT.

DeleteConnection ejemplo de entrada de registro

```
{
    "timestamp": "2025-08-09 15:37:23.476",
    "logLevel": "INFO",
    "traceId": "20b23f3f-d7f1-feae-169f-82263394fbdb",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "DeleteConnection",
    "protocol": "HTTP",
    "clientId": "abf27092886e49a8a5c1922749736453",
```

```
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
    "sourceIp": "205.251.233.181",
    "sourcePort": 13490
}
```

Además deAtributos comunes CloudWatch de los registros, las entradas de `DeleteConnection` registro contienen los siguientes atributos:

clientId

El identificador del cliente que se desconectará.

principalId

El ID de la entidad principal que realiza la solicitud. Para obtener más información sobre cómo identificar la entidad principal mediante el ID de la entidad principal, consulte Comparación de identidades y credenciales de IAM en la Guía del usuario de IAM.

protocol

El protocolo utilizado para realizar la solicitud. El valor válido es `HTTP`.

sourceIp

La dirección IP en la que se originó la solicitud.

sourcePort

El puerto en el que se originó la solicitud.

GetRetainedMessage entrada de registro

El agente de AWS IoT mensajes genera una entrada de registro con una `eventType` de `GetRetainedMessage` cuándo `GetRetainedMessage`se llama.

GetRetainedMessage ejemplo de entrada de registro

```
{
    "timestamp": "2017-08-07 18:47:56.664",
    "logLevel": "INFO",
    "traceId": "1a60d02e-15b9-605b-7096-a9f584a6ad3f",
    "accountId": "123456789012",
    "status": "Success",
```

```
    "eventType": "GetRetainedMessage",
    "protocol": "HTTP",
    "topicName": "a/b/c",
    "qos": "1",
    "lastModifiedDate": "2017-08-07 18:47:56.664"
}
```

Además deAtributos comunes CloudWatch de los registros, las entradas de `GetRetainedMessage` registro contienen los siguientes atributos:

lastModifiedDate

La fecha y hora de la época, en milisegundos, en la que se almacenó el mensaje retenido. AWS IoT

protocol

El protocolo utilizado para realizar la solicitud. Valor válido: `HTTP`.

qos

El nivel de calidad de servicio (QoS) utilizado en la solicitud de publicación. Los valores válidos son `0` o `1`.

topicName

El nombre del tema suscrito.

ListRetainedMessage entrada de registro

El agente de AWS IoT mensajes genera una entrada de registro con una `eventType` de `ListRetainedMessage` cuándo `ListRetainedMessages`se llama.

ListRetainedMessage ejemplo de entrada de registro

```
{
    "timestamp": "2017-08-07 18:47:56.664",
    "logLevel": "INFO",
    "traceId": "1a60d02e-15b9-605b-7096-a9f584a6ad3f",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "ListRetainedMessage",
    "protocol": "HTTP"
```

```
}
```

Además deAtributos comunes CloudWatch de los registros, las entradas de `ListRetainedMessage` registro contienen el siguiente atributo:

protocol

El protocolo utilizado para realizar la solicitud. Valor válido: `HTTP`.

Entrada de registro Publish-In

Cuando el agente de AWS IoT mensajes recibe un mensaje MQTT, genera una entrada de registro con un número `eventType` de`Publish-In`.

Ejemplo de entrada de registro Publish-In

```
{
        "timestamp": "2017-08-10 15:39:30.961",
        "logLevel": "INFO",
        "traceId": "672ec480-31ce-fd8b-b5fb-22e3ac420699",
        "accountId": "123456789012",
        "status": "Success",
        "eventType": "Publish-In",
        "protocol": "MQTT",
        "topicName": "$aws/things/MyThing/shadow/get",
        "clientId": "abf27092886e49a8a5c1922749736453",
        "principalId":
 "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
        "sourceIp": "205.251.233.181",
        "sourcePort": 13490,
        "retain": "True"
    }
```

Además deAtributos comunes CloudWatch de los registros, las entradas de `Publish-In` registro contienen los siguientes atributos:

clientId

El ID del cliente que realiza la solicitud.

principalId

El ID de la entidad principal que realiza la solicitud.

protocol

El protocolo utilizado para realizar la solicitud. Los valores válidos son `MQTT` o `HTTP`.

retain

El atributo que se usa cuando un mensaje tiene la marca RETAIN establecida con un valor de `True`. Si el mensaje no tiene la marca RETAIN establecida, este atributo no aparece en la entrada de registro. Para obtener más información, consulte Mensajes retenidos de MQTT .

sourceIp

La dirección IP en la que se originó la solicitud.

sourcePort

El puerto en el que se originó la solicitud.

topicName

El nombre del tema suscrito.

Entrada de registro Publish-Out

Cuando el agente de mensajes publica un mensaje MQTT, genera una entrada de registro con un `eventType` de `Publish-Out`.

Ejemplo de entrada de registro Publish-Out

```
{
    "timestamp": "2017-08-10 15:39:30.961",
    "logLevel": "INFO",
    "traceId": "672ec480-31ce-fd8b-b5fb-22e3ac420699",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "Publish-Out",
    "protocol": "MQTT",
    "topicName": "$aws/things/MyThing/shadow/get",
    "clientId": "abf27092886e49a8a5c1922749736453",
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
    "sourceIp": "205.251.233.181",
    "sourcePort": 13490
}
```

Además deAtributos comunes CloudWatch de los registros, las entradas de `Publish-Out` registro contienen los siguientes atributos:

clientId

El ID del cliente suscrito que recibe mensajes sobre ese tema de MQTT.

principalId

El ID de la entidad principal que realiza la solicitud.

protocol

El protocolo utilizado para realizar la solicitud. Los valores válidos son `MQTT` o `HTTP`.

sourceIp

La dirección IP en la que se originó la solicitud.

sourcePort

El puerto en el que se originó la solicitud.

topicName

El nombre del tema suscrito.

Entrada de registro en cola

Cuando se desconecta un dispositivo con una sesión persistente, el agente de mensajes MQTT almacena los mensajes del dispositivo y AWS IoT genera entradas de registro con un `Queued` EventType de. Para obtener más información sobre las sesiones persistentes de MQTT, consulte. Sesiones persistentes de MQTT

Ejemplo de entrada de registro de error del servidor en cola

```
{
    "timestamp": "2022-08-10 15:39:30.961",
    "logLevel": "ERROR",
    "traceId": "672ec480-31ce-fd8b-b5fb-22e3ac420699",
    "accountId": "123456789012",
    "topicName": "$aws/things/MyThing/get",
    "clientId": "123123123",
    "qos": "1",
```

```
    "protocol": "MQTT",
    "eventType": "Queued",
    "status": "Failure",
    "details": "Server Error"
}
```

Además de Atributos comunes CloudWatch de los registros, las entradas del registro de errores `Queued` del servidor contienen los siguientes atributos:

clientId

El ID del cliente al que se pone el mensaje en cola.

details

`Server Error`

Un error del servidor ha impedido que se almacene el mensaje.

protocol

El protocolo utilizado para realizar la solicitud. El valor será siempre `MQTT`.

qos

El nivel de calidad de servicio (QoS) de la solicitud. El valor siempre será 1 porque los mensajes con una QoS de 0 no se almacenan.

topicName

El nombre del tema suscrito.

Ejemplo de entrada de registro de éxito en cola

```
{
    "timestamp": "2022-08-10 15:39:30.961",
    "logLevel": "INFO",
    "traceId": "672ec480-31ce-fd8b-b5fb-22e3ac420699",
    "accountId": "123456789012",
    "topicName": "$aws/things/MyThing/get",
    "clientId": "123123123",
    "qos": "1",
    "protocol": "MQTT",
```

```
    "eventType": "Queued",
    "status": "Success"
}
```

Además deAtributos comunes CloudWatch de los registros, las entradas del registro de `Queued` éxitos contienen los siguientes atributos:

clientId

El ID del cliente al que se pone el mensaje en cola.

protocol

El protocolo utilizado para realizar la solicitud. El valor será siempre `MQTT`.

qos

El nivel de calidad de servicio (QoS) de la solicitud. El valor siempre será 1 porque los mensajes con una QoS de 0 no se almacenan.

topicName

El nombre del tema suscrito.

Ejemplo de entrada de registro limitada en cola

```
{
    "timestamp": "2022-08-10 15:39:30.961",
    "logLevel": "ERROR",
    "traceId": "672ec480-31ce-fd8b-b5fb-22e3ac420699",
    "accountId": "123456789012",
    "topicName": "$aws/things/MyThing/get",
    "clientId": "123123123",
    "qos": "1",
    "protocol": "MQTT",
    "eventType": "Queued",
    "status": "Failure",
    "details": "Throttled while queueing offline message"
}
```

Además de esoAtributos comunes CloudWatch de los registros, las entradas de `Queued` registro limitadas contienen los siguientes atributos:

clientId

El ID del cliente al que se pone el mensaje en cola.

details

`Throttled while queueing offline message`

El cliente superó el límite `Queued messages per second per account` , por lo que el mensaje no se almacenó.

protocol

El protocolo utilizado para realizar la solicitud. El valor será siempre `MQTT`.

qos

El nivel de calidad de servicio (QoS) de la solicitud. El valor siempre será 1 porque los mensajes con una QoS de 0 no se almacenan.

topicName

El nombre del tema suscrito.

Entrada de registro de suscripción

El agente de AWS IoT mensajes genera una entrada de registro con una `eventType` de `Subscribe` cuando un cliente MQTT se suscribe a un tema.

Ejemplo de entrada de registro de suscripción de MQTT 3

```
{
    "timestamp": "2017-08-10 15:39:04.413",
    "logLevel": "INFO",
    "traceId": "7aa5c38d-1b49-3753-15dc-513ce4ab9fa6",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "Subscribe",
    "protocol": "MQTT",
    "topicName": "$aws/things/MyThing/shadow/#",
    "clientId": "abf27092886e49a8a5c1922749736453",
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
    "sourceIp": "205.251.233.181",
```

```
    "sourcePort": 13490
}
```

Además de[Atributos comunes CloudWatch de los registros](), las entradas de `Subscribe` registro contienen los siguientes atributos:

clientId

El ID del cliente que realiza la solicitud.

principalId

El ID de la entidad principal que realiza la solicitud.

protocol

El protocolo utilizado para realizar la solicitud. El valor será siempre `MQTT`.

sourceIp

La dirección IP en la que se originó la solicitud.

sourcePort

El puerto en el que se originó la solicitud.

topicName

El nombre del tema suscrito.

Ejemplo de entrada de registro de suscripción de MQTT 5

```
{
 "timestamp": "2022-11-30 16:24:15.628",
 "logLevel": "INFO",
 "traceId": "7aa5c38d-1b49-3753-15dc-513ce4ab9fa6",
 "accountId": "123456789012",
 "status": "Success",
 "eventType": "Subscribe",
 "protocol": "MQTT",
 "topicName": "test/topic1,$invalid/reserved/topic",
 "subscriptions": [
  {
   "topicName": "test/topic1",
```

```
    "reasonCode": 1
   },
   {
    "topicName": "$invalid/reserved/topic",
    "reasonCode": 143
   }
  ],
  "clientId": "abf27092886e49a8a5c1922749736453",
  "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
  "sourceIp": "205.251.233.181",
  "sourcePort": 13490
}
```

En el caso de las operaciones de suscripción de MQTT 5, además del Atributos comunes CloudWatch de los registros y los atributos de entrada de registro de suscripción de MQTT 3, las entradas de registro `Subscribe` de MQTT 5 contienen el siguiente atributo:

subscriptions

Una lista de asignaciones entre los temas solicitados en la solicitud de suscripción y el código de motivo individual de MQTT 5. Para obtener más información, consulte Códigos de motivo de MQTT.

Entrada de registro Unsubscribe

El agente de AWS IoT mensajes genera una entrada de registro con una `eventType` de `Unsubscribe` cuando un cliente de MQTT cancela la suscripción a un tema de MQTT.

Ejemplo de entrada de registro de cancelación de suscripción de MQTT

```
{
    "timestamp": "2024-08-20 22:53:32.844",
    "logLevel": "INFO",
    "traceId": "db6bd09a-2c3f-1cd2-27cc-fd6b1ce03b58",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "Unsubscribe",
    "protocol": "MQTT",
    "clientId": "abf27092886e49a8a5c1922749736453",
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
    "sourceIp": "205.251.233.181",
    "sourcePort": 13490
```

```
}
```

Además de esoAtributos comunes CloudWatch de los registros, las entradas de `Unsubscribe` registro contienen los siguientes atributos:

protocol

El protocolo utilizado para realizar la solicitud. El valor será siempre `MQTT`.

clientId

El ID del cliente que realiza la solicitud.

principalId

El ID de la entidad principal que realiza la solicitud.

sourceIp

La dirección IP en la que se originó la solicitud.

sourcePort

El puerto en el que se originó la solicitud.

Entradas de registro de OCSP de certificados de servidor

AWS IoT Core genera entradas de registro para el siguiente evento:

Temas

- Recupera la entrada del registro de OCSPStaple datos
- Recupere la entrada del registro de OCSPStaple datos para puntos finales privados

Recupera la entrada del registro de OCSPStaple datos

AWS IoT Core genera una entrada de registro con un `eventType` de `RetrieveOCSPStapleData` cuando el servidor recupera los datos básicos del OCSP.

Recupere ejemplos OCSPStaple de entradas del registro de datos

Lo que sigue es una entrada de registro de `Success` de ejemplo.

```
{
 "timestamp": "2024-01-30 15:39:30.961",
 "logLevel": "INFO",
 "traceId": "180532b7-0cc7-057b-687a-5ca1824838f5",
 "accountId": "123456789012",
 "status": "Success",
 "eventType": "RetrieveOCSPStapleData",
 "domainConfigName": "test-domain-config-name",
 "connectionDetails": {
  "httpStatusCode": "200",
  "ocspResponderUri": "http://ocsp.example.com",
  "sourceIp": "205.251.233.181",
  "targetIp": "250.15.5.3"
 },
 "ocspRequestDetails": {
  "requesterName": "iot.amazonaws.com",
  "requestCertId":
 "30:3A:30:09:06:05:2B:0E:03:02:1A:05:00:04:14:9C:FF:90:A1:97:B0:4D:6C:01:B9:69:96:D8:3E:E7:A2:
 },
 "ocspResponseDetails": {
  "responseCertId":
 "30:3A:30:09:06:05:2B:0E:03:02:1A:05:00:04:14:9C:FF:90:A1:97:B0:4D:6C:01:B9:69:96:D8:3E:E7:A2:
  "ocspResponseStatus": "successful",
  "certStatus": "good",
  "signature":
 "4C:6F:63:61:6C:20:52:65:73:70:6F:6E:64:65:72:20:53:69:67:6E:61:74:75:72:65",
  "thisUpdateTime": "Jan 31 01:21:02 2024 UTC",
  "nextUpdateTime": "Feb 02 00:21:02 2024 UTC",
  "producedAtTime": "Jan 31 01:37:03 2024 UTC",
  "stapledDataPayloadSize": "XXX"
 }
}
```


Lo que sigue es una entrada de registro de `Failure` de ejemplo.


```
{
 "timestamp": "2024-01-30 15:39:30.961",
 "logLevel": "ERROR",
 "traceId": "180532b7-0cc7-057b-687a-5ca1824838f5",
 "accountId": "123456789012",
 "status": "Failure",
 "reason": "A non 2xx HTTP response was received from the OCSP responder.",
 "eventType": "RetrieveOCSPStapleData",
```

```
 "domainConfigName": "test-domain-config-name",
 "connectionDetails": {
  "httpStatusCode": "444",
  "ocspResponderUri": "http://ocsp.example.com",
  "sourceIp": "205.251.233.181",
  "targetIp": "250.15.5.3"
 },
 "ocspRequestDetails": {
  "requesterName": "iot.amazonaws.com",
  "requestCertId":
 "30:3A:30:09:06:05:2B:0E:03:02:1A:05:00:04:14:9C:FF:90:A1:97:B0:4D:6C:01:B9:69:96:D8:3E:E7:A2:
 }
}
```


Para la `RetrieveOCSPStaple` operación, además deAtributos comunes CloudWatch de los registros, las entradas de registro contienen los siguientes atributos:

razón

Es el motivo por el que la operación falla.

domainConfigName

Es el nombre de la configuración del dominio.

connectionDetails

Es una explicación breve de los detalles de la conexión.

- httpStatusCode

  Códigos de estado de HTTP que devuelve el respondedor de OCSP en respuesta a la solicitud del cliente realizada al servidor.

- ocspResponderUri

  El URI del respondedor OCSP que se AWS IoT Core obtiene del certificado del servidor.

- sourceIp

  La dirección IP de origen del AWS IoT Core servidor.

- targetIp

  Es la dirección IP de destino del respondedor de OCSP.

ocspRequestDetails

Detalles de la solicitud de OCSP.

- requesterName

  El identificador del AWS IoT Core servidor que envía una solicitud al respondedor OCSP.

- requestCertId

  Es el ID del certificado de la solicitud. Este es el ID del certificado para el que se solicita la respuesta del OCSP.

ocspResponseDetails

Son los detalles de la respuesta del OCSP.

- responseCertId

  Es el ID del certificado de la respuesta del OCSP.

- ocspResponseStatus

  Es el estado de la respuesta del OCSP.

- certStatus

  El estado del certificado.

- firma

  Es la firma que una entidad de confianza aplica a la respuesta.

- thisUpdateTime

  Es el momento en el que se sabe que el estado que se indica es correcto.

- nextUpdateTime

  Es la hora a la que estará disponible la información más reciente sobre el estado del certificado.

- producedAtTime

  Es la hora en la que el respondedor del OCSP firmó esta respuesta.

- stapledDataPayloadTamaño

  Es el tamaño de la carga útil de los datos asociados.

Recupere la entrada del registro de OCSPStaple datos para puntos finales privados

AWS IoT Core genera una entrada de registro con un `eventType` de `RetrieveOCSPStapleData` cuando el servidor recupera los datos básicos del OCSP.

Recupere ejemplos de entradas del registro de OCSPStaple datos para puntos finales privados

Lo que sigue es una entrada de registro de `Success` de ejemplo.


```
{
 "timestamp": "2024-01-30 15:39:30.961",
 "logLevel": "INFO",
 "traceId": "180532b7-0cc7-057b-687a-5ca1824838f5",
 "accountId": "123456789012",
 "status": "Success",
 "eventType": "RetrieveOCSPStapleData",
 "domainConfigName": "test-domain-config-name",
        "lambdaDetails": {
        "lambdaArn": "arn:aws:lambda:us-west-2:123456789012:function:my-function",
        "sourceArn": "arn:aws:iot:us-west-2:123456789012:domainconfiguration/
testDomainConfigure/6bzfg"
      },
        "authorizedResponderArn": "arn:aws:acm:us-west-2:123456789012:certificate/
certificate_ID",
 "ocspRequestDetails": {
  "requesterName": "iot.amazonaws.com",
  "requestCertId":
 "30:3A:30:09:06:05:2B:0E:03:02:1A:05:00:04:14:9C:FF:90:A1:97:B0:4D:6C:01:B9:69:96:D8:3E:E7:A2:
 },
 "ocspResponseDetails": {
  "responderId": "04:C1:3F:8F:27:D6:49:13:F8:DE:B2:36:9D:85:8E:F8:31:3B:A6:D0"
              "responseCertId":
 "30:3A:30:09:06:05:2B:0E:03:02:1A:05:00:04:14:9C:FF:90:A1:97:B0:4D:6C:01:B9:69:96:D8:3E:E7:A2:
  "ocspResponseStatus": "successful",
  "certStatus": "good",
  "signature":
 "4C:6F:63:61:6C:20:52:65:73:70:6F:6E:64:65:72:20:53:69:67:6E:61:74:75:72:65",
  "thisUpdateTime": "Jan 31 01:21:02 2024 UTC",
  "nextUpdateTime": "Feb 02 00:21:02 2024 UTC",
  "producedAtTime": "Jan 31 01:37:03 2024 UTC",
  "stapledDataPayloadSize": "XXX"
 }
}
```

Lo que sigue es una entrada de registro de `Failure` de ejemplo.

```
{
 "timestamp": "2024-01-30 15:39:30.961",
 "logLevel": "ERROR",
 "traceId": "180532b7-0cc7-057b-687a-5ca1824838f5",
 "accountId": "123456789012",
 "status": "Failure",
 "reason": "The payload returned by the Lambda function exceeds the maximum response
 size of 7 kilobytes.",
 "eventType": "RetrieveOCSPStapleData",
 "domainConfigName": "test-domain-config-name",
        "lambdaDetails": {
        "lambdaArn": "arn:aws:lambda:us-west-2:123456789012:function:my-function",
        "sourceArn": "arn:aws:iot:us-west-2:123456789012:domainconfiguration/
testDomainConfigure/6bzfg"
      },
        "authorizedResponderArn": "arn:aws:acm:us-west-2:123456789012:certificate/
certificate_ID",
  "ocspRequestDetails": {
  "requesterName": "iot.amazonaws.com",
  "requestCertId":
 "30:3A:30:09:06:05:2B:0E:03:02:1A:05:00:04:14:9C:FF:90:A1:97:B0:4D:6C:01:B9:69:96:D8:3E:E7:A2:
 }
}
```

Para la `RetrieveOCSPStaple` operación, además de los atributos de la Atributos comunes CloudWatch de los registros entrada de registro de recuperación de OCSPStaple datos, las entradas de registro de los puntos finales privados contienen los siguientes atributos:

lambdaDetails

Detalles de la función de Lambda.

- lambdaArn

  El ARN de la función de Lambda.

- sourceArn

  El ARN de la configuración del dominio.

authorizedResponderArn

El ARN del respondedor del autorizador, si hay uno configurado en la configuración del dominio.

Entradas de registro de sombre de dispositivo

El servicio AWS IoT Device Shadow genera entradas de registro para los siguientes eventos:

Temas

- DeleteThingShadow entrada de registro
- GetThingShadow entrada de registro
- UpdateThingShadow entrada de registro

DeleteThingShadow entrada de registro

El servicio de sombra de dispositivo genera una entrada de registro con un `eventType` de `DeleteThingShadow` cuando se recibe una solicitud de eliminación de la sombra de un dispositivo.

DeleteThingShadow ejemplo de entrada de registro

```
{
    "timestamp": "2017-08-07 18:47:56.664",
    "logLevel": "INFO",
    "traceId": "1a60d02e-15b9-605b-7096-a9f584a6ad3f",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "DeleteThingShadow",
    "protocol": "MQTT",
    "deviceShadowName": "Jack",
    "topicName": "$aws/things/Jack/shadow/delete"
}
```

Además deAtributos comunes CloudWatch de los registros, las entradas de `DeleteThingShadow` registro contienen los siguientes atributos:

deviceShadowName

Nombre de la sombra que se va a actualizar.

protocol

El protocolo utilizado para realizar la solicitud. Los valores válidos son `MQTT` o `HTTP`.

topicName

El nombre del tema en el que se publicó la solicitud.

GetThingShadow entrada de registro

El servicio de sombra de dispositivo genera una entrada de registro con un `eventType` de `GetThingShadow` cuando se recibe una solicitud de obtención para una sombra.

GetThingShadow ejemplo de entrada de registro

```
{
    "timestamp": "2017-08-09 17:56:30.941",
    "logLevel": "INFO",
    "traceId": "b575f19a-97a2-cf72-0ed0-c64a783a2504",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "GetThingShadow",
    "protocol": "MQTT",
    "deviceShadowName": "MyThing",
    "topicName": "$aws/things/MyThing/shadow/get"
}
```

Además deAtributos comunes CloudWatch de los registros, las entradas de `GetThingShadow` registro contienen los siguientes atributos:

deviceShadowName

El nombre de la sombra solicitada.

protocol

El protocolo utilizado para realizar la solicitud. Los valores válidos son `MQTT` o `HTTP`.

topicName

El nombre del tema en el que se publicó la solicitud.

UpdateThingShadow entrada de registro

El servicio de sombra de dispositivo genera una entrada de registro con un `eventType` de `UpdateThingShadow` cuando se recibe una solicitud de actualización de la sombra de un dispositivo.

UpdateThingShadow ejemplo de entrada de registro

```
{
    "timestamp": "2017-08-07 18:43:59.436",
    "logLevel": "INFO",
    "traceId": "d0074ba8-0c4b-a400-69df-76326d414c28",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "UpdateThingShadow",
    "protocol": "MQTT",
    "deviceShadowName": "Jack",
    "topicName": "$aws/things/Jack/shadow/update"
}
```

Además deAtributos comunes CloudWatch de los registros, las entradas de `UpdateThingShadow` registro contienen los siguientes atributos:

deviceShadowName

Nombre de la sombra que se va a actualizar.

protocol

El protocolo utilizado para realizar la solicitud. Los valores válidos son `MQTT` o `HTTP`.

topicName

El nombre del tema en el que se publicó la solicitud.

Entradas del registro del motor de reglas

El motor de AWS IoT reglas genera registros para los siguientes eventos:

Temas

- FunctionExecution entrada de registro
- RuleExecution entrada de registro

- RuleMatch entrada de registro
- RuleExecutionThrottled entrada de registro
- RuleNotFound entrada de registro
- StartingRuleExecution entrada de registro

FunctionExecution entrada de registro

El motor de reglas genera una entrada de registro con un `eventType` de `FunctionExecution` cuando la consulta SQL de una regla llama a una función externa. Se llama a una función externa cuando la acción de una regla realiza una solicitud HTTP a AWS IoT otro servicio web (por ejemplo, llamando a `get_thing_shadow` o`machinelearning_predict`).

FunctionExecution ejemplo de entrada de registro

```
{
    "timestamp": "2017-07-13 18:33:51.903",
    "logLevel": "DEBUG",
    "traceId": "180532b7-0cc7-057b-687a-5ca1824838f5",
    "status": "Success",
    "eventType": "FunctionExecution",
    "clientId": "N/A",
    "topicName":"rules/test",
    "ruleName": "ruleTestPredict",
    "ruleAction": "MachinelearningPredict",
    "resources": {
        "ModelId": "predict-model"
    },
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167"
}
```

Además deAtributos comunes CloudWatch de los registros, las entradas de `FunctionExecution` registro contienen los siguientes atributos:

clientId

`N/A` para registros `FunctionExecution`.

principalId

El ID de la entidad principal que realiza la solicitud.

recursos

Un conjunto de recursos utilizados por las acciones de la regla.

ruleName

El nombre de la regla que coincide.

topicName

El nombre del tema suscrito.

RuleExecution entrada de registro

Cuando el motor de AWS IoT reglas activa la acción de una regla, genera una entrada de `RuleExecution` registro.

RuleExecution ejemplo de entrada de registro

```
{
    "timestamp": "2017-08-10 16:32:46.070",
    "logLevel": "INFO",
    "traceId": "30aa7ccc-1d23-0b97-aa7b-76196d83537e",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "RuleExecution",
    "clientId": "abf27092886e49a8a5c1922749736453",
    "topicName": "rules/test",
    "ruleName": "JSONLogsRule",
    "ruleAction": "RepublishAction",
    "resources": {
        "RepublishTopic": "rules/republish"
    },
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167"
}
```

RuleExecution con un ejemplo de entrada de registro por lotes

```
{
    "logLevel": "INFO",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "RuleExecution",
    "ruleName": "rule_test",
```

```
    "ruleAction": "HttpAction",
    "resources": {
        "Url": "https://example.com",
        "ConfirmationUrl": "https://example.com"
    },
    "details": "HttpAction made a request to the specified endpoint",
    "batchDetails": {
        "timestamps": [
            "1234567890123",
            "1234567890123",
            "1234567890123"
        ],
        "traceIds": [
            "30aa7ccc-1d23-0b97-aa7b-76196d83537e",
            "30aa7ccc-1d23-0b97-aa7b-76196d83537c",
            "30aa7ccc-1d23-0b97-aa7b-76196d83537d"
        ],
        "clientIds": [
            "N/A",
            "N/A",
            "N/A"
        ],
        "topicNames": [
            "topic/ruletest",
            "topic/ruletest",
            "topic/ruletest"
        ],
        "principalIds": [
            "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
            "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
            "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167"
        ],
        "batchSize": 3,
        "batchSizeInBytes": 114
    }
}
```


Además de Atributos comunes CloudWatch de los registros, las entradas de registro `RuleExecution` incluyen los siguientes atributos:

clientId

El ID del cliente que realiza la solicitud.

principalId

El ID de la entidad principal que realiza la solicitud.

recursos

Un conjunto de recursos utilizados por las acciones de la regla.

ruleAction

El nombre de la acción activada.

ruleName

El nombre de la regla que coincide.

topicName

El nombre del tema suscrito.

RuleMatch entrada de registro

El motor de AWS IoT reglas genera una entrada de registro con un `eventType` de `RuleMatch` cuando el agente de mensajes recibe un mensaje que coincide con una regla.

RuleMatch ejemplo de entrada de registro

```
{
    "timestamp": "2017-08-10 16:32:46.002",
    "logLevel": "INFO",
    "traceId": "30aa7ccc-1d23-0b97-aa7b-76196d83537e",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "RuleMatch",
    "clientId": "abf27092886e49a8a5c1922749736453",
    "topicName": "rules/test",
    "ruleName": "JSONLogsRule",
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167"
}
```

Además deAtributos comunes CloudWatch de los registros, las entradas de `RuleMatch` registro contienen los siguientes atributos:

clientId

El ID del cliente que realiza la solicitud.

principalId

El ID de la entidad principal que realiza la solicitud.

ruleName

El nombre de la regla que coincide.

topicName

El nombre del tema suscrito.

RuleExecutionThrottled entrada de registro

Cuando se limita una ejecución, el motor de AWS IoT reglas genera una entrada de registro con un `eventType` de. `RuleExecutionThrottled`

RuleExecutionThrottled ejemplo de entrada de registro

```
{
    "timestamp": "2017-10-04 19:25:46.070",
    "logLevel": "ERROR",
    "traceId": "30aa7ccc-1d23-0b97-aa7b-76196d83537e",
    "accountId": "123456789012",
    "status": "Failure",
    "eventType": "RuleExecutionThrottled",
    "clientId": "abf27092886e49a8a5c1922749736453",
    "topicName": "$aws/rules/example_rule",
    "ruleName": "example_rule",
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
    "reason": "RuleExecutionThrottled",
    "details": "Exection of Rule example_rule throttled"
}
```

Además de[Atributos comunes CloudWatch de los registros](), las entradas de `RuleExecutionThrottled` registro contienen los siguientes atributos:

clientId

El ID del cliente que realiza la solicitud.

details

Una breve explicación del error.

principalId

El ID de la entidad principal que realiza la solicitud.

razón

La cadena "RuleExecutionThrottled».

ruleName

El nombre de la regla que se debe activar.

topicName

El nombre del tema publicado.

RuleNotFound entrada de registro

Cuando el motor de AWS IoT reglas no puede encontrar una regla con un nombre determinado, genera una entrada de registro con un número `eventType` de`RuleNotFound`.

RuleNotFound ejemplo de entrada de registro

```
{
    "timestamp": "2017-10-04 19:25:46.070",
    "logLevel": "ERROR",
    "traceId": "30aa7ccc-1d23-0b97-aa7b-76196d83537e",
    "accountId": "123456789012",
    "status": "Failure",
    "eventType": "RuleNotFound",
    "clientId": "abf27092886e49a8a5c1922749736453",
    "topicName": "$aws/rules/example_rule",
    "ruleName": "example_rule",
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
    "reason": "RuleNotFound",
    "details": "Rule example_rule not found"
}
```

Además de[Atributos comunes CloudWatch de los registros](), las entradas de `RuleNotFound` registro contienen los siguientes atributos:

clientId

El ID del cliente que realiza la solicitud.

details

Una breve explicación del error.

principalId

El ID de la entidad principal que realiza la solicitud.

razón

La cadena "RuleNotFound».

ruleName

El nombre de la regla que no se pudo encontrar.

topicName

El nombre del tema publicado.

StartingRuleExecution entrada de registro

Cuando el motor de AWS IoT reglas comienza a activar la acción de una regla, genera una entrada de registro con un número `eventType` de`StartingRuleExecution`.

StartingRuleExecution ejemplo de entrada de registro

```
{
    "timestamp": "2017-08-10 16:32:46.002",
    "logLevel": "DEBUG",
    "traceId": "30aa7ccc-1d23-0b97-aa7b-76196d83537e",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "StartingRuleExecution",
    "clientId": "abf27092886e49a8a5c1922749736453",
    "topicName": "rules/test",
    "ruleName": "JSONLogsRule",
    "ruleAction": "RepublishAction",
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167"
}
```

Además deAtributos comunes CloudWatch de los registros, las entradas de `rule-` registro contienen los siguientes atributos:

clientId

El ID del cliente que realiza la solicitud.

principalId

El ID de la entidad principal que realiza la solicitud.

ruleAction

El nombre de la acción activada.

ruleName

El nombre de la regla que coincide.

topicName

El nombre del tema suscrito.

Entradas del registro de Job

El servicio AWS IoT Job genera entradas de registro para los siguientes eventos. Las entradas de registro se generan cuando se recibe una solicitud de MQTT o HTTP procedente del dispositivo.

Temas

- DescribeJobExecution entrada de registro
- GetPendingJobExecution entrada de registro
- ReportFinalJobExecutionCount entrada de registro
- StartNextPendingJobExecution entrada de registro
- UpdateJobExecution entrada de registro

DescribeJobExecution entrada de registro

El servicio AWS IoT Jobs genera una entrada de registro con un `eventType` de `DescribeJobExecution` cuando el servicio recibe una solicitud para describir la ejecución de un trabajo.

DescribeJobExecution ejemplo de entrada de registro

```
{
    "timestamp": "2017-08-10 19:13:22.841",
    "logLevel": "DEBUG",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "DescribeJobExecution",
    "protocol": "MQTT",
    "clientId": "thingOne",
    "jobId": "002",
    "topicName": "$aws/things/thingOne/jobs/002/get",
    "clientToken": "myToken",
    "details": "The request status is SUCCESS."
}
```

Además deAtributos comunes CloudWatch de los registros, las entradas de `GetJobExecution` registro contienen los siguientes atributos:

clientId

El ID del cliente que realiza la solicitud.

clientToken

Identificador único con distinción entre mayúsculas y minúsculas que permite garantizar la idempotencia de la solicitud. Para obtener más información, consulte How to Ensure Idempotency.

details

Información adicional del servicio Jobs.

jobId

El ID de trabajo para la ejecución de trabajos.

protocol

El protocolo utilizado para realizar la solicitud. Los valores válidos son `MQTT` o `HTTP`.

topicName

El tema utilizado para realizar la solicitud.

GetPendingJobExecution entrada de registro

El servicio AWS IoT Jobs genera una entrada de registro con un `eventType` de `GetPendingJobExecution` cuando el servicio recibe una solicitud de ejecución de un trabajo.

GetPendingJobExecution ejemplo de entrada de registro

```
{
    "timestamp": "2018-06-13 17:45:17.197",
    "logLevel": "DEBUG",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "GetPendingJobExecution",
    "protocol": "MQTT",
    "clientId": "299966ad-54de-40b4-99d3-4fc8b52da0c5",
    "topicName": "$aws/things/299966ad-54de-40b4-99d3-4fc8b52da0c5/jobs/get",
    "clientToken": "24b9a741-15a7-44fc-bd3c-1ff2e34e5e82",
    "details": "The request status is SUCCESS."
}
```

Además de Atributos comunes CloudWatch de los registros, las entradas de `GetPendingJobExecution` registro contienen los siguientes atributos:

clientId

El ID del cliente que realiza la solicitud.

clientToken

Identificador único con distinción entre mayúsculas y minúsculas que permite garantizar la idempotencia de la solicitud. Para obtener más información, consulte How to Ensure Idempotency.

details

Información adicional del servicio Jobs.

protocol

El protocolo utilizado para realizar la solicitud. Los valores válidos son `MQTT` o `HTTP`.

topicName

El nombre del tema suscrito.

ReportFinalJobExecutionCount entrada de registro

El servicio AWS IoT Jobs genera una entrada de registro con una `entryType` de `ReportFinalJobExecutionCount` cuando se completa un trabajo.

ReportFinalJobExecutionCount ejemplo de entrada de registro

```
{
    "timestamp": "2017-08-10 19:44:16.776",
    "logLevel": "INFO",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "ReportFinalJobExecutionCount",
    "jobId": "002",
    "details": "Job 002 completed. QUEUED job execution count: 0 IN_PROGRESS job
 execution count: 0 FAILED job execution count: 0 SUCCEEDED job execution count: 1
 CANCELED job execution count: 0 REJECTED job execution count: 0 REMOVED job execution
 count: 0"
}
```

Además deAtributos comunes CloudWatch de los registros, las entradas de `ReportFinalJobExecutionCount` registro contienen los siguientes atributos:

details

Información adicional del servicio Jobs.

jobId

El ID de trabajo para la ejecución de trabajos.

StartNextPendingJobExecution entrada de registro

Cuando recibe una solicitud para iniciar la siguiente ejecución de un trabajo pendiente, el servicio AWS IoT Jobs genera una entrada de registro con un número `eventType` de`StartNextPendingJobExecution`.

StartNextPendingJobExecution ejemplo de entrada de registro

```
{
    "timestamp": "2018-06-13 17:49:51.036",
    "logLevel": "DEBUG",
```

```
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "StartNextPendingJobExecution",
    "protocol": "MQTT",
    "clientId": "95c47808-b1ca-4794-bc68-a588d6d9216c",
    "topicName": "$aws/things/95c47808-b1ca-4794-bc68-a588d6d9216c/jobs/start-next",
    "clientToken": "bd7447c4-3a05-49f4-8517-dd89b2c68d94",
    "details": "The request status is SUCCESS."
}
```

Además de Atributos comunes CloudWatch de los registros, las entradas de `StartNextPendingJobExecution` registro contienen los siguientes atributos:

clientId

El ID del cliente que realiza la solicitud.

clientToken

Identificador único con distinción entre mayúsculas y minúsculas que permite garantizar la idempotencia de la solicitud. Para obtener más información, consulte How to Ensure Idempotency.

details

Información adicional del servicio Jobs.

protocol

El protocolo utilizado para realizar la solicitud. Los valores válidos son `MQTT` o `HTTP`.

topicName

El tema utilizado para realizar la solicitud.

UpdateJobExecution entrada de registro

El servicio AWS IoT Jobs genera una entrada de registro con un `eventType` de `UpdateJobExecution` cuando el servicio recibe una solicitud para actualizar la ejecución de un trabajo.

UpdateJobExecution ejemplo de entrada de registro

```
{
```

```
    "timestamp": "2017-08-10 19:25:14.758",
    "logLevel": "DEBUG",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "UpdateJobExecution",
    "protocol": "MQTT",
    "clientId": "thingOne",
    "jobId": "002",
    "topicName": "$aws/things/thingOne/jobs/002/update",
    "clientToken": "myClientToken",
    "versionNumber": "1",
    "details": "The destination status is IN_PROGRESS. The request status is SUCCESS."
}
```

Además deAtributos comunes CloudWatch de los registros, las entradas de `UpdateJobExecution` registro contienen los siguientes atributos:

clientId

El ID del cliente que realiza la solicitud.

clientToken

Identificador único con distinción entre mayúsculas y minúsculas que permite garantizar la idempotencia de la solicitud. Para obtener más información, consulte How to Ensure Idempotency.

details

Información adicional del servicio Jobs.

jobId

El ID de trabajo para la ejecución de trabajos.

protocol

El protocolo utilizado para realizar la solicitud. Los valores válidos son `MQTT` o `HTTP`.

topicName

El tema utilizado para realizar la solicitud.

versionNumber

La versión de la ejecución de trabajos.

Entradas de registro de aprovisionamiento de dispositivos

El servicio de aprovisionamiento de AWS IoT dispositivos genera registros para los siguientes eventos.

Temas

- GetDeviceCredentials entrada de registro
- ProvisionDevice entrada de registro

GetDeviceCredentials entrada de registro

El servicio de aprovisionamiento de AWS IoT dispositivos genera una entrada de registro con una `eventType` de las `GetDeviceCredential` llamadas `GetDeviceCredential` de un cliente.

GetDeviceCredentials ejemplo de entrada de registro

```
{
  "timestamp" : "2019-02-20 20:31:22.932",
  "logLevel" : "INFO",
  "traceId" : "8d9c016f-6cc7-441e-8909-7ee3d5563405",
  "accountId" : "123456789101",
  "status" : "Success",
  "eventType" : "GetDeviceCredentials",
  "deviceCertificateId" :
 "e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855",
  "details" : "Additional details about this log."
}
```

Además deAtributos comunes CloudWatch de los registros, las entradas de `GetDeviceCredentials` registro contienen los siguientes atributos:

details

Una breve explicación del error.

deviceCertificateId

El ID del certificado del dispositivo.

ProvisionDevice entrada de registro

El servicio de aprovisionamiento de AWS IoT dispositivos genera una entrada de registro con una `eventType` de las `ProvisionDevice` llamadas `ProvisionDevice` de un cliente.

ProvisionDevice ejemplo de entrada de registro

```
{
  "timestamp" : "2019-02-20 20:31:22.932",
  "logLevel" : "INFO",
  "traceId" : "8d9c016f-6cc7-441e-8909-7ee3d5563405",
  "accountId" : "123456789101",
  "status" : "Success",
  "eventType" : "ProvisionDevice",
  "provisioningTemplateName" : "myTemplate",
  "deviceCertificateId" :
 "e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855",
  "details" : "Additional details about this log."
 }
```

Además de Atributos comunes CloudWatch de los registros, las entradas de `ProvisionDevice` registro contienen los siguientes atributos:

details

Una breve explicación del error.

deviceCertificateId

El ID del certificado del dispositivo.

provisioningTemplateName

El nombre de la plantilla de aprovisionamiento.

Entradas de registro de grupo de objetos dinámicos

AWS IoT Los grupos de cosas dinámicos generan registros para el siguiente evento.

Temas

- AddThingToDynamicThingGroupsFailed entrada de registro

AddThingToDynamicThingGroupsFailed entrada de registro

Cuando no AWS IoT se ha podido añadir algo a los grupos dinámicos especificados, se genera una entrada de registro con un `eventType` de`AddThingToDynamicThingGroupsFailed`. Esto ocurre cuando un objeto cumplía los criterios para estar en el grupo de objetos dinámico, pero no se pudo agregar a este grupo o se eliminó de él. Esto puede suceder por los motivos siguientes:

- El objeto ya es miembro del número máximo de grupos.
- Se utilizó la opción --override-dynamic-groups para agregar el objeto a un grupo de objetos estático. Se eliminó de un grupo de objetos dinámico para hacerlo posible.

Para obtener más información, consulte este artículo sobre las limitaciones y conflictos de los grupos de objetos dinámicos.

AddThingToDynamicThingGroupsFailed ejemplo de entrada de registro

En este ejemplo, se muestra la entrada de registro de un error `AddThingToDynamicThingGroupsFailed`. En este ejemplo, TestThingcumplía los criterios para formar parte de los grupos de elementos dinámicos enumerados en`dynamicThingGroupNames`, pero no se podía añadir a esos grupos dinámicos, tal y como se describe en`reason`.

```
{
 "timestamp": "2020-03-16 22:24:43.804",
 "logLevel": "ERROR",
 "traceId": "70b1f2f5-d95e-f897-9dcc-31e68c3e1a30",
 "accountId": "57EXAMPLE833",
 "status": "Failure",
 "eventType": "AddThingToDynamicThingGroupsFailed",
 "thingName": "TestThing",
 "dynamicThingGroupNames": [
  "DynamicThingGroup11",
  "DynamicThingGroup12",
  "DynamicThingGroup13",
  "DynamicThingGroup14"
 ],
 "reason": "The thing failed to be added to the given dynamic thing group(s) because
 the thing already belongs to the maximum allowed number of groups."
}
```

Además deAtributos comunes CloudWatch de los registros, las entradas de `AddThingToDynamicThingGroupsFailed` registro contienen los siguientes atributos:

dynamicThingGroupNombres

Matriz de los grupos de objetos dinámicos a los que no pudo agregarse el objeto.

razón

Razón por la cual el objeto no pudo agregarse a los grupos dinámicos.

thingName

Nombre del objeto que no pudo agregarse a un grupo de objetos dinámico.

Entradas de registro de indexación de flotas

AWS IoT La indexación de flotas genera entradas de registro para los siguientes eventos.

Temas

- NamedShadowCountForDynamicGroupQueryLimitExceeded entrada de registro

NamedShadowCountForDynamicGroupQueryLimitExceeded entrada de registro

Se procesan un máximo de 25 sombras con nombre por objeto para los términos de consulta que no son específicos del origen de datos en los grupos dinámicos. Cuando se supere este límite para un objeto, se emitirá el tipo de evento `NamedShadowCountForDynamicGroupQueryLimitExceeded`.

NamedShadowCountForDynamicGroupQueryLimitExceeded ejemplo de entrada de registro

En este ejemplo, se muestra la entrada de registro de un error `NamedShadowCountForDynamicGroupQueryLimitExceeded`. En este ejemplo, los resultados `DynamicGroup` basados en todos los valores pueden ser imprecisos, como se describe en el campo `reason`.


```
{
"timestamp": "2020-03-16 22:24:43.804",
"logLevel": "ERROR",
"traceId": "70b1f2f5-d95e-f897-9dcc-31e68c3e1a30",
"accountId": "571032923833",
"status": "Failure",
"eventType": "NamedShadowCountForDynamicGroupQueryLimitExceeded",
"thingName": "TestThing",
```

```
"reason": "A maximum of 25 named shadows per thing are processed for non-data source
 specific query terms in dynamic groups."
}
```

Atributos comunes CloudWatch de los registros

Todas las entradas del registro de CloudWatch registros incluyen los siguientes atributos:

accountId

Tu Cuenta de AWS ID.

eventType

El tipo de evento para el que se generó el registro. El valor del tipo de evento depende del evento que generó la entrada de registro. Cada descripción de entrada de registro incluye el valor de `eventType` para esa entrada de registro.

logLevel

El nivel de registro que se está utilizando. Para obtener más información, consulte the section called “Niveles de registro” .

status

El estado de la solicitud.

timestamp

La marca de tiempo UTC legible para el ser humano de cuando el cliente se conectó al agente de mensajes de AWS IoT .

traceId

Un identificador generado aleatoriamente que puede utilizarse para correlacionar todos los registros para una solicitud específica.

Sube registros del lado del dispositivo a Amazon CloudWatch

Puedes cargar registros históricos del lado del dispositivo en Amazon CloudWatch para monitorear y analizar la actividad de un dispositivo sobre el terreno. Los registros del dispositivo pueden incluir archivos de registros del sistema, de la aplicación y del dispositivo. Este proceso utiliza un parámetro de acción de las reglas de registro para publicar los CloudWatch registros del dispositivo en un grupo de registros definido por el cliente.

Funcionamiento

El proceso comienza cuando un AWS IoT dispositivo envía mensajes MQTT que contienen archivos de registro formateados a un tema. AWS IoT Una AWS IoT regla supervisa el tema del mensaje y envía los archivos de registro al grupo de CloudWatch registros que usted defina. A continuación, puede revisar y analizar la información.

Temas

- Temas de MQTT
- Acción de regla

Temas de MQTT

Elija un espacio de nombres de temas de MQTT que utilizará para publicar los registros. Se recomienda utilizar este formato para el espacio de temas común, `$aws/rules/things/thing_name/logs`, y este formato para los temas de error, `$aws/rules/things/thing_name/logs/errors`. Se recomienda utilizar la estructura de nomenclatura para los registros y los temas de error, pero no es obligatoria. Para obtener más información, consulte Diseño de temas de MQTT para AWS IoT Core.

Al utilizar el espacio de temas común recomendado, se utilizan los temas reservados de AWS IoT Basic Ingest. AWS IoT Basic Ingest envía de forma segura los datos del dispositivo a los AWS servicios compatibles con las acciones de las AWS IoT reglas. Elimina el agente de publish/subscribe mensajes de la ruta de ingestión, lo que lo hace más rentable. Para obtener más información, consulte Reducción de los costos de mensajería con Basic Ingest.

Si usa batchMode para cargar archivos de registro, sus mensajes deben seguir un formato específico que incluya una marca temporal y un mensaje de UNIX. Para obtener más información, consulte el tema sobre los requisitos de formato de los mensajes MQTT para BatchMode en CloudWatch la acción de la regla Logs.

Acción de regla

Cuando AWS IoT recibe los mensajes MQTT de los dispositivos cliente, una AWS IoT regla supervisa el tema definido por el cliente y publica el contenido en un CloudWatch grupo de registros que usted defina. Este proceso utiliza una acción de regla de CloudWatch registros para supervisar el MQTT en busca de lotes de archivos de registro. Para obtener más información, consulte la acción de la AWS IoT regla CloudWatch Registros.

Modo lote

`batchMode`es un parámetro booleano de la acción de la regla de AWS IoT CloudWatch registros. Este parámetro es opcional y está desactivado (`false`) de forma predeterminada. Para cargar archivos de registro del dispositivo en lotes, debes activar este parámetro (`true`) al crear la regla. AWS IoT Para obtener más información, consulta CloudWatch los registros en la sección de acciones de la AWS IoT regla.

Carga de registros del lado del dispositivo mediante reglas AWS IoT

Puedes usar el motor de AWS IoT reglas para cargar registros de archivos de registro existentes del lado del dispositivo (registros del sistema, de la aplicación y del dispositivo-cliente) a Amazon. CloudWatch Cuando los registros del dispositivo se publican en un tema de MQTT, la acción Registrar las reglas transfiere los mensajes a CloudWatch Logs. CloudWatch Este proceso describe cómo cargar los registros de los dispositivos en lotes utilizando el parámetro `batchMode` de acción de reglas activado (establecido en `true`).

Para empezar a cargar los registros del lado del dispositivo, complete los siguientes requisitos previos. CloudWatch

Requisitos previos

Antes de comenzar, haga lo siguiente:

- Crea al menos un dispositivo IoT objetivo que esté registrado AWS IoT Core como una AWS IoT cosa. Para obtener más información, consulte Crear un objeto.
- Determine el espacio de tema de MQTT para la ingesta y los errores. Para obtener más información sobre los temas de MQTT y las convenciones de nomenclatura recomendadas, consulte la sección de temas de Temas de MQTT MQTT en Subir registros del lado del dispositivo a Amazon. CloudWatch

Para obtener más información sobre estos requisitos previos, consulte Cargar registros del lado del dispositivo a. CloudWatch

Crear un grupo de registros CloudWatch

Para crear un grupo de CloudWatch registros, complete los siguientes pasos. Elija la pestaña adecuada en función de si prefiere realizar los pasos con Consola de administración de AWS o con AWS Command Line Interface (AWS CLI).

Consola de administración de AWS

Para crear un grupo de CloudWatch registros mediante el Consola de administración de AWS

1. Abra Consola de administración de AWS y navegue hasta CloudWatch.
2. En la barra de navegación, elija Registros y, luego, Grupos de registros.
3. Elija Crear grupo de registros.
4. Actualice el Nombre del grupo de registro y, si lo desea, actualice los campos de Configuración de retención.
5. Seleccione Crear.

AWS CLI

Para crear un grupo de CloudWatch registros mediante el AWS CLI

1. Para crear el grupo de registro, ejecute el siguiente comando. Para obtener más información, consulte la Referencia `create-log-group` de comandos AWS CLI de la versión 2.

   Sustituya el nombre del grupo de registro del ejemplo (`uploadLogsGroup`) por el que prefiera.

   ```
   aws logs create-log-group --log-group-name uploadLogsGroup
   ```

2. Para confirmar que el grupo de registro se ha creado correctamente, ejecute el siguiente comando.

   ```
   aws logs describe-log-groups --log-group-name-prefix uploadLogsGroup
   ```

   Código de salida de ejemplo:

   ```
   {
       "logGroups": [
           {
               "logGroupName": "uploadLogsGroup",
               "creationTime": 1674521804657,
               "metricFilterCount": 0,
               "arn": "arn:aws:logs:us-east-1:111122223333:log-
   group:uploadLogsGroup:*",
               "storedBytes": 0
   ```

```
        }
    ]
}
```

Creación de una regla del tema

Para crear una AWS IoT regla, complete los siguientes pasos. Elija la pestaña adecuada en función de si prefiere realizar los pasos con Consola de administración de AWS o con AWS Command Line Interface (AWS CLI).

Consola de administración de AWS

Para crear una regla temática mediante el Consola de administración de AWS

1. Abra el Centro de reglas.
   a. Abra la Consola de administración de AWS y vaya a AWS IoT.
   b. En la barra de navegación, seleccione Enrutamiento de mensajes y, a continuación, Reglas.
   c. Seleccione Creación de regla.
2. Introduzca las propiedades de la regla.
   a. Introduzca un Nombre de la regla alfanumérico.
   b. (Opcional) Introduzca una Descripción de regla y Etiquetas.
   c. Elija Siguiente.
3. Introduzca una instrucción SQL.
   a. Introduzca una instrucción SQL utilizando el tema de MQTT que definió para la ingesta.

      Por ejemplo, `SELECT * FROM '$aws/rules/things/thing_name/logs'`
   b. Elija Siguiente.
4. Introduzca las acciones de regla.
   a. En el menú Acción 1, selecciona CloudWatchregistros.
   b. Elija el Nombre del grupo de registro y, a continuación, el grupo de registro que ha creado.
   c. Seleccione Usar el modo de lotes.

d. Especifique el rol de IAM para la regla.

Si dispone de un rol de IAM para la regla, haga lo siguiente.

1. En el menú Rol de IAM, elija el rol de IAM.

Si no dispone de un rol de IAM para la regla, haga lo siguiente.

1. Elija Crear nuevo rol.
2. En Nombre del rol, introduzca un nombre único y elija Crear.
3. Confirme que el nombre del rol de IAM es correcto en el campo Rol de IAM.

e. Elija Siguiente.

5. Revise la configuración de la plantilla.

a. Revise la configuración de la plantilla de trabajo para comprobar que es correcta.

b. Cuando haya terminado, elija Crear.

AWS CLI

Para crear un rol de IAM y una regla temática mediante el AWS CLI

1. Cree un rol de IAM que otorgue derechos a la AWS IoT regla.

a. Cree una política de IAM.

Para crear una política de IAM, ejecute el siguiente comando. Asegúrese de actualizar el valor del parámetro `policy-name`. Para obtener más información, consulte la Referencia `create-policy`de comandos de AWS CLI la versión 2.

> **Note**
>
> Si utiliza un sistema operativo Microsoft Windows, puede que tenga que sustituir el marcador de final de línea (\) por una marca (`) u otro carácter.

```
aws iam create-policy \
    --policy-name uploadLogsPolicy \
    --policy-document \
```

```
'{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "iot:CreateTopicRule",
            "iot:Publish",
            "logs:CreateLogGroup",
            "logs:CreateLogStream",
            "logs:PutLogEvents",
            "logs:GetLogEvents"
        ],
        "Resource": "*"
    }
}'
```

b. Copie el ARN de la política de su salida en un editor de texto.

Código de salida de ejemplo:

```
{
    "Policy": {
        "PolicyName": "uploadLogsPolicy",
        "PermissionsBoundaryUsageCount": 0,
        "CreateDate": "2023-01-23T18:30:10Z",
        "AttachmentCount": 0,
        "IsAttachable": true,
        "PolicyId": "AAABBBCCCDDDEEEFFFGGG",
        "DefaultVersionId": "v1",
        "Path": "/",
        "Arn": "arn:aws:iam::111122223333:policy/uploadLogsPolicy",
        "UpdateDate": "2023-01-23T18:30:10Z"
    }
}
```

c. Cree un rol de IAM y una política de confianza.

Para crear una política de IAM, ejecute el siguiente comando. Asegúrese de actualizar el valor del parámetro `role-name`. Para obtener más información, consulte la Referencia `create-role`de comandos AWS CLI de la versión 2.

```
aws iam create-role \
--role-name uploadLogsRole \
```

```
--assume-role-policy-document \
'{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": "iot.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}'
```

d. Adjunte la política de IAM al rol.

Para crear una política de IAM, ejecute el siguiente comando. Asegúrese de actualizar los valores de los parámetros `role-name` y `policy-arn`. Para obtener más información, consulte la Referencia `attach-role-policy`de comandos AWS CLI de la versión 2.

```
aws iam attach-role-policy \
--role-name uploadLogsRole \
--policy-arn arn:aws:iam::111122223333:policy/uploadLogsPolicy
```

e. Revise el rol.

Para confirmar que el rol de IAM se ha creado correctamente, ejecute el siguiente comando. Asegúrese de actualizar el valor del parámetro `role-name`. Para obtener más información, consulte la Referencia `get-role`de comandos AWS CLI de la versión 2.

```
aws iam get-role --role-name uploadLogsRole
```

Código de salida de ejemplo:

```
{
    "Role": {
        "Path": "/",
        "RoleName": "uploadLogsRole",
```

```
        "RoleId": "AAABBBCCCDDDEEEFFFGGG",
        "Arn": "arn:aws:iam::111122223333:role/uploadLogsRole",
        "CreateDate": "2023-01-23T19:17:15+00:00",
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Sid": "Statement1",
                    "Effect": "Allow",
                    "Principal": {
                        "Service": "iot.amazonaws.com"
                    },
                    "Action": "sts:AssumeRole"
                }
            ]
        },
        "Description": "",
        "MaxSessionDuration": 3600,
        "RoleLastUsed": {}
    }
}
```

2. Cree una regla AWS IoT temática en AWS CLI.

   a. Para crear una regla AWS IoT temática, ejecute el siguiente comando. Asegúrese de actualizar los valores de la `sql` sentencia `--rule-name` `descriptionroleARN` , y de los `logGroupName` parámetros. Para obtener más información, consulte la Referencia create-topic-rulede comandos AWS CLI de la versión 2.

```
aws iot create-topic-rule \
--rule-name uploadLogsRule \
--topic-rule-payload \
 '{
  "sql":"SELECT * FROM 'rules/things/thing_name/logs'",
  "description":"Upload logs test rule",
  "ruleDisabled":false,
  "awsIotSqlVersion":"2016-03-23",
  "actions":[
   {"cloudwatchLogs":
    {"roleArn":"arn:aws:iam::111122223333:role/uploadLogsRole",
    "logGroupName":"uploadLogsGroup",
    "batchMode":true}
    }
```

```
   ]
}'
```

b. Para confirmar que la regla se ha creado correctamente, ejecute el siguiente comando. Asegúrese de actualizar el valor del parámetro `role-name`. Para obtener más información, consulte la Referencia get-topic-rulede comandos AWS CLI de la versión 2.

```
aws iot get-topic-rule --rule-name uploadLogsRule
```

Código de salida de ejemplo:

```
{
    "ruleArn": "arn:aws:iot:us-east-1:111122223333:rule/uploadLogsRule",
    "rule": {
        "ruleName": "uploadLogsRule",
        "sql": "SELECT * FROM rules/things/thing_name/logs",
        "description": "Upload logs test rule",
        "createdAt": "2023-01-24T16:28:15+00:00",
        "actions": [
            {
                "cloudwatchLogs": {
                    "roleArn": "arn:aws:iam::111122223333:role/
uploadLogsRole",
                    "logGroupName": "uploadLogsGroup",
                    "batchMode": true
                }
            }
        ],
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23"
    }
}
```

Enviar registros del lado del dispositivo a AWS IoT

Para enviar registros del lado del dispositivo a AWS IoT

1. Para enviar los registros históricos AWS IoT, comunícate con tus dispositivos para garantizar lo siguiente.

- La información del registro se envía al espacio de nombres del tema correcto, tal como se especifica en la sección Requisitos previos de este procedimiento.

  Por ejemplo, `$aws/rules/things/thing_name/logs`

- La carga del mensaje MQTT está formateada correctamente. Para obtener más información sobre el tema de MQTT y la convención de nomenclatura recomendada, consulta la Temas de MQTT sección correspondiente. Sube registros del lado del dispositivo a Amazon CloudWatch

2. Confirme que los mensajes MQTT se reciben en el cliente AWS IoT MQTT.

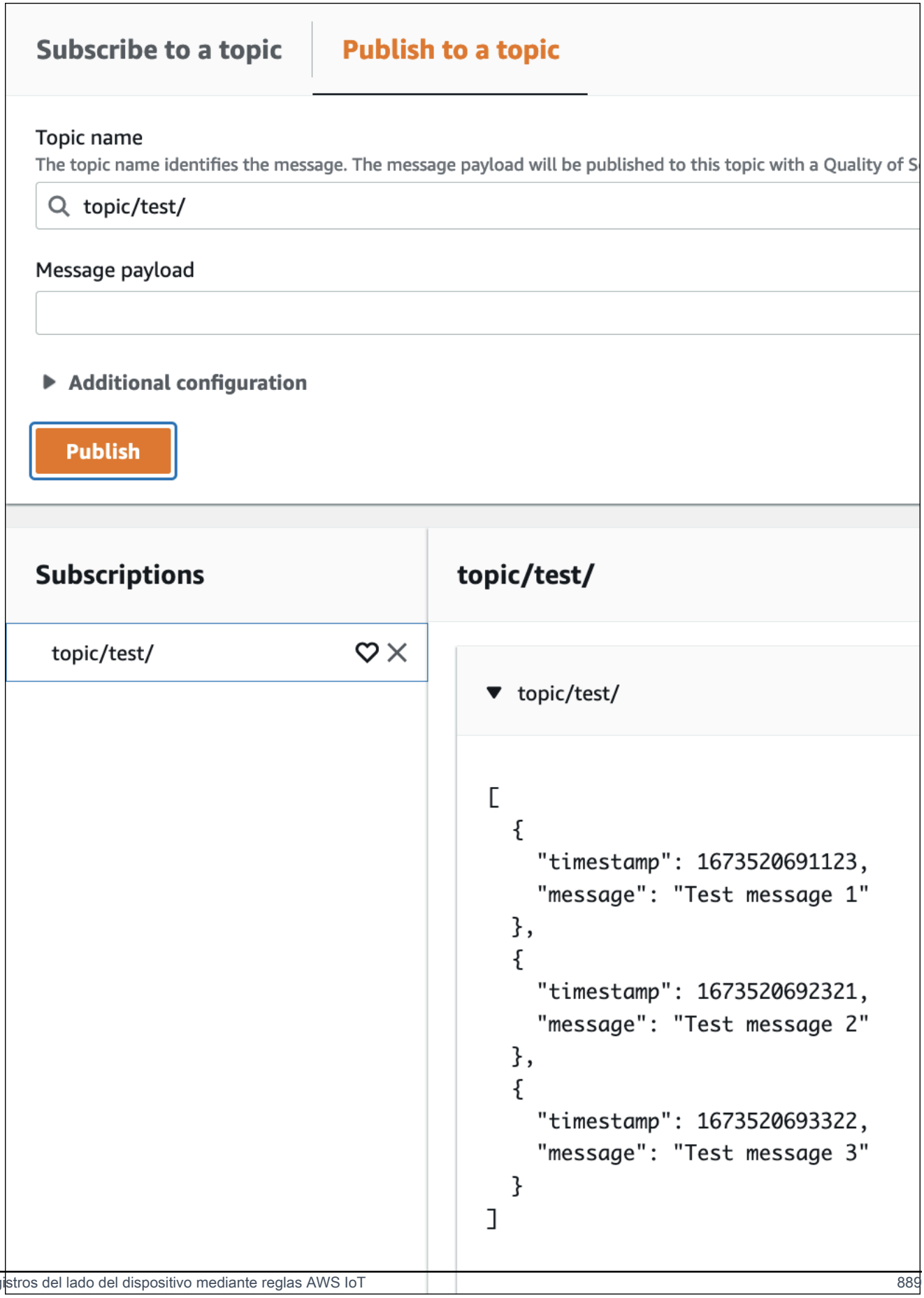
   a. Abre Consola de administración de AWS y navega hasta. AWS IoT

   b. Para ver el cliente de pruebas de MQTT, en la barra de navegación, elija Test, cliente de pruebas de MQTT.

   c. En Suscribirse a un tema, en Filtro de temas, introduzca el espacio de nombres del tema.

   d. Elija Suscribirse.

      Los mensajes MQTT aparecen en la tabla Suscripciones y Tema, como se muestra a continuación. Estos mensajes pueden tardar hasta cinco minutos en aparecer.


Subscribe to a topic
Publish to a topic
Topic name
The topic name identifies the message. The message payload will be published to this topic with a Quality of S
topic/test/
Message payload
Additional configuration
Publish
Subscriptions
topic/test/
topic/test/
topic/test/
[
{
"timestamp": 1673520691123,
"message": "Test message 1"
},
{
"timestamp": 1673520692321,
"message": "Test message 2"
},
{
"timestamp": 1673520693322,
"message": "Test message 3"
}
]

Visualización de los datos de registro

Para revisar tus registros en Logs CloudWatch

1. Abre Consola de administración de AWS y navega hasta CloudWatch.
2. En la barra de navegación, selecciona Logs, Logs Insights.
3. En el menú Seleccionar grupos de registros, elija el grupo de registros que especificó en la AWS IoT regla.
4. En la página Logs Insights, seleccione Ejecutar la consulta.

Registrar las llamadas a la AWS IoT API mediante AWS CloudTrail

AWS IoT está integrado con AWS CloudTrail un servicio que proporciona un registro de las acciones realizadas por un usuario, un rol o un AWS servicio en AWS IoT. CloudTrail captura todas las llamadas a la API AWS IoT como eventos, incluidas las llamadas desde la AWS IoT consola y desde las llamadas de código a AWS IoT APIs. Si crea una ruta, puede habilitar la entrega continua de CloudTrail eventos a un bucket de Amazon S3, incluidos los eventos para AWS IoT. Si no configura una ruta, podrá ver los eventos más recientes en la CloudTrail consola, en el historial de eventos. Con la información recopilada por usted CloudTrail, puede determinar el destinatario de la solicitud AWS IoT, la dirección IP desde la que se realizó la solicitud, quién la realizó, cuándo se realizó y otros detalles.

Para obtener más información CloudTrail, consulta la Guía AWS CloudTrail del usuario.

AWS IoT información en CloudTrail

CloudTrail está habilitada en tu cuenta Cuenta de AWS al crear la cuenta. Cuando se produce una actividad en AWS IoT, esa actividad se registra en un CloudTrail evento junto con otros eventos de AWS servicio en el historial de eventos. Puede ver, buscar y descargar eventos recientes en su Cuenta de AWS. Para obtener más información, consulte Visualización de eventos con el historial de CloudTrail eventos.

Para mantener un registro continuo de eventos en la Cuenta de AWS, incluidos los eventos de AWS IoT, cree un registro de seguimiento. Un rastro permite CloudTrail entregar archivos de registro a un bucket de Amazon S3. De forma predeterminada, cuando crea una ruta en la consola, la ruta se aplica a todos los Región de AWS s. La ruta registra los eventos de todas las Región de AWS s de la AWS partición y entrega los archivos de registro al bucket de Amazon S3 que especifique. Puede

configurar otros AWS servicios para analizar más a fondo los datos de eventos recopilados en los CloudTrail registros y actuar en función de ellos. Para obtener más información, consulte lo siguiente:

- Introducción a la creación de registros de seguimiento
- CloudTrail Servicios e integraciones compatibles
- Configuración de las notificaciones de Amazon SNS para CloudTrail
- Recibir archivos de CloudTrail registro de varias regiones y recibir archivos de CloudTrail registro de varias cuentas

> **Note**
>
> AWS IoT las acciones del plano de datos (del lado del dispositivo) no se registran en CloudTrail. Se utiliza CloudWatch para supervisar estas acciones.

En términos generales, las acciones del plano de AWS IoT control que realizan cambios se registran mediante CloudTrail. Las llamadas como CreateThing«y» UpdateCertificatedejan CloudTrail entradas, mientras que las llamadas «como» ListThingsy «ListTopicRulesno». CreateKeysAndCertificate

Cada entrada de registro o evento contiene información sobre quién generó la solicitud. La información de identidad del usuario lo ayuda a determinar lo siguiente:

- Si la solicitud se realizó con las credenciales 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 AWS servicio.

Para obtener más información, consulte el Elemento userIdentity de CloudTrail .

AWS IoT las acciones se documentan en la referencia AWS IoT de la API. AWS IoT Wireless las acciones se documentan en la Referencia AWS IoT Wireless de la API.

Descripción de las entradas de los archivos de AWS IoT registro

Un rastro es una configuración que permite la entrega de eventos como archivos de registro a un bucket de Amazon S3 que usted especifique. CloudTrail Los archivos de registro contienen una o

más entradas de registro. Un evento representa una solicitud única de cualquier fuente e incluye información sobre la acción solicitada, la fecha y la hora de la acción, los parámetros de la solicitud, etc. CloudTrail Los archivos de registro no son un registro ordenado de las llamadas a la API pública, por lo que no aparecen en ningún orden específico.

En el siguiente ejemplo, se muestra una entrada de CloudTrail registro que demuestra la `AttachPolicy` acción.


```
{
    "timestamp":"1460159496",
    "AdditionalEventData":"",
    "Annotation":"",
    "ApiVersion":"",
    "ErrorCode":"",
    "ErrorMessage":"",
    "EventID":"8bff4fed-c229-4d2d-8264-4ab28a487505",
    "EventName":"AttachPolicy",
    "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":""
}
```

Reglas para AWS IoT

Las reglas permiten que tus dispositivos interactúen con ellas Servicios de AWS. Las reglas se analizan y las acciones se ejecutan en función del flujo de temas MQTT. Puede utilizar reglas para realizar las siguientes tareas:

- Incrementar o filtrar los datos recibidos desde un dispositivo.
- Escribir los datos recibidos de un dispositivo en una base de datos de Amazon DynamoDB.
- Guardar un archivo en Amazon S3.
- Enviar una notificación push a todos los usuarios que utilicen Amazon SNS.
- Publicar datos en una cola de Amazon SQS.
- Invocar una función de Lambda para extraer datos.
- Procese mensajes de un gran número de dispositivos utilizando Amazon Kinesis.
- Envía datos a Amazon OpenSearch Service.
- Captura una CloudWatch métrica.
- Cambia una CloudWatch alarma.
- Envíe los datos de un mensaje MQTT a Amazon SageMaker AI para realizar predicciones basadas en un modelo de aprendizaje automático (ML).
- Enviar un mensaje a un flujo de entrada de Salesforce IoT
- Iniciar un proceso de una máquina de estado de funciones escalonadas.
- Envíe los datos del mensaje a una AWS IoT Events entrada.
- Enviar datos de mensaje de una propiedad de recurso de AWS IoT SiteWise.
- Envíe datos de mensaje a una aplicación o servicio web.

Sus reglas pueden usar mensajes MQTT que pasen por el publish/subscribe the section called “Protocolos de comunicación de dispositivos” protocolo compatible con. También puede utilizar la función de ingesta básica para enviar de forma segura los datos del dispositivo a los Servicios de AWS listados anteriormente, sin incurrir en gastos de mensajería. La función de ingesta básica optimiza el flujo de datos al eliminar el agente de publish/subscribe mensajes de la ruta de ingesta. Esto la hace rentable y, al mismo tiempo, mantiene las funciones de seguridad y procesamiento de datos de. AWS IoT

Antes de AWS IoT poder realizar estas acciones, debes concederle permiso para acceder a tus AWS recursos en tu nombre. Cuando se realicen las acciones, incurrirá en los cargos estándar por las Servicios de AWS que utilice.

Contenido

- Otorgar a una AWS IoT regla el acceso que requiere
- Transmisión de los permisos de rol
- Crear una AWS IoT regla
- Administrar una regla AWS IoT
- AWS IoT acciones de reglas
- Solución de problemas de las reglas
- Acceder a los recursos de varias cuentas mediante reglas AWS IoT
- Control de errores (acción de error)
- Reducción de los costes de mensajería con Basic Ingest
- AWS IoT Referencia de SQL

Otorgar a una AWS IoT regla el acceso que requiere

Utilice las funciones de IAM para controlar los AWS recursos a los que tiene acceso cada regla. Antes de crear una regla, debe crear un rol de IAM con una política que permita el acceso a los recursos necesarios AWS . AWS IoT asume esta función al implementar una regla.

Complete los siguientes pasos para crear la función y la AWS IoT política de IAM que concedan a una AWS IoT regla el acceso que necesita (AWS CLI).

1. Guarde el siguiente documento de política de confianza, que otorga el AWS IoT permiso para asumir el rol, en un archivo denominado`iot-role-trust.json`.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "iot.amazonaws.com"
        },
```

```
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "123456789012"
            },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:iot:us-east-1:123456789012:rule/
rulename"
            }
        }
        }
    ]
}
```

Utilice el comando create-role para crear un rol de IAM especificando 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 un aspecto similar al siguiente.

```
{
 "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. Copie el siguiente JSON en un archivo denominado `my-iot-policy.json`.

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

```
    "Action": "dynamodb:*",
    "Resource": "*"
   }
  ]
}
```

Este JSON es un ejemplo de documento de política que otorga acceso de AWS IoT administrador a DynamoDB.

Use el comando create-policy para conceder AWS IoT acceso a sus AWS recursos al asumir el rol, transfiriendo el archivo: `my-iot-policy.json`

```
aws iam create-policy --policy-name my-iot-policy --policy-document file://my-iot-policy.json
```

Para obtener más información sobre cómo conceder acceso a las políticas Servicios de AWS internas AWS IoT, consulte. Crear una AWS IoT regla

La salida del comando create-policy contendrá el ARN de la política. Asociar la política a un rol.

```
{
 "Policy": {
  "PolicyName": "my-iot-policy",
  "CreateDate": "2015-09-30T19:31:18.620Z",
  "AttachmentCount": 0,
  "IsAttachable": true,
  "PolicyId": "ZXR6A36LTYANPAI7NJ5UV",
  "DefaultVersionId": "v1",
  "Path": "/",
  "Arn": "arn:aws:iam::123456789012:policy/my-iot-policy",
  "UpdateDate": "2015-09-30T19:31:18.620Z"
 }
}
```

3. Utilice el attach-role-policycomando para adjuntar la política a su función:

```
aws iam attach-role-policy --role-name my-iot-role --policy-arn
 "arn:aws:iam::123456789012:policy/my-iot-policy"
```

Revocación del acceso al motor de reglas

Para revocar inmediatamente el acceso al motor de reglas, realice lo siguiente:

1. Eliminar iot.amazonaws.com de la política de confianza
2. Seguir los pasos para revocar las sesiones de roles de iot

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 activa la acción de la regla. El rol debe definirse de la misma manera Cuenta de AWS que la regla.

De hecho, cuando crea o sustituye una regla, está transfiriendo un rol al motor de reglas. El permiso `iam:PassRole` es necesario para realizar esta operación. Para comprobar que dispone de este permiso, cree una política que conceda a `iam:PassRole` el permiso y adjúntela 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. Adjunte esta política a su usuario de IAM o al rol al que pertenece su usuario. Para obtener más información, consulte Uso de políticas administradas.

> **Note**
>
> Las funciones de Lambda utilizan una política basada en recursos. Esta política está directamente asociada a la función de Lambda en sí. Cuando se crea una regla que invoca una función de Lambda, no se pasa un rol, por lo que el usuario que crea la regla no necesita el permiso `iam:PassRole`. Para obtener más información sobre la autorización de funciones de Lambda, consulte Concesión de permisos mediante una política de recursos.

Crear una AWS IoT regla

Puedes crear AWS IoT reglas para enrutar los datos de tus dispositivos conectados para que interactúen con otros AWS servicios. Una AWS IoT regla consta de los siguientes componentes:

Componentes de una regla

Componente	Description (Descripción)	Obligatoria u opcional
Nombre de la regla	El nombre de la regla. No se recomienda utilizar información personal identificable en los nombres de las reglas.	Obligatorio.
Descripción de la regla	Descripción textual de la regla. No se recomienda utilizar información personal identificable en las descripciones de las reglas.	Opcional.
Instrucción de SQL	Sintaxis de SQL simplificada para filtrar los mensajes recibidos sobre un tema de MQTT e insertar los datos en otro punto. Para obtener más información, consulte AWS IoT Referencia de SQL.	Obligatorio.
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. La AWS IoT Core consola establece esta propiedad en de forma `2016-03-23` predeterminada. Si no se	Obligatorio.

Componente	Description (Descripción)	Obligatoria u opcional
	establece esta propiedad, por ejemplo, en un AWS CLI comando o una CloudFormation plantilla , `2015-10-08` se utiliza. Para obtener más información, consulte Versiones de SQL.	
Una o varias acciones	Las acciones se AWS IoT realizan al promulgar 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 de Lambda.	Obligatorio.
Una acción de error	La acción se AWS IoT ejecuta cuando no puede realizar la acción de una regla.	Opcional.

Antes de crear una AWS IoT regla, debe crear un rol de IAM con una política que permita el acceso a los AWS recursos necesarios. AWS IoT asume esta función al implementar una regla. Para obtener más información, consulte Otorgar a una AWS IoT regla el acceso que requiere y Transferir permisos de rol.

Cuando cree una regla, debe tener en cuenta la cantidad de datos que publica en los temas. Si crea reglas que incluyen un patrón de tema comodín, es posible que coincidan con un gran porcentaje de sus mensajes. Si este es el caso, es posible que necesite aumentar la capacidad de los recursos AWS que utilizan las acciones objetivo. Recomendamos evitar los patrones de temas comodín en las reglas de republicación para evitar el procesamiento duplicado y reducir los costos.

> **Note**
>
> La creación y la actualización de reglas son acciones de administrador. Todo usuario que tenga permiso para crear o actualizar reglas podrá tener acceso a los datos procesados por las reglas.

Creación de una regla (consola)

Para crear una regla (Consola de administración de AWS)

Utilice el comando Consola de administración de AWS para crear una regla:

1. Abra la consola de AWS IoT.
2. En el menú de navegación de la izquierda, seleccione Enrutamiento de mensajes en la sección Administrar. A continuación, elija Reglas.
3. En la página Reglas, seleccione Crear una regla.
4. En la página Especificar las propiedades de la regla, escriba un nombre para la regla. Los campos Descripción de la regla y Etiquetas son opcionales. Elija Siguiente.
5. En la página Configurar la instrucción SQL, elija una versión de SQL e introduzca una instrucción SQL. Un ejemplo de instrucción SQL puede ser `SELECT temperature FROM 'iot/topic' WHERE temperature > 50`. Para obtener más información, consulte SQL versions y AWS IoT SQL reference.
6. En la página Adjuntar acciones de regla, añada acciones de regla para enrutar los datos a otros AWS servicios.

    1. En Acciones de la regla, seleccione una acción de la regla de la lista desplegable. Por ejemplo, puede elegir Kinesis Stream. Para obtener más información sobre las acciones de las reglas, consulte AWS IoT rule actions.
    2. En función de la acción de la regla que elija, introduzca los detalles de configuración relacionados. Por ejemplo, si elige Kinesis Stream, tendrá que elegir o crear un recurso de flujo de datos y, si lo desea, introducir detalles de configuración, como Clave de partición, que se utiliza para agrupar los datos por particiones en un flujo.
    3. En el rol de IAM, elija o cree un rol para conceder el AWS IoT acceso a su punto final. Tenga en cuenta que se AWS IoT creará automáticamente una política con el prefijo correspondiente a la función de `aws-iot-rule` IAM que haya seleccionado. Puede seleccionar Ver para ver su rol de IAM y la política desde la consola de IAM. El campo Acción de error es opcional. Encontrará más información en Error handling (error action). Para obtener más información sobre la creación de un rol de IAM para su regla, consulte Grant a rule the access it requires. Elija Siguiente.
7. En la página Revisar y crear, revise toda la configuración y realice las modificaciones necesarias. Seleccione Crear.

Después de crear una regla correctamente, la verá en la página Reglas. Puede seleccionar una regla para abrir la página Detalles, donde puede ver una regla, editarla, desactivarla y eliminarla.

Creación de una regla (CLI)

Para crear una regla (AWS CLI)

Utilice el comando create-topic-rule para crear una regla:

```
aws iot create-topic-rule --rule-name myrule --topic-rule-payload file://myrule.json
```

Ejemplo de archivo de carga con una regla que inserta todos los mensajes enviados al tema `iot/test` en la tabla de DynamoDB especificada. La sentencia SQL filtra los mensajes y el ARN del rol concede AWS IoT permiso para escribir en la tabla de 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 sentencia SQL filtra los mensajes y el ARN del rol concede 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": "amzn-s3-demo-bucket",
    "key": "myS3Key"
   }
  }
 ]
}
```

El siguiente es un ejemplo de archivo de carga útil con una regla que envía datos a Amazon OpenSearch Service:

```
{
 "sql": "SELECT *, timestamp() as timestamp FROM 'iot/test'",
 "ruleDisabled": false,
 "awsIotSqlVersion": "2016-03-23",
 "actions": [
  {
   "OpenSearch": {
    "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 de 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 útil con una regla que inserta datos en un flujo de Amazon Kinesis Data 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"
   }
  }
 ]
}
```

A continuación se muestra un ejemplo de archivo de carga útil con una regla que utiliza la `machinelearning_predict` función Amazon SageMaker AI para volver a publicar en un tema si los datos de la carga útil de MQTT están clasificados 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": [
  {
   "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"
   }
  }
 ]
}
```

Ejemplo de archivo de carga con una regla que inicia la ejecución de una máquina de estado de Step Functions.

```
{
 "sql": "expression",
 "ruleDisabled": false,
 "awsIotSqlVersion": "2016-03-23",
 "actions": [
  {
   "stepFunctions": {
    "stateMachineName": "myCoolStateMachine",
    "executionNamePrefix": "coolRunning",
    "roleArn": "arn:aws:iam::123456789012:role/my-iot-role"
   }
  }
 ]
}
```

Administrar una regla AWS IoT

Puede utilizar las siguientes acciones para gestionar AWS IoT las reglas.

En este tema:

- Etiquetado de una regla
- Visualización de una regla
- Eliminación de una regla

Etiquetado de una regla

Para añadir otro nivel de especificidad a sus reglas nuevas o existentes, puede aplicar el etiquetado. El etiquetado aprovecha los pares clave-valor de sus reglas para proporcionarle un mayor control

sobre cómo y dónde se aplican las reglas a sus AWS IoT recursos y servicios. Por ejemplo, puede limitar el alcance de la regla para que solo se aplique en su entorno beta para las pruebas previas al lanzamiento (`Key=environment, Value=beta`) o para capturar todos los mensajes enviados al `iot/test` tema únicamente desde un punto de conexión específico y almacenarlos en un bucket de Amazon S3.

Ejemplo de política de IAM

Para ver un ejemplo que muestre cómo conceder permisos de etiquetado a una regla, piense en un usuario que ejecute el siguiente comando para crear una regla y etiquetarla para aplicarla únicamente a su entorno beta.

En el ejemplo, sustituya:

- *`MyTopicRuleName`*con el nombre de la regla.
- *`myrule.json`*con el nombre del documento de política.

```
aws iot create-topic-rule
    --rule-name MyTopicRuleName
    --topic-rule-payload file://myrule.json
    --tags "environment=beta"
```

Para este ejemplo, debe utilizar la siguiente política de IAM:

```
{
    "Version":"2012-10-17",
    "Statement":
    {
        "Action": [ "iot:CreateTopicRule", "iot:TagResource" ],
        "Effect": "Allow",
        "Resource": [
            "arn:aws:iot:us-east-1:123456789012:rule/MyTopicRuleName"
        ]
    }
}
```

El ejemplo anterior muestra una regla recién creada llamada `MyTopicRuleName` que solo se aplica a su entorno beta. El `iot:TagResource` en la declaración de política con `MyTopicRuleName`

específicamente llamado permite el etiquetado al crear o actualizar `MyTopicRuleName`. El parámetro `--tags "environment=beta"` utilizado al crear la regla limita el alcance de `MyTopicRuleName` únicamente a su entorno beta. Si elimina el parámetro`--tags "environment=beta"`, `MyTopicRuleName` se aplicará a todos los entornos.

Para obtener más información sobre la creación de funciones y políticas de IAM específicas para una regla AWS IoT , consulte Otorgar a una AWS IoT regla el acceso que requiere

Para obtener información general sobre el etiquetado de recursos, consulte Etiquetar sus recursos AWS IoT.

Visualización de una regla

Usa el list-topic-rulescomando para enumerar tus reglas:

```
aws iot list-topic-rules
```

Usa el get-topic-rulecomando para obtener información sobre una regla:

```
aws iot get-topic-rule --rule-name myrule
```

Eliminación de una regla

Cuando ya no necesite una regla, puede eliminarla.

Para eliminar una regla (AWS CLI)

Usa el delete-topic-rulecomando para eliminar una regla:

```
aws iot delete-topic-rule --rule-name myrule
```

AWS IoT acciones de reglas

AWS IoT las acciones de regla especifican qué hacer cuando se invoca una regla. Puede definir acciones para enviar datos a una base de datos de Amazon DynamoDB, enviar datos a Amazon Kinesis Data Streams, AWS Lambda invocar una función, etc. AWS IoT admite las siguientes acciones Regiones de AWS cuando el servicio de la acción está disponible.

Acción de regla	Description (Descripción)	Nombre en la API
Apache Kafka	Envía un mensaje a un clúster de Apache Kafka.	`kafka`
CloudWatch alarmas	Cambia el estado de una CloudWatch alarma de Amazon.	`cloudwatchAlarm`
CloudWatch Registros	Envía un mensaje a Amazon CloudWatch Logs.	`cloudwatchLogs`
CloudWatch métricas	Envía un mensaje a una CloudWatch métrica.	`cloudwatchMetric`
DynamoDB	Envía un mensaje a una tabla DynamoDB.	`dynamoDB`
Dinamo DBv2	Envía los datos de los mensajes a varias columnas de una tabla de DynamoDB.	`dynamoDBv2`
Elasticsearch	Envía un mensaje a un OpenSearch punto final.	`OpenSearch`
HTTP	Publica un mensaje en un punto de conexión HTTPS.	`http`
AWS IoT Events	Envía un mensaje a una AWS IoT Events entrada.	`iotEvents`
AWS IoT SiteWise	Envía los datos del mensaje a las propiedades AWS IoT SiteWise de los activos.	`iotSiteWise`
Firehose	Envía un mensaje a un flujo de entrega de Firehose.	`firehose`

Acción de regla	Description (Descripción)	Nombre en la API
Kinesis Data Streams	Envía un mensaje a un flujo de datos Kinesis.	`kinesis`
Lambda	Invoca una función de Lambda con datos de mensaje como entrada.	`lambda`
Ubicación	Envía los datos de ubicación a Amazon Location Service.	`location`
OpenSearch	Envía un mensaje a un punto final OpenSearch de Amazon Service.	`OpenSearch`
Volver a publicar	Vuelve a publicar un mensaje en otro tema MQTT.	`republish`
S3	Almacena un mensaje en un bucket de Amazon Simple Storage Service (Amazon S3).	`s3`
IoT de Salesforce	Envía un mensaje a un flujo de entrada de Salesforce IoT.	`salesforce`
SNS	Publica un mensaje como notificación push de Amazon Simple Notification Service (Amazon SNS).	`sns`
SQS	Envía un mensaje a una cola de Amazon Simple Queue Service (Amazon SQS).	`sqs`
Step Functions	Inicia una máquina de AWS Step Functions estados.	`stepFunctions`

Acción de regla	Description (Descripción)	Nombre en la API
the section called “Timestream”	Envía un mensaje a una tabla de base de datos de Amazon Timestream.	`timestream`

Notas

- Defina la regla al Región de AWS igual que el recurso de otro servicio para que la acción de la regla pueda interactuar con ese recurso.
- El motor de AWS IoT reglas puede realizar varios intentos para realizar una acción si se producen errores intermitentes. Si todos los intentos fallan, el mensaje se descarta y el error aparece en tus CloudWatch registros. Es posible especificar una acción de error para cada regla que se invoca cuando se produce un error. Para obtener más información, consulte Control de errores (acción de error).
- Algunas acciones de regla desencadenan acciones en servicios que se integran con AWS Key Management Service (AWS KMS) para admitir el cifrado de datos en reposo. Si utilizas una clave KMS administrada por el cliente AWS KMS key para cifrar los datos en reposo, el servicio debe tener permiso para usar la clave KMS en nombre de la persona que llama. Para saber cómo gestionar los permisos de su clave KMS gestionada por el cliente, consulte los temas de cifrado de datos en la guía de servicio correspondiente. Para obtener más información sobre las claves KMS gestionadas por el cliente, consulte los conceptos de AWS Key Management Service en la Guía del desarrollador de AWS Key Management Service .

  Puede utilizar cualquier función o plantilla de sustitución en la sentencia SQL de una acción de error, incluidas las funciones externas: `aws_lambda()`,, `get_dynamodb()`, `get_registry_data()get_thing_shadow()`, `get_secret()`y. `machinelearning_predict()decode()` Si una acción de error requiere llamar a una función externa, la invocación de la acción de error puede generar una factura adicional por la función externa.

Apache Kafka

La acción de Apache Kafka (Kafka) envía mensajes directamente a Amazon Managed Streaming para Apache Kafka (Amazon MSK), a clústeres de Apache Kafka autoadministrados por proveedores externos, como Confluent Cloud o a clústeres de Apache Kafka autoadministrados. Con la acción de reglas de Kafka, puede enrutar sus datos de IoT a los clústeres de Kafka. Esto le permite crear canalizaciones de datos de alto rendimiento para diversos fines, como el análisis de flujos, la integración de datos, la visualización y las aplicaciones empresariales esenciales.

> **Note**
>
> En este tema se presupone estar familiarizado con la plataforma Apache Kafka y los conceptos relacionados. Para obtener más información sobre Apache Kafka, consulte Apache Kafka. No se admite MSK sin servidor. Los clústeres de MSK sin servidor solo se pueden crear mediante la autenticación de IAM, que la acción de la regla de Apache Kafka no admite actualmente. Para obtener más información sobre cómo configurar AWS IoT Core con Confluent, consulte Aprovechar Confluent y AWS resolver los desafíos de la administración de datos y dispositivos de IoT.

Requisitos

Esta regla tiene los siguientes requisitos:

- Una función de IAM que AWS IoT puede asumir para realizar las `ec2:CreateNetworkInterface` operaciones`ec2:DescribeNetworkInterfaces`,,`ec2:CreateNetworkInterfacePermission`, `ec2:DeleteNetworkInterface` `ec2:DescribeSubnets`, `ec2:DescribeVpcs`, `ec2:DescribeVpcAttribute` y. `ec2:DescribeSecurityGroups` Este rol crea y administra interfaces de red elásticas para su Amazon Virtual Private Cloud para comunicarse con su agente de Kafka. Para obtener más información, consulte Otorgar a una AWS IoT regla el acceso que requiere.

  En la AWS IoT consola, puede elegir o crear un rol que permita AWS IoT Core realizar esta acción de regla.

  A fin de obtener más información sobre las interfaces de red, consulte Interfaces de red elástica en la Guía del usuario de Amazon EC2.

La política asociada al rol especificado debería verse de manera similar al siguiente ejemplo.

```
{
    "Version":"2012-10-17",
    "Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "ec2:CreateNetworkInterface",
            "ec2:DescribeNetworkInterfaces",
            "ec2:CreateNetworkInterfacePermission",
            "ec2:DeleteNetworkInterface",
            "ec2:DescribeSubnets",
            "ec2:DescribeVpcs",
            "ec2:DescribeVpcAttribute",
            "ec2:DescribeSecurityGroups"
            ],
            "Resource": "*"
        }
    ]
}
```

- Si lo utiliza AWS Secrets Manager para almacenar las credenciales necesarias para conectarse a su agente de Kafka, debe crear un rol de IAM que AWS IoT Core pueda asumir para realizar las operaciones `secretsmanager:GetSecretValue` y`secretsmanager:DescribeSecret`.

  La política asociada al rol especificado debería verse de manera similar al siguiente ejemplo.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue",
                "secretsmanager:DescribeSecret"
            ],
            "Resource": [
                "arn:aws:secretsmanager:us-
east-1:123456789012:secret:kafka_client_truststore-*",
                "arn:aws:secretsmanager:us-
east-1:123456789012:secret:kafka_keytab-*"
```

```
            ]
        }
    ]
}
```

- Puede ejecutar sus clústeres de Apache Kafka dentro de Amazon Virtual Private Cloud (Amazon VPC). Debe crear un destino de Amazon VPC y utilizar una puerta de enlace NAT en las subredes para reenviar los mensajes desde un clúster público de AWS IoT Kafka. El motor de reglas AWS IoT crea una interfaz de red en cada una de las subredes enumeradas en el destino de la VPC para enrutar el tráfico directamente a la VPC. Al crear un destino de VPC, el motor de AWS IoT reglas crea automáticamente una acción de regla de VPC. Para obtener más información sobre las acciones de las reglas, consulte Destinos de nube privada virtual (VPC).
- Si utilizas una clave KMS gestionada AWS KMS key por el cliente para cifrar los datos en reposo, el servicio debe tener permiso para utilizar la clave KMS en nombre de la persona que llama. Para obtener más información, consulte el cifrado de Amazon MSK en la guía para desarrolladores Amazon Managed Streaming for Apache Kafka..

Parameters

Al crear una AWS IoT regla con esta acción, debe especificar la siguiente información:

destinationArn

El nombre de recurso de Amazon (ARN) de la VPC de destino. Para obtener información sobre la creación de una VPC de destino, consulte Destinos de nube privada virtual (VPC).

tema

Tema de Kafka para que los mensajes se envíen al agente de Kafka.

Puede sustituir este campo mediante una plantilla de sustitución. Para obtener más información, consulte the section called “Plantillas de sustitución”.

clave (opcional)

Clave de mensajes de Kafka.

Puede sustituir este campo mediante una plantilla de sustitución. Para obtener más información, consulte the section called “Plantillas de sustitución”.

encabezados (opcional)

La lista de cabeceras de Kafka que usted especifique. Cada encabezado es un par clave-valor que puede especificar al crear una acción de Kafka. Puede usar estos encabezados para enrutar los datos de los clientes de IoT a los clústeres de Kafka descendentes sin modificar la carga útil de los mensajes.

Puede sustituir este campo mediante una plantilla de sustitución. Para saber cómo pasar una función de regla en línea como plantilla de sustitución en el encabezado de la acción Kafka, consulte los ejemplos. Para obtener más información, consulte the section called “Plantillas de sustitución”.

> **Note**
>
> Los encabezados en formato binario no son compatibles.

partición (opcional)

Partición de mensajes de Kafka.

Puede sustituir este campo mediante una plantilla de sustitución. Para obtener más información, consulte the section called “Plantillas de sustitución”.

clientProperties

Un objeto que define las propiedades del cliente productor de Apache Kafka.

acks (opcional)

El número de reconocimientos que el productor requiere que el servidor haya recibido antes de considerar completa una solicitud.

Si especifica 0 como valor, el productor no esperará ningún acuse de recibo del servidor. Si el servidor no recibe el mensaje, el productor no volverá a intentar enviarlo.

Valores válidos: `-1`, `0`, `1`, `all`. El valor predeterminado es `1`.

bootstrap.servers

Una lista de pares de host y puerto (por ejemplo `host1:port1`, `host2:port2`) que se utilizan para establecer la conexión inicial con el clúster de Kafka.

compression.type (opcional)

El tipo de compresión de todos los datos generados por el productor.

Valores válidos: `none`, `gzip`, `snappy`, `lz4`, `zstd`. El valor predeterminado es `none`.

security.protocol

El protocolo de seguridad utilizado para conectarse a su agente de Kafka.

Valores válidos: `SSL`, `SASL_SSL`. El valor predeterminado es `SSL`.

key.serializer

Especifica cómo convertir los objetos clave que usted proporciona con el `ProducerRecord` en bytes.

Valor válido: `StringSerializer`.

value.serializer

Especifica cómo convertir los objetos de valor que proporcione con el `ProducerRecord` en bytes.

Valor válido: `ByteBufferSerializer`.

ssl.truststore

El archivo truststore en formato base64 o la ubicación del archivo truststore en AWS Secrets Manager Este valor no es necesario si su almacén de confianza cuenta con la confianza de las autoridades de certificación (CA) de Amazon.

Este campo admite plantillas de sustitución. Si usa Secrets Manager para almacenar las credenciales necesarias para conectarse a su agente de Kafka, puede usar la función SQL `get_secret` para recuperar el valor de este campo. Para obtener más información sobre las plantillas de sustitución, consulte the section called “Plantillas de sustitución”. Para obtener más información sobre la función `get_secret`, consulte the section called “get_secret(secretId, secretType, key, roleArn)”. Si el almacén de confianza tiene la forma de un archivo, utilice el parámetro `SecretBinary`. Si el almacén de confianza tiene la forma de una cadena, utilice el parámetro `SecretString`.

El valor máximo de este recuento es 65 KB.

ssl.truststore.password

La contraseña del almacén de confianza. Este valor solo es obligatorio si ha creado una contraseña para el almacén de confianza.

ssl.keystore

El archivo del almacén de claves. Este valor es obligatorio cuando se especifica SSL como valor para `security.protocol`.

Este campo admite plantillas de sustitución. Utilice Secrets Manager para almacenar las credenciales necesarias para conectarse a su agente de Kafka. Para recuperar el valor de este campo, utilice la función SQL `get_secret`. Para obtener más información sobre las plantillas de sustitución, consulte the section called “Plantillas de sustitución”. Para obtener más información sobre la función `get_secret`, consulte the section called “get_secret(secretId, secretType, key, roleArn)”. Utilice el parámetro `SecretBinary`.

ssl.keystore.password

La contraseña del almacén para el archivo keystore. Este valor es necesario si especifica un valor para `ssl.keystore`.

El valor de este campo puede ser texto sin formato. Este campo también admite plantillas de sustitución. Utilice Secrets Manager para almacenar las credenciales necesarias para conectarse a su agente de Kafka. Para recuperar el valor de este campo, utilice la función SQL `get_secret`. Para obtener más información sobre las plantillas de sustitución, consulte the section called “Plantillas de sustitución”. Para obtener más información sobre la función `get_secret`, consulte the section called “get_secret(secretId, secretType, key, roleArn)”. Utilice el parámetro `SecretString`.

ssl.key.password

La contraseña de la clave privada del archivo del almacén de claves.

Este campo admite plantillas de sustitución. Utilice Secrets Manager para almacenar las credenciales necesarias para conectarse a su agente de Kafka. Para recuperar el valor de este campo, utilice la función SQL `get_secret`. Para obtener más información sobre las plantillas de sustitución, consulte the section called “Plantillas de sustitución”. Para obtener más información sobre la función `get_secret`, consulte the section called “get_secret(secretId, secretType, key, roleArn)”. Utilice el parámetro `SecretString`.

sasl.mechanism

El mecanismo de seguridad utilizado para conectarse a su agente de Kafka. Este valor es obligatorio cuando se especifica `SASL_SSL` para `security.protocol`.

Valores válidos: `PLAIN`, `SCRAM-SHA-512`, `GSSAPI`.

> **Note**
>
> `SCRAM-SHA-512`es el único mecanismo de seguridad compatible en las regiones cn-north-1, cn-northwest-1, -1 y -1. us-gov-east us-gov-west

sasl.plain.username

El nombre de usuario utilizado para recuperar la cadena secreta de Secrets Manager. Este valor es obligatorio cuando se especifica `SASL_SSL` para `security.protocol` y `PLAIN` para `sasl.mechanism`.

sasl.plain.password

La contraseña utilizada para recuperar la cadena secreta de Secrets Manager. Este valor es obligatorio cuando se especifica `SASL_SSL` para `security.protocol` y `PLAIN` para `sasl.mechanism`.

sasl.scram.username

El nombre de usuario utilizado para recuperar la cadena secreta de Secrets Manager. Este valor es obligatorio cuando se especifica `SASL_SSL` para `security.protocol` y `SCRAM-SHA-512` para `sasl.mechanism`.

sasl.scram.password

La contraseña utilizada para recuperar la cadena secreta de Secrets Manager. Este valor es obligatorio cuando se especifica `SASL_SSL` para `security.protocol` y `SCRAM-SHA-512` para `sasl.mechanism`.

sasl.kerberos.keytab

El archivo keytab para la autenticación de Kerberos en Secrets Manager. Este valor es obligatorio cuando se especifica `SASL_SSL` para `security.protocol` y `GSSAPI` para `sasl.mechanism`.

Este campo admite plantillas de sustitución. Utilice Secrets Manager para almacenar las credenciales necesarias para conectarse a su agente de Kafka. Para recuperar el valor de este campo, utilice la función SQL `get_secret`. Para obtener más información sobre las plantillas de sustitución, consulte the section called “Plantillas de sustitución”. Para obtener más información sobre la función `get_secret`, consulte the section called “get_secret(secretId, secretType, key, roleArn)”. Utilice el parámetro `SecretBinary`.

sasl.kerberos.service.name

El nombre principal de Kerberos con el que se ejecuta Apache Kafka. Este valor es obligatorio cuando se especifica `SASL_SSL` para `security.protocol` y `GSSAPI` para `sasl.mechanism`.

sasl.kerberos.krb5.kdc

El nombre de host del centro de distribución de claves (KDC) al que se conecta su cliente productor de Apache Kafka. Este valor es obligatorio cuando se especifica `SASL_SSL` para `security.protocol` y `GSSAPI` para `sasl.mechanism`.

sasl.kerberos.krb5.realm

El ámbito al que se conecta su cliente productor de Apache Kafka. Este valor es obligatorio cuando se especifica `SASL_SSL` para `security.protocol` y `GSSAPI` para `sasl.mechanism`.

sasl.kerberos.principal

La identidad única de Kerberos a la que Kerberos puede asignar tickets para acceder a los servicios compatibles con Kerberos. Este valor es obligatorio cuando se especifica `SASL_SSL` para `security.protocol` y `GSSAPI` para `sasl.mechanism`.

Ejemplos

El siguiente ejemplo de JSON define una acción de Apache Kafka en una regla. AWS IoT El siguiente ejemplo pasa la función en línea sourceIP () como plantilla de sustitución en el encabezado Kafka Action.

```
{
 "topicRulePayload": {
  "sql": "SELECT * FROM 'some/topic'",
  "ruleDisabled": false,
  "awsIotSqlVersion": "2016-03-23",
  "actions": [
```

```
   {
    "kafka": {
     "destinationArn": "arn:aws:iot:region:123456789012:ruledestination/vpc/
VPCDestinationARN",
     "topic": "TopicName",
     "clientProperties": {
      "bootstrap.servers": "kafka.com:9092",
      "security.protocol": "SASL_SSL",
      "ssl.truststore": "${get_secret('kafka_client_truststore',
 'SecretBinary','arn:aws:iam::123456789012:role/kafka-get-secret-role-name')}",
      "ssl.truststore.password": "kafka password",
      "sasl.mechanism": "GSSAPI",
      "sasl.kerberos.service.name": "kafka",
      "sasl.kerberos.krb5.kdc": "kerberosdns.com",
      "sasl.kerberos.keytab": "${get_secret('kafka_keytab','SecretBinary',
 'arn:aws:iam::123456789012:role/kafka-get-secret-role-name')}",
      "sasl.kerberos.krb5.realm": "KERBEROSREALM",
      "sasl.kerberos.principal": "kafka-keytab/kafka-keytab.com"
     },
     "headers": [
      {
       "key": "static_header_key",
       "value": "static_header_value"
      },
      {
       "key": "substitutable_header_key",
       "value": "${value_from_payload}"
      },
      {
       "key": "source_ip",
       "value": "${sourceIp()}"
      }
     ]
    }
   }
  ]
 }
}
```

Notas importantes sobre la configuración de Kerberos

- Su centro de distribución de claves (KDC) debe poder resolverse mediante un sistema de nombres de dominio (DNS) privado dentro de la VPC de destino. Un enfoque posible consiste en añadir

la entrada DNS del KDC a una zona alojada privada. Para más información sobre este enfoque, consulte Trabajar con zonas alojadas privadas.

- Cada VPC debe tener habilitada la resolución DNS. Para obtener más información, consulte Utilización de DNS con su VPC.
- Los grupos de seguridad de la interfaz de red y los grupos de seguridad a nivel de instancia del destino de la VPC deben permitir el tráfico desde dentro de la VPC en los siguientes puertos.
  - El tráfico TCP en el puerto oyente intermediario bootstrap (normalmente 9092, pero debe estar dentro del rango de 9000 a 9100)
  - Tráfico TCP y UDP en el puerto 88 del KDC
- `SCRAM-SHA-512`es el único mecanismo de seguridad compatible en las regiones cn-north-1, cn-northwest-1, -1 y -1. us-gov-east us-gov-west

Destinos de nube privada virtual (VPC)

La acción de regla de Apache Kafka enruta los datos a un clúster de Apache Kafka en una Amazon Virtual Private Cloud (Amazon VPC). La configuración de la VPC utilizada por la acción de regla de Apache Kafka se habilita automáticamente cuando se especifica el destino de la VPC para la acción de regla.

Un destino de VPC contiene una lista de subredes dentro de la VPC. El motor de reglas crea una interfaz de red elástica en cada subred especificada en esta lista. A fin de obtener más información sobre las interfaces de red, consulte Interfaces de red elástica en la Guía del usuario de Amazon EC2.

Requisitos y consideraciones

- Si utiliza un clúster de Apache Kafka autogestionado al que se accederá mediante un punto de conexión público a través de Internet:
  - Cree una puerta de enlace NAT para las instancias de sus subredes. La puerta de enlace NAT tiene una dirección IP pública que puede conectarse a Internet, lo que permite al motor de reglas reenviar sus mensajes al clúster público de Kafka.
  - Asigne una dirección IP elástica con las interfaces de red elásticas (ENIs) que crea el destino de la VPC. Los grupos de seguridad que utilice deben estar configurados para bloquear el tráfico entrante.

> **Note**
>
> Si el destino de la VPC está deshabilitado y, a continuación, se vuelve a habilitar, debe volver a asociar el elástico IPs al nuevo. ENIs

- Si el destino de una regla temática de VPC no recibe tráfico durante 30 días seguidos, se deshabilitará.
- Si algún recurso utilizado por el destino de la VPC cambia, el destino se deshabilitará y no se podrá utilizar.
- Algunos cambios que pueden deshabilitar un destino de VPC incluyen:
  - Eliminar la VPC, las subredes, los grupos de seguridad o el rol utilizado.
  - Modificar el rol para que deje de tener los permisos necesarios.
  - Alcanzar casi la capacidad de la subred, lo que nos impide aplicar los parches de FedRAMP.
  - Deshabilitar el destino.

Precios

A efectos de fijación de precios, se mide una acción de regla de VPC además de la acción que envía un mensaje a un recurso cuando el recurso está en su VPC. Para obtener información sobre precios, consulte Precios de AWS IoT Core.

Creación de destinos de reglas temáticas de nube privada virtual (VPC)

Puede crear un destino de nube privada virtual (VPC) mediante la CreateTopicRuleDestinationAPI o la AWS IoT Core consola.

Cuando cree un destino de VPC, debe especificar la siguiente información.

vpcId

El ID único del destino de VPC.

subnetIds

Una lista de subredes en las que el motor de reglas crea interfaces de red elásticas. El motor de reglas asigna una única interfaz de red para cada subred de la lista.

securityGroups (opcional)

Lista de grupos de seguridad que se aplicarán a las interfaces de red.

roleArn

El nombre de recurso de Amazon (ARN) de un rol que tiene permiso para crear interfaces de red en su nombre.

Este ARN debería tener asociada una política similar al siguiente ejemplo.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeVpcs",
                "ec2:DeleteNetworkInterface",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcAttribute",
                "ec2:DescribeSecurityGroups"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "ec2:CreateNetworkInterfacePermission",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "ec2:ResourceTag/VPCDestinationENI": "true"
            }
        }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "ec2:CreateAction": "CreateNetworkInterface",
```

```
                    "aws:RequestTag/VPCDestinationENI": "true"
                }
            }
            }
    ]
}
```

Crear un destino de VPC mediante AWS CLI

El siguiente ejemplo muestra cómo crear un destino de VPC utilizando AWS CLI.

```
aws --region regions iot create-topic-rule-destination --destination-configuration
 'vpcConfiguration={subnetIds=["subnet-
123456789101230456"],securityGroups=[],vpcId="vpc-
123456789101230456",roleArn="arn:aws:iam::123456789012:role/role-name"}'
```

Cuando ejecuta este comando, el estado de destino de VPC será `IN_PROGRESS`. Transcurridos unos minutos, su estado cambiará a `ERROR` (si el comando no se ejecuta correctamente) o `ENABLED`. Cuando el estado de destino es `ENABLED`, está lista para su uso.

Puede utilizar el siguiente comando para obtener el estado del destino de la VPC.

```
aws --region region iot get-topic-rule-destination --arn "VPCDestinationARN"
```

Crear un destino de VPC mediante la consola AWS IoT Core

Los siguientes pasos describen cómo crear un destino de VPC mediante la AWS IoT Core consola.

1. Navegue hasta la AWS IoT Core consola. En el panel izquierdo, en la pestaña Actuar, seleccione Destinos.
2. Escriba valores en los siguientes campos:
   - ID de VPC
   - Subred IDs

- Security Group

3. Seleccione un rol que tenga los permisos necesarios para crear interfaces de red. El ejemplo anterior de política contiene estos permisos.

Cuando el estado del destino de la VPC es HABILITADO, está listo para su uso.

CloudWatch alarmas

La acción CloudWatch alarm (`cloudWatchAlarm`) cambia el estado de una CloudWatch alarma de Amazon. Puede especificar el motivo del cambio de estado y el valor de esta llamada.

Requisitos

Esta regla tiene los siguientes requisitos:

- Una función de IAM que AWS IoT puede asumir para realizar la `cloudwatch:SetAlarmState` operación. Para obtener más información, consulte Otorgar a una AWS IoT regla el acceso que requiere.

  En la AWS IoT consola, puede elegir o crear un rol que permita AWS IoT realizar esta acción de regla.

Parameters

Al crear una AWS IoT regla con esta acción, debe especificar la siguiente información:

`alarmName`

El nombre CloudWatch de la alarma.

Soporta plantillas de sustitución: API y AWS CLI únicamente

`stateReason`

El motivo del cambio de alarma.

Admite plantillas de sustitución: Sí

`stateValue`

El valor del estado de alarma. Valores válidos: `OK`, `ALARM`, `INSUFFICIENT_DATA`.

Admite plantillas de sustitución: Sí

`roleArn`

La función de IAM que permite el acceso a la CloudWatch alarma. Para obtener más información, consulte Requisitos.

Admite plantillas de sustitución: No

Ejemplos

El siguiente ejemplo de JSON define una acción CloudWatch de alarma en una AWS IoT regla.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "cloudwatchAlarm": {
                    "alarmName": "IotAlarm",
                    "stateReason": "Temperature stabilized.",
                    "stateValue": "OK",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_cw"
                }
            }
        ]
    }
}
```

Véase también

- ¿Qué es Amazon CloudWatch? en la Guía del CloudWatch usuario de Amazon
- Uso de CloudWatch las alarmas de Amazon en la Guía del CloudWatch usuario de Amazon

CloudWatch Registros

La acción CloudWatch Logs (`cloudwatchLogs`) envía datos a Amazon CloudWatch Logs. Puede utilizar `batchMode` para cargar y marcar la hora de varios registros de dispositivos en un solo mensaje. También puede especificar el grupo de registro al que la acción envía los datos.

Requisitos

Esta regla tiene los siguientes requisitos:

- Una función de IAM que AWS IoT puede asumir la responsabilidad de realizar `logs:CreateLogStream` las `logs:DescribeLogStreams` `logs:PutLogEvents` operaciones y. Para obtener más información, consulte Otorgar a una AWS IoT regla el acceso que requiere.

  En la AWS IoT consola, puede elegir o crear un rol que permita AWS IoT realizar esta acción de regla.
- Si utilizas una clave KMS administrada AWS KMS key por el cliente para cifrar los datos de registro en CloudWatch Logs, el servicio debe tener permiso para usar la clave KMS en nombre de la persona que llama. Para obtener más información, consulte Cifrar datos de registro en CloudWatch Logs utilizando AWS KMS la Guía del usuario de Amazon CloudWatch Logs.

Requisitos de formato de mensaje MQTT para **`batchMode`**

Si utilizas la acción de la regla de CloudWatch registros `batchMode` desactivada, no hay requisitos de formato de mensajes en MQTT. (Nota: el valor por defecto del parámetro `batchMode` es `false`.) Sin embargo, si utiliza la acción de la regla CloudWatch Registros con la opción `batchMode` activada (el valor del parámetro es`true`), los mensajes MQTT que contienen registros del dispositivo deben formatearse de forma que contengan una marca de tiempo y una carga útil de mensajes. Nota: `timestamp` representa la hora en que se produjo el suceso y se expresa como un número de milisegundos después del 1 de enero de 1970 00:00:00 UTC.

A continuación se muestra un ejemplo del formato de publicación:

```
[
  {"timestamp": 1673520691093, "message": "Test message 1"},
  {"timestamp": 1673520692879, "message": "Test message 2"},
  {"timestamp": 1673520693442, "message": "Test message 3"}
]
```

Según cómo se generen los registros del dispositivo, es posible que sea necesario filtrarlos y reformatearlos antes de enviarlos para cumplir con este requisito. Para obtener más información, consulte Carga de mensajes MQTT.

Independientemente del `batchMode` parámetro, el `message` contenido debe cumplir con las limitaciones de tamaño de los mensajes. AWS IoT Para obtener más información, consulte Puntos de conexión y cuotas de AWS IoT Core.

Parameters

Al crear una AWS IoT regla con esta acción, debe especificar la siguiente información:

`logGroupName`

El grupo de CloudWatch registros al que la acción envía los datos.

Admite plantillas de sustitución: API y AWS CLI únicamente

`roleArn`

La función de IAM que permite el acceso al grupo de CloudWatch registros. Para obtener más información, consulte Requisitos.

Admite plantillas de sustitución: No

(opcional) `batchMode`

Indica si los lotes de registros se van a extraer y cargar en CloudWatch ellos. Los valores incluyen `true` o `false` (predeterminado). Para obtener más información, consulte Requisitos.

Admite plantillas de sustitución: No

Ejemplos

El siguiente ejemplo de JSON define una acción de CloudWatch registros en una AWS IoT regla.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "cloudwatchLogs": {
                    "logGroupName": "IotLogs",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_cw",
                    "batchMode": false
                }
```

```
            }
        ]
    }
}
```

Véase también

- ¿Qué es Amazon CloudWatch Logs? en la Guía del usuario CloudWatch de Amazon Logs

CloudWatch métricas

La acción CloudWatch metric (`cloudwatchMetric`) captura una CloudWatch métrica de Amazon. Puede especificar el espacio de nombres, el nombre, el valor, la unidad y la marca de tiempo de la métrica.

Requisitos

Esta regla tiene los siguientes requisitos:

- Una función de IAM que AWS IoT puede asumir para realizar la `cloudwatch:PutMetricData` operación. Para obtener más información, consulte Otorgar a una AWS IoT regla el acceso que requiere.

  En la AWS IoT consola, puede elegir o crear un rol que permita AWS IoT realizar esta acción de regla.

Parameters

Al crear una AWS IoT regla con esta acción, debe especificar la siguiente información:

`metricName`

El nombre de la CloudWatch métrica.

Admite plantillas de sustitución: Sí

`metricNamespace`

El nombre del espacio de nombres de la CloudWatch métrica.

Admite plantillas de sustitución: Sí

`metricUnit`

La unidad métrica admitida por. CloudWatch

Admite plantillas de sustitución: Sí

`metricValue`

Cadena que contiene el valor CloudWatch métrico.

Admite plantillas de sustitución: Sí

`metricTimestamp`

(Opcional) Una cadena que contiene la marca de tiempo, expresada en segundos en tiempo de época Unix. El valor predeterminado es la época actual de Unix.

Admite plantillas de sustitución: Sí

`roleArn`

La función de IAM que permite el acceso a la CloudWatch métrica. Para obtener más información, consulte Requisitos.

Admite plantillas de sustitución: No

Ejemplos

El siguiente ejemplo de JSON define una acción CloudWatch métrica en una AWS IoT regla.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "cloudwatchMetric": {
                    "metricName": "IotMetric",
                    "metricNamespace": "IotNamespace",
                    "metricUnit": "Count",
                    "metricValue": "1",
                    "metricTimestamp": "1456821314",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_cw"
                }
```

```
            }
        ]
    }
}
```

El siguiente ejemplo de JSON define una acción CloudWatch métrica con plantillas de sustitución en una AWS IoT regla.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "cloudwatchMetric": {
                    "metricName": "${topic()}",
                    "metricNamespace": "${namespace}",
                    "metricUnit": "${unit}",
                    "metricValue": "${value}",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_cw"
                }
            }
        ]
    }
}
```

Véase también

- ¿Qué es Amazon CloudWatch? en la Guía del CloudWatch usuario de Amazon
- Uso de CloudWatch las métricas de Amazon en la Guía del CloudWatch usuario de Amazon

DynamoDB

La acción DynamoDB (`dynamoDB`) escribe la totalidad o parte de un mensaje MQTT en una tabla de Amazon DynamoDB.

Puede seguir un tutorial que muestra cómo crear y probar una regla con una acción de DynamoDB. Para obtener más información, consulte Tutorial: Almacenamiento de datos de dispositivos en una tabla de DynamoDB.

> **Note**
>
> Esta regla escribe datos no JSON en DynamoDB como datos binarios. La consola de DynamoDB mostrará los datos como texto codificado en Base64.

Requisitos

Esta regla tiene los siguientes requisitos:

- Un rol de IAM que AWS IoT puede asumir para realizar la operación. `dynamodb:PutItem` Para obtener más información, consulte Otorgar a una AWS IoT regla el acceso que requiere.

  En la AWS IoT consola, puede elegir o crear un rol que permita AWS IoT realizar esta acción de regla.
- Si utiliza una clave KMS gestionada AWS KMS key por el cliente para cifrar los datos en reposo en DynamoDB, el servicio debe tener permiso para utilizar la clave KMS en nombre de la persona que llama. Para obtener más información, consulte la clave de KMS gestionada por el cliente en la Guía de introducción de Amazon DynamoDB.

Parameters

Al crear una AWS IoT regla con esta acción, debe especificar la siguiente información:

`tableName`

El nombre de la tabla de DynamoDB.

Admite plantillas de sustitución: API y AWS CLI únicamente

`hashKeyField`

El nombre de la clave hash (también denominada clave de partición).

Admite plantillas de sustitución: API y AWS CLI únicamente

`hashKeyType`

(Opcional) El tipo de datos de la clave hash (también denominado clave de partición). Valores válidos: `STRING`, `NUMBER`.

Admite plantillas de sustitución: API y AWS CLI únicamente

`hashKeyValue`

El valor de la clave hash. Considere la posibilidad de utilizar una plantilla de sustitución como `${topic()}` o `${timestamp()}`.

Admite plantillas de sustitución: Sí

`rangeKeyField`

(Opcional) El nombre de la clave de rango (también denominada clave de clasificación).

Admite plantillas de sustitución: API y AWS CLI únicamente

`rangeKeyType`

(Opcional) El tipo de datos de la clave de rango (también denominada clave de clasificación). Valores válidos: `STRING`, `NUMBER`.

Admite plantillas de sustitución: API y AWS CLI únicamente

`rangeKeyValue`

(Opcional) El valor de la clave de rango. Considere la posibilidad de utilizar una plantilla de sustitución como `${topic()}` o `${timestamp()}`.

Admite plantillas de sustitución: Sí

`payloadField`

(Opcional) El nombre de la columna donde se escribe la carga útil. Si omite este valor, la carga útil se escribe en la columna denominada `payload`.

Admite plantillas de sustitución: Sí

`operation`

(Opcional) El tipo de operación que se va a realizar. Valores válidos: `INSERT`, `UPDATE`, `DELETE`.

Admite plantillas de sustitución: Sí

`roleARN`

El rol de IAM que permite tener acceso a la tabla de DynamoDB. Para obtener más información, consulte Requisitos.

Admite plantillas de sustitución: No

Los datos que se escriben en la tabla de DynamoDB son el resultado de la instrucción SQL de la regla.

Ejemplos

El siguiente ejemplo de JSON define una acción de DynamoDB en una regla. AWS IoT

```
{
    "topicRulePayload": {
        "sql": "SELECT * AS message FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "dynamoDB": {
                    "tableName": "my_ddb_table",
                    "hashKeyField": "key",
                    "hashKeyValue": "${topic()}",
                    "rangeKeyField": "timestamp",
                    "rangeKeyValue": "${timestamp()}",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_dynamoDB"
                }
            }
        ]
    }
}
```

Véase también

- ¿Qué es Amazon DynamoDB? en la Guía para desarrolladores de Amazon DynamoDB.
- Introducción a DynamoDB en la Guía para desarrolladores de Amazon DynamoDB
- Tutorial: Almacenamiento de datos de dispositivos en una tabla de DynamoDB

Dinamo DBv2

La acción Dynamo DBv2 (`dynamoDBv2`) escribe todo o parte de un mensaje MQTT en una tabla de Amazon DynamoDB. Cada atributo de la carga útil se escribe en una columna independiente de la base de datos de DynamoDB.

Requisitos

Esta regla tiene los siguientes requisitos:

- Una función de IAM que AWS IoT puede asumir para realizar la operación. `dynamodb:PutItem` Para obtener más información, consulte Otorgar a una AWS IoT regla el acceso que requiere.

  En la AWS IoT consola, puede elegir o crear un rol que permita AWS IoT realizar esta acción de regla.
- 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 clasificación principal de la tabla, si hay una definida.
- Si utiliza una clave KMS gestionada AWS KMS key por el cliente para cifrar los datos en reposo en DynamoDB, el servicio debe tener permiso para utilizar la clave KMS en nombre de la persona que llama. Para obtener más información, consulte la clave de KMS gestionada por el cliente en la Guía de introducción de Amazon DynamoDB.

Parameters

Al crear una AWS IoT regla con esta acción, debe especificar la siguiente información:

`putItem`

Un objeto que especifica la tabla de DynamoDB en la que se escribirán los datos del mensaje. Este objeto debe contener la siguiente información:

`tableName`

El nombre de la tabla de DynamoDB.

Admite plantillas de sustitución: API y AWS CLI únicamente

`roleARN`

El rol de IAM que permite tener acceso a la tabla de DynamoDB. Para obtener más información, consulte Requisitos.

Admite plantillas de sustitución: No

Los datos que se escriben en la tabla de DynamoDB son el resultado de la instrucción SQL de la regla.

Ejemplos

El siguiente ejemplo de JSON define una DBv2 acción de Dynamo en una AWS IoT regla.

```
{
    "topicRulePayload": {
        "sql": "SELECT * AS message FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "dynamoDBv2": {
                    "putItem": {
                        "tableName": "my_ddb_table"
                    },
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_dynamoDBv2",
                }
            }
        ]
    }
}
```

El siguiente ejemplo de JSON define una acción de DynamoDB con plantillas de sustitución en una regla. AWS IoT

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2015-10-08",
        "actions": [
            {
                "dynamoDBv2": {
                    "putItem": {
                        "tableName": "${topic()}"
                    },
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_dynamoDBv2"
                }
            }
        ]
    }
}
```

Véase también

- ¿Qué es Amazon DynamoDB? en la Guía para desarrolladores de Amazon DynamoDB.
- Introducción a DynamoDB en la Guía para desarrolladores de Amazon DynamoDB

Elasticsearch

La acción Elasticsearch (`elasticsearch`) escribe los datos de los mensajes MQTT en un dominio de Amazon OpenSearch Service. A continuación, puede utilizar herramientas como los OpenSearch paneles de control para consultar y visualizar los datos en Service. OpenSearch

> **Warning**
>
> La acción `Elasticsearch` solo puede ser utilizado por acciones de reglas existentes. Para crear una nueva acción de regla o actualizar una acción de regla existente, use la acción de la regla `OpenSearch` en su lugar. Para obtener más información, consulte OpenSearch.

Requisitos

Esta regla tiene los siguientes requisitos:

- Una función de IAM que AWS IoT puede asumir para realizar la `es:ESHttpPut` operación. Para obtener más información, consulte Otorgar a una AWS IoT regla el acceso que requiere.

  En la AWS IoT consola, puede elegir o crear un rol que permita AWS IoT realizar esta acción de regla.
- Si utilizas una clave KMS gestionada AWS KMS key por el cliente para cifrar los datos en reposo OpenSearch, el servicio debe tener permiso para utilizar la clave KMS en nombre de la persona que llama. Para obtener más información, consulta Cifrado de datos en reposo para Amazon OpenSearch Service en la Guía para desarrolladores de Amazon OpenSearch Service.

Parameters

Al crear una AWS IoT regla con esta acción, debe especificar la siguiente información:

`endpoint`

El punto de conexión de su dominio de servicio.

Admite plantillas de sustitución: API y AWS CLI únicamente

`index`

Índice de donde se van a almacenar los datos.

Admite plantillas de sustitución: Sí

`type`

Tipo de documento que está almacenando.

Admite plantillas de sustitución: Sí

`id`

Identificador único de cada documento.

Admite plantillas de sustitución: Sí

`roleARN`

La función de IAM que permite el acceso al dominio del OpenSearch servicio. Para obtener más información, consulte Requisitos.

Admite plantillas de sustitución: No

Ejemplos

El siguiente ejemplo de JSON define una acción de Elasticsearch en una AWS IoT regla y cómo se pueden especificar los campos de la acción. `elasticsearch` Para obtener más información, consulte ElasticsearchAction.

```
{
    "topicRulePayload": {
        "sql": "SELECT *, timestamp() as timestamp FROM 'iot/test'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "elasticsearch": {
                    "endpoint": "https://my-endpoint",
```

```
                    "index": "my-index",
                    "type": "my-type",
                    "id": "${newuuid()}",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_es"
                }
            }
        ]
    }
}
```

El siguiente ejemplo de JSON define una acción de Elasticsearch con plantillas de sustitución en una regla. AWS IoT

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "elasticsearch": {
                    "endpoint": "https://my-endpoint",
                    "index": "${topic()}",
                    "type": "${type}",
                    "id": "${newuuid()}",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_es"
                }
            }
        ]
    }
}
```

Véase también

- OpenSearch
- ¿Qué es Amazon OpenSearch Service?

HTTP

La acción HTTPS (`http`) envía los datos de un mensaje MQTT a un punto final HTTPS, que puede apuntar a una aplicación o servicio web.

Requisitos

Esta regla tiene los siguientes requisitos:

- Debe confirmar y habilitar los puntos de conexión HTTPS antes de que el motor de reglas pueda usarlos. Para obtener más información, consulte Trabajar con destinos de reglas temática HTTP.

Parameters

Al crear una AWS IoT regla con esta acción, debe especificar la siguiente información:

`url`

El punto de conexión HTTPS al que se envía el mensaje mediante el método HTTP POST. Si utiliza una dirección IP en lugar de un nombre de host, debe ser una IPv4 dirección. IPv6 no se admiten direcciones.

Admite plantillas de sustitución: Sí

`confirmationUrl`

(Opcional) Si se especifica, AWS IoT utiliza la URL de confirmación para crear un destino de regla temática coincidente. Debe habilitar el destino de la regla del tema para poder usarlo en una acción HTTP. Para obtener más información, consulte Trabajar con destinos de reglas temática HTTP. Si utiliza plantillas de sustitución, debe crear manualmente destinos de reglas de tema para poder utilizar la acción `http`. `confirmationUrl` debe ser un prefijo de `url`.

La relación entre `url` y `confirmationUrl` se describe de la siguiente manera:

- Si `url` está codificado y no `confirmationUrl` se proporciona, tratamos implícitamente el `url` campo como. `confirmationUrl` AWS IoT crea un tema de destino para la regla. `url`
- Si `url` y `confirmationUrl` están codificados, `url` deben empezar `confirmationUrl` por. AWS IoT crea un tema, regla de destino para`confirmationUrl`.
- Si `url` contiene una plantilla de sustitución, debe especificar `confirmationUrl` y `url` debe comenzar por `confirmationUrl`. Si `confirmationUrl` contiene plantillas de sustitución, debe crear manualmente destinos de reglas del tema para poder utilizar la acción `http`. Si `confirmationUrl` no contiene plantillas de sustitución, AWS IoT crea un destino de regla temática para`confirmationUrl`.

Admite plantillas de sustitución: Sí

`headers`

(Opcional) La lista de encabezados que se van a incluir en las solicitudes HTTP al punto de conexión. Cada encabezado debe contener la siguiente información:

`key`

La clave del encabezado.

Admite plantillas de sustitución: No

`value`

El valor del encabezado.

Admite plantillas de sustitución: Sí

> **Note**
>
> El tipo de contenido predeterminado es application/json cuando la carga útil está en formato JSON. De lo contrario, es application/octet-stream. Puede sobrescribirlo especificando el tipo de contenido exacto en el encabezado con el tipo de contenido de la clave (sin distinción entre mayúsculas y minúsculas).

`auth`

(Opcional) Autenticación utilizada por el motor de reglas para conectarse a la dirección URL del punto de conexión especificada en el argumento `url`. Actualmente, Signature Version 4 es el único tipo de autenticación admitido. Para obtener más información, consulte Autorización HTTP.

Admite plantillas de sustitución: No

`enableBatching`

(Opcional) Si se deben procesar los mensajes de acción HTTP en una sola solicitud para una URL determinada. El valor puede ser verdadero o falso. Para obtener más información sobre el procesamiento por lotes, consulte Procesamiento por lotes de mensajes de acción HTTP.

Valor booleano

Admite plantillas de sustitución: No

`batchConfig`

(Opcional) Ajustes de configuración para el procesamiento por lotes. Una vez habilitados, se deben especificar `batchConfig` los parámetros. Si no se especifican los `batchConfig` parámetros, se utilizarán los valores predeterminados.

`maxBatchOpenMs`

El tiempo máximo (en milisegundos) que un mensaje saliente espera a que otros mensajes creen el lote. Cuanto más alta sea la configuración, mayor será la latencia de la acción HTTP agrupada en lotes.

Valor mínimo: 5 ms. Valor máximo: 200 ms.

Valor predeterminado: 20 ms

Admite plantillas de sustitución: No

`maxBatchSize`

El número máximo de mensajes que se agrupan en lotes en la ejecución de una sola acción.

Admite plantillas de sustitución: No

Valor mínimo: 2 mensajes. Valor máximo: 10 mensajes

Valor predeterminado: 10 mensajes

`maxBatchSizeBytes`

Tamaño máximo de un lote de mensajes, en bytes.

Valor mínimo: 100 bytes. Valor máximo: 131.072 bytes

Valor predeterminado: 5.120 bytes

Admite plantillas de sustitución: No

> **Note**
>
> El tipo de contenido predeterminado es application/json cuando la carga útil está en formato JSON. De lo contrario, es application/octet-stream. Puede sobrescribirlo

especificando el tipo de contenido exacto en el encabezado con el tipo de contenido de la clave (sin distinción entre mayúsculas y minúsculas).

Ejemplos

El siguiente ejemplo de JSON define una AWS IoT regla con una acción HTTP.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "http": {
                    "url": "https://www.example.com/subpath",
                    "confirmationUrl": "https://www.example.com",
                    "headers": [
                        {
                            "key": "static_header_key",
                            "value": "static_header_value"
                        },
                        {
                            "key": "substitutable_header_key",
                            "value": "${value_from_payload}"
                        }
                    ]
                }
            }
        ]
    }
}
```

```
"http": {
    "url": "https://www.example.com/subpath",
    "confirmationUrl": "https://www.example.com",
    "headers": [
        {
            "key": "Content-Type",
            "value": "application/json"
          }
```

```
    ],
    "enableBatching": true,
    "batchConfig": {
      "maxBatchOpenMs": 123,
      "maxBatchSize": 5,
      "maxBatchSizeBytes": 131072,
     }
 },
 "errorAction": {
        "http": {
            "url": "https://www.example.com/subpath",
            "confirmationUrl": "https://www.example.com"
            // batchConfig is not allowed here
        }
}
```

Lógica de reintento de acciones HTTP

El motor de AWS IoT reglas vuelve a intentar la acción HTTP de acuerdo con estas reglas:

- El motor de reglas intenta enviar un mensaje al menos una vez.
- El motor de reglas realiza como máximo dos reintentos. El número máximo de intentos es tres.
- El motor de reglas no realiza un reintento si:
  - El intento anterior proporcionó una respuesta mayor de 16.384 bytes.
  - El servicio o la aplicación web de salida cierra la conexión TCP después del intento.
  - El tiempo total para completar una solicitud con reintentos superó el límite de tiempo de espera de la solicitud.
  - La solicitud devuelve un código de estado HTTP distinto de 429, 500-599.

> **Note**
>
> Se aplican costos estándar de transferencia de datos a los reintentos.

Véase también

- Procesamiento por lotes de mensajes de acción HTTP
- Trabajar con destinos de reglas temática HTTP

- Enrute los datos directamente desde AWS IoT Core sus servicios web en el blog de Internet de las AWS cosas

Procesamiento por lotes de mensajes de acción HTTP

Puedes usar el procesamiento por lotes para enviar varios mensajes de acción HTTP en una sola solicitud.

Descripción general de

El procesamiento por lotes le permite enviar mensajes desde AWS IoT Core Rules Engine a sus puntos de enlace HTTP por lotes. Esta funcionalidad puede ayudarle a reducir los costes al reducir el número de ejecuciones de acciones HTTP, así como a mejorar la eficiencia al reducir la sobrecarga asociada al establecimiento de nuevas conexiones.

> **Note**
>
> La acción HTTP por lotes se mide como una acción única. Se mide en incrementos de 5 KiB, en función del tamaño de la carga útil saliente por lotes emitida por el motor de reglas al AWS IoT Core servicio descendente. Para obtener más información, consulte la página de precios de AWS IoT Core.

Cuando habilita el procesamiento por lotes en la definición de su acción de regla de IoT, los siguientes parámetros estarán disponibles para la configuración:

`maxBatchOpenMs`

El tiempo máximo (en milisegundos) que un mensaje saliente espera a que otros mensajes creen el lote. Cuanto más alta sea la configuración, mayor será la latencia de la acción HTTP agrupada en lotes.

Valor mínimo: 5 ms. Valor máximo: 200 ms.

Valor predeterminado: 20 ms

Admite plantillas de sustitución: No

`maxBatchSize`

La cantidad máxima de mensajes que se agrupan por lotes en una sola ejecución de acción de regla de IoT.

Valor mínimo: 2 mensajes. Valor máximo: 10 mensajes

Valor predeterminado: 10 mensajes

Admite plantillas de sustitución: No

`maxBatchSizeBytes`

Tamaño máximo de un lote de mensajes, en bytes.

Valor mínimo: 100 bytes. Valor máximo: 131.072 bytes

Valor predeterminado: 5120 bytes

Admite plantillas de sustitución: No

> **Important**
>
> Al especificar varios parámetros de lote, el procesamiento por lotes finaliza cuando se alcanza el primer límite. Por ejemplo, si especifica 100 ms como el tiempo máximo de apertura del lote y 5 KiB como el tamaño máximo del lote, y Rules Engine solo graba 2 KiB en 100 ms, se creará y enviará un lote de 2 KiB.

Uso de encabezados HTTP en un lote

Cuando utilizas encabezados en tu acción HTTP, la solicitud agrupada usa el valor del encabezado del último mensaje que se agregó al lote (no necesariamente el último mensaje que publicaste). Recomendamos utilizar valores de encabezado que sean:

- Idénticos en todos los mensajes del lote
- Aplicable a todos los mensajes (por ejemplo, las credenciales de autenticación)

Los encabezados se envían con la solicitud HTTP y no forman parte del cuerpo del mensaje.

> **Note**
>
> Cuando el procesamiento por lotes está habilitado:
>
> - La solicitud agrupada incluye automáticamente el `Content-Type: application/json` encabezado, ya que el lote se envía como una matriz JSON.
> - No podemos garantizar que el último mensaje del lote sea el último mensaje que hayas publicado. Es el último mensaje que se incluyó en el lote.

Ejemplo de carga útil

El siguiente ejemplo muestra la estructura de la carga útil de un mensaje por lotes que se envía a tu punto final HTTP:

```
[
  {
    "user_id": "user1",
    "steps_today": 1000
  },
  {
    "user_id": "user2",
    "steps_today": 21000
  },
  {
    "user_id": "user8",
    "steps_today": 1500
  },
  ...
]
```

Limitaciones

Las siguientes son las limitaciones del procesamiento por lotes:

- AWS IoT Core no garantiza el orden general de los mensajes. El procesamiento por lotes se realiza de forma local en cada host, lo que puede provocar que los mensajes de un lote se procesen en un orden diferente al que se recibieron.

- AWS IoT Core no proporciona soporte para el procesamiento de mensajes en el lado del receptor. Usted es responsable de garantizar que su servicio descendente esté configurado para aceptar y procesar los datos en lotes.
- No se admite el procesamiento por lotes entre cuentas, incluso si los mensajes están destinados al mismo identificador de recurso (URL HTTP o ARN del recurso).
- AWS IoT Core no garantiza que el tamaño del lote cumpla con la configuración que especificó. Los lotes pueden ser más pequeños que los límites configurados en función del tiempo y el flujo de mensajes.
- Cuando el procesamiento por lotes está activado, no se admiten las cargas útiles binarias (datos que no sean UTF-8). Solo se aceptan cargas útiles de texto en UTF-8 (como JSON). Para enviar datos binarios, base64 los codifica antes de enviarlos a la acción HTTP y, a continuación, los decodifica en el punto final receptor. Por ejemplo, puede usar la función de codificación en las reglas de IoT para codificar la carga binaria. Como alternativa, puedes codificar la carga binaria en tu dispositivo IoT y publicarla en. AWS IoT Core

Acciones de error al procesar por lotes

No podrá definir una lógica de procesamiento por lotes independiente en la definición de la acción de error. Sin embargo, su acción de error admitirá el procesamiento por lotes si ha definido la lógica de procesamiento por lotes en su acción principal.

Cuando se produce un error en una solicitud por lotes, el motor de AWS IoT Core reglas seguirá la lógica de reintento de una acción HTTP. Tras el último intento, se ejecutará una acción de error para todo el lote fallido.

El siguiente es un ejemplo de un mensaje de acción de error con el procesamiento por lotes activado:


```
{
    "ruleName": "FailedTopicRule",
    "topic": "topic/rulesengine",
    "payloadsWithMetadata": [
        {
            "id": 1,
            "cloudwatchTraceId": "bebd6d93-6d4a-899e-9e40-56e82252d2be",
            "clientId": "Test",
            "sourceIp": "10.0.0.0",
            "base64OriginalPayload":
 "eyJ1c2VyX2lkIjogInVzZXI1NjQ3IiwgInN0ZXBzX3RvZGF5IjogMTMzNjUsICJ0aW1lc3RhbXAiOiAiMjAyNS0xMC0wO
        },
```

```
        {
            "id": 2,
            "cloudwatchTraceId": "af94d3b8-0b18-1dbf-2c7d-513f5cb9e2e1",
            "clientId": "Test",
            "sourceIp": "10.0.0.0",
            "base64OriginalPayload":
"eyJ1c2VyX2lkIjogInVzZXI1NjQ3IiwgInN0ZXBzX3RvZGF5IjogMTMzNjUsICJ0aW1lc3RhbXAiOiAiMjAyNS0xMC0wO
        },
        {
            "id": 3,
            "cloudwatchTraceId": "ca441266-c2ce-c916-6aee-b9e5c7831675",
            "clientId": "Test",
            "sourceIp": "10.0.0.0",
            "base64OriginalPayload":
"eyJ1c2VyX2lkIjogInVzZXI1NjQ3IiwgInN0ZXBzX3RvZGF5IjogMTMzNjUsICJ0aW1lc3RhbXAiOiAiMjAyNS0xMC0wO
        }
    ],
    "failures": [
        {
            "affectedIds": [
                1,
                2,
                3
            ],
            "failedAction": "HttpAction",
            "failedResource": "https://example.foobar.com/HttpAction",
            "errorMessage": "HttpAction failed to make a request to the specified
endpoint. StatusCode: 500. Reason: Internal Server Error."
        },
        {
            "affectedIds": [
                3
            ],
            "failedAction": "S3Action",
            "failedResource": "amzn-s3-demo-bucket",
            "errorMessage": "Failed to put S3 object. The error received was The
specified bucket does not exist"
        },
        {
            "affectedIds": [
                3
            ],
            "failedAction": "LambdaAction",
            "failedResource": "arn:aws:lambda:us-west-2:123456789012:function:dummy",
```

```
            "errorMessage": "Failed to invoke lambda function. Received Server error
 from Lambda. The error code is 403"
        }
    ]
}
```

> **Note**
>
> Los errores de las acciones por lotes también generan una mayor carga útil de las acciones de error, lo que, debido al tamaño, puede aumentar la probabilidad de que se produzcan errores en las acciones de error. Puede monitorizar los fallos en las acciones de error mediante la `ErrorActionFailure` métrica. Para obtener más información, consulte Métricas de acciones de reglas.

Agrupar por lotes los mensajes de acción HTTP con el AWS CLI

Crear o actualizar una acción de regla con procesamiento por lotes

1. Utilice el AWS CLI comando correspondiente para crear o actualizar una regla:
   - Para crear una regla nueva, utilice el create-topic-rulecomando:

     ```
     aws iot create-topic-rule --rule-name myrule --topic-rule-payload
      file://myrule.json
     ```

   - Para actualizar una regla existente, utilice el replace-topic-rulecomando:

     ```
     aws iot replace-topic-rule --rule-name myrule --topic-rule-payload
      file://myrule.json
     ```

2. Habilite las capacidades de procesamiento por lotes configurando el parámetro enableBatching en true en la carga útil de la regla temática:

```
{
        "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "http": {
```

```
                    "url": "https://www.example.com/subpath",
                    "confirmationUrl": "https://www.example.com",
                    "headers": [
                        {
                            "key": "static_header_key",
                            "value": "static_header_value"
                        },
                        {
                            "key": "substitutable_header_key",
                            "value": "${value_from_payload}"
                         }
                    ],
                    "enableBatching": true,
                    "batchConfig": {
                       "maxBatchOpenMs": 100,
                       "maxBatchSize": 5,
                       "maxBatchSizeBytes": 1024
                    }
                }
            }
        ]
}
```

3. Configure los parámetros de procesamiento por lotes. No es necesario especificar todos los parámetros del lote. Puede elegir especificar 1, 2 o los 3 parámetros del lote. Si no especifica un parámetro de lote, Rules Engine actualizará ese parámetro con los valores predeterminados. Para obtener más información sobre los parámetros de procesamiento por lotes y sus valores predeterminados, consulte Parámetros HTTP.

Trabajar con destinos de reglas temática HTTP

El destino de una regla temática HTTP es un servicio web al que el motor de reglas puede enrutar los datos de una regla temática. Un AWS IoT Core recurso describe el servicio web para AWS IoT. Los recursos de destino de las reglas temáticas se pueden compartir mediante diferentes reglas.

Antes de AWS IoT Core poder enviar datos a otro servicio web, debe confirmar que puede acceder al punto final del servicio.

En este capítulo:

- Descripción general del destino de la regla del tema HTTP
- Administración de los destinos de las reglas temáticas HTTP

- Las autoridades de certificación son compatibles con puntos de conexión HTTPS en los destinos de las reglas temáticas

Descripción general del destino de la regla del tema HTTP

El destino de una regla temática HTTP hace referencia a un servicio web que admite una URL de confirmación y una o varias recopilaciones de datos URLs. El recurso de destino de la regla temática HTTP contiene la URL de confirmación de su servicio web. Al configurar una acción de regla temática HTTP, se especifica la URL real del punto de conexión que debe recibir los datos junto con la URL de confirmación del servicio web. Una vez confirmado el destino, la regla temática envía el resultado de la sentencia SQL al punto de conexión HTTPS (y no a la URL de confirmación).

Un destino de regla temática HTTP puede estar en uno de los siguientes estados:

ENABLED

El destino se ha confirmado y se puede utilizar mediante una acción de regla. Un destino debe tener el estado `ENABLED` para que se utilice en una regla. Solo puede habilitar destinos con el estado DISABLED.

DISABLED

El destino se ha confirmado y se puede utilizar mediante una acción de regla. Esto es útil si desea impedir temporalmente el tráfico a su punto de conexión sin tener que pasar de nuevo por el proceso de confirmación. Solo puede deshabilitar un destino con el estado ENABLED.

IN_PROGRESS

La confirmación del destino se está realizando.

ERROR

Se ha agotado el tiempo de espera de confirmación del destino.

Una vez confirmado y habilitado el destino de una regla temática HTTP, puede utilizarse con cualquier regla de su cuenta.

Las siguientes secciones describen acciones comunes en destinos de reglas temáticas HTTP.

Administración de los destinos de las reglas temáticas HTTP

Puede utilizar las siguientes operaciones para administrar sus destinos de reglas temáticas HTTP.

En este tema:

- Creación y confirmación de los destinos de las reglas temáticas HTTP
- Confirmación de los destinos de las reglas temáticas HTTP
- Enviar una nueva solicitud de confirmación
- Desactivación y eliminación de un destino de regla temática

Creación y confirmación de los destinos de las reglas temáticas HTTP

Para crear un destino de regla temática HTTP, llame a la operación `CreateTopicRuleDestination` o utilice la consola de AWS IoT .

Tras crear un destino, AWS IoT envía una solicitud de confirmación a la URL de confirmación. La solicitud de confirmación tiene el siguiente formato:

```
HTTP POST {confirmationUrl}/?confirmationToken={confirmationToken}
Headers:
x-amz-rules-engine-message-type: DestinationConfirmation
x-amz-rules-engine-destination-arn:"arn:aws:iot:us-east-1:123456789012:ruledestination/
http/7a280e37-b9c6-47a2-a751-0703693f46e4"
Content-Type: application/json
Body:
{
    "arn":"arn:aws:iot:us-east-1:123456789012:ruledestination/http/7a280e37-b9c6-47a2-
a751-0703693f46e4",
    "confirmationToken": "AYADeMXLrPrNY2wqJAKsFNn-…NBJndA",
    "enableUrl": "https://iot.us-east-1.amazonaws.com/confirmdestination/
AYADeMXLrPrNY2wqJAKsFNn-…NBJndA",
    "messageType": "DestinationConfirmation"
}
```

El contenido de la solicitud de confirmación incluye la siguiente información:

arn

El nombre de recurso de Amazon (ARN) para el destino de la regla temática a confirmar.

confirmationToken

El token de confirmación enviado por AWS IoT Core. El token del ejemplo está truncado. Su token será mayor. Necesitará este token para confirmar su destino con AWS IoT Core.

enableUrl

La dirección URL a la que se desplaza para confirmar el destino de una regla del tema.

messageType

Tipo de mensaje.

Confirmación de los destinos de las reglas temáticas HTTP

Para completar el proceso de confirmación del punto de conexión, si utiliza la AWS CLI, debe realizar uno de los siguientes pasos después de que su URL de confirmación reciba la solicitud de confirmación.

1. Confirmación de que el destino está dispuesto a recibir mensajes

   Para confirmar que el destino de la regla temática está dispuesto a recibir mensajes de IoT, llame a la `enableUrl` de la solicitud de confirmación o ejecute la operación de la API `ConfirmTopicRuleDestination` y transfiera el `confirmationToken` desde la solicitud de confirmación.
2. Definición del estado de la regla temática como activado

   Después de confirmar que el destino puede recibir mensajes, debe ejecutar la operación de la API `UpdateTopicRuleDestination` para establecer el estado de la regla temática en `ENABLED`.

Si utilizas la AWS IoT consola, cópiala `confirmationToken` y pégala en el cuadro de diálogo de confirmación del destino en la AWS IoT consola. A continuación, ya podrá activar la regla temática.

Enviar una nueva solicitud de confirmación

Para activar un nuevo mensaje de confirmación para un destino, llame a `UpdateTopicRuleDestination` y establezca el estado del destino de la regla del tema en `IN_PROGRESS`.

Repita el proceso de confirmación después de enviar una nueva solicitud de confirmación.

Desactivación y eliminación de un destino de regla temática

Para deshabilitar un destino, llame a `UpdateTopicRuleDestination` y establezca el estado del destino de la regla del tema en `DISABLED`. Una regla temática en estado DESACTIVADO se puede volver a activar sin necesidad de enviar una nueva solicitud de confirmación.

Para eliminar un destino de regla del tema, llame a `DeleteTopicRuleDestination`.

Las autoridades de certificación son compatibles con puntos de conexión HTTPS en los destinos de las reglas temáticas

Las autoridades de certificación son compatibles con puntos de conexión HTTPS en los destinos de las reglas temáticas Puede elegir una de estas autoridades de certificación compatibles. Las firmas sirven de referencia. Tenga en cuenta que no puede usar certificados autofirmados porque no funcionarán.

Ayúdenos a mejorar este tema

Denos su opinión

```
Alias name: swisssignplatinumg2ca
Certificate fingerprints:
  MD5:  C9:98:27:77:28:1E:3D:0E:15:3C:84:00:B8:85:03:E6
  SHA1: 56:E0:FA:C0:3B:8F:18:23:55:18:E5:D3:11:CA:E8:C2:43:31:AB:66
  SHA256:
 3B:22:2E:56:67:11:E9:92:30:0D:C0:B1:5A:B9:47:3D:AF:DE:F8:C8:4D:0C:EF:7D:33:17:B4:C1:82:1D:14:3

Alias name: hellenicacademicandresearchinstitutionsrootca2011
Certificate fingerprints:
  MD5:  73:9F:4C:4B:73:5B:79:E9:FA:BA:1C:EF:6E:CB:D5:C9
  SHA1: FE:45:65:9B:79:03:5B:98:A1:61:B5:51:2E:AC:DA:58:09:48:22:4D
  SHA256:
 BC:10:4F:15:A4:8B:E7:09:DC:A5:42:A7:E1:D4:B9:DF:6F:05:45:27:E8:02:EA:A9:2D:59:54:44:25:8A:FE:7

Alias name: teliasonerarootcav1
Certificate fingerprints:
  MD5:  37:41:49:1B:18:56:9A:26:F5:AD:C2:66:FB:40:A5:4C
  SHA1: 43:13:BB:96:F1:D5:86:9B:C1:4E:6A:92:F6:CF:F6:34:69:87:82:37
  SHA256:
 DD:69:36:FE:21:F8:F0:77:C1:23:A1:A5:21:C1:22:24:F7:22:55:B7:3E:03:A7:26:06:93:E8:A2:4B:0F:A3:8

Alias name: geotrustprimarycertificationauthority
Certificate fingerprints:
  MD5:  02:26:C3:01:5E:08:30:37:43:A9:D0:7D:CF:37:E6:BF
  SHA1: 32:3C:11:8E:1B:F7:B8:B6:52:54:E2:E2:10:0D:D6:02:90:37:F0:96
  SHA256:
 37:D5:10:06:C5:12:EA:AB:62:64:21:F1:EC:8C:92:01:3F:C5:F8:2A:E9:8E:E5:33:EB:46:19:B8:DE:B4:D0:6
```

```
Alias name: trustisfpsrootca
Certificate fingerprints:
  MD5:  30:C9:E7:1E:6B:E6:14:EB:65:B2:16:69:20:31:67:4D
  SHA1: 3B:C0:38:0B:33:C3:F6:A6:0C:86:15:22:93:D9:DF:F5:4B:81:C0:04
  SHA256:
 C1:B4:82:99:AB:A5:20:8F:E9:63:0A:CE:55:CA:68:A0:3E:DA:5A:51:9C:88:02:A0:D3:A6:73:BE:8F:8E:55:7

Alias name: quovadisrootca3g3
Certificate fingerprints:
  MD5:  DF:7D:B9:AD:54:6F:68:A1:DF:89:57:03:97:43:B0:D7
  SHA1: 48:12:BD:92:3C:A8:C4:39:06:E7:30:6D:27:96:E6:A4:CF:22:2E:7D
  SHA256:
 88:EF:81:DE:20:2E:B0:18:45:2E:43:F8:64:72:5C:EA:5F:BD:1F:C2:D9:D2:05:73:07:09:C5:D8:B8:69:0F:4

Alias name: buypassclass2ca
Certificate fingerprints:
  MD5:  46:A7:D2:FE:45:FB:64:5A:A8:59:90:9B:78:44:9B:29
  SHA1: 49:0A:75:74:DE:87:0A:47:FE:58:EE:F6:C7:6B:EB:C6:0B:12:40:99
  SHA256:
 9A:11:40:25:19:7C:5B:B9:5D:94:E6:3D:55:CD:43:79:08:47:B6:46:B2:3C:DF:11:AD:A4:A0:0E:FF:15:FB:4

Alias name: secureglobalca
Certificate fingerprints:
  MD5:  CF:F4:27:0D:D4:ED:DC:65:16:49:6D:3D:DA:BF:6E:DE
  SHA1: 3A:44:73:5A:E5:81:90:1F:24:86:61:46:1E:3B:9C:C4:5F:F5:3A:1B
  SHA256:
 42:00:F5:04:3A:C8:59:0E:BB:52:7D:20:9E:D1:50:30:29:FB:CB:D4:1C:A1:B5:06:EC:27:F1:5A:DE:7D:AC:6

Alias name: chunghwaepkirootca
Certificate fingerprints:
  MD5:  1B:2E:00:CA:26:06:90:3D:AD:FE:6F:15:68:D3:6B:B3
  SHA1: 67:65:0D:F1:7E:8E:7E:5B:82:40:A4:F4:56:4B:CF:E2:3D:69:C6:F0
  SHA256:
 C0:A6:F4:DC:63:A2:4B:FD:CF:54:EF:2A:6A:08:2A:0A:72:DE:35:80:3E:2F:F5:FF:52:7A:E5:D8:72:06:DF:D

Alias name: verisignclass2g2ca
Certificate fingerprints:
  MD5:  2D:BB:E5:25:D3:D1:65:82:3A:B7:0E:FA:E6:EB:E2:E1
  SHA1: B3:EA:C4:47:76:C9:C8:1C:EA:F2:9D:95:B6:CC:A0:08:1B:67:EC:9D
  SHA256:
 3A:43:E2:20:FE:7F:3E:A9:65:3D:1E:21:74:2E:AC:2B:75:C2:0F:D8:98:03:05:BC:50:2C:AF:8C:2D:9B:41:A

Alias name: szafirrootca2
```

```
Certificate fingerprints:
  MD5:  11:64:C1:89:B0:24:B1:8C:B1:07:7E:89:9E:51:9E:99
  SHA1: E2:52:FA:95:3F:ED:DB:24:60:BD:6E:28:F3:9C:CC:CF:5E:B3:3F:DE
  SHA256:
 A1:33:9D:33:28:1A:0B:56:E5:57:D3:D3:2B:1C:E7:F9:36:7E:B0:94:BD:5F:A7:2A:7E:50:04:C8:DE:D7:CA:F

Alias name: quovadisrootca1g3
Certificate fingerprints:
  MD5:  A4:BC:5B:3F:FE:37:9A:FA:64:F0:E2:FA:05:3D:0B:AB
  SHA1: 1B:8E:EA:57:96:29:1A:C9:39:EA:B8:0A:81:1A:73:73:C0:93:79:67
  SHA256:
 8A:86:6F:D1:B2:76:B5:7E:57:8E:92:1C:65:82:8A:2B:ED:58:E9:F2:F2:88:05:41:34:B7:F1:F4:BF:C9:CC:7

Alias name: utndatacorpsgcca
Certificate fingerprints:
  MD5:  B3:A5:3E:77:21:6D:AC:4A:C0:C9:FB:D5:41:3D:CA:06
  SHA1: 58:11:9F:0E:12:82:87:EA:50:FD:D9:87:45:6F:4F:78:DC:FA:D6:D4
  SHA256:
 85:FB:2F:91:DD:12:27:5A:01:45:B6:36:53:4F:84:02:4A:D6:8B:69:B8:EE:88:68:4F:F7:11:37:58:05:B3:4

Alias name: autoridaddecertificacionfirmaprofesionalcifa62634068
Certificate fingerprints:
  MD5:  73:3A:74:7A:EC:BB:A3:96:A6:C2:E4:E2:C8:9B:C0:C3
  SHA1: AE:C5:FB:3F:C8:E1:BF:C4:E5:4F:03:07:5A:9A:E8:00:B7:F7:B6:FA
  SHA256:
 04:04:80:28:BF:1F:28:64:D4:8F:9A:D4:D8:32:94:36:6A:82:88:56:55:3F:3B:14:30:3F:90:14:7F:5D:40:E

Alias name: securesignrootca11
Certificate fingerprints:
  MD5:  B7:52:74:E2:92:B4:80:93:F2:75:E4:CC:D7:F2:EA:26
  SHA1: 3B:C4:9F:48:F8:F3:73:A0:9C:1E:BD:F8:5B:B1:C3:65:C7:D8:11:B3
  SHA256:
 BF:0F:EE:FB:9E:3A:58:1A:D5:F9:E9:DB:75:89:98:57:43:D2:61:08:5C:4D:31:4F:6F:5D:72:59:AA:42:16:1

Alias name: amazon-ca-g4-acm2
Certificate fingerprints:
  MD5:  B2:F1:03:2B:93:64:05:80:B8:A8:17:36:B9:1B:52:3C
  SHA1: A7:E6:45:32:1F:7A:B7:AD:C0:70:EA:73:5F:AB:ED:C3:DA:B4:D0:C8
  SHA256:
 D7:A8:7C:69:95:D0:E2:04:2A:32:70:A7:E2:87:FE:A7:E8:F4:C1:70:62:F7:90:C3:EB:BB:53:F2:AC:39:26:B

Alias name: isrgrootx1
Certificate fingerprints:
  MD5:  0C:D2:F9:E0:DA:17:73:E9:ED:86:4D:A5:E3:70:E7:4E
```

```
  SHA1: CA:BD:2A:79:A1:07:6A:31:F2:1D:25:36:35:CB:03:9D:43:29:A5:E8
  SHA256:
 96:BC:EC:06:26:49:76:F3:74:60:77:9A:CF:28:C5:A7:CF:E8:A3:C0:AA:E1:1A:8F:FC:EE:05:C0:BD:DF:08:C

Alias name: amazon-ca-g4-acm1
Certificate fingerprints:
  MD5:  E2:F1:18:19:61:5C:43:E0:D4:A8:5D:0B:FA:7C:89:1B
  SHA1: F2:0D:28:B6:29:C2:2C:5E:84:05:E6:02:4D:97:FE:8F:A0:84:93:A0
  SHA256:
 B0:11:A4:F7:29:6C:74:D8:2B:F5:62:DF:87:D7:28:C7:1F:B5:8C:F4:E6:73:F2:78:FC:DA:F3:FF:83:A6:8C:8

Alias name: etugracertificationauthority
Certificate fingerprints:
  MD5:  B8:A1:03:63:B0:BD:21:71:70:8A:6F:13:3A:BB:79:49
  SHA1: 51:C6:E7:08:49:06:6E:F3:92:D4:5C:A0:0D:6D:A3:62:8F:C3:52:39
  SHA256:
 B0:BF:D5:2B:B0:D7:D9:BD:92:BF:5D:4D:C1:3D:A2:55:C0:2C:54:2F:37:83:65:EA:89:39:11:F5:5E:55:F2:3

Alias name: geotrustuniversalca2
Certificate fingerprints:
  MD5:  34:FC:B8:D0:36:DB:9E:14:B3:C2:F2:DB:8F:E4:94:C7
  SHA1: 37:9A:19:7B:41:85:45:35:0C:A6:03:69:F3:3C:2E:AF:47:4F:20:79
  SHA256:
 A0:23:4F:3B:C8:52:7C:A5:62:8E:EC:81:AD:5D:69:89:5D:A5:68:0D:C9:1D:1C:B8:47:7F:33:F8:78:B9:5B:0

Alias name: digicertglobalrootca
Certificate fingerprints:
  MD5:  79:E4:A9:84:0D:7D:3A:96:D7:C0:4F:E2:43:4C:89:2E
  SHA1: A8:98:5D:3A:65:E5:E5:C4:B2:D7:D6:6D:40:C6:DD:2F:B1:9C:54:36
  SHA256:
 43:48:A0:E9:44:4C:78:CB:26:5E:05:8D:5E:89:44:B4:D8:4F:96:62:BD:26:DB:25:7F:89:34:A4:43:C7:01:6

Alias name: staatdernederlandenevrootca
Certificate fingerprints:
  MD5:  FC:06:AF:7B:E8:1A:F1:9A:B4:E8:D2:70:1F:C0:F5:BA
  SHA1: 76:E2:7E:C1:4F:DB:82:C1:C0:A6:75:B5:05:BE:3D:29:B4:ED:DB:BB
  SHA256:
 4D:24:91:41:4C:FE:95:67:46:EC:4C:EF:A6:CF:6F:72:E2:8A:13:29:43:2F:9D:8A:90:7A:C4:CB:5D:AD:C1:5

Alias name: utnuserfirstclientauthemailca
Certificate fingerprints:
  MD5:  D7:34:3D:EF:1D:27:09:28:E1:31:02:5B:13:2B:DD:F7
  SHA1: B1:72:B1:A5:6D:95:F9:1F:E5:02:87:E1:4D:37:EA:6A:44:63:76:8A
```

```
  SHA256:
 43:F2:57:41:2D:44:0D:62:74:76:97:4F:87:7D:A8:F1:FC:24:44:56:5A:36:7A:E6:0E:DD:C2:7A:41:25:31:A

Alias name: actalisauthenticationrootca
Certificate fingerprints:
  MD5:  69:C1:0D:4F:07:A3:1B:C3:FE:56:3D:04:BC:11:F6:A6
  SHA1: F3:73:B3:87:06:5A:28:84:8A:F2:F3:4A:CE:19:2B:DD:C7:8E:9C:AC
  SHA256:
 55:92:60:84:EC:96:3A:64:B9:6E:2A:BE:01:CE:0B:A8:6A:64:FB:FE:BC:C7:AA:B5:AF:C1:55:B3:7F:D7:60:6

Alias name: amazonrootca4
Certificate fingerprints:
  MD5:  89:BC:27:D5:EB:17:8D:06:6A:69:D5:FD:89:47:B4:CD
  SHA1: F6:10:84:07:D6:F8:BB:67:98:0C:C2:E2:44:C2:EB:AE:1C:EF:63:BE
  SHA256:
 E3:5D:28:41:9E:D0:20:25:CF:A6:90:38:CD:62:39:62:45:8D:A5:C6:95:FB:DE:A3:C2:2B:0B:FB:25:89:70:9

Alias name: amazonrootca3
Certificate fingerprints:
  MD5:  A0:D4:EF:0B:F7:B5:D8:49:95:2A:EC:F5:C4:FC:81:87
  SHA1: 0D:44:DD:8C:3C:8C:1A:1A:58:75:64:81:E9:0F:2E:2A:FF:B3:D2:6E
  SHA256:
 18:CE:6C:FE:7B:F1:4E:60:B2:E3:47:B8:DF:E8:68:CB:31:D0:2E:BB:3A:DA:27:15:69:F5:03:43:B4:6D:B3:A

Alias name: amazonrootca2
Certificate fingerprints:
  MD5:  C8:E5:8D:CE:A8:42:E2:7A:C0:2A:5C:7C:9E:26:BF:66
  SHA1: 5A:8C:EF:45:D7:A6:98:59:76:7A:8C:8B:44:96:B5:78:CF:47:4B:1A
  SHA256:
 1B:A5:B2:AA:8C:65:40:1A:82:96:01:18:F8:0B:EC:4F:62:30:4D:83:CE:C4:71:3A:19:C3:9C:01:1E:A4:6D:B

Alias name: amazonrootca1
Certificate fingerprints:
  MD5:  43:C6:BF:AE:EC:FE:AD:2F:18:C6:88:68:30:FC:C8:E6
  SHA1: 8D:A7:F9:65:EC:5E:FC:37:91:0F:1C:6E:59:FD:C1:CC:6A:6E:DE:16
  SHA256:
 8E:CD:E6:88:4F:3D:87:B1:12:5B:A3:1A:C3:FC:B1:3D:70:16:DE:7F:57:CC:90:4F:E1:CB:97:C6:AE:98:19:6

Alias name: affirmtrustpremium
Certificate fingerprints:
  MD5:  C4:5D:0E:48:B6:AC:28:30:4E:0A:BC:F9:38:16:87:57
  SHA1: D8:A6:33:2C:E0:03:6F:B1:85:F6:63:4F:7D:6A:06:65:26:32:28:27
  SHA256:
 70:A7:3F:7F:37:6B:60:07:42:48:90:45:34:B1:14:82:D5:BF:0E:69:8E:CC:49:8D:F5:25:77:EB:F2:E9:3B:9
```

```
Alias name: keynectisrootca
Certificate fingerprints:
  MD5:  CC:4D:AE:FB:30:6B:D8:38:FE:50:EB:86:61:4B:D2:26
  SHA1: 9C:61:5C:4D:4D:85:10:3A:53:26:C2:4D:BA:EA:E4:A2:D2:D5:CC:97
  SHA256:
 42:10:F1:99:49:9A:9A:C3:3C:8D:E0:2B:A6:DB:AA:14:40:8B:DD:8A:6E:32:46:89:C1:92:2D:06:97:15:A3:3

Alias name: equifaxsecureglobalebusinessca1
Certificate fingerprints:
  MD5:  51:F0:2A:33:F1:F5:55:39:07:F2:16:7A:47:C7:5D:63
  SHA1: 3A:74:CB:7A:47:DB:70:DE:89:1F:24:35:98:64:B8:2D:82:BD:1A:36
  SHA256:
 86:AB:5A:65:71:D3:32:9A:BC:D2:E4:E6:37:66:8B:A8:9C:73:1E:C2:93:B6:CB:A6:0F:71:63:40:A0:91:CE:A

Alias name: affirmtrustpremiumca
Certificate fingerprints:
  MD5:  C4:5D:0E:48:B6:AC:28:30:4E:0A:BC:F9:38:16:87:57
  SHA1: D8:A6:33:2C:E0:03:6F:B1:85:F6:63:4F:7D:6A:06:65:26:32:28:27
  SHA256:
 70:A7:3F:7F:37:6B:60:07:42:48:90:45:34:B1:14:82:D5:BF:0E:69:8E:CC:49:8D:F5:25:77:EB:F2:E9:3B:9

Alias name: baltimorecodesigningca
Certificate fingerprints:
  MD5:  90:F5:28:49:56:D1:5D:2C:B0:53:D4:4B:EF:6F:90:22
  SHA1: 30:46:D8:C8:88:FF:69:30:C3:4A:FC:CD:49:27:08:7C:60:56:7B:0D
  SHA256:
 A9:15:45:DB:D2:E1:9C:4C:CD:F9:09:AA:71:90:0D:18:C7:35:1C:89:B3:15:F0:F1:3D:05:C1:3A:8F:FB:46:8

Alias name: gdcatrustauthr5root
Certificate fingerprints:
  MD5:  63:CC:D9:3D:34:35:5C:6F:53:A3:E2:08:70:48:1F:B4
  SHA1: 0F:36:38:5B:81:1A:25:C3:9B:31:4E:83:CA:E9:34:66:70:CC:74:B4
  SHA256:
 BF:FF:8F:D0:44:33:48:7D:6A:8A:A6:0C:1A:29:76:7A:9F:C2:BB:B0:5E:42:0F:71:3A:13:B9:92:89:1D:38:9

Alias name: certinomisrootca
Certificate fingerprints:
  MD5:  14:0A:FD:8D:A8:28:B5:38:69:DB:56:7E:61:22:03:3F
  SHA1: 9D:70:BB:01:A5:A4:A0:18:11:2E:F7:1C:01:B9:32:C5:34:E7:88:A8
  SHA256:
 2A:99:F5:BC:11:74:B7:3C:BB:1D:62:08:84:E0:1C:34:E5:1C:CB:39:78:DA:12:5F:0E:33:26:88:83:BF:41:5

Alias name: verisignclass3publicprimarycertificationauthorityg5
```

```
Certificate fingerprints:
  MD5:  CB:17:E4:31:67:3E:E2:09:FE:45:57:93:F3:0A:FA:1C
  SHA1: 4E:B6:D5:78:49:9B:1C:CF:5F:58:1E:AD:56:BE:3D:9B:67:44:A5:E5
  SHA256:
 9A:CF:AB:7E:43:C8:D8:80:D0:6B:26:2A:94:DE:EE:E4:B4:65:99:89:C3:D0:CA:F1:9B:AF:64:05:E4:1A:B7:D

Alias name: verisignclass3publicprimarycertificationauthorityg4
Certificate fingerprints:
  MD5:  3A:52:E1:E7:FD:6F:3A:E3:6F:F3:6F:99:1B:F9:22:41
  SHA1: 22:D5:D8:DF:8F:02:31:D1:8D:F7:9D:B7:CF:8A:2D:64:C9:3F:6C:3A
  SHA256:
 69:DD:D7:EA:90:BB:57:C9:3E:13:5D:C8:5E:A6:FC:D5:48:0B:60:32:39:BD:C4:54:FC:75:8B:2A:26:CF:7F:7

Alias name: verisignclass3publicprimarycertificationauthorityg3
Certificate fingerprints:
  MD5:  CD:68:B6:A7:C7:C4:CE:75:E0:1D:4F:57:44:61:92:09
  SHA1: 13:2D:0D:45:53:4B:69:97:CD:B2:D5:C3:39:E2:55:76:60:9B:5C:C6
  SHA256:
 EB:04:CF:5E:B1:F3:9A:FA:76:2F:2B:B1:20:F2:96:CB:A5:20:C1:B9:7D:B1:58:95:65:B8:1C:B9:A1:7B:72:4

Alias name: swisssignsilverg2ca
Certificate fingerprints:
  MD5:  E0:06:A1:C9:7D:CF:C9:FC:0D:C0:56:75:96:D8:62:13
  SHA1: 9B:AA:E5:9F:56:EE:21:CB:43:5A:BE:25:93:DF:A7:F0:40:D1:1D:CB
  SHA256:
 BE:6C:4D:A2:BB:B9:BA:59:B6:F3:93:97:68:37:42:46:C3:C0:05:99:3F:A9:8F:02:0D:1D:ED:BE:D4:8A:81:D

Alias name: swisssignsilvercag2
Certificate fingerprints:
  MD5:  E0:06:A1:C9:7D:CF:C9:FC:0D:C0:56:75:96:D8:62:13
  SHA1: 9B:AA:E5:9F:56:EE:21:CB:43:5A:BE:25:93:DF:A7:F0:40:D1:1D:CB
  SHA256:
 BE:6C:4D:A2:BB:B9:BA:59:B6:F3:93:97:68:37:42:46:C3:C0:05:99:3F:A9:8F:02:0D:1D:ED:BE:D4:8A:81:D

Alias name: atostrustedroot2011
Certificate fingerprints:
  MD5:  AE:B9:C4:32:4B:AC:7F:5D:66:CC:77:94:BB:2A:77:56
  SHA1: 2B:B1:F5:3E:55:0C:1D:C5:F1:D4:E6:B7:6A:46:4B:55:06:02:AC:21
  SHA256:
 F3:56:BE:A2:44:B7:A9:1E:B3:5D:53:CA:9A:D7:86:4A:CE:01:8E:2D:35:D5:F8:F9:6D:DF:68:A6:F4:1A:A4:7

Alias name: comodoecccertificationauthority
Certificate fingerprints:
  MD5:  7C:62:FF:74:9D:31:53:5E:68:4A:D5:78:AA:1E:BF:23
```

```
  SHA1: 9F:74:4E:9F:2B:4D:BA:EC:0F:31:2C:50:B6:56:3B:8E:2D:93:C3:11
  SHA256:
 17:93:92:7A:06:14:54:97:89:AD:CE:2F:8F:34:F7:F0:B6:6D:0F:3A:E3:A3:B8:4D:21:EC:15:DB:BA:4F:AD:C

Alias name: securetrustca
Certificate fingerprints:
  MD5:  DC:32:C3:A7:6D:25:57:C7:68:09:9D:EA:2D:A9:A2:D1
  SHA1: 87:82:C6:C3:04:35:3B:CF:D2:96:92:D2:59:3E:7D:44:D9:34:FF:11
  SHA256:
 F1:C1:B5:0A:E5:A2:0D:D8:03:0E:C9:F6:BC:24:82:3D:D3:67:B5:25:57:59:B4:E7:1B:61:FC:E9:F7:37:5D:7

Alias name: soneraclass1ca
Certificate fingerprints:
  MD5:  33:B7:84:F5:5F:27:D7:68:27:DE:14:DE:12:2A:ED:6F
  SHA1: 07:47:22:01:99:CE:74:B9:7C:B0:3D:79:B2:64:A2:C8:55:E9:33:FF
  SHA256:
 CD:80:82:84:CF:74:6F:F2:FD:6E:B5:8A:A1:D5:9C:4A:D4:B3:CA:56:FD:C6:27:4A:89:26:A7:83:5F:32:31:3

Alias name: cadisigrootr2
Certificate fingerprints:
  MD5:  26:01:FB:D8:27:A7:17:9A:45:54:38:1A:43:01:3B:03
  SHA1: B5:61:EB:EA:A4:DE:E4:25:4B:69:1A:98:A5:57:47:C2:34:C7:D9:71
  SHA256:
 E2:3D:4A:03:6D:7B:70:E9:F5:95:B1:42:20:79:D2:B9:1E:DF:BB:1F:B6:51:A0:63:3E:AA:8A:9D:C5:F8:07:0

Alias name: cadisigrootr1
Certificate fingerprints:
  MD5:  BE:EC:11:93:9A:F5:69:21:BC:D7:C1:C0:67:89:CC:2A
  SHA1: 8E:1C:74:F8:A6:20:B9:E5:8A:F4:61:FA:EC:2B:47:56:51:1A:52:C6
  SHA256:
 F9:6F:23:F4:C3:E7:9C:07:7A:46:98:8D:5A:F5:90:06:76:A0:F0:39:CB:64:5D:D1:75:49:B2:16:C8:24:40:C

Alias name: verisignclass3g5ca
Certificate fingerprints:
  MD5:  CB:17:E4:31:67:3E:E2:09:FE:45:57:93:F3:0A:FA:1C
  SHA1: 4E:B6:D5:78:49:9B:1C:CF:5F:58:1E:AD:56:BE:3D:9B:67:44:A5:E5
  SHA256:
 9A:CF:AB:7E:43:C8:D8:80:D0:6B:26:2A:94:DE:EE:E4:B4:65:99:89:C3:D0:CA:F1:9B:AF:64:05:E4:1A:B7:D

Alias name: utnuserfirsthardwareca
Certificate fingerprints:
  MD5:  4C:56:41:E5:0D:BB:2B:E8:CA:A3:ED:18:08:AD:43:39
  SHA1: 04:83:ED:33:99:AC:36:08:05:87:22:ED:BC:5E:46:00:E3:BE:F9:D7
```

```
  SHA256:
 6E:A5:47:41:D0:04:66:7E:ED:1B:48:16:63:4A:A3:A7:9E:6E:4B:96:95:0F:82:79:DA:FC:8D:9B:D8:81:21:3

Alias name: addtrustqualifiedca
Certificate fingerprints:
  MD5:  27:EC:39:47:CD:DA:5A:AF:E2:9A:01:65:21:A9:4C:BB
  SHA1: 4D:23:78:EC:91:95:39:B5:00:7F:75:8F:03:3B:21:1E:C5:4D:8B:CF
  SHA256:
 80:95:21:08:05:DB:4B:BC:35:5E:44:28:D8:FD:6E:C2:CD:E3:AB:5F:B9:7A:99:42:98:8E:B8:F4:DC:D0:60:1

Alias name: verisignclass3g3ca
Certificate fingerprints:
  MD5:  CD:68:B6:A7:C7:C4:CE:75:E0:1D:4F:57:44:61:92:09
  SHA1: 13:2D:0D:45:53:4B:69:97:CD:B2:D5:C3:39:E2:55:76:60:9B:5C:C6
  SHA256:
 EB:04:CF:5E:B1:F3:9A:FA:76:2F:2B:B1:20:F2:96:CB:A5:20:C1:B9:7D:B1:58:95:65:B8:1C:B9:A1:7B:72:4

Alias name: thawtepersonalfreemailca
Certificate fingerprints:
  MD5:  53:4B:1D:17:58:58:1A:30:A1:90:F8:6E:5C:F2:CF:65
  SHA1: E6:18:83:AE:84:CA:C1:C1:CD:52:AD:E8:E9:25:2B:45:A6:4F:B7:E2
  SHA256:
 5B:38:BD:12:9E:83:D5:A0:CA:D2:39:21:08:94:90:D5:0D:4A:AE:37:04:28:F8:DD:FF:FF:FA:4C:15:64:E1:8

Alias name: certplusclass3pprimaryca
Certificate fingerprints:
  MD5:  E1:4B:52:73:D7:1B:DB:93:30:E5:BD:E4:09:6E:BE:FB
  SHA1: 21:6B:2A:29:E6:2A:00:CE:82:01:46:D8:24:41:41:B9:25:11:B2:79
  SHA256:
 CC:C8:94:89:37:1B:AD:11:1C:90:61:9B:EA:24:0A:2E:6D:AD:D9:9F:9F:6E:1D:4D:41:E5:8E:D6:DE:3D:02:8

Alias name: swisssigngoldg2ca
Certificate fingerprints:
  MD5:  24:77:D9:A8:91:D1:3B:FA:88:2D:C2:FF:F8:CD:33:93
  SHA1: D8:C5:38:8A:B7:30:1B:1B:6E:D4:7A:E6:45:25:3A:6F:9F:1A:27:61
  SHA256:
 62:DD:0B:E9:B9:F5:0A:16:3E:A0:F8:E7:5C:05:3B:1E:CA:57:EA:55:C8:68:8F:64:7C:68:81:F2:C8:35:7B:9

Alias name: swisssigngoldcag2
Certificate fingerprints:
  MD5:  24:77:D9:A8:91:D1:3B:FA:88:2D:C2:FF:F8:CD:33:93
  SHA1: D8:C5:38:8A:B7:30:1B:1B:6E:D4:7A:E6:45:25:3A:6F:9F:1A:27:61
  SHA256:
 62:DD:0B:E9:B9:F5:0A:16:3E:A0:F8:E7:5C:05:3B:1E:CA:57:EA:55:C8:68:8F:64:7C:68:81:F2:C8:35:7B:9
```

```
Alias name: dtrustrootclass3ca22009
Certificate fingerprints:
  MD5:  CD:E0:25:69:8D:47:AC:9C:89:35:90:F7:FD:51:3D:2F
  SHA1: 58:E8:AB:B0:36:15:33:FB:80:F7:9B:1B:6D:29:D3:FF:8D:5F:00:F0
  SHA256:
 49:E7:A4:42:AC:F0:EA:62:87:05:00:54:B5:25:64:B6:50:E4:F4:9E:42:E3:48:D6:AA:38:E0:39:E9:57:B1:C

Alias name: acraizfnmtrcm
Certificate fingerprints:
  MD5:  E2:09:04:B4:D3:BD:D1:A0:14:FD:1A:D2:47:C4:57:1D
  SHA1: EC:50:35:07:B2:15:C4:95:62:19:E2:A8:9A:5B:42:99:2C:4C:2C:20
  SHA256:
 EB:C5:57:0C:29:01:8C:4D:67:B1:AA:12:7B:AF:12:F7:03:B4:61:1E:BC:17:B7:DA:B5:57:38:94:17:9B:93:F

Alias name: securitycommunicationevrootca1
Certificate fingerprints:
  MD5:  22:2D:A6:01:EA:7C:0A:F7:F0:6C:56:43:3F:77:76:D3
  SHA1: FE:B8:C4:32:DC:F9:76:9A:CE:AE:3D:D8:90:8F:FD:28:86:65:64:7D
  SHA256:
 A2:2D:BA:68:1E:97:37:6E:2D:39:7D:72:8A:AE:3A:9B:62:96:B9:FD:BA:60:BC:2E:11:F6:47:F2:C6:75:FB:3

Alias name: starfieldclass2ca
Certificate fingerprints:
  MD5:  32:4A:4B:BB:C8:63:69:9B:BE:74:9A:C6:DD:1D:46:24
  SHA1: AD:7E:1C:28:B0:64:EF:8F:60:03:40:20:14:C3:D0:E3:37:0E:B5:8A
  SHA256:
 14:65:FA:20:53:97:B8:76:FA:A6:F0:A9:95:8E:55:90:E4:0F:CC:7F:AA:4F:B7:C2:C8:67:75:21:FB:5F:B6:5

Alias name: opentrustrootcag3
Certificate fingerprints:
  MD5:  21:37:B4:17:16:92:7B:67:46:70:A9:96:D7:A8:13:24
  SHA1: 6E:26:64:F3:56:BF:34:55:BF:D1:93:3F:7C:01:DE:D8:13:DA:8A:A6
  SHA256:
 B7:C3:62:31:70:6E:81:07:8C:36:7C:B8:96:19:8F:1E:32:08:DD:92:69:49:DD:8F:57:09:A4:10:F7:5B:62:9

Alias name: opentrustrootcag2
Certificate fingerprints:
  MD5:  57:24:B6:59:24:6B:AE:C8:FE:1C:0C:20:F2:C0:4E:EB
  SHA1: 79:5F:88:60:C5:AB:7C:3D:92:E6:CB:F4:8D:E1:45:CD:11:EF:60:0B
  SHA256:
 27:99:58:29:FE:6A:75:15:C1:BF:E8:48:F9:C4:76:1D:B1:6C:22:59:29:25:7B:F4:0D:08:94:F2:9E:A8:BA:F

Alias name: buypassclass2rootca
```

```
Certificate fingerprints:
  MD5:  46:A7:D2:FE:45:FB:64:5A:A8:59:90:9B:78:44:9B:29
  SHA1: 49:0A:75:74:DE:87:0A:47:FE:58:EE:F6:C7:6B:EB:C6:0B:12:40:99
  SHA256:
 9A:11:40:25:19:7C:5B:B9:5D:94:E6:3D:55:CD:43:79:08:47:B6:46:B2:3C:DF:11:AD:A4:A0:0E:FF:15:FB:4

Alias name: opentrustrootcag1
Certificate fingerprints:
  MD5:  76:00:CC:81:29:CD:55:5E:88:6A:7A:2E:F7:4D:39:DA
  SHA1: 79:91:E8:34:F7:E2:EE:DD:08:95:01:52:E9:55:2D:14:E9:58:D5:7E
  SHA256:
 56:C7:71:28:D9:8C:18:D9:1B:4C:FD:FF:BC:25:EE:91:03:D4:75:8E:A2:AB:AD:82:6A:90:F3:45:7D:46:0E:B

Alias name: globalsignr2ca
Certificate fingerprints:
  MD5:  94:14:77:7E:3E:5E:FD:8F:30:BD:41:B0:CF:E7:D0:30
  SHA1: 75:E0:AB:B6:13:85:12:27:1C:04:F8:5F:DD:DE:38:E4:B7:24:2E:FE
  SHA256:
 CA:42:DD:41:74:5F:D0:B8:1E:B9:02:36:2C:F9:D8:BF:71:9D:A1:BD:1B:1E:FC:94:6F:5B:4C:99:F4:2C:1B:9

Alias name: buypassclass3rootca
Certificate fingerprints:
  MD5:  3D:3B:18:9E:2C:64:5A:E8:D5:88:CE:0E:F9:37:C2:EC
  SHA1: DA:FA:F7:FA:66:84:EC:06:8F:14:50:BD:C7:C2:81:A5:BC:A9:64:57
  SHA256:
 ED:F7:EB:BC:A2:7A:2A:38:4D:38:7B:7D:40:10:C6:66:E2:ED:B4:84:3E:4C:29:B4:AE:1D:5B:93:32:E6:B2:4

Alias name: ecacc
Certificate fingerprints:
  MD5:  EB:F5:9D:29:0D:61:F9:42:1F:7C:C2:BA:6D:E3:15:09
  SHA1: 28:90:3A:63:5B:52:80:FA:E6:77:4C:0B:6D:A7:D6:BA:A6:4A:F2:E8
  SHA256:
 88:49:7F:01:60:2F:31:54:24:6A:E2:8C:4D:5A:EF:10:F1:D8:7E:BB:76:62:6F:4A:E0:B7:F9:5B:A7:96:87:9

Alias name: epkirootcertificationauthority
Certificate fingerprints:
  MD5:  1B:2E:00:CA:26:06:90:3D:AD:FE:6F:15:68:D3:6B:B3
  SHA1: 67:65:0D:F1:7E:8E:7E:5B:82:40:A4:F4:56:4B:CF:E2:3D:69:C6:F0
  SHA256:
 C0:A6:F4:DC:63:A2:4B:FD:CF:54:EF:2A:6A:08:2A:0A:72:DE:35:80:3E:2F:F5:FF:52:7A:E5:D8:72:06:DF:D

Alias name: verisignclass1g2ca
Certificate fingerprints:
  MD5:  DB:23:3D:F9:69:FA:4B:B9:95:80:44:73:5E:7D:41:83
```

```
  SHA1: 27:3E:E1:24:57:FD:C4:F9:0C:55:E8:2B:56:16:7F:62:F5:32:E5:47
  SHA256:
 34:1D:E9:8B:13:92:AB:F7:F4:AB:90:A9:60:CF:25:D4:BD:6E:C6:5B:9A:51:CE:6E:D0:67:D0:0E:C7:CE:9B:7

Alias name: certigna
Certificate fingerprints:
  MD5:  AB:57:A6:5B:7D:42:82:19:B5:D8:58:26:28:5E:FD:FF
  SHA1: B1:2E:13:63:45:86:A4:6F:1A:B2:60:68:37:58:2D:C4:AC:FD:94:97
  SHA256:
 E3:B6:A2:DB:2E:D7:CE:48:84:2F:7A:C5:32:41:C7:B7:1D:54:14:4B:FB:40:C1:1F:3F:1D:0B:42:F5:EE:A1:2

Alias name: camerfirmaglobalchambersignroot
Certificate fingerprints:
  MD5:  C5:E6:7B:BF:06:D0:4F:43:ED:C4:7A:65:8A:FB:6B:19
  SHA1: 33:9B:6B:14:50:24:9B:55:7A:01:87:72:84:D9:E0:2F:C3:D2:D8:E9
  SHA256:
 EF:3C:B4:17:FC:8E:BF:6F:97:87:6C:9E:4E:CE:39:DE:1E:A5:FE:64:91:41:D1:02:8B:7D:11:C0:B2:29:8C:E

Alias name: cfcaevroot
Certificate fingerprints:
  MD5:  74:E1:B6:ED:26:7A:7A:44:30:33:94:AB:7B:27:81:30
  SHA1: E2:B8:29:4B:55:84:AB:6B:58:C2:90:46:6C:AC:3F:B8:39:8F:84:83
  SHA256:
 5C:C3:D7:8E:4E:1D:5E:45:54:7A:04:E6:87:3E:64:F9:0C:F9:53:6D:1C:CC:2E:F8:00:F3:55:C4:C5:FD:70:F

Alias name: soneraclass2rootca
Certificate fingerprints:
  MD5:  A3:EC:75:0F:2E:88:DF:FA:48:01:4E:0B:5C:48:6F:FB
  SHA1: 37:F7:6D:E6:07:7C:90:C5:B1:3E:93:1A:B7:41:10:B4:F2:E4:9A:27
  SHA256:
 79:08:B4:03:14:C1:38:10:0B:51:8D:07:35:80:7F:FB:FC:F8:51:8A:00:95:33:71:05:BA:38:6B:15:3D:D9:2

Alias name: certumtrustednetworkca
Certificate fingerprints:
  MD5:  D5:E9:81:40:C5:18:69:FC:46:2C:89:75:62:0F:AA:78
  SHA1: 07:E0:32:E0:20:B7:2C:3F:19:2F:06:28:A2:59:3A:19:A7:0F:06:9E
  SHA256:
 5C:58:46:8D:55:F5:8E:49:7E:74:39:82:D2:B5:00:10:B6:D1:65:37:4A:CF:83:A7:D4:A3:2D:B7:68:C4:40:8

Alias name: securitycommunicationrootca2
Certificate fingerprints:
  MD5:  6C:39:7D:A4:0E:55:59:B2:3F:D6:41:B1:12:50:DE:43
  SHA1: 5F:3B:8C:F2:F8:10:B3:7D:78:B4:CE:EC:19:19:C3:73:34:B9:C7:74
```

```
  SHA256:
 51:3B:2C:EC:B8:10:D4:CD:E5:DD:85:39:1A:DF:C6:C2:DD:60:D8:7B:B7:36:D2:B5:21:48:4A:A4:7A:0E:BE:F

Alias name: globalsigneccrootcar5
Certificate fingerprints:
  MD5:  9F:AD:3B:1C:02:1E:8A:BA:17:74:38:81:0C:A2:BC:08
  SHA1: 1F:24:C6:30:CD:A4:18:EF:20:69:FF:AD:4F:DD:5F:46:3A:1B:69:AA
  SHA256:
 17:9F:BC:14:8A:3D:D0:0F:D2:4E:A1:34:58:CC:43:BF:A7:F5:9C:81:82:D7:83:A5:13:F6:EB:EC:10:0C:89:2

Alias name: globalsigneccrootcar4
Certificate fingerprints:
  MD5:  20:F0:27:68:D1:7E:A0:9D:0E:E6:2A:CA:DF:5C:89:8E
  SHA1: 69:69:56:2E:40:80:F4:24:A1:E7:19:9F:14:BA:F3:EE:58:AB:6A:BB
  SHA256:
 BE:C9:49:11:C2:95:56:76:DB:6C:0A:55:09:86:D7:6E:3B:A0:05:66:7C:44:2C:97:62:B4:FB:B7:73:DE:22:8

Alias name: chambersofcommerceroot2008
Certificate fingerprints:
  MD5:  5E:80:9E:84:5A:0E:65:0B:17:02:F3:55:18:2A:3E:D7
  SHA1: 78:6A:74:AC:76:AB:14:7F:9C:6A:30:50:BA:9E:A8:7E:FE:9A:CE:3C
  SHA256:
 06:3E:4A:FA:C4:91:DF:D3:32:F3:08:9B:85:42:E9:46:17:D8:93:D7:FE:94:4E:10:A7:93:7E:E2:9D:96:93:C

Alias name: pscprocert
Certificate fingerprints:
  MD5:  E6:24:E9:12:01:AE:0C:DE:8E:85:C4:CE:A3:12:DD:EC
  SHA1: 70:C1:8D:74:B4:28:81:0A:E4:FD:A5:75:D7:01:9F:99:B0:3D:50:74
  SHA256:
 3C:FC:3C:14:D1:F6:84:FF:17:E3:8C:43:CA:44:0C:00:B9:67:EC:93:3E:8B:FE:06:4C:A1:D7:2C:90:F2:AD:B

Alias name: thawteprimaryrootcag3
Certificate fingerprints:
  MD5:  FB:1B:5D:43:8A:94:CD:44:C6:76:F2:43:4B:47:E7:31
  SHA1: F1:8B:53:8D:1B:E9:03:B6:A6:F0:56:43:5B:17:15:89:CA:F3:6B:F2
  SHA256:
 4B:03:F4:58:07:AD:70:F2:1B:FC:2C:AE:71:C9:FD:E4:60:4C:06:4C:F5:FF:B6:86:BA:E5:DB:AA:D7:FD:D3:4

Alias name: quovadisrootca
Certificate fingerprints:
  MD5:  27:DE:36:FE:72:B7:00:03:00:9D:F4:F0:1E:6C:04:24
  SHA1: DE:3F:40:BD:50:93:D3:9B:6C:60:F6:DA:BC:07:62:01:00:89:76:C9
  SHA256:
 A4:5E:DE:3B:BB:F0:9C:8A:E1:5C:72:EF:C0:72:68:D6:93:A2:1C:99:6F:D5:1E:67:CA:07:94:60:FD:6D:88:7
```

```
Alias name: thawteprimaryrootcag2
Certificate fingerprints:
  MD5:  74:9D:EA:60:24:C4:FD:22:53:3E:CC:3A:72:D9:29:4F
  SHA1: AA:DB:BC:22:23:8F:C4:01:A1:27:BB:38:DD:F4:1D:DB:08:9E:F0:12
  SHA256:
 A4:31:0D:50:AF:18:A6:44:71:90:37:2A:86:AF:AF:8B:95:1F:FB:43:1D:83:7F:1E:56:88:B4:59:71:ED:15:5

Alias name: deprecateditsecca
Certificate fingerprints:
  MD5:  A5:96:0C:F6:B5:AB:27:E5:01:C6:00:88:9E:60:33:E5
  SHA1: 12:12:0B:03:0E:15:14:54:F4:DD:B3:F5:DE:13:6E:83:5A:29:72:9D
  SHA256:
 9A:59:DA:86:24:1A:FD:BA:A3:39:FA:9C:FD:21:6A:0B:06:69:4D:E3:7E:37:52:6B:BE:63:C8:BC:83:74:2E:C

Alias name: usertrustrsacertificationauthority
Certificate fingerprints:
  MD5:  1B:FE:69:D1:91:B7:19:33:A3:72:A8:0F:E1:55:E5:B5
  SHA1: 2B:8F:1B:57:33:0D:BB:A2:D0:7A:6C:51:F7:0E:E9:0D:DA:B9:AD:8E
  SHA256:
 E7:93:C9:B0:2F:D8:AA:13:E2:1C:31:22:8A:CC:B0:81:19:64:3B:74:9C:89:89:64:B1:74:6D:46:C3:D4:CB:D

Alias name: entrustrootcag2
Certificate fingerprints:
  MD5:  4B:E2:C9:91:96:65:0C:F4:0E:5A:93:92:A0:0A:FE:B2
  SHA1: 8C:F4:27:FD:79:0C:3A:D1:66:06:8D:E8:1E:57:EF:BB:93:22:72:D4
  SHA256:
 43:DF:57:74:B0:3E:7F:EF:5F:E4:0D:93:1A:7B:ED:F1:BB:2E:6B:42:73:8C:4E:6D:38:41:10:3D:3A:A7:F3:3

Alias name: networksolutionscertificateauthority
Certificate fingerprints:
  MD5:  D3:F3:A6:16:C0:FA:6B:1D:59:B1:2D:96:4D:0E:11:2E
  SHA1: 74:F8:A3:C3:EF:E7:B3:90:06:4B:83:90:3C:21:64:60:20:E5:DF:CE
  SHA256:
 15:F0:BA:00:A3:AC:7A:F3:AC:88:4C:07:2B:10:11:A0:77:BD:77:C0:97:F4:01:64:B2:F8:59:8A:BD:83:86:0

Alias name: trustcenterclass4caii
Certificate fingerprints:
  MD5:  9D:FB:F9:AC:ED:89:33:22:F4:28:48:83:25:23:5B:E0
  SHA1: A6:9A:91:FD:05:7F:13:6A:42:63:0B:B1:76:0D:2D:51:12:0C:16:50
  SHA256:
 32:66:96:7E:59:CD:68:00:8D:9D:D3:20:81:11:85:C7:04:20:5E:8D:95:FD:D8:4F:1C:7B:31:1E:67:04:FC:3

Alias name: oistewisekeyglobalrootgaca
```

```
Certificate fingerprints:
  MD5:  BC:6C:51:33:A7:E9:D3:66:63:54:15:72:1B:21:92:93
  SHA1: 59:22:A1:E1:5A:EA:16:35:21:F8:98:39:6A:46:46:B0:44:1B:0F:A9
  SHA256:
 41:C9:23:86:6A:B4:CA:D6:B7:AD:57:80:81:58:2E:02:07:97:A6:CB:DF:4F:FF:78:CE:83:96:B3:89:37:D7:F

Alias name: verisignuniversalrootcertificationauthority
Certificate fingerprints:
  MD5:  8E:AD:B5:01:AA:4D:81:E4:8C:1D:D1:E1:14:00:95:19
  SHA1: 36:79:CA:35:66:87:72:30:4D:30:A5:FB:87:3B:0F:A7:7B:B7:0D:54
  SHA256:
 23:99:56:11:27:A5:71:25:DE:8C:EF:EA:61:0D:DF:2F:A0:78:B5:C8:06:7F:4E:82:82:90:BF:B8:60:E8:4B:3

Alias name: ttelesecglobalrootclass3ca
Certificate fingerprints:
  MD5:  CA:FB:40:A8:4E:39:92:8A:1D:FE:8E:2F:C4:27:EA:EF
  SHA1: 55:A6:72:3E:CB:F2:EC:CD:C3:23:74:70:19:9D:2A:BE:11:E3:81:D1
  SHA256:
 FD:73:DA:D3:1C:64:4F:F1:B4:3B:EF:0C:CD:DA:96:71:0B:9C:D9:87:5E:CA:7E:31:70:7A:F3:E9:6D:52:2B:B

Alias name: starfieldservicesrootg2ca
Certificate fingerprints:
  MD5:  17:35:74:AF:7B:61:1C:EB:F4:F9:3C:E2:EE:40:F9:A2
  SHA1: 92:5A:8F:8D:2C:6D:04:E0:66:5F:59:6A:FF:22:D8:63:E8:25:6F:3F
  SHA256:
 56:8D:69:05:A2:C8:87:08:A4:B3:02:51:90:ED:CF:ED:B1:97:4A:60:6A:13:C6:E5:29:0F:CB:2A:E6:3E:DA:B

Alias name: addtrustexternalroot
Certificate fingerprints:
  MD5:  1D:35:54:04:85:78:B0:3F:42:42:4D:BF:20:73:0A:3F
  SHA1: 02:FA:F3:E2:91:43:54:68:60:78:57:69:4D:F5:E4:5B:68:85:18:68
  SHA256:
 68:7F:A4:51:38:22:78:FF:F0:C8:B1:1F:8D:43:D5:76:67:1C:6E:B2:BC:EA:B4:13:FB:83:D9:65:D0:6D:2F:F

Alias name: turktrustelektroniksertifikahizmetsaglayicisih5
Certificate fingerprints:
  MD5:  DA:70:8E:F0:22:DF:93:26:F6:5F:9F:D3:15:06:52:4E
  SHA1: C4:18:F6:4D:46:D1:DF:00:3D:27:30:13:72:43:A9:12:11:C6:75:FB
  SHA256:
 49:35:1B:90:34:44:C1:85:CC:DC:5C:69:3D:24:D8:55:5C:B2:08:D6:A8:14:13:07:69:9F:4A:F0:63:19:9D:7

Alias name: camerfirmachambersca
Certificate fingerprints:
  MD5:  5E:80:9E:84:5A:0E:65:0B:17:02:F3:55:18:2A:3E:D7
```

```
  SHA1: 78:6A:74:AC:76:AB:14:7F:9C:6A:30:50:BA:9E:A8:7E:FE:9A:CE:3C
  SHA256:
 06:3E:4A:FA:C4:91:DF:D3:32:F3:08:9B:85:42:E9:46:17:D8:93:D7:FE:94:4E:10:A7:93:7E:E2:9D:96:93:C

Alias name: certsignrootca
Certificate fingerprints:
  MD5:  18:98:C0:D6:E9:3A:FC:F9:B0:F5:0C:F7:4B:01:44:17
  SHA1: FA:B7:EE:36:97:26:62:FB:2D:B0:2A:F6:BF:03:FD:E8:7C:4B:2F:9B
  SHA256:
 EA:A9:62:C4:FA:4A:6B:AF:EB:E4:15:19:6D:35:1C:CD:88:8D:4F:53:F3:FA:8A:E6:D7:C4:66:A9:4E:60:42:B

Alias name: verisignuniversalrootca
Certificate fingerprints:
  MD5:  8E:AD:B5:01:AA:4D:81:E4:8C:1D:D1:E1:14:00:95:19
  SHA1: 36:79:CA:35:66:87:72:30:4D:30:A5:FB:87:3B:0F:A7:7B:B7:0D:54
  SHA256:
 23:99:56:11:27:A5:71:25:DE:8C:EF:EA:61:0D:DF:2F:A0:78:B5:C8:06:7F:4E:82:82:90:BF:B8:60:E8:4B:3

Alias name: geotrustuniversalca
Certificate fingerprints:
  MD5:  92:65:58:8B:A2:1A:31:72:73:68:5C:B4:A5:7A:07:48
  SHA1: E6:21:F3:35:43:79:05:9A:4B:68:30:9D:8A:2F:74:22:15:87:EC:79
  SHA256:
 A0:45:9B:9F:63:B2:25:59:F5:FA:5D:4C:6D:B3:F9:F7:2F:F1:93:42:03:35:78:F0:73:BF:1D:1B:46:CB:B9:1



Alias name: luxtrustglobalroot2
Certificate fingerprints:
  MD5:  B2:E1:09:00:61:AF:F7:F1:91:6F:C4:AD:8D:5E:3B:7C
  SHA1: 1E:0E:56:19:0A:D1:8B:25:98:B2:04:44:FF:66:8A:04:17:99:5F:3F
  SHA256:
 54:45:5F:71:29:C2:0B:14:47:C4:18:F9:97:16:8F:24:C5:8F:C5:02:3B:F5:DA:5B:E2:EB:6E:1D:D8:90:2E:D



Alias name: twcaglobalrootca
Certificate fingerprints:
  MD5:  F9:03:7E:CF:E6:9E:3C:73:7A:2A:90:07:69:FF:2B:96
  SHA1: 9C:BB:48:53:F6:A4:F6:D3:52:A4:E8:32:52:55:60:13:F5:AD:AF:65
  SHA256:
 59:76:90:07:F7:68:5D:0F:CD:50:87:2F:9F:95:D5:75:5A:5B:2B:45:7D:81:F3:69:2B:61:0A:98:67:2F:0E:1

Alias name: tubitakkamusmsslkoksertifikasisurum1
```

```
Certificate fingerprints:
  MD5:  DC:00:81:DC:69:2F:3E:2F:B0:3B:F6:3D:5A:91:8E:49
  SHA1: 31:43:64:9B:EC:CE:27:EC:ED:3A:3F:0B:8F:0D:E4:E8:91:DD:EE:CA
  SHA256:
 46:ED:C3:68:90:46:D5:3A:45:3F:B3:10:4A:B8:0D:CA:EC:65:8B:26:60:EA:16:29:DD:7E:86:79:90:64:87:1

Alias name: affirmtrustnetworkingca
Certificate fingerprints:
  MD5:  42:65:CA:BE:01:9A:9A:4C:A9:8C:41:49:CD:C0:D5:7F
  SHA1: 29:36:21:02:8B:20:ED:02:F5:66:C5:32:D1:D6:ED:90:9F:45:00:2F
  SHA256:
 0A:81:EC:5A:92:97:77:F1:45:90:4A:F3:8D:5D:50:9F:66:B5:E2:C5:8F:CD:B5:31:05:8B:0E:17:F3:F0:B4:1

Alias name: affirmtrustcommercialca
Certificate fingerprints:
  MD5:  82:92:BA:5B:EF:CD:8A:6F:A6:3D:55:F9:84:F6:D6:B7
  SHA1: F9:B5:B6:32:45:5F:9C:BE:EC:57:5F:80:DC:E9:6E:2C:C7:B2:78:B7
  SHA256:
 03:76:AB:1D:54:C5:F9:80:3C:E4:B2:E2:01:A0:EE:7E:EF:7B:57:B6:36:E8:A9:3C:9B:8D:48:60:C9:6F:5F:A

Alias name: godaddyrootcertificateauthorityg2
Certificate fingerprints:
  MD5:  80:3A:BC:22:C1:E6:FB:8D:9B:3B:27:4A:32:1B:9A:01
  SHA1: 47:BE:AB:C9:22:EA:E8:0E:78:78:34:62:A7:9F:45:C2:54:FD:E6:8B
  SHA256:
 45:14:0B:32:47:EB:9C:C8:C5:B4:F0:D7:B5:30:91:F7:32:92:08:9E:6E:5A:63:E2:74:9D:D3:AC:A9:19:8E:D

Alias name: starfieldrootg2ca
Certificate fingerprints:
  MD5:  D6:39:81:C6:52:7E:96:69:FC:FC:CA:66:ED:05:F2:96
  SHA1: B5:1C:06:7C:EE:2B:0C:3D:F8:55:AB:2D:92:F4:FE:39:D4:E7:0F:0E
  SHA256:
 2C:E1:CB:0B:F9:D2:F9:E1:02:99:3F:BE:21:51:52:C3:B2:DD:0C:AB:DE:1C:68:E5:31:9B:83:91:54:DB:B7:F

Alias name: dtrustrootclass3ca2ev2009
Certificate fingerprints:
  MD5:  AA:C6:43:2C:5E:2D:CD:C4:34:C0:50:4F:11:02:4F:B6
  SHA1: 96:C9:1B:0B:95:B4:10:98:42:FA:D0:D8:22:79:FE:60:FA:B9:16:83
  SHA256:
 EE:C5:49:6B:98:8C:E9:86:25:B9:34:09:2E:EC:29:08:BE:D0:B0:F3:16:C2:D4:73:0C:84:EA:F1:F3:D3:48:8

Alias name: buypassclass3ca
Certificate fingerprints:
  MD5:  3D:3B:18:9E:2C:64:5A:E8:D5:88:CE:0E:F9:37:C2:EC
```

```
  SHA1: DA:FA:F7:FA:66:84:EC:06:8F:14:50:BD:C7:C2:81:A5:BC:A9:64:57
  SHA256:
 ED:F7:EB:BC:A2:7A:2A:38:4D:38:7B:7D:40:10:C6:66:E2:ED:B4:84:3E:4C:29:B4:AE:1D:5B:93:32:E6:B2:4

Alias name: verisignclass2g3ca
Certificate fingerprints:
  MD5:  F8:BE:C4:63:22:C9:A8:46:74:8B:B8:1D:1E:4A:2B:F6
  SHA1: 61:EF:43:D7:7F:CA:D4:61:51:BC:98:E0:C3:59:12:AF:9F:EB:63:11
  SHA256:
 92:A9:D9:83:3F:E1:94:4D:B3:66:E8:BF:AE:7A:95:B6:48:0C:2D:6C:6C:2A:1B:E6:5D:42:36:B6:08:FC:A1:B

Alias name: digicerttrustedrootg4
Certificate fingerprints:
  MD5:  78:F2:FC:AA:60:1F:2F:B4:EB:C9:37:BA:53:2E:75:49
  SHA1: DD:FB:16:CD:49:31:C9:73:A2:03:7D:3F:C8:3A:4D:7D:77:5D:05:E4
  SHA256:
 55:2F:7B:DC:F1:A7:AF:9E:6C:E6:72:01:7F:4F:12:AB:F7:72:40:C7:8E:76:1A:C2:03:D1:D9:D2:0A:C8:99:8

Alias name: quovadisrootca2g3
Certificate fingerprints:
  MD5:  AF:0C:86:6E:BF:40:2D:7F:0B:3E:12:50:BA:12:3D:06
  SHA1: 09:3C:61:F3:8B:8B:DC:7D:55:DF:75:38:02:05:00:E1:25:F5:C8:36
  SHA256:
 8F:E4:FB:0A:F9:3A:4D:0D:67:DB:0B:EB:B2:3E:37:C7:1B:F3:25:DC:BC:DD:24:0E:A0:4D:AF:58:B4:7E:18:4

Alias name: geotrustprimarycertificationauthorityg3
Certificate fingerprints:
  MD5:  B5:E8:34:36:C9:10:44:58:48:70:6D:2E:83:D4:B8:05
  SHA1: 03:9E:ED:B8:0B:E7:A0:3C:69:53:89:3B:20:D2:D9:32:3A:4C:2A:FD
  SHA256:
 B4:78:B8:12:25:0D:F8:78:63:5C:2A:A7:EC:7D:15:5E:AA:62:5E:E8:29:16:E2:CD:29:43:61:88:6C:D1:FB:D

Alias name: geotrustprimarycertificationauthorityg2
Certificate fingerprints:
  MD5:  01:5E:D8:6B:BD:6F:3D:8E:A1:31:F8:12:E0:98:73:6A
  SHA1: 8D:17:84:D5:37:F3:03:7D:EC:70:FE:57:8B:51:9A:99:E6:10:D7:B0
  SHA256:
 5E:DB:7A:C4:3B:82:A0:6A:87:61:E8:D7:BE:49:79:EB:F2:61:1F:7D:D7:9B:F9:1C:1C:6B:56:6A:21:9E:D7:6

Alias name: godaddyclass2ca
Certificate fingerprints:
  MD5:  91:DE:06:25:AB:DA:FD:32:17:0C:BB:25:17:2A:84:67
  SHA1: 27:96:BA:E6:3F:18:01:E2:77:26:1B:A0:D7:77:70:02:8F:20:EE:E4
```

```
  SHA256:
 C3:84:6B:F2:4B:9E:93:CA:64:27:4C:0E:C6:7C:1E:CC:5E:02:4F:FC:AC:D2:D7:40:19:35:0E:81:FE:54:6A:E

Alias name: trustcoreca1
Certificate fingerprints:
  MD5:  27:92:23:1D:0A:F5:40:7C:E9:E6:6B:9D:D8:F5:E7:6C
  SHA1: 58:D1:DF:95:95:67:6B:63:C0:F0:5B:1C:17:4D:8B:84:0B:C8:78:BD
  SHA256:
 5A:88:5D:B1:9C:01:D9:12:C5:75:93:88:93:8C:AF:BB:DF:03:1A:B2:D4:8E:91:EE:15:58:9B:42:97:1D:03:9

Alias name: hellenicacademicandresearchinstitutionseccrootca2015
Certificate fingerprints:
  MD5:  81:E5:B4:17:EB:C2:F5:E1:4B:0D:41:7B:49:92:FE:EF
  SHA1: 9F:F1:71:8D:92:D5:9A:F3:7D:74:97:B4:BC:6F:84:68:0B:BA:B6:66
  SHA256:
 44:B5:45:AA:8A:25:E6:5A:73:CA:15:DC:27:FC:36:D2:4C:1C:B9:95:3A:06:65:39:B1:15:82:DC:48:7B:48:3

Alias name: utnuserfirstobjectca
Certificate fingerprints:
  MD5:  A7:F2:E4:16:06:41:11:50:30:6B:9C:E3:B4:9C:B0:C9
  SHA1: E1:2D:FB:4B:41:D7:D9:C3:2B:30:51:4B:AC:1D:81:D8:38:5E:2D:46
  SHA256:
 6F:FF:78:E4:00:A7:0C:11:01:1C:D8:59:77:C4:59:FB:5A:F9:6A:3D:F0:54:08:20:D0:F4:B8:60:78:75:E5:8

Alias name: ttelesecglobalrootclass3
Certificate fingerprints:
  MD5:  CA:FB:40:A8:4E:39:92:8A:1D:FE:8E:2F:C4:27:EA:EF
  SHA1: 55:A6:72:3E:CB:F2:EC:CD:C3:23:74:70:19:9D:2A:BE:11:E3:81:D1
  SHA256:
 FD:73:DA:D3:1C:64:4F:F1:B4:3B:EF:0C:CD:DA:96:71:0B:9C:D9:87:5E:CA:7E:31:70:7A:F3:E9:6D:52:2B:B

Alias name: ttelesecglobalrootclass2
Certificate fingerprints:
  MD5:  2B:9B:9E:E4:7B:6C:1F:00:72:1A:CC:C1:77:79:DF:6A
  SHA1: 59:0D:2D:7D:88:4F:40:2E:61:7E:A5:62:32:17:65:CF:17:D8:94:E9
  SHA256:
 91:E2:F5:78:8D:58:10:EB:A7:BA:58:73:7D:E1:54:8A:8E:CA:CD:01:45:98:BC:0B:14:3E:04:1B:17:05:25:5

Alias name: addtrustclass1ca
Certificate fingerprints:
  MD5:  1E:42:95:02:33:92:6B:B9:5F:C0:7F:DA:D6:B2:4B:FC
  SHA1: CC:AB:0E:A0:4C:23:01:D6:69:7B:DD:37:9F:CD:12:EB:24:E3:94:9D
  SHA256:
 8C:72:09:27:9A:C0:4E:27:5E:16:D0:7F:D3:B7:75:E8:01:54:B5:96:80:46:E3:1F:52:DD:25:76:63:24:E9:A
```

```
Alias name: amzninternalrootca
Certificate fingerprints:
  MD5:  08:09:73:AC:E0:78:41:7C:0A:26:33:51:E8:CF:E6:60
  SHA1: A7:B7:F6:15:8A:FF:1E:C8:85:13:38:BC:93:EB:A2:AB:A4:09:EF:06
  SHA256:
 0E:DE:63:C1:DC:7A:8E:11:F1:AB:BC:05:4F:59:EE:49:9D:62:9A:2F:DE:9C:A7:16:32:A2:64:29:3E:8B:66:A

Alias name: starfieldrootcertificateauthorityg2
Certificate fingerprints:
  MD5:  D6:39:81:C6:52:7E:96:69:FC:FC:CA:66:ED:05:F2:96
  SHA1: B5:1C:06:7C:EE:2B:0C:3D:F8:55:AB:2D:92:F4:FE:39:D4:E7:0F:0E
  SHA256:
 2C:E1:CB:0B:F9:D2:F9:E1:02:99:3F:BE:21:51:52:C3:B2:DD:0C:AB:DE:1C:68:E5:31:9B:83:91:54:DB:B7:F

Alias name: camerfirmachambersignca
Certificate fingerprints:
  MD5:  9E:80:FF:78:01:0C:2E:C1:36:BD:FE:96:90:6E:08:F3
  SHA1: 4A:BD:EE:EC:95:0D:35:9C:89:AE:C7:52:A1:2C:5B:29:F6:D6:AA:0C
  SHA256:
 13:63:35:43:93:34:A7:69:80:16:A0:D3:24:DE:72:28:4E:07:9D:7B:52:20:BB:8F:BD:74:78:16:EE:BE:BA:C

Alias name: secomscrootca2
Certificate fingerprints:
  MD5:  6C:39:7D:A4:0E:55:59:B2:3F:D6:41:B1:12:50:DE:43
  SHA1: 5F:3B:8C:F2:F8:10:B3:7D:78:B4:CE:EC:19:19:C3:73:34:B9:C7:74
  SHA256:
 51:3B:2C:EC:B8:10:D4:CD:E5:DD:85:39:1A:DF:C6:C2:DD:60:D8:7B:B7:36:D2:B5:21:48:4A:A4:7A:0E:BE:F

Alias name: entrustevca
Certificate fingerprints:
  MD5:  D6:A5:C3:ED:5D:DD:3E:00:C1:3D:87:92:1F:1D:3F:E4
  SHA1: B3:1E:B1:B7:40:E3:6C:84:02:DA:DC:37:D4:4D:F5:D4:67:49:52:F9
  SHA256:
 73:C1:76:43:4F:1B:C6:D5:AD:F4:5B:0E:76:E7:27:28:7C:8D:E5:76:16:C1:E6:E6:14:1A:2B:2C:BC:7D:8E:4

Alias name: secomscrootca1
Certificate fingerprints:
  MD5:  F1:BC:63:6A:54:E0:B5:27:F5:CD:E7:1A:E3:4D:6E:4A
  SHA1: 36:B1:2B:49:F9:81:9E:D7:4C:9E:BC:38:0F:C6:56:8F:5D:AC:B2:F7
  SHA256:
 E7:5E:72:ED:9F:56:0E:EC:6E:B4:80:00:73:A4:3F:C3:AD:19:19:5A:39:22:82:01:78:95:97:4A:99:02:6B:6

Alias name: affirmtrustcommercial
```

```
Certificate fingerprints:
  MD5:  82:92:BA:5B:EF:CD:8A:6F:A6:3D:55:F9:84:F6:D6:B7
  SHA1: F9:B5:B6:32:45:5F:9C:BE:EC:57:5F:80:DC:E9:6E:2C:C7:B2:78:B7
  SHA256:
 03:76:AB:1D:54:C5:F9:80:3C:E4:B2:E2:01:A0:EE:7E:EF:7B:57:B6:36:E8:A9:3C:9B:8D:48:60:C9:6F:5F:A

Alias name: digicertassuredidrootg3
Certificate fingerprints:
  MD5:  7C:7F:65:31:0C:81:DF:8D:BA:3E:99:E2:5C:AD:6E:FB
  SHA1: F5:17:A2:4F:9A:48:C6:C9:F8:A2:00:26:9F:DC:0F:48:2C:AB:30:89
  SHA256:
 7E:37:CB:8B:4C:47:09:0C:AB:36:55:1B:A6:F4:5D:B8:40:68:0F:BA:16:6A:95:2D:B1:00:71:7F:43:05:3F:C

Alias name: affirmtrustnetworking
Certificate fingerprints:
  MD5:  42:65:CA:BE:01:9A:9A:4C:A9:8C:41:49:CD:C0:D5:7F
  SHA1: 29:36:21:02:8B:20:ED:02:F5:66:C5:32:D1:D6:ED:90:9F:45:00:2F
  SHA256:
 0A:81:EC:5A:92:97:77:F1:45:90:4A:F3:8D:5D:50:9F:66:B5:E2:C5:8F:CD:B5:31:05:8B:0E:17:F3:F0:B4:1

Alias name: izenpecom
Certificate fingerprints:
  MD5:  A6:B0:CD:85:80:DA:5C:50:34:A3:39:90:2F:55:67:73
  SHA1: 2F:78:3D:25:52:18:A7:4A:65:39:71:B5:2C:A2:9C:45:15:6F:E9:19
  SHA256:
 25:30:CC:8E:98:32:15:02:BA:D9:6F:9B:1F:BA:1B:09:9E:2D:29:9E:0F:45:48:BB:91:4F:36:3B:C0:D4:53:1

Alias name: amazon-ca-g4-legacy
Certificate fingerprints:
  MD5:  6C:E5:BD:67:A4:4F:E3:FD:C2:4C:46:E6:06:5B:6D:55
  SHA1: EA:E7:DE:F9:0A:BE:9F:0B:68:CE:B7:24:0D:80:74:03:BF:6E:B1:6E
  SHA256:
 CD:72:C4:7F:B4:AD:28:A4:67:2B:E1:86:47:D4:40:E9:3B:16:2D:95:DB:3C:2F:94:BB:81:D9:09:F7:91:24:5

Alias name: digicertassuredidrootg2
Certificate fingerprints:
  MD5:  92:38:B9:F8:63:24:82:65:2C:57:33:E6:FE:81:8F:9D
  SHA1: A1:4B:48:D9:43:EE:0A:0E:40:90:4F:3C:E0:A4:C0:91:93:51:5D:3F
  SHA256:
 7D:05:EB:B6:82:33:9F:8C:94:51:EE:09:4E:EB:FE:FA:79:53:A1:14:ED:B2:F4:49:49:45:2F:AB:7D:2F:C1:8

Alias name: comodoaaaservicesroot
Certificate fingerprints:
  MD5:  49:79:04:B0:EB:87:19:AC:47:B0:BC:11:51:9B:74:D0
```

```
  SHA1: D1:EB:23:A4:6D:17:D6:8F:D9:25:64:C2:F1:F1:60:17:64:D8:E3:49
  SHA256:
 D7:A7:A0:FB:5D:7E:27:31:D7:71:E9:48:4E:BC:DE:F7:1D:5F:0C:3E:0A:29:48:78:2B:C8:3E:E0:EA:69:9E:F

Alias name: entrustnetpremium2048secureserverca
Certificate fingerprints:
  MD5:  EE:29:31:BC:32:7E:9A:E6:E8:B5:F7:51:B4:34:71:90
  SHA1: 50:30:06:09:1D:97:D4:F5:AE:39:F7:CB:E7:92:7D:7D:65:2D:34:31
  SHA256:
 6D:C4:71:72:E0:1C:BC:B0:BF:62:58:0D:89:5F:E2:B8:AC:9A:D4:F8:73:80:1E:0C:10:B9:C8:37:D2:1E:B1:7

Alias name: trustcorrootcertca2
Certificate fingerprints:
  MD5:  A2:E1:F8:18:0B:BA:45:D5:C7:41:2A:BB:37:52:45:64
  SHA1: B8:BE:6D:CB:56:F1:55:B9:63:D4:12:CA:4E:06:34:C7:94:B2:1C:C0
  SHA256:
 07:53:E9:40:37:8C:1B:D5:E3:83:6E:39:5D:AE:A5:CB:83:9E:50:46:F1:BD:0E:AE:19:51:CF:10:FE:C7:C9:6

Alias name: entrust2048ca
Certificate fingerprints:
  MD5:  EE:29:31:BC:32:7E:9A:E6:E8:B5:F7:51:B4:34:71:90
  SHA1: 50:30:06:09:1D:97:D4:F5:AE:39:F7:CB:E7:92:7D:7D:65:2D:34:31
  SHA256:
 6D:C4:71:72:E0:1C:BC:B0:BF:62:58:0D:89:5F:E2:B8:AC:9A:D4:F8:73:80:1E:0C:10:B9:C8:37:D2:1E:B1:7

Alias name: trustcorrootcertca1
Certificate fingerprints:
  MD5:  6E:85:F1:DC:1A:00:D3:22:D5:B2:B2:AC:6B:37:05:45
  SHA1: FF:BD:CD:E7:82:C8:43:5E:3C:6F:26:86:5C:CA:A8:3A:45:5B:C3:0A
  SHA256:
 D4:0E:9C:86:CD:8F:E4:68:C1:77:69:59:F4:9E:A7:74:FA:54:86:84:B6:C4:06:F3:90:92:61:F4:DC:E2:57:5

Alias name: baltimorecybertrustroot
Certificate fingerprints:
  MD5:  AC:B6:94:A5:9C:17:E0:D7:91:52:9B:B1:97:06:A6:E4
  SHA1: D4:DE:20:D0:5E:66:FC:53:FE:1A:50:88:2C:78:DB:28:52:CA:E4:74
  SHA256:
 16:AF:57:A9:F6:76:B0:AB:12:60:95:AA:5E:BA:DE:F2:2A:B3:11:19:D6:44:AC:95:CD:4B:93:DB:F3:F2:6A:B

Alias name: eecertificationcentrerootca
Certificate fingerprints:
  MD5:  43:5E:88:D4:7D:1A:4A:7E:FD:84:2E:52:EB:01:D4:6F
  SHA1: C9:A8:B9:E7:55:80:5E:58:E3:53:77:A7:25:EB:AF:C3:7B:27:CC:D7
```

```
  SHA256:
 3E:84:BA:43:42:90:85:16:E7:75:73:C0:99:2F:09:79:CA:08:4E:46:85:68:1F:F1:95:CC:BA:8A:22:9B:8A:7

Alias name: dstacescax6
Certificate fingerprints:
  MD5:  21:D8:4C:82:2B:99:09:33:A2:EB:14:24:8D:8E:5F:E8
  SHA1: 40:54:DA:6F:1C:3F:40:74:AC:ED:0F:EC:CD:DB:79:D1:53:FB:90:1D
  SHA256:
 76:7C:95:5A:76:41:2C:89:AF:68:8E:90:A1:C7:0F:55:6C:FD:6B:60:25:DB:EA:10:41:6D:7E:B6:83:1F:8C:4

Alias name: comodocertificationauthority
Certificate fingerprints:
  MD5:  5C:48:DC:F7:42:72:EC:56:94:6D:1C:CC:71:35:80:75
  SHA1: 66:31:BF:9E:F7:4F:9E:B6:C9:D5:A6:0C:BA:6A:BE:D1:F7:BD:EF:7B
  SHA256:
 0C:2C:D6:3D:F7:80:6F:A3:99:ED:E8:09:11:6B:57:5B:F8:79:89:F0:65:18:F9:80:8C:86:05:03:17:8B:AF:6

Alias name: thawteserverca
Certificate fingerprints:
  MD5:  EE:FE:61:69:65:6E:F8:9C:C6:2A:F4:D7:2B:63:EF:A2
  SHA1: 9F:AD:91:A6:CE:6A:C6:C5:00:47:C4:4E:C9:D4:A5:0D:92:D8:49:79
  SHA256:
 87:C6:78:BF:B8:B2:5F:38:F7:E9:7B:33:69:56:BB:CF:14:4B:BA:CA:A5:36:47:E6:1A:23:25:BC:10:55:31:6

Alias name: secomvalicertclass1ca
Certificate fingerprints:
  MD5:  65:58:AB:15:AD:57:6C:1E:A8:A7:B5:69:AC:BF:FF:EB
  SHA1: E5:DF:74:3C:B6:01:C4:9B:98:43:DC:AB:8C:E8:6A:81:10:9F:E4:8E
  SHA256:
 F4:C1:49:55:1A:30:13:A3:5B:C7:BF:FE:17:A7:F3:44:9B:C1:AB:5B:5A:0A:E7:4B:06:C2:3B:90:00:4C:01:0

Alias name: godaddyrootg2ca
Certificate fingerprints:
  MD5:  80:3A:BC:22:C1:E6:FB:8D:9B:3B:27:4A:32:1B:9A:01
  SHA1: 47:BE:AB:C9:22:EA:E8:0E:78:78:34:62:A7:9F:45:C2:54:FD:E6:8B
  SHA256:
 45:14:0B:32:47:EB:9C:C8:C5:B4:F0:D7:B5:30:91:F7:32:92:08:9E:6E:5A:63:E2:74:9D:D3:AC:A9:19:8E:D

Alias name: globalchambersignroot2008
Certificate fingerprints:
  MD5:  9E:80:FF:78:01:0C:2E:C1:36:BD:FE:96:90:6E:08:F3
  SHA1: 4A:BD:EE:EC:95:0D:35:9C:89:AE:C7:52:A1:2C:5B:29:F6:D6:AA:0C
  SHA256:
 13:63:35:43:93:34:A7:69:80:16:A0:D3:24:DE:72:28:4E:07:9D:7B:52:20:BB:8F:BD:74:78:16:EE:BE:BA:C
```

```
Alias name: equifaxsecureebusinessca1
Certificate fingerprints:
  MD5:  14:C0:08:E5:A3:85:03:A3:BE:78:E9:67:4F:27:CA:EE
  SHA1: AE:E6:3D:70:E3:76:FB:C7:3A:EB:B0:A1:C1:D4:C4:7A:A7:40:B3:F4
  SHA256:
 2E:3A:2B:B5:11:25:05:83:6C:A8:96:8B:E2:CB:37:27:CE:9B:56:84:5C:6E:E9:8E:91:85:10:4A:FB:9A:F5:9

Alias name: quovadisrootca3
Certificate fingerprints:
  MD5:  31:85:3C:62:94:97:63:B9:AA:FD:89:4E:AF:6F:E0:CF
  SHA1: 1F:49:14:F7:D8:74:95:1D:DD:AE:02:C0:BE:FD:3A:2D:82:75:51:85
  SHA256:
 18:F1:FC:7F:20:5D:F8:AD:DD:EB:7F:E0:07:DD:57:E3:AF:37:5A:9C:4D:8D:73:54:6B:F4:F1:FE:D1:E1:8D:3

Alias name: usertrustecccertificationauthority
Certificate fingerprints:
  MD5:  FA:68:BC:D9:B5:7F:AD:FD:C9:1D:06:83:28:CC:24:C1
  SHA1: D1:CB:CA:5D:B2:D5:2A:7F:69:3B:67:4D:E5:F0:5A:1D:0C:95:7D:F0
  SHA256:
 4F:F4:60:D5:4B:9C:86:DA:BF:BC:FC:57:12:E0:40:0D:2B:ED:3F:BC:4D:4F:BD:AA:86:E0:6A:DC:D2:A9:AD:7

Alias name: quovadisrootca2
Certificate fingerprints:
  MD5:  5E:39:7B:DD:F8:BA:EC:82:E9:AC:62:BA:0C:54:00:2B
  SHA1: CA:3A:FB:CF:12:40:36:4B:44:B2:16:20:88:80:48:39:19:93:7C:F7
  SHA256:
 85:A0:DD:7D:D7:20:AD:B7:FF:05:F8:3D:54:2B:20:9D:C7:FF:45:28:F7:D6:77:B1:83:89:FE:A5:E5:C4:9E:8

Alias name: soneraclass2ca
Certificate fingerprints:
  MD5:  A3:EC:75:0F:2E:88:DF:FA:48:01:4E:0B:5C:48:6F:FB
  SHA1: 37:F7:6D:E6:07:7C:90:C5:B1:3E:93:1A:B7:41:10:B4:F2:E4:9A:27
  SHA256:
 79:08:B4:03:14:C1:38:10:0B:51:8D:07:35:80:7F:FB:FC:F8:51:8A:00:95:33:71:05:BA:38:6B:15:3D:D9:2

Alias name: twcarootcertificationauthority
Certificate fingerprints:
  MD5:  AA:08:8F:F6:F9:7B:B7:F2:B1:A7:1E:9B:EA:EA:BD:79
  SHA1: CF:9E:87:6D:D3:EB:FC:42:26:97:A3:B5:A3:7A:A0:76:A9:06:23:48
  SHA256:
 BF:D8:8F:E1:10:1C:41:AE:3E:80:1B:F8:BE:56:35:0E:E9:BA:D1:A6:B9:BD:51:5E:DC:5C:6D:5B:87:11:AC:4

Alias name: baltimorecybertrustca
```

```
Certificate fingerprints:
  MD5:  AC:B6:94:A5:9C:17:E0:D7:91:52:9B:B1:97:06:A6:E4
  SHA1: D4:DE:20:D0:5E:66:FC:53:FE:1A:50:88:2C:78:DB:28:52:CA:E4:74
  SHA256:
 16:AF:57:A9:F6:76:B0:AB:12:60:95:AA:5E:BA:DE:F2:2A:B3:11:19:D6:44:AC:95:CD:4B:93:DB:F3:F2:6A:E

Alias name: cia-crt-g3-01-ca
Certificate fingerprints:
  MD5:  E3:66:DD:D6:A0:D5:40:8F:FF:29:E2:C0:CB:6E:62:1A
  SHA1: 2B:EE:2C:BA:A3:1D:B5:FE:60:40:41:95:08:ED:46:82:39:4D:ED:E2
  SHA256:
 20:48:AD:4C:EC:90:7F:FA:4A:15:D4:CE:45:E3:C8:E4:2C:EA:78:33:DC:C7:D3:40:48:FC:60:47:27:42:99:E

Alias name: entrustrootcertificationauthorityg2
Certificate fingerprints:
  MD5:  4B:E2:C9:91:96:65:0C:F4:0E:5A:93:92:A0:0A:FE:B2
  SHA1: 8C:F4:27:FD:79:0C:3A:D1:66:06:8D:E8:1E:57:EF:BB:93:22:72:D4
  SHA256:
 43:DF:57:74:B0:3E:7F:EF:5F:E4:0D:93:1A:7B:ED:F1:BB:2E:6B:42:73:8C:4E:6D:38:41:10:3D:3A:A7:F3:3

Alias name: verisignclass3g4ca
Certificate fingerprints:
  MD5:  3A:52:E1:E7:FD:6F:3A:E3:6F:F3:6F:99:1B:F9:22:41
  SHA1: 22:D5:D8:DF:8F:02:31:D1:8D:F7:9D:B7:CF:8A:2D:64:C9:3F:6C:3A
  SHA256:
 69:DD:D7:EA:90:BB:57:C9:3E:13:5D:C8:5E:A6:FC:D5:48:0B:60:32:39:BD:C4:54:FC:75:8B:2A:26:CF:7F:7

Alias name: xrampglobalcaroot
Certificate fingerprints:
  MD5:  A1:0B:44:B3:CA:10:D8:00:6E:9D:0F:D8:0F:92:0A:D1
  SHA1: B8:01:86:D1:EB:9C:86:A5:41:04:CF:30:54:F3:4C:52:B7:E5:58:C6
  SHA256:
 CE:CD:DC:90:50:99:D8:DA:DF:C5:B1:D2:09:B7:37:CB:E2:C1:8C:FB:2C:10:C0:FF:0B:CF:0D:32:86:FC:1A:A

Alias name: identrustcommercialrootca1
Certificate fingerprints:
  MD5:  B3:3E:77:73:75:EE:A0:D3:E3:7E:49:63:49:59:BB:C7
  SHA1: DF:71:7E:AA:4A:D9:4E:C9:55:84:99:60:2D:48:DE:5F:BC:F0:3A:25
  SHA256:
 5D:56:49:9B:E4:D2:E0:8B:CF:CA:D0:8A:3E:38:72:3D:50:50:3B:DE:70:69:48:E4:2F:55:60:30:19:E5:28:A

Alias name: camerfirmachamberscommerceca
Certificate fingerprints:
  MD5:  B0:01:EE:14:D9:AF:29:18:94:76:8E:F1:69:33:2A:84
```

```
  SHA1: 6E:3A:55:A4:19:0C:19:5C:93:84:3C:C0:DB:72:2E:31:30:61:F0:B1
  SHA256:
 0C:25:8A:12:A5:67:4A:EF:25:F2:8B:A7:DC:FA:EC:EE:A3:48:E5:41:E6:F5:CC:4E:E6:3B:71:B3:61:60:6A:C

Alias name: verisignclass3g2ca
Certificate fingerprints:
  MD5:  A2:33:9B:4C:74:78:73:D4:6C:E7:C1:F3:8D:CB:5C:E9
  SHA1: 85:37:1C:A6:E5:50:14:3D:CE:28:03:47:1B:DE:3A:09:E8:F8:77:0F
  SHA256:
 83:CE:3C:12:29:68:8A:59:3D:48:5F:81:97:3C:0F:91:95:43:1E:DA:37:CC:5E:36:43:0E:79:C7:A8:88:63:8

Alias name: deutschetelekomrootca2
Certificate fingerprints:
  MD5:  74:01:4A:91:B1:08:C4:58:CE:47:CD:F0:DD:11:53:08
  SHA1: 85:A4:08:C0:9C:19:3E:5D:51:58:7D:CD:D6:13:30:FD:8C:DE:37:BF
  SHA256:
 B6:19:1A:50:D0:C3:97:7F:7D:A9:9B:CD:AA:C8:6A:22:7D:AE:B9:67:9E:C7:0B:A3:B0:C9:D9:22:71:C1:70:D

Alias name: certumca
Certificate fingerprints:
  MD5:  2C:8F:9F:66:1D:18:90:B1:47:26:9D:8E:86:82:8C:A9
  SHA1: 62:52:DC:40:F7:11:43:A2:2F:DE:9E:F7:34:8E:06:42:51:B1:81:18
  SHA256:
 D8:E0:FE:BC:1D:B2:E3:8D:00:94:0F:37:D2:7D:41:34:4D:99:3E:73:4B:99:D5:65:6D:97:78:D4:D8:14:36:2

Alias name: cybertrustglobalroot
Certificate fingerprints:
  MD5:  72:E4:4A:87:E3:69:40:80:77:EA:BC:E3:F4:FF:F0:E1
  SHA1: 5F:43:E5:B1:BF:F8:78:8C:AC:1C:C7:CA:4A:9A:C6:22:2B:CC:34:C6
  SHA256:
 96:0A:DF:00:63:E9:63:56:75:0C:29:65:DD:0A:08:67:DA:0B:9C:BD:6E:77:71:4A:EA:FB:23:49:AB:39:3D:A

Alias name: globalsignrootca
Certificate fingerprints:
  MD5:  3E:45:52:15:09:51:92:E1:B7:5D:37:9F:B1:87:29:8A
  SHA1: B1:BC:96:8B:D4:F4:9D:62:2A:A8:9A:81:F2:15:01:52:A4:1D:82:9C
  SHA256:
 EB:D4:10:40:E4:BB:3E:C7:42:C9:E3:81:D3:1E:F2:A4:1A:48:B6:68:5C:96:E7:CE:F3:C1:DF:6C:D4:33:1C:9

Alias name: secomevrootca1
Certificate fingerprints:
  MD5:  22:2D:A6:01:EA:7C:0A:F7:F0:6C:56:43:3F:77:76:D3
  SHA1: FE:B8:C4:32:DC:F9:76:9A:CE:AE:3D:D8:90:8F:FD:28:86:65:64:7D
```

```
  SHA256:
 A2:2D:BA:68:1E:97:37:6E:2D:39:7D:72:8A:AE:3A:9B:62:96:B9:FD:BA:60:BC:2E:11:F6:47:F2:C6:75:FB:3

Alias name: globalsignr3ca
Certificate fingerprints:
  MD5:  C5:DF:B8:49:CA:05:13:55:EE:2D:BA:1A:C3:3E:B0:28
  SHA1: D6:9B:56:11:48:F0:1C:77:C5:45:78:C1:09:26:DF:5B:85:69:76:AD
  SHA256:
 CB:B5:22:D7:B7:F1:27:AD:6A:01:13:86:5B:DF:1C:D4:10:2E:7D:07:59:AF:63:5A:7C:F4:72:0D:C9:63:C5:3

Alias name: staatdernederlandenrootcag3
Certificate fingerprints:
  MD5:  0B:46:67:07:DB:10:2F:19:8C:35:50:60:D1:0B:F4:37
  SHA1: D8:EB:6B:41:51:92:59:E0:F3:E7:85:00:C0:3D:B6:88:97:C9:EE:FC
  SHA256:
 3C:4F:B0:B9:5A:B8:B3:00:32:F4:32:B8:6F:53:5F:E1:72:C1:85:D0:FD:39:86:58:37:CF:36:18:7F:A6:F4:2

Alias name: staatdernederlandenrootcag2
Certificate fingerprints:
  MD5:  7C:A5:0F:F8:5B:9A:7D:6D:30:AE:54:5A:E3:42:A2:8A
  SHA1: 59:AF:82:79:91:86:C7:B4:75:07:CB:CF:03:57:46:EB:04:DD:B7:16
  SHA256:
 66:8C:83:94:7D:A6:3B:72:4B:EC:E1:74:3C:31:A0:E6:AE:D0:DB:8E:C5:B3:1B:E3:77:BB:78:4F:91:B6:71:6

Alias name: aolrootca2
Certificate fingerprints:
  MD5:  D6:ED:3C:CA:E2:66:0F:AF:10:43:0D:77:9B:04:09:BF
  SHA1: 85:B5:FF:67:9B:0C:79:96:1F:C8:6E:44:22:00:46:13:DB:17:92:84
  SHA256:
 7D:3B:46:5A:60:14:E5:26:C0:AF:FC:EE:21:27:D2:31:17:27:AD:81:1C:26:84:2D:00:6A:F3:73:06:CC:80:B

Alias name: dstrootcax3
Certificate fingerprints:
  MD5:  41:03:52:DC:0F:F7:50:1B:16:F0:02:8E:BA:6F:45:C5
  SHA1: DA:C9:02:4F:54:D8:F6:DF:94:93:5F:B1:73:26:38:CA:6A:D7:7C:13
  SHA256:
 06:87:26:03:31:A7:24:03:D9:09:F1:05:E6:9B:CF:0D:32:E1:BD:24:93:FF:C6:D9:20:6D:11:BC:D6:77:07:3

Alias name: trustcenteruniversalcai
Certificate fingerprints:
  MD5:  45:E1:A5:72:C5:A9:36:64:40:9E:F5:E4:58:84:67:8C
  SHA1: 6B:2F:34:AD:89:58:BE:62:FD:B0:6B:5C:CE:BB:9D:D9:4F:4E:39:F3
  SHA256:
 EB:F3:C0:2A:87:89:B1:FB:7D:51:19:95:D6:63:B7:29:06:D9:13:CE:0D:5E:10:56:8A:8A:77:E2:58:61:67:E
```

```
Alias name: aolrootca1
Certificate fingerprints:
  MD5:  14:F1:08:AD:9D:FA:64:E2:89:E7:1C:CF:A8:AD:7D:5E
  SHA1: 39:21:C1:15:C1:5D:0E:CA:5C:CB:5B:C4:F0:7D:21:D8:05:0B:56:6A
  SHA256:
 77:40:73:12:C6:3A:15:3D:5B:C0:0B:4E:51:75:9C:DF:DA:C2:37:DC:2A:33:B6:79:46:E9:8E:9B:FA:68:0A:E

Alias name: affirmtrustpremiumecc
Certificate fingerprints:
  MD5:  64:B0:09:55:CF:B1:D5:99:E2:BE:13:AB:A6:5D:EA:4D
  SHA1: B8:23:6B:00:2F:1D:16:86:53:01:55:6C:11:A4:37:CA:EB:FF:C3:BB
  SHA256:
 BD:71:FD:F6:DA:97:E4:CF:62:D1:64:7A:DD:25:81:B0:7D:79:AD:F8:39:7E:B4:EC:BA:9C:5E:84:88:82:14:2

Alias name: microseceszignorootca2009
Certificate fingerprints:
  MD5:  F8:49:F4:03:BC:44:2D:83:BE:48:69:7D:29:64:FC:B1
  SHA1: 89:DF:74:FE:5C:F4:0F:4A:80:F9:E3:37:7D:54:DA:91:E1:01:31:8E
  SHA256:
 3C:5F:81:FE:A5:FA:B8:2C:64:BF:A2:EA:EC:AF:CD:E8:E0:77:FC:86:20:A7:CA:E5:37:16:3D:F3:6E:DB:F3:7

Alias name: verisignclass1g3ca
Certificate fingerprints:
  MD5:  B1:47:BC:18:57:D1:18:A0:78:2D:EC:71:E8:2A:95:73
  SHA1: 20:42:85:DC:F7:EB:76:41:95:57:8E:13:6B:D4:B7:D1:E9:8E:46:A5
  SHA256:
 CB:B5:AF:18:5E:94:2A:24:02:F9:EA:CB:C0:ED:5B:B8:76:EE:A3:C1:22:36:23:D0:04:47:E4:F3:BA:55:4B:6

Alias name: certplusrootcag2
Certificate fingerprints:
  MD5:  A7:EE:C4:78:2D:1B:EE:2D:B9:29:CE:D6:A7:96:32:31
  SHA1: 4F:65:8E:1F:E9:06:D8:28:02:E9:54:47:41:C9:54:25:5D:69:CC:1A
  SHA256:
 6C:C0:50:41:E6:44:5E:74:69:6C:4C:FB:C9:F8:0F:54:3B:7E:AB:BB:44:B4:CE:6F:78:7C:6A:99:71:C4:2F:1

Alias name: certplusrootcag1
Certificate fingerprints:
  MD5:  7F:09:9C:F7:D9:B9:5C:69:69:56:D5:37:3E:14:0D:42
  SHA1: 22:FD:D0:B7:FD:A2:4E:0D:AC:49:2C:A0:AC:A6:7B:6A:1F:E3:F7:66
  SHA256:
 15:2A:40:2B:FC:DF:2C:D5:48:05:4D:22:75:B3:9C:7F:CA:3E:C0:97:80:78:B0:F0:EA:76:E5:61:A6:C7:43:3

Alias name: addtrustexternalca
```

```
Certificate fingerprints:
  MD5:  1D:35:54:04:85:78:B0:3F:42:42:4D:BF:20:73:0A:3F
  SHA1: 02:FA:F3:E2:91:43:54:68:60:78:57:69:4D:F5:E4:5B:68:85:18:68
  SHA256:
 68:7F:A4:51:38:22:78:FF:F0:C8:B1:1F:8D:43:D5:76:67:1C:6E:B2:BC:EA:B4:13:FB:83:D9:65:D0:6D:2F:F

Alias name: entrustrootcertificationauthority
Certificate fingerprints:
  MD5:  D6:A5:C3:ED:5D:DD:3E:00:C1:3D:87:92:1F:1D:3F:E4
  SHA1: B3:1E:B1:B7:40:E3:6C:84:02:DA:DC:37:D4:4D:F5:D4:67:49:52:F9
  SHA256:
 73:C1:76:43:4F:1B:C6:D5:AD:F4:5B:0E:76:E7:27:28:7C:8D:E5:76:16:C1:E6:E6:14:1A:2B:2C:BC:7D:8E:4

Alias name: verisignclass3ca
Certificate fingerprints:
  MD5:  EF:5A:F1:33:EF:F1:CD:BB:51:02:EE:12:14:4B:96:C4
  SHA1: A1:DB:63:93:91:6F:17:E4:18:55:09:40:04:15:C7:02:40:B0:AE:6B
  SHA256:
 A4:B6:B3:99:6F:C2:F3:06:B3:FD:86:81:BD:63:41:3D:8C:50:09:CC:4F:A3:29:C2:CC:F0:E2:FA:1B:14:03:0

Alias name: digicertassuredidrootca
Certificate fingerprints:
  MD5:  87:CE:0B:7B:2A:0E:49:00:E1:58:71:9B:37:A8:93:72
  SHA1: 05:63:B8:63:0D:62:D7:5A:BB:C8:AB:1E:4B:DF:B5:A8:99:B2:4D:43
  SHA256:
 3E:90:99:B5:01:5E:8F:48:6C:00:BC:EA:9D:11:1E:E7:21:FA:BA:35:5A:89:BC:F1:DF:69:56:1E:3D:C6:32:5

Alias name: globalsignrootcar3
Certificate fingerprints:
  MD5:  C5:DF:B8:49:CA:05:13:55:EE:2D:BA:1A:C3:3E:B0:28
  SHA1: D6:9B:56:11:48:F0:1C:77:C5:45:78:C1:09:26:DF:5B:85:69:76:AD
  SHA256:
 CB:B5:22:D7:B7:F1:27:AD:6A:01:13:86:5B:DF:1C:D4:10:2E:7D:07:59:AF:63:5A:7C:F4:72:0D:C9:63:C5:3

Alias name: globalsignrootcar2
Certificate fingerprints:
  MD5:  94:14:77:7E:3E:5E:FD:8F:30:BD:41:B0:CF:E7:D0:30
  SHA1: 75:E0:AB:B6:13:85:12:27:1C:04:F8:5F:DD:DE:38:E4:B7:24:2E:FE
  SHA256:
 CA:42:DD:41:74:5F:D0:B8:1E:B9:02:36:2C:F9:D8:BF:71:9D:A1:BD:1B:1E:FC:94:6F:5B:4C:99:F4:2C:1B:9

Alias name: verisignclass1ca
Certificate fingerprints:
  MD5:  86:AC:DE:2B:C5:6D:C3:D9:8C:28:88:D3:8D:16:13:1E
```

```
  SHA1: CE:6A:64:A3:09:E4:2F:BB:D9:85:1C:45:3E:64:09:EA:E8:7D:60:F1
  SHA256:
 51:84:7C:8C:BD:2E:9A:72:C9:1E:29:2D:2A:E2:47:D7:DE:1E:3F:D2:70:54:7A:20:EF:7D:61:0F:38:B8:84:2

Alias name: thawtepremiumserverca
Certificate fingerprints:
  MD5:  A6:6B:60:90:23:9B:3F:2D:BB:98:6F:D6:A7:19:0D:46
  SHA1: E0:AB:05:94:20:72:54:93:05:60:62:02:36:70:F7:CD:2E:FC:66:66
  SHA256:
 3F:9F:27:D5:83:20:4B:9E:09:C8:A3:D2:06:6C:4B:57:D3:A2:47:9C:36:93:65:08:80:50:56:98:10:5D:BC:E

Alias name: verisigntsaca
Certificate fingerprints:
  MD5:  F2:89:95:6E:4D:05:F0:F1:A7:21:55:7D:46:11:BA:47
  SHA1: 20:CE:B1:F0:F5:1C:0E:19:A9:F3:8D:B1:AA:8E:03:8C:AA:7A:C7:01
  SHA256:
 CB:6B:05:D9:E8:E5:7C:D8:82:B1:0B:4D:B7:0D:E4:BB:1D:E4:2B:A4:8A:7B:D0:31:8B:63:5B:F6:E7:78:1A:9

Alias name: thawteprimaryrootca
Certificate fingerprints:
  MD5:  8C:CA:DC:0B:22:CE:F5:BE:72:AC:41:1A:11:A8:D8:12
  SHA1: 91:C6:D6:EE:3E:8A:C8:63:84:E5:48:C2:99:29:5C:75:6C:81:7B:81
  SHA256:
 8D:72:2F:81:A9:C1:13:C0:79:1D:F1:36:A2:96:6D:B2:6C:95:0A:97:1D:B4:6B:41:99:F4:EA:54:B7:8B:FB:9

Alias name: visaecommerceroot
Certificate fingerprints:
  MD5:  FC:11:B8:D8:08:93:30:00:6D:23:F9:7E:EB:52:1E:02
  SHA1: 70:17:9B:86:8C:00:A4:FA:60:91:52:22:3F:9F:3E:32:BD:E0:05:62
  SHA256:
 69:FA:C9:BD:55:FB:0A:C7:8D:53:BB:EE:5C:F1:D5:97:98:9F:D0:AA:AB:20:A2:51:51:BD:F1:73:3E:E7:D1:2

Alias name: digicertglobalrootg3
Certificate fingerprints:
  MD5:  F5:5D:A4:50:A5:FB:28:7E:1E:0F:0D:CC:96:57:56:CA
  SHA1: 7E:04:DE:89:6A:3E:66:6D:00:E6:87:D3:3F:FA:D9:3B:E8:3D:34:9E
  SHA256:
 31:AD:66:48:F8:10:41:38:C7:38:F3:9E:A4:32:01:33:39:3E:3A:18:CC:02:29:6E:F9:7C:2A:C9:EF:67:31:D

Alias name: xrampglobalca
Certificate fingerprints:
  MD5:  A1:0B:44:B3:CA:10:D8:00:6E:9D:0F:D8:0F:92:0A:D1
  SHA1: B8:01:86:D1:EB:9C:86:A5:41:04:CF:30:54:F3:4C:52:B7:E5:58:C6
```

```
  SHA256:
 CE:CD:DC:90:50:99:D8:DA:DF:C5:B1:D2:09:B7:37:CB:E2:C1:8C:FB:2C:10:C0:FF:0B:CF:0D:32:86:FC:1A:A

Alias name: digicertglobalrootg2
Certificate fingerprints:
  MD5:  E4:A6:8A:C8:54:AC:52:42:46:0A:FD:72:48:1B:2A:44
  SHA1: DF:3C:24:F9:BF:D6:66:76:1B:26:80:73:FE:06:D1:CC:8D:4F:82:A4
  SHA256:
 CB:3C:CB:B7:60:31:E5:E0:13:8F:8D:D3:9A:23:F9:DE:47:FF:C3:5E:43:C1:14:4C:EA:27:D4:6A:5A:B1:CB:5

Alias name: valicertclass2ca
Certificate fingerprints:
  MD5:  A9:23:75:9B:BA:49:36:6E:31:C2:DB:F2:E7:66:BA:87
  SHA1: 31:7A:2A:D0:7F:2B:33:5E:F5:A1:C3:4E:4B:57:E8:B7:D8:F1:FC:A6
  SHA256:
 58:D0:17:27:9C:D4:DC:63:AB:DD:B1:96:A6:C9:90:6C:30:C4:E0:87:83:EA:E8:C1:60:99:54:D6:93:55:59:6

Alias name: geotrustprimaryca
Certificate fingerprints:
  MD5:  02:26:C3:01:5E:08:30:37:43:A9:D0:7D:CF:37:E6:BF
  SHA1: 32:3C:11:8E:1B:F7:B8:B6:52:54:E2:E2:10:0D:D6:02:90:37:F0:96
  SHA256:
 37:D5:10:06:C5:12:EA:AB:62:64:21:F1:EC:8C:92:01:3F:C5:F8:2A:E9:8E:E5:33:EB:46:19:B8:DE:B4:D0:6

Alias name: netlockaranyclassgoldfotanusitvany
Certificate fingerprints:
  MD5:  C5:A1:B7:FF:73:DD:D6:D7:34:32:18:DF:FC:3C:AD:88
  SHA1: 06:08:3F:59:3F:15:A1:04:A0:69:A4:6B:A9:03:D0:06:B7:97:09:91
  SHA256:
 6C:61:DA:C3:A2:DE:F0:31:50:6B:E0:36:D2:A6:FE:40:19:94:FB:D1:3D:F9:C8:D4:66:59:92:74:C4:46:EC:9

Alias name: geotrustglobalca
Certificate fingerprints:
  MD5:  F7:75:AB:29:FB:51:4E:B7:77:5E:FF:05:3C:99:8E:F5
  SHA1: DE:28:F4:A4:FF:E5:B9:2F:A3:C5:03:D1:A3:49:A7:F9:96:2A:82:12
  SHA256:
 FF:85:6A:2D:25:1D:CD:88:D3:66:56:F4:50:12:67:98:CF:AB:AA:DE:40:79:9C:72:2D:E4:D2:B5:DB:36:A7:3

Alias name: oistewisekeyglobalrootgbca
Certificate fingerprints:
  MD5:  A4:EB:B9:61:28:2E:B7:2F:98:B0:35:26:90:99:51:1D
  SHA1: 0F:F9:40:76:18:D3:D7:6A:4B:98:F0:A8:35:9E:0C:FD:27:AC:CC:ED
  SHA256:
 6B:9C:08:E8:6E:B0:F7:67:CF:AD:65:CD:98:B6:21:49:E5:49:4A:67:F5:84:5E:7B:D1:ED:01:9F:27:B8:6B:D
```

```
Alias name: certumtrustednetworkca2
Certificate fingerprints:
  MD5:  6D:46:9E:D9:25:6D:08:23:5B:5E:74:7D:1E:27:DB:F2
  SHA1: D3:DD:48:3E:2B:BF:4C:05:E8:AF:10:F5:FA:76:26:CF:D3:DC:30:92
  SHA256:
 B6:76:F2:ED:DA:E8:77:5C:D3:6C:B0:F6:3C:D1:D4:60:39:61:F4:9E:62:65:BA:01:3A:2F:03:07:B6:D0:B8:0

Alias name: starfieldservicesrootcertificateauthorityg2
Certificate fingerprints:
  MD5:  17:35:74:AF:7B:61:1C:EB:F4:F9:3C:E2:EE:40:F9:A2
  SHA1: 92:5A:8F:8D:2C:6D:04:E0:66:5F:59:6A:FF:22:D8:63:E8:25:6F:3F
  SHA256:
 56:8D:69:05:A2:C8:87:08:A4:B3:02:51:90:ED:CF:ED:B1:97:4A:60:6A:13:C6:E5:29:0F:CB:2A:E6:3E:DA:B

Alias name: comodorsacertificationauthority
Certificate fingerprints:
  MD5:  1B:31:B0:71:40:36:CC:14:36:91:AD:C4:3E:FD:EC:18
  SHA1: AF:E5:D2:44:A8:D1:19:42:30:FF:47:9F:E2:F8:97:BB:CD:7A:8C:B4
  SHA256:
 52:F0:E1:C4:E5:8E:C6:29:29:1B:60:31:7F:07:46:71:B8:5D:7E:A8:0D:5B:07:27:34:63:53:4B:32:B4:02:3

Alias name: comodoaaaca
Certificate fingerprints:
  MD5:  49:79:04:B0:EB:87:19:AC:47:B0:BC:11:51:9B:74:D0
  SHA1: D1:EB:23:A4:6D:17:D6:8F:D9:25:64:C2:F1:F1:60:17:64:D8:E3:49
  SHA256:
 D7:A7:A0:FB:5D:7E:27:31:D7:71:E9:48:4E:BC:DE:F7:1D:5F:0C:3E:0A:29:48:78:2B:C8:3E:E0:EA:69:9E:F

Alias name: identrustpublicsectorrootca1
Certificate fingerprints:
  MD5:  37:06:A5:B0:FC:89:9D:BA:F4:6B:8C:1A:64:CD:D5:BA
  SHA1: BA:29:41:60:77:98:3F:F4:F3:EF:F2:31:05:3B:2E:EA:6D:4D:45:FD
  SHA256:
 30:D0:89:5A:9A:44:8A:26:20:91:63:55:22:D1:F5:20:10:B5:86:7A:CA:E1:2C:78:EF:95:8F:D4:F4:38:9F:2

Alias name: certplusclass2primaryca
Certificate fingerprints:
  MD5:  88:2C:8C:52:B8:A2:3C:F3:F7:BB:03:EA:AE:AC:42:0B
  SHA1: 74:20:74:41:72:9C:DD:92:EC:79:31:D8:23:10:8D:C2:81:92:E2:BB
  SHA256:
 0F:99:3C:8A:EF:97:BA:AF:56:87:14:0E:D5:9A:D1:82:1B:B4:AF:AC:F0:AA:9A:58:B5:D5:7A:33:8A:3A:FB:C

Alias name: ttelesecglobalrootclass2ca
```

```
Certificate fingerprints:
  MD5:  2B:9B:9E:E4:7B:6C:1F:00:72:1A:CC:C1:77:79:DF:6A
  SHA1: 59:0D:2D:7D:88:4F:40:2E:61:7E:A5:62:32:17:65:CF:17:D8:94:E9
  SHA256:
 91:E2:F5:78:8D:58:10:EB:A7:BA:58:73:7D:E1:54:8A:8E:CA:CD:01:45:98:BC:0B:14:3E:04:1B:17:05:25:5

Alias name: accvraiz1
Certificate fingerprints:
  MD5:  D0:A0:5A:EE:05:B6:09:94:21:A1:7D:F1:B2:29:82:02
  SHA1: 93:05:7A:88:15:C6:4F:CE:88:2F:FA:91:16:52:28:78:BC:53:64:17
  SHA256:
 9A:6E:C0:12:E1:A7:DA:9D:BE:34:19:4D:47:8A:D7:C0:DB:18:22:FB:07:1D:F1:29:81:49:6E:D1:04:38:41:1

Alias name: digicerthighassuranceevrootca
Certificate fingerprints:
  MD5:  D4:74:DE:57:5C:39:B2:D3:9C:85:83:C5:C0:65:49:8A
  SHA1: 5F:B7:EE:06:33:E2:59:DB:AD:0C:4C:9A:E6:D3:8F:1A:61:C7:DC:25
  SHA256:
 74:31:E5:F4:C3:C1:CE:46:90:77:4F:0B:61:E0:54:40:88:3B:A9:A0:1E:D0:0B:A6:AB:D7:80:6E:D3:B1:18:C

Alias name: amzninternalinfoseccag3
Certificate fingerprints:
  MD5:  E9:34:94:02:BA:BB:31:6B:22:E6:2B:A9:C4:F0:26:04
  SHA1: B9:B1:CA:38:F7:BF:9C:D2:D4:95:E7:B6:5E:75:32:9B:A8:78:2E:F6
  SHA256:
 81:03:0B:C7:E2:54:DA:7B:F8:B7:45:DB:DD:41:15:89:B5:A3:81:86:FB:4B:29:77:1F:84:0A:18:D9:67:6D:6

Alias name: cia-crt-g3-02-ca
Certificate fingerprints:
  MD5:  FD:B9:23:FD:D3:EB:2D:3E:57:EF:56:FF:DB:D3:E4:B9
  SHA1: 96:4A:BB:A7:BD:DA:FC:97:34:C0:0A:2D:F0:05:98:F7:E6:C6:6F:09
  SHA256:
 93:F1:72:FB:BA:43:31:5C:06:EE:0F:9F:04:89:B8:F6:88:BC:75:15:3C:BE:B4:80:AC:A7:14:3A:F6:FC:4A:C

Alias name: entrustrootcertificationauthorityec1
Certificate fingerprints:
  MD5:  B6:7E:1D:F0:58:C5:49:6C:24:3B:3D:ED:98:18:ED:BC
  SHA1: 20:D8:06:40:DF:9B:25:F5:12:25:3A:11:EA:F7:59:8A:EB:14:B5:47
  SHA256:
 02:ED:0E:B2:8C:14:DA:45:16:5C:56:67:91:70:0D:64:51:D7:FB:56:F0:B2:AB:1D:3B:8E:B0:70:E5:6E:DF:F

Alias name: securitycommunicationrootca
Certificate fingerprints:
  MD5:  F1:BC:63:6A:54:E0:B5:27:F5:CD:E7:1A:E3:4D:6E:4A
```

```
  SHA1: 36:B1:2B:49:F9:81:9E:D7:4C:9E:BC:38:0F:C6:56:8F:5D:AC:B2:F7
  SHA256:
 E7:5E:72:ED:9F:56:0E:EC:6E:B4:80:00:73:A4:3F:C3:AD:19:19:5A:39:22:82:01:78:95:97:4A:99:02:6B:6

Alias name: globalsignca
Certificate fingerprints:
  MD5:  3E:45:52:15:09:51:92:E1:B7:5D:37:9F:B1:87:29:8A
  SHA1: B1:BC:96:8B:D4:F4:9D:62:2A:A8:9A:81:F2:15:01:52:A4:1D:82:9C
  SHA256:
 EB:D4:10:40:E4:BB:3E:C7:42:C9:E3:81:D3:1E:F2:A4:1A:48:B6:68:5C:96:E7:CE:F3:C1:DF:6C:D4:33:1C:9

Alias name: trustcenterclass2caii
Certificate fingerprints:
  MD5:  CE:78:33:5C:59:78:01:6E:18:EA:B9:36:A0:B9:2E:23
  SHA1: AE:50:83:ED:7C:F4:5C:BC:8F:61:C6:21:FE:68:5D:79:42:21:15:6E
  SHA256:
 E6:B8:F8:76:64:85:F8:07:AE:7F:8D:AC:16:70:46:1F:07:C0:A1:3E:EF:3A:1F:F7:17:53:8D:7A:BA:D3:91:B

Alias name: camerfirmachambersofcommerceroot
Certificate fingerprints:
  MD5:  B0:01:EE:14:D9:AF:29:18:94:76:8E:F1:69:33:2A:84
  SHA1: 6E:3A:55:A4:19:0C:19:5C:93:84:3C:C0:DB:72:2E:31:30:61:F0:B1
  SHA256:
 0C:25:8A:12:A5:67:4A:EF:25:F2:8B:A7:DC:FA:EC:EE:A3:48:E5:41:E6:F5:CC:4E:E6:3B:71:B3:61:60:6A:C

Alias name: geotrustprimarycag3
Certificate fingerprints:
  MD5:  B5:E8:34:36:C9:10:44:58:48:70:6D:2E:83:D4:B8:05
  SHA1: 03:9E:ED:B8:0B:E7:A0:3C:69:53:89:3B:20:D2:D9:32:3A:4C:2A:FD
  SHA256:
 B4:78:B8:12:25:0D:F8:78:63:5C:2A:A7:EC:7D:15:5E:AA:62:5E:E8:29:16:E2:CD:29:43:61:88:6C:D1:FB:D

Alias name: geotrustprimarycag2
Certificate fingerprints:
  MD5:  01:5E:D8:6B:BD:6F:3D:8E:A1:31:F8:12:E0:98:73:6A
  SHA1: 8D:17:84:D5:37:F3:03:7D:EC:70:FE:57:8B:51:9A:99:E6:10:D7:B0
  SHA256:
 5E:DB:7A:C4:3B:82:A0:6A:87:61:E8:D7:BE:49:79:EB:F2:61:1F:7D:D7:9B:F9:1C:1C:6B:56:6A:21:9E:D7:6

Alias name: hongkongpostrootca1
Certificate fingerprints:
  MD5:  A8:0D:6F:39:78:B9:43:6D:77:42:6D:98:5A:CC:23:CA
  SHA1: D6:DA:A8:20:8D:09:D2:15:4D:24:B5:2F:CB:34:6E:B2:58:B2:8A:58
```

```
  SHA256:
 F9:E6:7D:33:6C:51:00:2A:C0:54:C6:32:02:2D:66:DD:A2:E7:E3:FF:F1:0A:D0:61:ED:31:D8:BB:B4:10:CF:B

Alias name: affirmtrustpremiumeccca
Certificate fingerprints:
  MD5:  64:B0:09:55:CF:B1:D5:99:E2:BE:13:AB:A6:5D:EA:4D
  SHA1: B8:23:6B:00:2F:1D:16:86:53:01:55:6C:11:A4:37:CA:EB:FF:C3:BB
  SHA256:
 BD:71:FD:F6:DA:97:E4:CF:62:D1:64:7A:DD:25:81:B0:7D:79:AD:F8:39:7E:B4:EC:BA:9C:5E:84:88:82:14:2

Alias name: hellenicacademicandresearchinstitutionsrootca2015
Certificate fingerprints:
  MD5:  CA:FF:E2:DB:03:D9:CB:4B:E9:0F:AD:84:FD:7B:18:CE
  SHA1: 01:0C:06:95:A6:98:19:14:FF:BF:5F:C6:B0:B6:95:EA:29:E9:12:A6
  SHA256:
 A0:40:92:9A:02:CE:53:B4:AC:F4:F2:FF:C6:98:1C:E4:49:6F:75:5E:6D:45:FE:0B:2A:69:2B:CD:52:52:3F:3
```

AWS IoT Events

La acción AWS IoT Events (`iotEvents`) envía los datos de un mensaje MQTT a una AWS IoT Events entrada.

> **Important**
>
> Si la carga útil se envía AWS IoT Core sin la`Input attribute Key`, o si la clave no está en la misma ruta JSON especificada en la clave, se producirá un error `Failed to send message to Iot Events` en la regla de IoT.

Requisitos

Esta regla tiene los siguientes requisitos:

- Un rol de IAM que AWS IoT puede asumir para realizar la `iotevents:BatchPutMessage` operación. Para obtener más información, consulte Otorgar a una AWS IoT regla el acceso que requiere.

  En la AWS IoT consola, puede elegir o crear un rol que permita AWS IoT realizar esta acción de regla.

Parameters

Al crear una AWS IoT regla con esta acción, debe especificar la siguiente información:

`batchMode`

(Opcional) Si se procesan las acciones de evento como un lote. El valor predeterminado es `false`.

Cuando `batchMode` es `true` y la sentencia SQL de la regla se evalúa como una matriz, cada elemento de la matriz se trata como un mensaje independiente cuando se envía a AWS IoT Events mediante una llamada `BatchPutMessage`. La matriz resultante no puede tener más de 10 mensajes.

Cuando `batchMode` es `true`, no puede especificar un `messageId`.

Admite plantillas de sustitución: No

`inputName`

El nombre de la AWS IoT Events entrada.

Soporta plantillas de sustitución: API y AWS CLI únicamente

`messageId`

(Opcional) Use esto para verificar que un AWS IoT Events detector procese solo una entrada (mensaje) con un dato determinado`messageId`. Puede utilizar la plantilla de sustitución `${newuuid()}` para generar un identificador único para cada solicitud.

Cuando `batchMode` es `true`, no puede especificar un `messageId`. Se asignará un nuevo valor UUID.

Admite plantillas de sustitución: Sí

`roleArn`

La función de IAM que permite AWS IoT enviar una entrada a un AWS IoT Events detector. Para obtener más información, consulte Requisitos.

Admite plantillas de sustitución: No

Ejemplos

El siguiente ejemplo JSON define una acción de Eventos IoT en una regla AWS IoT .

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "iotEvents": {
                    "inputName": "MyIoTEventsInput",
                    "messageId": "${newuuid()}",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_events"
                }
            }
        ]
    }
}
```

Véase también

- ¿Qué es? AWS IoT Events en la Guía AWS IoT Events para desarrolladores

AWS IoT SiteWise

La acción AWS IoT SiteWise (`iotSiteWise`) envía los datos de un mensaje MQTT a las propiedades del activo. AWS IoT SiteWise

Puedes seguir un tutorial que te muestra cómo ingerir datos de AWS IoT cosas. Para obtener más información, consulta el tutorial sobre cómo ingerir datos AWS IoT SiteWise desde AWS IoT cosas o la sección Cómo ingerir datos mediante las reglas AWS IoT básicas de la Guía del AWS IoT SiteWise usuario.

Requisitos

Esta regla tiene los siguientes requisitos:

- Una función de IAM que AWS IoT puede asumir para realizar la operación. `iotsitewise:BatchPutAssetPropertyValue` Para obtener más información, consulte Otorgar a una AWS IoT regla el acceso que requiere.

  Puede asociar el siguiente ejemplo de política de confianza al rol.

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

Para mejorar la seguridad, puede especificar una ruta jerárquica de AWS IoT SiteWise activos en la `Condition` propiedad. El siguiente ejemplo es una política de confianza que especifica una ruta jerárquica de activos.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iotsitewise:BatchPutAssetPropertyValue",
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "iotsitewise:assetHierarchyPath": [
                        "/root node asset ID",
                        "/root node asset ID/*"
                    ]
            }
        }
        }
    ]
}
```

- Al enviar datos a AWS IoT SiteWise mediante esta acción, estos deben cumplir los requisitos de la `BatchPutAssetPropertyValue` operación. Para obtener más información, consulta BatchPutAssetPropertyValue en la AWS IoT SiteWise Referencia de la API de .

Parameters

Al crear una AWS IoT regla con esta acción, debe especificar la siguiente información:

`putAssetPropertyValueEntries`

Una lista de entradas de valor de propiedad de recurso, en la que cada entrada contiene la siguiente información:

`propertyAlias`

(Opcional) El alias de propiedad asociado a su propiedad de activo. Debe especificar un `propertyAlias` o tanto un `assetId` como un `propertyId`. Para obtener más información acerca de los alias de propiedades, consulte Mapeo de flujos de datos industriales a propiedades de recursos en la Guía del usuario de AWS IoT SiteWise .

Admite plantillas de sustitución: Sí

`assetId`

(Opcional) El ID del AWS IoT SiteWise activo. Debe especificar un `propertyAlias` o tanto un `assetId` como un `propertyId`.

Admite plantillas de sustitución: Sí

`propertyId`

(Opcional) El ID de la propiedad de activo. Debe especificar un `propertyAlias` o tanto un `assetId` como un `propertyId`.

Admite plantillas de sustitución: Sí

`entryId`

(Opcional) Un identificador único para esta entrada. Defina el `entryId` para rastrear mejor qué mensaje causó un error si se produce un fallo. El valor predeterminado es un nuevo UUID.

Admite plantillas de sustitución: Sí

`propertyValues`

Una lista de valores de propiedad que se van a insertar, en la que cada valor contiene una marca temporal, la calidad y el valor (TQV) en el formato siguiente:

`timestamp`

Una estructura de marca temporal que contiene la siguiente información:

`timeInSeconds`

Una cadena que contiene el tiempo en segundos en formato de tiempo Unix. Si su carga de mensajes no tiene una marca temporal, puede usar timestamp(), que devuelve la hora actual en milisegundos. Para convertir ese tiempo en segundos, puede utilizar la siguiente plantilla de sustitución: **`${floor(timestamp() / 1E3)}`**.

Admite plantillas de sustitución: Sí

`offsetInNanos`

(Opcional) Una cadena que contiene el desfase de tiempo en nanosegundos a partir del tiempo en segundos. Si su carga de mensajes no tiene una marca temporal, puede usar timestamp(), que devuelve la hora actual en milisegundos. Para calcular el desfase en nanosegundos a partir de ese tiempo, puede utilizar la siguiente plantilla de sustitución: **`${(timestamp() % 1E3) * 1E6}`**.

Admite plantillas de sustitución: Sí

Con respecto a la época de Unix, solo AWS IoT SiteWise acepta entradas que tengan una marca de tiempo de hasta 7 días en el pasado y hasta 5 minutos en el futuro.

`quality`

(Opcional) Una cadena que describe la calidad del valor. Valores válidos: `GOOD`, `BAD`, `UNCERTAIN`.

Admite plantillas de sustitución: Sí

`value`

Una estructura de valores que contiene uno de los siguientes campos de valor, en función del tipo de datos de la propiedad del recurso:

`booleanValue`

(Opcional) Una cadena que contiene el valor booleano de la entrada del valor.

Admite plantillas de sustitución: Sí

`doubleValue`

(Opcional) Una cadena que contiene el valor “double” (doble) de la entrada del valor.

Admite plantillas de sustitución: Sí

`integerValue`

(Opcional) Una cadena que contiene el valor entero de la entrada del valor.

Admite plantillas de sustitución: Sí

`stringValue`

(Opcional) El valor de cadena de la entrada del valor.

Admite plantillas de sustitución: Sí

`roleArn`

El ARN de la función de IAM que concede el AWS IoT permiso para enviar el valor de una propiedad de un activo a. AWS IoT SiteWise Para obtener más información, consulte Requisitos.

Admite plantillas de sustitución: No

Ejemplos

El siguiente ejemplo de JSON define una SiteWise acción básica de IoT en una AWS IoT regla.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "iotSiteWise": {
                    "putAssetPropertyValueEntries": [
                        {
                            "propertyAlias": "/some/property/alias",
                            "propertyValues": [
                                {
                                    "timestamp": {
                                        "timeInSeconds": "${my.payload.timeInSeconds}"
                                    },
                                    "value": {
                                        "integerValue": "${my.payload.value}"
```

```
                                    }
                                }
                            ]
                        }
                    ],
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_sitewise"
                }
            }
        ]
    }
}
```

El siguiente ejemplo de JSON define una SiteWise acción de IoT en una AWS IoT regla. En este ejemplo se utiliza el tema como alias de propiedad y función `timestamp()`. Por ejemplo, si publica datos en `/company/windfarm/3/turbine/7/rpm`, esta acción envía los datos a la propiedad del recurso con un alias de propiedad que es el mismo que el tema especificado.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM '/company/windfarm/+/turbine/+/+'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "iotSiteWise": {
                    "putAssetPropertyValueEntries": [
                        {
                            "propertyAlias": "${topic()}",
                            "propertyValues": [
                                {
                                    "timestamp": {
                                        "timeInSeconds": "${floor(timestamp() / 1E3)}",
                                        "offsetInNanos": "${(timestamp() % 1E3) * 1E6}"
                                    },
                                    "value": {
                                        "doubleValue": "${my.payload.value}"
                                    }
                                }
                            ]
                        }
                    ],
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_sitewise"
                }
```

```
            }
        ]
    }
}
```

Véase también

- ¿Qué es AWS IoT SiteWise? en la Guía del usuario de AWS IoT SiteWise
- Ingerir datos mediante AWS IoT Core las reglas de la Guía del AWS IoT SiteWise usuario
- Ingerir datos a AWS IoT SiteWise partir de AWS IoT elementos de la Guía del usuario AWS IoT SiteWise
- Solución de problemas de una AWS IoT SiteWise acción regulada en la Guía del AWS IoT SiteWise usuario

Firehose

La acción de Firehose (`firehose`) envía datos de un mensaje MQTT a un flujo de Amazon Data Firehose.

Requisitos

Esta regla tiene los siguientes requisitos:

- Un rol de IAM que AWS IoT puede asumir para realizar la operación. `firehose:PutRecord` Para obtener más información, consulte Otorgar a una AWS IoT regla el acceso que requiere.

  En la AWS IoT consola, puede elegir o crear un rol que permita AWS IoT realizar esta acción de regla.
- Si utilizas Firehose para enviar datos a un bucket de Amazon S3 y utilizas un AWS KMS cliente gestionado AWS KMS key para cifrar los datos en reposo en Amazon S3, Firehose debe tener acceso a tu bucket y permiso para usarlo AWS KMS key en nombre de la persona que llama. Para obtener más información, consulte Grant Firehose access to an Amazon S3 destination en la Guía para desarrolladores de Amazon Data Firehose.

Parameters

Al crear una AWS IoT regla con esta acción, debe especificar la siguiente información:

`batchMode`

(Opcional) Si entregar o no el flujo de Firehose como lote con `PutRecordBatch`. El valor predeterminado es `false`.

Cuando `batchMode` es `true` y la instrucción de SQL de la regla se evalúa como una matriz, cada elemento de la matriz forma un registro en la solicitud `PutRecordBatch`. La matriz resultante no puede tener más de 500 registros.

Admite plantillas de sustitución: No

`deliveryStreamName`

El flujo de Firehose en el que deben escribirse los datos del mensaje.

Admite plantillas de sustitución: API y AWS CLI únicamente

`separator`

(Opcional) Un separador de caracteres que se utilizará para separar registros escritos en el flujo de Firehose. Si omite este parámetro, el flujo no utiliza ningún separador. Valores válidos: `,` (coma), `\t` (tabulador), (nueva línea), `\n` (nueva línea de Windows). `\r\n`

Admite plantillas de sustitución: No

`roleArn`

El rol de IAM que permite acceder al flujo de Firehose. Para obtener más información, consulte Requisitos.

Admite plantillas de sustitución: No

Ejemplos

El siguiente ejemplo de JSON define una acción Firehose en una AWS IoT regla.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
```

```
            {
                "firehose": {
                    "deliveryStreamName": "my_firehose_stream",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_firehose"
                }
            }
        ]
    }
}
```

El siguiente ejemplo de JSON define una acción Firehose con plantillas de sustitución en una AWS IoT regla.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "firehose": {
                    "deliveryStreamName": "${topic()}",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_firehose"
                }
            }
        ]
    }
}
```

Véase también

- ¿What is Amazon Data Firehose? en la Guía para desarrolladores de Amazon Data Firehose

Kinesis Data Streams

La acción Kinesis Data Streams (`kinesis`) escribe datos de un mensaje MQTT en Amazon Kinesis Data Streams.

Requisitos

Esta regla tiene los siguientes requisitos:

- Una función de IAM que AWS IoT puede asumir para realizar la operación. `kinesis:PutRecord` Para obtener más información, consulte Otorgar a una AWS IoT regla el acceso que requiere.

  En la AWS IoT consola, puede elegir o crear un rol que permita AWS IoT realizar esta acción de regla.
- Si utiliza una clave KMS AWS KMS administrada por el cliente AWS KMS key para cifrar los datos en reposo en Kinesis Data Streams, el servicio debe tener permiso para utilizarla en nombre de AWS KMS key la persona que llama. Para obtener más información, consulte Permisos de uso generados por el usuario AWS KMS keys en la guía para desarrolladores de Amazon Kinesis Data Streams.

Parameters

Al crear una AWS IoT regla con esta acción, debe especificar la siguiente información:

`stream`

El flujo de datos Kinesis en el que escribir los datos.

Admite plantillas de sustitución: API y AWS CLI únicamente

`partitionKey`

La clave de partición utilizada para determinar en qué fragmento se escriben los datos. La clave de partición suele estar compuesta por una expresión (por ejemplo, `${topic()}` o `${timestamp()}`).

Admite plantillas de sustitución: Sí

`roleArn`

El ARN del rol de IAM que concede el AWS IoT permiso para acceder a la transmisión de datos de Kinesis. Para obtener más información, consulte Requisitos.

Admite plantillas de sustitución: No

Ejemplos

El siguiente ejemplo de JSON define una acción de Kinesis Data Streams en AWS IoT una regla.

```
{
```

```
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "kinesis": {
                    "streamName": "my_kinesis_stream",
                    "partitionKey": "${topic()}",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_kinesis"
                }
            }
        ]
    }
}
```

El siguiente ejemplo de JSON define una acción de Kinesis con plantillas de sustitución en una AWS IoT regla.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "kinesis": {
                    "streamName": "${topic()}",
                    "partitionKey": "${timestamp()}",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_kinesis"
                }
            }
        ]
    }
}
```

Véase también

- ¿Qué es Amazon Kinesis Data Streams? en la Guía para desarrolladores de Amazon Kinesis Data Streams.

Lambda

Una acción Lambda (`lambda`) invoca una AWS Lambda función y pasa un mensaje MQTT. AWS IoT invoca funciones Lambda de forma asíncrona.

Puede seguir un tutorial que muestra cómo crear y probar una regla con una acción de Lambda. Para obtener más información, consulte Tutorial: Formatear una notificación mediante una AWS Lambda función.

Requisitos

Esta regla tiene los siguientes requisitos:

- AWS IoT Para invocar una función Lambda, debe configurar una política que conceda `lambda:InvokeFunction` el permiso a. AWS IoT Solo puede invocar una función de Lambda definida en el Región de AWS mismo lugar donde existe su política de Lambda. Las funciones de Lambda utilizan políticas basadas en recursos, por lo que debe asociar la política a la función de Lambda en sí.

  Utilice el siguiente AWS CLI comando para adjuntar una política que conceda el permiso. `lambda:InvokeFunction` En este comando, sustituya:

  - *`function_name`*con el nombre de la función Lambda. Agrega un nuevo permiso para actualizar la política de recursos de la función.
  - *`region`*con el Región de AWS de la función.
  - *`account-id`*con el Cuenta de AWS número en el que se define la regla.
  - *`rule-name`*con el nombre de la AWS IoT regla para la que está definiendo la acción Lambda.
  - *`unique_id`*con un identificador de declaración único.

  > **Important**
  >
  > Si agrega un permiso para un AWS IoT principal sin proporcionar el `source-arn` `osource-account`, cualquiera Cuenta de AWS que cree una regla con su acción de Lambda puede activar reglas desde las que invocar la función de Lambda. AWS IoT

  Para obtener más información, consulte Permisos de AWS Lambda.

```
aws lambda add-permission \
```

```
    --function-name function_name \
    --region region \
    --principal iot.amazonaws.com \
    --source-arn arn:aws:iot:region:account-id:rule/rule_name \
    --source-account account-id
    --statement-id unique_id
    --action "lambda:InvokeFunction"
```

- Si utiliza la AWS IoT consola para crear una regla para la acción de la regla Lambda, la función Lambda se activa automáticamente. Si la usa AWS CloudFormation en su lugar con `AWS::IoT::TopicRule LambdaAction`, debe agregar un `AWS::lambda::Permission`recurso. A continuación, el recurso le concede permiso para desencadenar la función de Lambda.

  En el siguiente código se muestra un ejemplo de cómo agregar este recurso. En este ejemplo, sustituya:

  - *`function_name`*con el nombre de la función Lambda.
  - *`region`*con el Región de AWS de la función.
  - *`account-id`*con el Cuenta de AWS número en el que se define la regla.
  - *`rule-name`*con el nombre de la AWS IoT regla para la que está definiendo la acción Lambda.

```
Type: AWS::Lambda::Permission
Properties:
  Action: lambda:InvokeFunction
  FunctionName: !Ref function_name
  Principal: "iot.amazonaws.com"
  SourceAccount: account-id
  SourceArn: arn:aws:iot:region:account-id:rule/rule_name
```

- Si utiliza un AWS KMS cliente gestionado AWS KMS key para cifrar datos en reposo en Lambda, el servicio debe tener permiso para utilizarlos en nombre de AWS KMS key la persona que llama. Para obtener más información, consulte Encryption at rest (Cifrado en reposo) en la Guía para desarrolladores de AWS Lambda .

Parameters

Al crear una AWS IoT regla con esta acción, debe especificar la siguiente información:

`functionArn`

El ARN de la función Lambda que se va a invocar. AWS IoT debe tener permiso para invocar la función. Para obtener más información, consulte Requisitos.

Si no especifica una versión o un alias para su función de Lambda, se cerrará la versión más reciente de la función. Puede especificar una versión o un alias si desea cerrar una versión específica de su función de Lambda. Para especificar una versión o alias, añada la versión o el alias en el ARN de la función de Lambda.

```
arn:aws:lambda:us-east-2:123456789012:function:myLambdaFunction:someAlias
```

Para obtener más información acerca del control de versiones y los alias, consulte Control de versiones y alias de las funciones de AWS Lambda.

Admite plantillas de sustitución: API y únicamente AWS CLI

Ejemplos

El siguiente ejemplo de JSON define una acción Lambda en una AWS IoT regla.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "lambda": {
                    "functionArn": "arn:aws:lambda:us-
east-2:123456789012:function:myLambdaFunction"
                }
            }
        ]
    }
}
```

El siguiente ejemplo de JSON define una acción Lambda con plantillas de sustitución en una AWS IoT regla.

```
{
```

```
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "lambda": {
                    "functionArn": "arn:aws:lambda:us-east-1:123456789012:function:
${topic()}"
                }
            }
        ]
    }
}
```

Véase también

- ¿Qué es? AWS Lambda en la Guía AWS Lambda para desarrolladores
- Tutorial: Formatear una notificación mediante una AWS Lambda función

Ubicación

La acción Ubicación (`location`) envía sus datos de ubicación geográfica a Amazon Location Service.

Requisitos

Esta regla tiene los siguientes requisitos:

- Una función de IAM que AWS IoT puede asumir para realizar la `geo:BatchUpdateDevicePosition` operación. Para obtener más información, consulte Otorgar a una AWS IoT regla el acceso que requiere.

  En la AWS IoT consola, puede elegir o crear un rol que permita AWS IoT realizar esta acción de regla.

Parameters

Al crear una AWS IoT regla con esta acción, debe especificar la siguiente información:

`deviceId`

El identificador único del dispositivo que proporciona los datos de ubicación. Para obtener más información, consulte la `DeviceId` de la Referencia de la API de Amazon Location Service.

Admite plantillas de sustitución: Sí

`latitude`

Una cadena que se evalúa como un valor doble que representa la latitud de la ubicación del dispositivo.

Admite plantillas de sustitución: Sí

`longitude`

Una cadena que se evalúa como un valor doble que representa la longitud de la ubicación del dispositivo.

Admite plantillas de sustitución: Sí

`roleArn`

El rol de IAM que permite el acceso al dominio de Amazon Location Service. Para obtener más información, consulte Requisitos.

`timestamp`

La hora en que se muestrearon los datos de ubicación. El valor predeterminado es la hora en que se procesó el mensaje MQTT.

El valor `timestamp` se compone de los siguientes dos valores:

- `value`: Una expresión que devuelve un valor de tiempo de época larga. Puede utilizar la función the section called “time_to_epoch(String, String)” para crear una marca de tiempo válida a partir de un valor de fecha u hora incluido en la carga útil del mensaje. Admite plantillas de sustitución: Sí
- `unit`: (Opcional) La precisión del valor de marca de tiempo que resulta de la expresión que se describe en `value`. Valores válidos: `SECONDS` | `MILLISECONDS` | `MICROSECONDS` | `NANOSECONDS` El valor predeterminado es `MILLISECONDS`. Admite plantillas de sustitución: API y AWS CLI únicamente.

`trackerName`

El nombre del recurso de seguimiento de Amazon Location en el que se actualiza la ubicación. Para obtener más información, consulte Rastreador en la Guía para desarrolladores de Amazon Location Service.

Admite plantillas de sustitución: API y AWS CLI únicamente

Ejemplos

El siguiente ejemplo de JSON define una acción de ubicación en una AWS IoT regla.

```
{
 "topicRulePayload": {
  "sql": "SELECT * FROM 'some/topic'",
  "ruleDisabled": false,
  "awsIotSqlVersion": "2016-03-23",
  "actions": [
   {
    "location": {
     "roleArn": "arn:aws:iam::123454962127:role/service-role/ExampleRole",
     "trackerName": "MyTracker",
     "deviceId": "001",
     "sampleTime": {
      "value": "${timestamp()}",
      "unit": "MILLISECONDS"
     },
     "latitude": "-12.3456",
     "longitude": "65.4321"
    }
   }
  ]
 }
}
```

El siguiente ejemplo de JSON define una acción de ubicación con plantillas de sustitución en una AWS IoT regla.

```
{
 "topicRulePayload": {
  "sql": "SELECT * FROM 'some/topic'",
  "ruleDisabled": false,
```

```
  "awsIotSqlVersion": "2016-03-23",
  "actions": [
   {
    "location": {
     "roleArn": "arn:aws:iam::123456789012:role/service-role/ExampleRole",
     "trackerName": "${TrackerName}",
     "deviceId": "${DeviceID}",
     "timestamp": {
      "value": "${timestamp()}",
      "unit": "MILLISECONDS"
     },
     "latitude": "${get(position, 0)}",
     "longitude": "${get(position, 1)}"
    }
   }
  ]
 }
}
```

El siguiente ejemplo de carga útil de MQTT muestra cómo las plantillas de sustitución del ejemplo anterior acceden a los datos. Puede usar el comando CLS get-device-position-history para verificar que los datos de carga útil MQTT se entreguen en su rastreador de ubicación.

```
{
 "TrackerName": "mytracker",
 "DeviceID": "001",
 "position": [
  "-12.3456",
  "65.4321"
 ]
}
```

```
aws location get-device-position-history --device-id 001 --tracker-name mytracker
```

```
{
 "DevicePositions": [
  {
   "DeviceId": "001",
   "Position": [
    -12.3456,
    65.4321
   ],
```

```
    "ReceivedTime": "2022-11-11T01:31:54.464000+00:00",
    "SampleTime": "2022-11-11T01:31:54.308000+00:00"
   }
  ]
}
```

Véase también

- ¿Qué es Amazon Location Service? en la Guía para desarrolladores de Amazon Location Service.

OpenSearch

La acción OpenSearch (`openSearch`) escribe los datos de los mensajes MQTT en un dominio de Amazon OpenSearch Service. A continuación, puede utilizar herramientas como los OpenSearch paneles de control para consultar y visualizar los datos en OpenSearch Service.

Requisitos

Esta regla tiene los siguientes requisitos:

- Una función de IAM que AWS IoT puede asumir para realizar la `es:ESHttpPut` operación. Para obtener más información, consulte Otorgar a una AWS IoT regla el acceso que requiere.

  En la AWS IoT consola, puede elegir o crear un rol que permita AWS IoT realizar esta acción de regla.
- Si utiliza un cliente que ha gestionado AWS KMS key el cifrado de datos en reposo en el OpenSearch Servicio, el servicio debe tener permiso para utilizar la clave KMS en nombre de la persona que llama. Para obtener más información, consulta Cifrado de datos en reposo para Amazon OpenSearch Service en la Guía para desarrolladores de Amazon OpenSearch Service.

Parameters

Al crear una AWS IoT regla con esta acción, debe especificar la siguiente información:

`endpoint`

El punto final de tu dominio OpenSearch de Amazon Service.

Admite plantillas de sustitución: API y AWS CLI únicamente

`index`

El OpenSearch índice en el que desea almacenar sus datos.

Admite plantillas de sustitución: Sí

`type`

Tipo de documento que está almacenando.

> **Note**
>
> Para OpenSearch las versiones posteriores a la 1.0, el valor del `type` parámetro debe ser`_doc`. Para obtener más información, consulte la Documentación de OpenSearch .

Admite plantillas de sustitución: Sí

`id`

Identificador único de cada documento.

Admite plantillas de sustitución: Sí

`roleARN`

La función de IAM que permite el acceso al dominio del OpenSearch servicio. Para obtener más información, consulte Requisitos.

Admite plantillas de sustitución: No

Limitaciones

La acción OpenSearch (`openSearch`) no se puede usar para entregar datos a los clústeres de Elasticsearch de VPC.

Ejemplos

El siguiente ejemplo de JSON define una OpenSearch acción en una AWS IoT regla y cómo se pueden especificar los campos de la acción. `OpenSearch` Para obtener más información, consulte OpenSearchAction.

```
{
    "topicRulePayload": {
        "sql": "SELECT *, timestamp() as timestamp FROM 'iot/test'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "openSearch": {
                    "endpoint": "https://my-endpoint",
                    "index": "my-index",
                    "type": "_doc",
                    "id": "${newuuid()}",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_os"
                }
            }
        ]
    }
}
```

El siguiente ejemplo de JSON define una OpenSearch acción con plantillas de sustitución en una AWS IoT regla.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "openSearch": {
                    "endpoint": "https://my-endpoint",
                    "index": "${topic()}",
                    "type": "${type}",
                    "id": "${newuuid()}",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_os"
                }
            }
        ]
    }
}
```

> **Note**
>
> El `type` campo sustituido funciona en la OpenSearch versión 1.0. Para cualquier versión posterior a la 1.0, el valor de `type` debe ser `_doc`.

Véase también

¿Qué es Amazon OpenSearch Service? en la Guía para desarrolladores OpenSearch de Amazon Service

Volver a publicar

La acción republish (`republish`) vuelve a publicar un mensaje MQTT en otro tema MQTT.

Requisitos

Esta regla tiene los siguientes requisitos:

- Un rol de IAM que AWS IoT puede asumir para realizar la operación. `iot:Publish` Para obtener más información, consulte Otorgar a una AWS IoT regla el acceso que requiere.

  En la AWS IoT consola, puede elegir o crear un rol que permita AWS IoT realizar esta acción de regla.

Parameters

Al crear una AWS IoT regla con esta acción, debe especificar la siguiente información:

`headers`

Información acerca de los encabezados de la versión 5.0 de MQTT.

Para obtener más información consulte RepublishAction y MqttHeaders en la Referencia de la API de AWS .

`topic`

El tema MQTT en el que se va a volver a publicar el mensaje.

Para volver a publicar en un tema reservado, que comience por $, utilice $$ en su lugar. Por ejemplo, para volver a publicar en el tema sombra del dispositivo `$aws/things/MyThing/shadow/update`, especifique el tema como `$$aws/things/MyThing/shadow/update`.

> **Note**
>
> No se permite volver a publicar en temas de trabajo reservados.
> AWS IoT Device Defender los temas de reserva no admiten la publicación en HTTP.

Admite plantillas de sustitución: Sí

`qos`

(Opcional) El nivel de calidad de servicio (QoS) que se utiliza para volver a publicar mensajes. Valores válidos: `0`, `1`. El valor predeterminado es `0`. Para obtener más información acerca de MQTT QoS, consulte MQTT.

Admite plantillas de sustitución: No

`roleArn`

La función de IAM que permite AWS IoT publicar en el tema MQTT. Para obtener más información, consulte Requisitos.

Admite plantillas de sustitución: No

Ejemplos

El siguiente ejemplo de JSON define una acción de republicación en una regla. AWS IoT

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "republish": {
                    "topic": "another/topic",
                    "qos": 1,
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_republish"
```

```
                }
            }
        ]
    }
}
```

El siguiente ejemplo de JSON define una acción de republicación con plantillas de sustitución en una AWS IoT regla.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "republish": {
                    "topic": "${topic()}/republish",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_republish"
                }
            }
        ]
    }
}
```

El siguiente ejemplo de JSON define una acción de republicación dentro de una `headers` AWS IoT regla.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "republish": {
                    "topic": "${topic()}/republish",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_republish",
                    "headers": {
                        "payloadFormatIndicator": "UTF8_DATA",
                        "contentType": "rule/contentType",
                        "correlationData": "cnVsZSBjb3JyZWxhdGlvbiBkYXRh",
```

```
                        "userProperties": [
                            {
                                "key": "ruleKey1",
                                "value": "ruleValue1"
                            },
                            {
                                "key": "ruleKey2",
                                "value": "ruleValue2"
                            }
                        ]
                    }
                }
            }
        ]
    }
}
```

> **Note**
>
> La IP de origen original no se transferirá a través de la acción Republicar.

S3

La acción S3 (`s3`) escribe los datos de un mensaje MQTT en un bucket de Amazon Simple Storage Service (Amazon S3).

Requisitos

Esta regla tiene los siguientes requisitos:

- Un rol de IAM que AWS IoT puede asumir para realizar la `s3:PutObject` operación. Para obtener más información, consulte Otorgar a una AWS IoT regla el acceso que requiere.

  En la AWS IoT consola, puede elegir o crear un rol que permita AWS IoT realizar esta acción de regla.
- Si utiliza un servicio AWS KMS gestionado por el cliente AWS KMS key para cifrar los datos en reposo en Amazon S3, el servicio debe tener permiso para utilizarlos AWS KMS key en nombre de la persona que llama. Para obtener más información, consulte AWS Administrado AWS KMS keys y administrado por el cliente AWS KMS keys en la Guía para desarrolladores de Amazon Simple Storage Service.

Parameters

Al crear una AWS IoT regla con esta acción, debe especificar la siguiente información:

`bucket`

El bucket de Amazon S3 en el que se escribirán los datos.

Admite plantillas de sustitución: API y AWS CLI únicamente

`cannedacl`

(Opcional) 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, incluidos los valores permitidos, consulte ACL predefinidas.

Admite plantillas de sustitución: No

`key`

La ruta al archivo en el que se escriben los datos.

Considere un ejemplo en el que se encuentra este parámetro `${topic()}/${timestamp()}` y la regla recibe un mensaje donde se encuentra el tema `some/topic`. Si la marca de tiempo actual es `1460685389`, esta acción escribe los datos en un archivo llamado `1460685389` en la carpeta `some/topic` del bucket de S3

> **Note**
>
> Si usa una clave estática, AWS IoT sobrescribe un solo archivo cada vez que se invoque la regla. Le recomendamos que utilice la marca de tiempo del mensaje u otro identificador de mensaje único para que se guarde un nuevo archivo en Amazon S3 por cada mensaje recibido.

Admite plantillas de sustitución: Sí

`roleArn`

El rol de IAM que permite el acceso al bucket de Amazon S3. Para obtener más información, consulte Requisitos.

Admite plantillas de sustitución: No

Ejemplos

El siguiente ejemplo de JSON define una acción de S3 en una AWS IoT regla.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "s3": {
                    "bucketName": "amzn-s3-demo-bucket",
                    "cannedacl": "public-read",
                    "key": "${topic()}/${timestamp()}",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_s3"
                }
            }
        ]
    }
}
```

Véase también

- ¿Qué es Amazon S3 en la Guía del usuario de Amazon Simple Storage Service

IoT de Salesforce

La acción Salesforce IoT (`salesforce`) envía los datos del mensaje MQTT que activó la regla a un flujo de entrada de Salesforce IoT.

Parameters

Al crear una AWS IoT regla con esta acción, debe especificar la siguiente información:

`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. Para obtener más información, consulte la documentación de Salesforce IoT.

Admite plantillas de sustitución: No

`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 se crea un flujo de entrada. Para obtener más información, consulte la documentación de Salesforce IoT.

Admite plantillas de sustitución: No

Ejemplos

El siguiente ejemplo JSON define una acción de Salesforce IoT en una regla AWS IoT .

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "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"
                }
            }
        ]
    }
}
```

SNS

La acción SNS (`sns`) envía los datos de un mensaje MQTT como una notificación push de Amazon Simple Notification Service (Amazon SNS).

Puede seguir un tutorial que le muestra cómo crear y probar una regla con una acción SNS. Para obtener más información, consulte Tutorial: Envío de una notificación de Amazon SNS.

> **Note**
>
> La acción SNS no es compatible con los temas FIFO (First-In-First-Out) de Amazon SNS. Dado que el motor de reglas es un servicio totalmente distribuido, no se garantiza el orden de los mensajes cuando se invoca la acción SNS.

Requisitos

Esta regla tiene los siguientes requisitos:

- Un rol de IAM que AWS IoT puede asumir para realizar la operación. `sns:Publish` Para obtener más información, consulte Otorgar a una AWS IoT regla el acceso que requiere.

  En la AWS IoT consola, puede elegir o crear un rol que permita AWS IoT realizar esta acción de regla.
- Si utilizas un AWS KMS servicio gestionado por el cliente AWS KMS key para cifrar los datos en reposo en Amazon SNS, el servicio debe tener permiso para utilizarlos AWS KMS key en nombre de la persona que llama. Para obtener más información, consulte Gestión de claves en la Guía para desarrolladores de Amazon Simple Notification Service.

Parameters

Al crear una AWS IoT regla con esta acción, debe especificar la siguiente información:

`targetArn`

El tema de SNS o el dispositivo individual al que se enviará la notificación de inserción.

Admite plantillas de sustitución: API y AWS CLI únicamente

`messageFormat`

(Opcional) El formato del mensaje. Amazon SNS utiliza esta configuración para determinar si la carga debe analizarse y las partes pertinentes de la carga específicas de la plataforma deben extraerse. Valores válidos: `JSON`, `RAW`. El valor predeterminado es `RAW`.

Admite plantillas de sustitución: No

`roleArn`

El rol de IAM que permite obtener acceso a SNS. Para obtener más información, consulte Requisitos.

Admite plantillas de sustitución: No

Ejemplos

El siguiente ejemplo de JSON define una acción de SNS en una AWS IoT regla.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "sns": {
                    "targetArn": "arn:aws:sns:us-east-2:123456789012:my_sns_topic",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_sns"
                }
            }
        ]
    }
}
```

El siguiente ejemplo de JSON define una acción de SNS con plantillas de sustitución en una AWS IoT regla.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "sns": {
                    "targetArn": "arn:aws:sns:us-east-1:123456789012:${topic()}",
                    "messageFormat": "JSON",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_sns"
```

```
                }
            }
        ]
    }
}
```

Véase también

- ¿Qué es Amazon Simple Notification Service? en la Guía para desarrolladores de Amazon Simple Notification Service
- Tutorial: Envío de una notificación de Amazon SNS

SQS

La acción SQS (`sqs`) envía datos de un mensaje MQTT a una cola de Amazon Simple Queue Service (Amazon SQS).

> **Note**
>
> La acción SQS no es compatible con las colas FIFO (First-In-First-Out) de Amazon SQS.. Dado que el motor de reglas es un servicio totalmente distribuido, no se garantiza el orden de los mensajes cuando se activa la acción de SQS.

Requisitos

Esta regla tiene los siguientes requisitos:

- Un rol de IAM que AWS IoT puede asumir para realizar la operación. `sqs:SendMessage` Para obtener más información, consulte Otorgar a una AWS IoT regla el acceso que requiere.

  En la AWS IoT consola, puede elegir o crear un rol que permita AWS IoT realizar esta acción de regla.
- Si utiliza un sistema gestionado por el AWS KMS cliente AWS KMS key para cifrar datos en reposo en Amazon SQS, el servicio debe tener permiso para utilizarlos en nombre de AWS KMS key la persona que llama. Para obtener más información, consulte Gestión de claves en la Guía para desarrolladores de Amazon Simple Queue Service.

Parameters

Al crear una AWS IoT regla con esta acción, debe especificar la siguiente información:

`queueUrl`

La dirección URL de la cola Amazon SQS en la que se escriben los datos. No es necesario que la región de esta URL sea la misma que la de tu Región de AWS AWS IoT regla.

> **Note**
>
> Puede haber cargos adicionales por la transferencia de datos entre Regiones de AWS si se utiliza la acción de la regla de SQS. Para obtener más información, consulte precios de Amazon SQS.

Admite plantillas de sustitución: API y AWS CLI únicamente

`useBase64`

Establezca este parámetro en `true` para configurar la acción de la regla para codificar en base64 los datos del mensaje antes de escribirlos en la cola de Amazon SQS. El valor predeterminado es `false`.

Admite plantillas de sustitución: No

`roleArn`

El rol de &IAM que permite tener acceso a la cola Amazon SQS. Para obtener más información, consulte Requisitos.

Admite plantillas de sustitución: No

Ejemplos

El siguiente ejemplo de JSON define una acción de SQS en una AWS IoT regla.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
```

```
        "actions": [
            {
                "sqs": {
                    "queueUrl": "https://sqs.us-east-2.amazonaws.com/123456789012/
my_sqs_queue",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_sqs"
                }
            }
        ]
    }
}
```

El siguiente ejemplo de JSON define una acción de SQS con plantillas de sustitución en una AWS IoT regla.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "sqs": {
                    "queueUrl": "https://sqs.us-east-2.amazonaws.com/123456789012/
${topic()}",
                    "useBase64": true,
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_sqs"
                }
            }
        ]
    }
}
```

Véase también

- ¿Qué es Amazon Simple Queue Service? en la Guía para desarrolladores de Amazon Simple Queue Service

Step Functions

La acción Step Functions (`stepFunctions`) inicia una máquina de AWS Step Functions estados.

Requisitos

Esta regla tiene los siguientes requisitos:

- Una función de IAM que AWS IoT puede asumir para realizar la `states:StartExecution` operación. Para obtener más información, consulte Otorgar a una AWS IoT regla el acceso que requiere.

  En la AWS IoT consola, puede elegir o crear un rol que permita AWS IoT realizar esta acción de regla.

Parameters

Al crear una AWS IoT regla con esta acción, debe especificar la siguiente información:

`stateMachineName`

El nombre de la máquina de estado de Step Functions a iniciar.

Admite plantillas de sustitución: API y AWS CLI únicamente

`executionNamePrefix`

(Opcional) El nombre dado a la ejecución de la máquina de estado consiste en este prefijo seguido de un UUID. Si no se facilita uno, Step Functions crea automáticamente un nombre exclusivo para cada ejecución de la máquina de estado.

Admite plantillas de sustitución: Sí

`roleArn`

El ARN del rol que concede el AWS IoT permiso para iniciar la máquina de estados. Para obtener más información, consulte Requisitos.

Admite plantillas de sustitución: No

Ejemplos

El siguiente ejemplo de JSON define una acción de Step Functions en una AWS IoT regla.

```
{
```

```
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "stepFunctions": {
                    "stateMachineName": "myStateMachine",
                    "executionNamePrefix": "myExecution",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_step_functions"
                }
            }
        ]
    }
}
```

Véase también

- ¿Qué es AWS Step Functions? en la Guía AWS Step Functions para desarrolladores

Timestream

La acción de regla Timestream escribe atributos (medidas) de un mensaje MQTT en una tabla de Amazon Timestream. Para obtener información completa sobre Amazon Timestream, consulte ¿Qué es Amazon Timestream?

> **Note**
>
> Amazon Timestream no está disponible en todos los s. Región de AWS Si Amazon Timestream no está disponible en su región, no aparecerá en la lista de acciones de reglas.

Los atributos que esta regla almacena en la base de datos de Timestream son los que resultan de la declaración de consulta de la regla. El valor de cada atributo del resultado de la declaración de consulta se analiza para deducir su tipo de datos (como en una acción the section called “Dinamo DBv2”). El valor de cada atributo se escribe en su propio registro de la tabla Timestream. Para especificar o cambiar el tipo de datos de un atributo, utilice la función `cast()` en la sentencia de consulta. Para obtener más información acerca del contenido de cada registro de Timestream, consulte the section called “Contenido de los registros de Timestream”.

> **Note**
>
> Con SQL V2 (2016-03-23), los valores numéricos que son números enteros, como `10.0`, se convierten su representación Entera (`10`). Convertirlos explícitamente en un valor `Decimal`, por ejemplo utilizando la función cast (), no evita este comportamiento: el resultado sigue siendo un valor `Integer`. Esto puede provocar errores de discordancia de tipos que impidan que los datos se registren en la base de datos de Timestream. Para procesar los valores numéricos de números enteros como valores `Decimal`, utilice SQL V1 (2015-10-08) como instrucción de consulta de regla.

> **Note**
>
> El número máximo de valores que una acción de regla Timestream puede escribir en una tabla de Amazon Timestream es 100. Para obtener más información, consulte la Referencia de Cuotas de Amazon Timestream.

Requisitos

Esta regla tiene los siguientes requisitos:

- Una función de IAM que AWS IoT puede asumir para realizar las `timestream:DescribeEndpoints` operaciones y. `timestream:WriteRecords` Para obtener más información, consulte Otorgar a una AWS IoT regla el acceso que requiere.

  En la AWS IoT consola, puede elegir, actualizar o crear un rol que permita AWS IoT realizar esta acción de regla.
- Si utilizas un cliente AWS KMS para cifrar los datos en reposo en Timestream, el servicio debe tener permiso para utilizarlos AWS KMS key en nombre de la persona que llama. Para obtener más información, consulte Cómo AWS utilizan los servicios KMS. AWS

Parameters

Al crear una AWS IoT regla con esta acción, debe especificar la siguiente información:

`databaseName`

Nombre de una base de datos de Amazon Timestream que contiene la tabla para recibir los registros que crea esta acción. Véase también **`tableName`**.

Admite plantillas de sustitución: API y AWS CLI únicamente

`dimensions`

Atributos de metadatos de las series de tiempo que se escriben en cada registro de medida. Por ejemplo, el nombre y la zona de disponibilidad de una instancia EC2 o el nombre del fabricante de un aerogenerador son dimensiones.

`name`

El nombre de la dimensión de metadatos. Es el nombre de la columna en el registro de tabla de la base de datos.

Las dimensiones no se pueden denominar:`measure_name`, `measure_value` o `time`. Estos nombres están reservados. Los nombres de las dimensiones no pueden empezar por `ts_` o ni `measure_value` pueden contener el carácter de dos puntos (`:`).

Admite plantillas de sustitución: No

`value`

El valor que se va a escribir en esta columna del registro de la base de datos.

Admite plantillas de sustitución: Sí

`roleArn`

El nombre de recurso de Amazon (ARN) del rol que concede permiso a AWS IoT para escribir en la tabla de base de datos Timestream. Para obtener más información, consulte Requisitos.

Admite plantillas de sustitución: No

`tableName`

El nombre de la tabla de la base de datos en la que escribir los registros de la medida. Véase también **`databaseName`**.

Admite plantillas de sustitución: API y AWS CLI únicamente

`timestamp`

El valor que se va a utilizar para la marca de tiempo de la entrada. Si está en blanco, se utiliza la hora en que se procesó la entrada.

`unit`

La precisión del valor de marca de tiempo que resulta de la expresión que se describe en `value`.

Valores válidos: `SECONDS` | `MILLISECONDS` | `MICROSECONDS` | `NANOSECONDS` El valor predeterminado es `MILLISECONDS`.

`value`

Una expresión que devuelve un valor de tiempo de época larga.

Puede utilizar la función the section called “time_to_epoch(String, String)” para crear una marca de tiempo válida a partir de un valor de fecha u hora incluido en la carga útil del mensaje.

Contenido de los registros de Timestream

Los datos que esta acción escribe en la tabla Amazon Timestream incluyen una marca de tiempo, los metadatos de la acción de la regla Timestream y el resultado de la declaración de consulta de la regla.

Para cada atributo (medida) del resultado de la declaración de consulta, esta acción de regla escribe un registro en la tabla Timestream especificada con estas columnas.

Nombre de la columna	Tipo de atributo	Valor	Comentarios
`dimension-name`	DIMENSION	El valor especific ado en la entrada de acción de la regla Timestream.	Cada dimensión especificada en la entrada de acción de la regla crea una columna en la base de datos de Timestrea m con el nombre de la dimensión.

Nombre de la columna	Tipo de atributo	Valor	Comentarios
measure_name	MEASURE_NAME	Nombre del atributo	El nombre del atributo en el resultado de la declaración de consulta cuyo valor se especifica en la columna `measure_value::` *`data-type`* .
valor_medida: *`data-type`*	MEASURE_VALUE	El valor del atributo en el resultado de la sentencia de consulta. El nombre del atributo está en la columna `measure_name` .	El valor se interpreta* y se emite como la coincidencia más adecuada de: `bigint`, `boolean`, `double` o `varchar`. Amazon Timestream crea una columna independiente para cada tipo de datos. El valor del mensaje se puede convertir en otro tipo de datos mediante la función `cast()` de la declaración de consulta de la regla.
tiempo	TIMESTAMP	Congela la fecha y la hora de la base de datos.	Este valor lo asigna el motor de reglas o la propiedad `timestamp` , si está definido.

* El valor del atributo leído en la carga útil del mensaje se interpreta de la siguiente manera. Consulte la the section called “Ejemplos” para ver una ilustración de cada uno de estos casos.

- Un valor sin comillas de `true` o `false` se interpreta como un tipo `boolean`.
- Un número decimal se interpreta como un tipo `double`.
- Un valor numérico sin punto decimal se interpreta como un tipo `bigint`.
- Una cadena entre comillas se interpreta como un tipo `varchar`.
- Los objetos y los valores de la matriz se convierten en cadenas JSON y se almacenan como un tipo `varchar`.

Ejemplos

El siguiente ejemplo de JSON define una acción de regla Timestream con una plantilla de sustitución en una regla. AWS IoT

```
{
  "topicRulePayload": {
    "sql": "SELECT * FROM 'iot/topic'",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [
      {
        "timestream": {
          "roleArn": "arn:aws:iam::123456789012:role/aws_iot_timestream",
          "tableName": "devices_metrics",
          "dimensions": [
            {
              "name": "device_id",
              "value": "${clientId()}"
            },
            {
              "name": "device_firmware_sku",
              "value": "My Static Metadata"
            }
          ],
          "databaseName": "record_devices"
        }
      }
    ]
  }
```

```
}
```

Si se utiliza la acción de regla del tema Timestream definida en el ejemplo anterior con la carga útil del mensaje siguiente, se obtienen los registros de Amazon Timestream escritos en la tabla siguiente.

```
{
  "boolean_value": true,
  "integer_value": 123456789012,
  "double_value": 123.456789012,
  "string_value": "String value",
  "boolean_value_as_string": "true",
  "integer_value_as_string": "123456789012",
  "double_value_as_string": "123.456789012",
  "array_of_integers": [23,36,56,72],
  "array of strings": ["red", "green","blue"],
  "complex_value": {
    "simple_element": 42,
    "array_of_integers": [23,36,56,72],
    "array of strings": ["red", "green","blue"]
  }
}
```

En la siguiente tabla se muestran las columnas y los registros de la base de datos que se crean al utilizar la acción de regla temática especificada para procesar la carga útil del mensaje anterior. Las columnas `device_firmware_sku` y `device_id` son las DIMENSIONES definidas en la acción de la regla temática. La acción de regla temática Timestream crea la columna `time` y las columnas `measure_name` y `measure_value::*`, que rellena con los valores del resultado de la declaración de consulta de la acción de regla temática.

device_firmware_sku	device_id	measure_name	measure_value::bigint	measure_value::varchar	measure_value::double	measure_value::boolean	hora
Mis metadatos estáticos	iotconsole-159 -0 EXAMPLE7 8	complex_value	-	{"simple_element":42,"array_of_integers":[23,36,56,72]	-	-	2020-08-26 22:42:16.423000000

device_firmware_sku	device_id	measure_name	measure_value::bigint	measure_value::varchar	measure_value::double	measure_value::boolean	hora
				,"array of strings": ["red","green","blue"]}			
Mis metadatos estáticos	consola iot-159 -0 EXAMPLE78	integer_value_as_string	-	123456789012	-	-	2020-08-26 22:42:16.423000000
Mis metadatos estáticos	consola iot-159 -0 EXAMPLE78	boolean_value	-	-	-	TRUE	2020-08-26 22:42:16.423000000
Mis metadatos estáticos	consola iot-159 -0 EXAMPLE78	integer_value	123456789012	-	-	-	2020-08-26 22:42:16.423000000
Mis metadatos estáticos	consola iot-159 -0 EXAMPLE78	string_value	-	Valor de cadena	-	-	2020-08-26 22:42:16.423000000
Mis metadatos estáticos	consola iot-159 -0 EXAMPLE78	matriz_de_enteros	-	[23,36,56,72]	-	-	2020-08-26 22:42:16.423000000

device_firmware_sku	device_id	measure_name	measure_value::bigint	measure_value::varchar	measure_value::double	measure_value::boolean	hora
Mis metadatos estáticos	consola iot-159 -0 EXAMPLE78	matriz de cadenas	-	["red","green","blue"]	-	-	2020-08-26 22:42:16.423000000
Mis metadatos estáticos	consola iot-159 -0 EXAMPLE78	boolean_value_as_string	-	TRUE	-	-	2020-08-26 22:42:16.423000000
Mis metadatos estáticos	consola iot-159 -0 EXAMPLE78	double_value	-	-	123.456789012	-	2020-08-26 22:42:16.423000000
Mis metadatos estáticos	consola IoT - 159 - 0 EXAMPLE78	double_value_as_string	-	123.45679	-	-	2020-08-26 22:42:16.423000000

Solución de problemas de las reglas

Si tiene algún problema con sus reglas, le recomendamos que active CloudWatch los registros. Puede analizar sus registros para determinar si el problema es de autorización o si, por ejemplo, la condición de una cláusula WHERE no coincidía. Para obtener más información, consulta Cómo configurar CloudWatch los registros.

Acceder a los recursos de varias cuentas mediante reglas AWS IoT

Puede configurar AWS IoT reglas para el acceso entre cuentas para que los datos ingeridos en los temas de MQTT de una cuenta puedan enrutarse a los AWS servicios, como Amazon SQS y

Lambda, de otra cuenta. A continuación, se explica cómo configurar AWS IoT reglas para la ingesta de datos entre cuentas, desde un tema de MQTT de una cuenta hasta un destino de otra cuenta.

Las reglas multicuenta se pueden configurar mediante permisos basados en recursos en el recurso de destino. Por lo tanto, solo los destinos que admiten permisos basados en recursos pueden habilitarse para el acceso entre cuentas mediante reglas. AWS IoT Entre los destinos admitidos se incluyen Amazon SQS, Amazon S3 y AWS Lambda.

> **Note**
>
> Para los destinos admitidos, excepto Amazon SQS, debe definir la regla al Región de AWS igual que el recurso de otro servicio para que la acción de la regla pueda interactuar con ese recurso. Para obtener más información sobre las acciones de las AWS IoT reglas, consulte las acciones de las AWS IoT reglas. Para obtener más información sobre la acción de SQS de una regla, consulte ???.

Requisitos previos

- Familiaridad con las reglas AWS IoT
- Comprensión de los usuarios, roles y permisos de IAM basados en recursos de IAM
- Después de haber instalado AWS CLI

Configuración de varias cuentas para Amazon SQS

Escenario: la cuenta A envía datos de un mensaje MQTT a la cola de Amazon SQS de la cuenta B.

Cuenta de AWS	Cuenta denominada	Description (Descripción)
1111-1111-1111	Cuenta A	Acción de la regla: `sqs:SendMessage`
2222-2222-2222	Cuenta B	Cola de Amazon SQS • ARN: *arn:aws:sqs:region:2222-2222-2222:ExampleQueue*

Cuenta de AWS	Cuenta denominada	Description (Descripción)
		• URL: *https://sqs.region.amazonaws.com/2222-2222-2222/ExampleQueue*

> **Note**
>
> La cola de Amazon SQS de destino no tiene por qué coincidir con la Región de AWS regla.AWS IoT Para obtener más información sobre la acción de SQS de una regla, consulte ???.

Realizar las tareas de la cuenta A

> **Nota**
>
> Para ejecutar los siguientes comandos, el usuario de IAM debe tener permisos para `iot:CreateTopicRule` con el nombre de recurso de Amazon (ARN) de la regla como recurso y permisos para `iam:PassRole` con un recurso como ARN del rol.

1. Configure AWS CLI con el usuario de IAM de la cuenta A.
2. Cree un rol de IAM que confíe en el motor de AWS IoT reglas y adjunte una política que permita el acceso a la cola Amazon SQS de la cuenta B. Consulte ejemplos de comandos y documentos de política en Concesión AWS IoT del acceso requerido.
3. Para crear una regla adjunta a un tema, ejecute el create-topic-rule comando.

   ```
   aws iot create-topic-rule --rule-name myRule --topic-rule-payload file://./my-rule.json
   ```

   A continuación se muestra un archivo de carga útil de ejemplo con una regla que inserta todos los mensajes enviados al tema `iot/test` en la cola de Amazon SQS especificada. La sentencia SQL filtra los mensajes y el rol ARN concede los permisos AWS IoT para añadir el mensaje a la cola de Amazon SQS.

   ```
   {
   ```

```
 "sql": "SELECT * FROM 'iot/test'",
 "ruleDisabled": false,
 "awsIotSqlVersion": "2016-03-23",
 "actions": [
  {
   "sqs": {
    "queueUrl": "https://sqs.region.amazonaws.com/2222-2222-2222/ExampleQueue",
    "roleArn": "arn:aws:iam::1111-1111-1111:role/my-iot-role",
    "useBase64": false
   }
  }
 ]
}
```

Para obtener más información sobre cómo definir una acción de Amazon SQS en una AWS IoT regla, consulte AWS IoT Rule actions: Amazon SQS.

Realizar las tareas de la cuenta B

1. Configure AWS CLI con el usuario de IAM de la cuenta B.
2. Para conceder permisos para el recurso de cola de Amazon SQS a la cuenta A, ejecute el comando add-permission.

```
aws sqs add-permission --queue-url https://sqs.region.amazonaws.com/2222-2222-2222/ExampleQueue --label SendMessagesToMyQueue --aws-account-ids 1111-1111-1111 --actions SendMessage
```

Configuración de varias cuentas para Amazon SNS

Escenario: La cuenta A envía datos de un mensaje MQTT a un tema de Amazon SNS de la cuenta B.

Cuenta de AWS	Cuenta denominada	Description (Descripción)
1111-1111-1111	Cuenta A	Acción de la regla: `sns:Publish`

Cuenta de AWS	Cuenta denominada	Description (Descripción)
2222-2222-2222	Cuenta B	ARN del tema de Amazon SNS: *arn:aws:sns:region:2222-2222-2222:ExampleTopic*

Realizar las tareas de la cuenta A

> **Notas**
>
> Para ejecutar los siguientes comandos, su usuario de IAM debe tener permisos para `iot:CreateTopicRule` con la regla ARN como recurso y permisos para la acción `iam:PassRole` con un recurso como rol ARN.

1. Configure AWS CLI con el usuario de IAM de la cuenta A.
2. Cree un rol de IAM que confíe en el motor de AWS IoT reglas y adjunte una política que permita el acceso al tema Amazon SNS de la cuenta B. Para ver ejemplos de comandos y documentos de políticas, consulte Concesión AWS IoT del acceso requerido.
3. Para crear una regla adjunta a un tema, ejecute el create-topic-rule comando.

   ```
   aws iot create-topic-rule --rule-name myRule --topic-rule-payload file://./my-rule.json
   ```

   A continuación se muestra un archivo de carga útil de ejemplo con una regla que inserta todos los mensajes enviados al tema `iot/test` en el tema de Amazon SNS especificado. La sentencia SQL filtra los mensajes y el rol ARN concede los permisos AWS IoT para enviar el mensaje al tema de Amazon SNS.

   ```
   {
    "sql": "SELECT * FROM 'iot/test'",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [
     {
      "sns": {
       "targetArn": "arn:aws:sns:region:2222-2222-2222:ExampleTopic",
       "roleArn": "arn:aws:iam::1111-1111-1111:role/my-iot-role"
   ```

```
    }
   }
  ]
}
```

Para obtener más información sobre cómo definir una acción de Amazon SNS en una AWS IoT regla, consulte AWS IoT Rule actions: Amazon SNS.

Realizar las tareas de la cuenta B

1. Configure AWS CLI con el usuario de IAM de la cuenta B.
2. Para dar permiso sobre el recurso de temas de Amazon SNS a la cuenta A, ejecute el comando add-permission.

```
aws sns add-permission --topic-arn arn:aws:sns:region:2222-2222-2222:ExampleTopic
 --label Publish-Permission --aws-account-id 1111-1111-1111 --action-name Publish
```

Configuración de varias cuentas para Amazon S3

Escenario: La cuenta A envía datos de un mensaje MQTT a un bucket de Amazon S3 de la cuenta B.

Cuenta de AWS	Cuenta denominada	Description (Descripción)
1111-1111-1111	Cuenta A	Acción de la regla: `s3:PutObject`
2222-2222-2222	Cuenta B	ARN del bucket de Amazon S3: *arn:aws:s3:::amzn-s3-demo-bucket*

Realizar las tareas de la cuenta A

> **Nota**
>
> Para ejecutar los siguientes comandos, su usuario de IAM debe tener permisos para `iot:CreateTopicRule` con la regla ARN como recurso y permisos para la acción `iam:PassRole` con un recurso como rol ARN.

1. Configure AWS CLI con el usuario de IAM de la cuenta A.
2. Cree un rol de IAM que confíe en el motor de AWS IoT reglas y adjunte una política que permita el acceso al bucket de Amazon S3 de la cuenta B. Para ver ejemplos de comandos y documentos de políticas, consulte Otorgar AWS IoT el acceso requerido.
3. Para crear una regla que se adjunte al bucket S3 de destino, ejecuta el create-topic-rule comando.

   ```
   aws iot create-topic-rule --rule-name my-rule --topic-rule-payload file://./my-rule.json
   ```

   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 Amazon S3 especificado. La sentencia SQL filtra los mensajes y el rol ARN concede los permisos AWS IoT para añadir el mensaje al bucket de Amazon S3.

   ```
   {
    "sql": "SELECT * FROM 'iot/test'",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [
     {
      "s3": {
       "bucketName": "amzn-s3-demo-bucket",
       "key": "${topic()}/${timestamp()}",
       "roleArn": "arn:aws:iam::1111-1111-1111:role/my-iot-role"
      }
     }
    ]
   }
   ```

   Para obtener más información sobre cómo definir una acción de Amazon S3 en una AWS IoT regla, consulte AWS IoT Rule actions: Amazon S3.

Realizar las tareas de la cuenta B

1. Configure AWS CLI con el usuario de IAM de la cuenta B.
2. Cree una política de bucket que confíe en la entidad principal de la cuenta A.

El siguiente es un ejemplo de archivo de carga útil que define una política de bucket que confía en la entidad principal de otra cuenta.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AddCannedAcl",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::123456789012:root"
                ]
        },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
        }
    ]
}
```

Para obtener más información, consulte Ejemplos de política de bucket.

3. Para adjuntar la política de bucket al bucket especificado, ejecute el put-bucket-policy comando.

```
aws s3api put-bucket-policy --bucket amzn-s3-demo-bucket --policy file://./amzn-s3-demo-bucket-policy.json
```

4. Para que el acceso entre cuentas funcione, asegúrese de que tiene la configuración correcta de Bloquear todo acceso público. Para obtener más información, consulte Prácticas recomendadas de seguridad para Amazon S3.

Configuración multicuenta para AWS Lambda

Escenario: la cuenta A invoca una AWS Lambda función de la cuenta B y envía un mensaje MQTT.

Cuenta de AWS	Cuenta denominada	Description (Descripción)
1111-1111-1111	Cuenta A	Acción de la regla: `lambda:InvokeFunction`
2222-2222-2222	Cuenta B	ARN de la función de Lambda: *arn:aws:lambda:region:2222-2222-2222:function:example-function*

Realizar las tareas de la cuenta A

> **Notas**
>
> Para ejecutar los siguientes comandos, su usuario de IAM debe tener permisos para `iot:CreateTopicRule` con la regla ARN como recurso y permisos para la acción `iam:PassRole` con un recurso como rol ARN.

1. Configure AWS CLI con el usuario de IAM de la cuenta A.
2. Ejecute el create-topic-rule comando para crear una regla que defina el acceso entre cuentas a la función Lambda de la cuenta B.

   ```
   aws iot create-topic-rule --rule-name my-rule --topic-rule-payload file://./my-rule.json
   ```

   A continuación se muestra un archivo de carga útil de ejemplo con una regla que inserta todos los mensajes enviados al tema `iot/test` en la función de Lambda especificada. La sentencia SQL filtra los mensajes y el ARN de rol concede permiso AWS IoT para pasar los datos a la función de Lambda.

   ```
   {
    "sql": "SELECT * FROM 'iot/test'",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [
     {
      "lambda": {
       "functionArn": "arn:aws:lambda:region:2222-2222-2222:function:example-function"
   ```

```
    }
   }
  ]
}
```

Para obtener más información sobre cómo definir una AWS Lambda acción en una AWS IoT regla, lee Acciones de la AWS IoT regla: Lambda.

Realizar las tareas de la cuenta B

1. Configure AWS CLI con el usuario de IAM de la cuenta B.
2. Ejecute el comando add-permission de Lambda para dar permiso a AWS IoT las reglas para activar la función Lambda. Para ejecutar el siguiente comando, su usuario de IAM debe tener permiso para una acción `lambda:AddPermission`.

```
aws lambda add-permission --function-name example-function --region us-east-1 --
principal iot.amazonaws.com --source-arn arn:aws:iot:region:1111-1111-1111:rule/
example-rule --source-account 1111-1111-1111 --statement-id "unique_id" --action
 "lambda:InvokeFunction"
```

Opciones:

--entidad principal

Este campo da permiso a AWS IoT (representado por`iot.amazonaws.com`) para llamar a la función Lambda.

--source-arn

Este campo confirma que solo `arn:aws:iot:region:1111-1111-1111:rule/example-rule` en AWS IoT activa esta función de Lambda y que ninguna otra regla de la misma cuenta o de otra diferente puede activar esta función de Lambda.

--source-account

Este campo confirma que AWS IoT activa esta función Lambda solo en nombre de la `1111-1111-1111` cuenta.

> **Notas**
>
> Si aparece el mensaje de error “No se ha encontrado la regla” en la consola de la función AWS Lambda , en la sección Configuración, ignore el mensaje de error y proceda a probar la conexión.

Control de errores (acción de error)

Cuando AWS IoT recibe un mensaje de un dispositivo, el motor de reglas comprueba si el mensaje coincide con una regla. Si es así, se evalúa la sentencia de consulta de la regla y se activan las acciones de la regla, pasando el resultado de la sentencia de consulta.

Si se produce un problema al activar una acción, el motor de reglas activa una acción de error, si se ha especificado una para la regla. Esto podría ocurrir cuando:

- Una regla no dispone de permiso para acceder al bucket de Amazon S3.
- Un error de usuario provoca que se supere el rendimiento aprovisionado de DynamoDB.

> **Note**
>
> El tratamiento de errores que se trata en este tema se refiere a las acciones de las reglas. Para depurar problemas de SQL, incluidas las funciones externas, puede configurar el AWS IoT registro. Para obtener más información, consulte ???.

Formato de mensaje de acción de error

Se genera un único mensaje por regla y mensaje. Por ejemplo, si se produce un error en dos acciones de regla en la misma regla, la acción de error recibe un mensaje con los dos errores.

El mensaje de acción de error se parece al siguiente ejemplo.

```
{
  "ruleName": "TestAction",
  "topic": "testme/action",
  "cloudwatchTraceId": "7e146a2c-95b5-6caf-98b9-50e3969734c7",
```

```
  "clientId": "iotconsole-1511213971966-0",
  "base64OriginalPayload":
 "ewogICJtZXNzYWdlIjogIkhlbGxvIHZyb20gQVdTIElvVCBjb25zb2xlIgp9",
  "failures": [
    {
      "failedAction": "S3Action",
      "failedResource": "us-east-1-s3-verify-user",
      "errorMessage": "Failed to put S3 object. The error received was The
 specified bucket does not exist (Service: Amazon S3; Status Code: 404; Error
 Code: NoSuchBucket; Request ID: 9DF5416B9B47B9AF; S3 Extended Request ID:
 yMah1cwPhqTH267QLPhTKeVPKJB8BO5ndBHzOmWtxLTM6uAvwYYuqieAKyb6qRPTxP1tHXCoR4Y=).
 Message arrived on: error/action, Action: s3, Bucket: us-
east-1-s3-verify-user, Key: \"aaa\". Value of x-amz-id-2:
 yMah1cwPhqTH267QLPhTKeVPKJB8BO5ndBHzOmWtxLTM6uAvwYYuqieAKyb6qRPTxP1tHXCoR4Y="
    }
  ]
}
```

ruleName

El nombre de la regla que activó la acción de error.

tema

El tema en el que se recibió el mensaje original.

cloudwatchTraceId

Inicia sesión con una identidad única que hace referencia al error. CloudWatch

clientId

El ID de cliente del publicador de mensajes.

base64 OriginalPayload

La carga del mensaje original codificada en Base64.

failures

failedAction

El nombre de la acción que no se pudo completar (por ejemplo "S3Action").

failedResource

El nombre del recurso (por ejemplo el nombre de un bucket de S3).

errorMessage

La descripción y explicación del error.

Ejemplo de acción de error

A continuación se muestra un ejemplo de una regla con una acción de error añadida. La siguiente regla tiene una acción que escribe datos de mensajes en una tabla de y una acción de error que escribe datos en un bucket de Amazon S3:

```
{
    "sql" : "SELECT * FROM ..."
    "actions" : [{
        "dynamoDB" : {
            "table" : "PoorlyConfiguredTable",
            "hashKeyField" : "AConstantString",
            "hashKeyValue" : "AHashKey"}}
    ],
    "errorAction" : {
        "s3" : {
            "roleArn": "arn:aws:iam::123456789012:role/aws_iot_s3",
            "bucketName" : "message-processing-errors",
            "key" : "${replace(topic(), '/', '-') + '-' + timestamp() + '-' +
 newuuid()}"
        }
    }
}
```

Puede utilizar cualquier función o plantilla de sustitución en la sentencia SQL de una acción de error, incluidas las funciones externas: `aws_lambda()get_dynamodb()`, `get_registry_data()`, `get_thing_shadow()`, `get_secret()machinelearning_predict()`, y `decode()`. Si una acción de error requiere llamar a una función externa, la invocación de la acción de error puede generar una factura adicional por la función externa.

Para obtener más información sobre las reglas y cómo especificar una acción de error, consulte Creación de una AWS IoT regla.

Para obtener más información sobre el uso CloudWatch para supervisar el éxito o el fracaso de las reglas, consulteAWS IoT métricas y dimensiones.

Reducción de los costes de mensajería con Basic Ingest

Puedes usar Basic Ingest para enviar de forma segura los datos del dispositivo a la entidad Servicios de AWS compatibleAWS IoT acciones de reglas, sin incurrir en gastos de mensajería. Basic Ingest optimiza el flujo de datos al eliminar el agente de publish/subscribe mensajes de la ruta de ingestión.

Basic Ingest puede enviar mensajes desde sus dispositivos o aplicaciones. Los mensajes tienen nombres de temas que comienzan por `$aws/rules/rule_name` en sus tres primeros niveles, y donde *`rule_name`* es el nombre de la regla de AWS IoT que quiere invocar.

Puede utilizar una regla existente con Basic Ingest agregando el prefijo de Basic Ingest (`$aws/rules/rule_name`) al tema del mensaje que utilizaría para invocar la regla. Por ejemplo, si tiene una regla llamada `BuildingManager` que se invoca mediante mensajes con temas como `Buildings/Building5/Floor2/Room201/Lights` (`"sql": "SELECT * FROM 'Buildings/#'"`), podrá invocar la misma regla con Basic Ingest enviando un mensaje con el tema `$aws/rules/BuildingManager/Buildings/Building5/Floor2/Room201/Lights`.

> **Note**
>
> - Sus dispositivos y sus reglas no pueden suscribirse a temas reservados para Basic Ingest. Por ejemplo, las AWS IoT Device Defender `num-messages-received` métricas no se emiten porque no permiten suscribirse a temas. Para obtener más información, consulte Temas reservados.
> - Si necesita un publish/subscribe intermediario para distribuir los mensajes a varios suscriptores (por ejemplo, para entregar los mensajes a otros dispositivos y al motor de reglas), debe seguir utilizando el intermediario de AWS IoT mensajes para gestionar la distribución de los mensajes. No obstante, debe publicar sus mensajes en temas que no sean temas de Basic Ingest.

Uso de Basic Ingest

Antes de usar Basic Ingest, compruebe que su dispositivo o aplicación utilice una política que tenga permisos de publicación en `$aws/rules/*`. También puede especificar permisos para reglas individuales que incluyan `$aws/rules/rule_name/*` en la política. De lo contrario, los dispositivos y las aplicaciones podrán seguir utilizando las conexiones existentes con AWS IoT Core.

Cuando el mensaje llega al motor de reglas, no existe diferencia alguna en la implementación o en la gestión de los errores entre reglas invocadas desde Basic Ingest y las invocadas a través suscripciones al agente de mensajes.

Puede crear reglas para usarlas con Basic Ingest. Tenga en cuenta lo siguiente:

- El prefijo inicial de un tema de Basic Ingest (`$aws/rules/rule_name`) no está disponible a la función topic(Decimal).
- Si define una regla que se invoca solo con Basic Ingest, la cláusula `FROM` es opcional en el campo `sql` de la definición `rule`. Seguirá siendo necesaria si la regla también se invoca a través de otros mensajes que deben enviarse por medio del agente de mensajes (por ejemplo, porque esos otros mensajes deban distribuirse a varios suscriptores). Para obtener más información, consulte AWS IoT Referencia de SQL.
- Los primeros tres niveles del tema de Basic Ingest (`$aws/rules/rule_name`) no se cuentan en cuanto al límite de longitud de 8 segmentos o en cuanto al límite de caracteres total de 256 de un tema. De lo contrario, se aplican las mismas restricciones que se documentan en la sección sobre límites de AWS IoT.
- Si se recibe un mensaje con un tema de Basic Ingest que especifica una regla inactiva o una regla que no existe, se crea un registro de errores en un CloudWatch registro de Amazon para ayudarte con la depuración. Para obtener más información, consulte Entradas del registro del motor de reglas. Se muestra una métrica `RuleNotFound`, en la cual podrá crear alarmas. Para obtener más información, consulte Métricas de la regla en Métricas de reglas.
- Seguirá pudiendo publicar con QoS 1 en temas de Basic Ingest. Recibirá PUBACK después de que el mensaje se entregue correctamente al motor de reglas. Recibir PUBACK no significa que las acciones asociadas a la regla se hayan realizado correctamente. Puede configurar una acción de error para gestionar los errores durante el proceso de ejecución. Para obtener más información, consulte Control de errores (acción de error).

AWS IoT Referencia de SQL

En AWS IoT, las reglas se definen mediante una sintaxis similar a la de SQL. Las instrucciones de SQL se componen de tres tipos de cláusulas:

SET

(Opcional) Define variables que se pueden reutilizar en las sentencias SQL y las plantillas de sustitución. Asigne valores a las variables mediante expresiones. Haga referencia a estas variables en las cláusulas SELECT y WHERE y en las plantillas de sustitución de acciones.

La cláusula SET admiteTipos de datos, OperadoresFunciones,Literales,Instrucciones case,Extensiones JSON, Variables yConsultas de objetos anidados.

SELECT

(Obligatorio) Extrae información de la carga de un mensaje de entrada y realiza transformaciones en la información. Los mensajes que se van a utilizar se identifican mediante el filtro de tema especificado en la cláusula FROM.

La cláusula SELECT admite Tipos de datos OperadoresFunciones,Literales,Instrucciones case,Extensiones JSON, Plantillas de sustituciónConsultas de objetos anidados, Variables yCargas binarias.

FROM

El filtro de tema de mensajes MQTT que identifica los mensajes de los que se van a extraer datos. La regla se activa para cada mensaje enviado a un tema de MQTT que coincide con el filtro de temas especificado aquí. Obligatorio para reglas que se activan mediante mensajes que pasan por el agente de mensajes. Opcional para reglas que solo se activan mediante la característica Basic Ingest.

WHERE

(Opcional) Agrega lógica condicional que determina si se llevan a cabo las acciones especificadas por una regla.

La cláusula WHERE admite Tipos de datos OperadoresFunciones,,Literales,Instrucciones case,Extensiones JSON, Variables yConsultas de objetos anidados.

Un ejemplo de instrucción SQL tiene este aspecto:

```
SELECT color AS rgb FROM 'topic/subtopic' 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 `'topic/subtopic'`, la regla se activa y se evalúa la instrucción SQL. La instrucción SQL extrae el valor de la propiedad `color` 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 AWS IoT acciones de reglas.

> **Note**
>
> Los comentarios no se admiten actualmente en la sintaxis de AWS IoT SQL.
> Los nombres de atributos con espacios no se pueden usar como nombres de campo en la instrucción SQL. Si bien la carga entrante puede tener nombres de atributos con espacios, dichos nombres no se pueden usar en la instrucción SQL. Sin embargo, se transferirán a la carga saliente si utiliza una especificación de nombre de campo comodín (*).

Cláusula SELECT

La cláusula AWS IoT SELECT es básicamente la misma que la cláusula SELECT de ANSI SQL, con algunas pequeñas diferencias.

La cláusula SELECT admite Tipos de datosOperadores,Funciones,Literales,Instrucciones case, Extensiones JSONConsultas de objetos anidados, Variables yCargas binarias.

Puede utilizar la cláusula SELECT para extraer información de los mensajes MQTT de entrada. También puede utilizar `SELECT *` para recuperar toda la carga del mensaje de entrada. Por ejemplo:

```
Incoming payload published on topic 'topic/subtopic': {"color":"red", "temperature":50}
```

```
SQL statement: SELECT * FROM 'topic/subtopic'
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 contiene el par clave-valor. Por ejemplo:

```
Incoming payload published on topic 'topic/subtopic': {"color":"red", "temperature":50}
SQL statement: SELECT color FROM 'topic/subtopic'
Outgoing payload: {"color":"red"}
```

Puede utilizar la palabra clave AS para cambiar el nombre de las claves. Por ejemplo:

```
Incoming payload published on topic 'topic/subtopic':{"color":"red", "temperature":50}
SQL:SELECT color AS my_color FROM 'topic/subtopic'
Outgoing payload: {"my_color":"red"}
```

Puede seleccionar varios elementos separándolos con una coma. Por ejemplo:

```
Incoming payload published on topic 'topic/subtopic': {"color":"red", "temperature":50}
SQL: SELECT color as my_color, temperature as fahrenheit FROM 'topic/subtopic'
Outgoing payload: {"my_color":"red","fahrenheit":50}
```

Puede seleccionar varios elementos incluido '*' para agregar elementos a la carga de entrada. Por ejemplo:

```
Incoming payload published on topic 'topic/subtopic': {"color":"red", "temperature":50}
SQL: SELECT *, 15 as speed FROM 'topic/subtopic'
Outgoing payload: {"color":"red", "temperature":50, "speed":15}
```

Puede utilizar la palabra clave "VALUE" para generar cargas de salida que no sean objetos JSON. Con la versión 2015-10-08 de SQL, solo se puede seleccionar un elemento. Con la versión SQL 2016-03-23 o posterior, también puede seleccionar una matriz para generar como objeto de nivel superior.

Example

```
Incoming payload published on topic 'topic/subtopic': {"color":"red", "temperature":50}
SQL: SELECT VALUE color FROM 'topic/subtopic'
Outgoing payload: "red"
```

Puede utilizar la sintaxis `'.'` para explorar objetos JSON anidados en la carga de entrada. Por ejemplo:

```
Incoming payload published on topic 'topic/subtopic': {"color":
{"red":255,"green":0,"blue":0}, "temperature":50}
SQL: SELECT color.red as red_value FROM 'topic/subtopic'
Outgoing payload: {"red_value":255}
```

Para obtener información sobre cómo utilizar nombres de objetos y propiedades JSON que incluyen caracteres reservados, como números o el carácter de guion (signo negativo), consulte Extensiones JSON

Puede utilizar funciones (consulte Funciones) para transformar la carga de entrada. Puede utilizar paréntesis para realizar agrupaciones. Por ejemplo:

```
Incoming payload published on topic 'topic/subtopic': {"color":"red", "temperature":50}
SQL: SELECT (temperature - 32) * 5 / 9 AS celsius, upper(color) as my_color FROM
 'topic/subtopic'
Outgoing payload: {"celsius":10,"my_color":"RED"}
```

Cláusula FROM

La cláusula FROM suscribe a su regla a un tema o un filtro de temas. Incluya el tema o el filtro de temas entre comillas simples ('). La regla se activa para cada mensaje enviado a un tema de MQTT que coincide con el filtro de temas especificado aquí. Mediante un filtro de temas, puede suscribirse a un grupo de temas similares.

Ejemplo:

Carga de entrada publicada en el tema `'topic/subtopic'`: `{temperature: 50}`

Carga de entrada publicada en el tema `'topic/subtopic-2'`: `{temperature: 50}`

SQL: `"SELECT temperature AS t FROM 'topic/subtopic'"`.

La regla se suscribe a `'topic/subtopic'`, por lo que la carga de entrada se pasa a la regla. La carga de salida, que se pasa a las acciones de regla, es: `{t: 50}`. La regla no está suscrita a `'topic/subtopic-2'`, por lo que la regla no se activa para el mensaje publicado en `'topic/subtopic-2'`.

Ejemplo de comodín #:

Puede utilizar el carácter comodín '#' (multinivel) para buscar coincidencias con uno o varios elementos de ruta en particular:

Carga de entrada publicada en el tema `'topic/subtopic'`: `{temperature: 50}`.

Carga de entrada publicada en el tema `'topic/subtopic-2'`: `{temperature: 60}`.

Carga de entrada publicada en el tema `'topic/subtopic-3/details'`: `{temperature: 70}`.

Carga de entrada publicada en el tema `'topic-2/subtopic-x'`: `{temperature: 80}`.

SQL: `"SELECT temperature AS t FROM 'topic/#'"`.

La regla se suscribe a cualquier tema que comience por`'topic'`, por lo que se ejecuta tres veces y envía cargas útiles salientes de `{t: 50}` (para el tema/subtema), (para el tema/subtema 2) y `{t: 60}` (para) a sus acciones. `{t: 70}` topic/subtopic-3/details No está suscrita a `'topic-2/subtopic-x'`, por lo que la regla no se activa para el mensaje `{temperature: 80}`.

Ejemplo de comodín +:

Puede utilizar el carácter comodín '+' (nivel único) para buscar coincidencias con cualquier elemento de ruta en particular:

Carga de entrada publicada en el tema `'topic/subtopic'`: `{temperature: 50}`.

Carga de entrada publicada en el tema `'topic/subtopic-2'`: `{temperature: 60}`.

Carga de entrada publicada en el tema `'topic/subtopic-3/details'`: `{temperature: 70}`.

Carga de entrada publicada en el tema `'topic-2/subtopic-x'`: `{temperature: 80}`.

SQL: `"SELECT temperature AS t FROM 'topic/+'"`.

La regla está suscrita a todos los temas que tengan dos elementos de ruta donde el primer elemento sea `'topic'`. La regla se ejecuta para los mensajes enviados a `'topic/subtopic'` y `'topic/subtopic-2'`, pero no `'topic/subtopic-3/details'` (tiene más niveles que el filtro de tema) o `'topic-2/subtopic-x'` (no empieza por `topic`).

Cláusula SET

Utilice la cláusula SET para definir las variables que almacenan los resultados de las expresiones. Puede reutilizar estas variables en las cláusulas SELECT y WHERE y en las plantillas de sustitución.

Esto le ayuda a evitar la duplicación de expresiones complejas y a reducir el número de llamadas a funciones en su sentencia SQL.

La cláusula SET admiteTipos de datos,Operadores,Funciones,Literales, Instrucciones caseExtensiones JSON, Variables yConsultas de objetos anidados.

Sintaxis de la cláusula SET

La cláusula SET debe aparecer antes que la cláusula SELECT en la sentencia SQL. Utilice la siguiente sintaxis:

```
SET @variable_name = expression [, @variable_name2 = expression2]
```

Reglas de sintaxis:

- Comience los nombres de las variables con @
- Los nombres de las variables pueden contener letras, números y guiones bajos
- Los nombres de las variables pueden tener una longitud máxima de 64 caracteres
- Se pueden establecer varias variables en una sola cláusula SET, separadas por comas
- Cada variable solo se puede asignar una vez (las variables son inmutables)
- La palabra clave SET solo se puede usar una vez por sentencia SQL

Uso de variables

Después de definir las variables, puede usarlas en:

- cláusulas SELECT
- cláusulas WHERE
- Otras asignaciones de variables SET
- Plantillas de sustitución de acciones
- Plantillas de sustitución de acciones de error
- Consultas SELECT anidadas
- Parámetros de función (algunos parámetros, como los parámetros de RoLearn y los parámetros que cambian el modo de una función de forma similar, `transform("enrichArray", attributes, values)` no admiten variables)

Se hace referencia a las variables con la misma `@variable_name` sintaxis utilizada en la cláusula SET. También puede usar la sintaxis de la extensión JSON para acceder a las propiedades de las variables que contienen objetos, como`@variable_name.property`.

Ejemplos de cláusulas SET

Uso básico de variables

El siguiente ejemplo muestra una carga útil publicada sobre el tema: `device/data` `{"temp_fahrenheit": 75, "humidity": 60}`

Instrucción SQL:

```
SET @temp_celsius = (temp_fahrenheit - 32) * 5 / 9
SELECT @temp_celsius AS celsius, humidity FROM 'device/data'
```

Carga útil saliente: `{"celsius": 23.89, "humidity": 60}`

Accede a los miembros en objetos JSON incrustados

En el siguiente ejemplo, se muestra una carga útil publicada sobre un tema: `device/data` `{"device1": {"deviceId":"weather_sensor", "deviceData": {"sensors": {"temp_fahrenheit": 75, "humidity": 60}, "location": [47.606,-122.332]}}}`

Instrucción SQL:

```
SET @device_sensor_data = device1.deviceData.sensors
SELECT @device_sensor_data.temp_fahrenheit AS temp_fahrenheit,
 @device_sensor_data.humidity as humidity, device1.deviceId as deviceId FROM 'device/
data'
```

Carga útil saliente: `{"temp_fahrenheit":75,"humidity":60,"deviceId":"weather_sensor"}`

para obtener más información sobre cómo trabajar con extensiones JSON, consulte Extensiones JSON

Evitar la duplicación de llamadas a funciones

Las variables SET ayudan a evitar la duplicación de operaciones de decodificación complejas:

```
SET @decoded_data = decode(encode(*, 'base64'), 'proto', 'schema', 'schema.desc',
 'message.proto', 'Message')
```

```
SELECT @decoded_data.sensor_id, @decoded_data.reading FROM 'device/protobuf'
WHERE @decoded_data.reading > 100
```

Sin las variables SET, tendría que repetir la función de decodificación tres veces, lo que supera los límites de llamadas a la función.

Variables múltiples

Puede definir varias variables en una sola cláusula SET separándolas con comas:

```
SET @user_data = get_user_properties(device_id), @threshold = 50
SELECT @user_data.name, temp_fahrenheit FROM 'sensors/+'
WHERE temp_fahrenheit > @threshold AND @user_data.active = true
```

Uso de variables en plantillas de sustitución

Las variables también se pueden utilizar en las plantillas de sustitución de acciones, lo que permite reutilizar los valores calculados tanto en las acciones de la sentencia SQL como en las de las reglas.

Instrucción SQL:

```
SET @temp_celsius = (temp_fahrenheit - 32) * 5 / 9
SELECT @temp_celsius AS celsius, humidity FROM 'device/data'
```

Configuración de acciones:

```
{
  "s3": {
    "roleArn": "arn:aws:iam::123456789012:role/testRuleRole",
    "bucketName": "bucket",
    "key": "temperature-data/${device_id}/temp-${@temp_celsius}C.json"
  }
}
```

En este ejemplo, la variable SET `@temp_celsius` se utiliza en una plantilla de sustitución para construir el campo clave de la acción de S3.

Uso de cargas que no son de JSON

Las variables SET no admiten directamente cargas útiles que no sean de JSON, por lo que la carga útil debe codificarse o decodificarse primero:

```
SET @encoded_payload = encode(*, 'base64')
SELECT @encoded_payload AS raw_data FROM 'device/binary'
```

para obtener más información sobre cómo trabajar con cargas útiles que no son de JSON, consulte Uso de las cargas binarias

LÍMITES DE CLÁUSULAS SET

Los siguientes límites se aplican a las variables SET:

- Máximo de 10 variables únicas por sentencia SQL
- Tamaño máximo de valor variable de 128 KiB (cadena JSON UTF-8 minificada)
- Tamaño máximo del valor total de 128 KiB para todas las variables
- Los nombres de las variables están limitados a 64 caracteres
- Las variables pueden aceptar cargas útiles de JSON directamente tal cual (las cargas que no son de JSON deben codificarse o decodificarse primero)

Cláusula WHERE

La cláusula WHERE determina si se llevan a cabo las acciones especificadas por una regla. Si la cláusula WHERE se evalúa en true, se llevan a cabo las acciones de la regla. De lo contrario, las acciones de la regla no se llevan a cabo.

La cláusula WHERE admite Tipos de datos OperadoresFunciones,,Literales,Instrucciones case,Extensiones JSON, Variables yConsultas de objetos anidados.

Ejemplo:

Carga de entrada publicada en `topic/subtopic`: `{"color":"red", "temperature":40}`.

SQL: `SELECT color AS my_color FROM 'topic/subtopic' WHERE temperature > 50 AND color <> 'red'`.

En este caso, la regla se activará, pero las acciones especificadas por la regla no se llevarán a cabo. No habrá carga de salida.

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.

Ejemplo con una carga que no es JSON:

Carga entrante no JSON publicada en el `tema/subtema`: `80`

SQL: `SELECT decode(encode(*, 'base64'), 'base64') AS value FROM 'topic/subtopic' WHERE decode(encode(*, 'base64'), 'base64') > 50`

En este caso, la regla se activará y las acciones especificadas por la regla se llevarán a cabo. La cláusula SELECT transformará la carga saliente en una carga `{"value":80}` JSON.

Tipos de datos

El motor de AWS IoT reglas admite todos los tipos de datos JSON.

Tipos de datos compatibles

Tipo	Significado
`Int`	Un discreto `Int`. 34 dígitos como máximo.
`Decimal`	Un valor `Decimal` con una precisión de 34 dígitos, con una magnitud no nula mínima de 1E-999 y una magnitud máxima de 9.999... E999. **Note** Algunas funciones devuelven valores `Decimal` con doble precisión (en lugar de una precisión de 34 dígitos). Con SQL V2 (2016-03-23), los valores numéricos que son números enteros, como `10.0`, se procesan como un valor `Int` (`10`) en lugar del valor `Decimal` esperado (`10.0`). Para procesar de forma fiable los valores numéricos de números enteros como valores `Decimal`, utilice SQL V1 (2015-10-08) como instrucción de consulta de regla.

Tipo	Significado
`Boolean`	`True` o bien `False`.
`String`	Una cadena UTF-8.
`Array`	Una serie de valores que no han de tener obligatoriamente el mismo tipo.
`Object`	Un valor JSON compuesto de una clave y un valor. Las claves deben ser cadenas. Los valores pueden ser de cualquier tipo.
`Null`	Valor `Null`, tal como lo define JSON. Es un valor real que representa la ausencia de valor. El usuario puede crear explícitamente un valor `Null` especificando la palabra clave `Null` en la instrucción SQL. Por ejemplo: `"SELECT NULL AS n FROM 'topic/subtopic'"`

Tipo	Significado

`Undefined`	No es un valor. No se representa explícita mente en JSON, salvo que se omita el valor. Por ejemplo, en el objeto `{"foo": null}`, la clave "foo" devuelve NULL, pero la clave "bar" devuelve `Undefined` . Internamente, el lenguaje SQL trata a `Undefined` como un valor, pero este no se puede representar en JSON, por lo que cuando se serializa en JSON, los resultados son `Undefined` .
	`{"foo":null, "bar":undefined}`
	se serializa en JSON como:
	`{"foo":null}`
	Del mismo modo, `Undefined` 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.) devuelven `Undefined` .

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
`Int`	Valor `Decimal` sin separador decimal.
`Decimal`	El valor de origen.
`Boolean`	`Undefined` . (Puede utilizar de forma explícita la función cast para transformar true = 1.0, false = 0.0.)
`String`	El motor SQL intenta analizar la cadena como un`Decimal`. AWS IoT intenta analizar las cadenas que coinciden con la expresión regular:`^-?\d+(\.\d+)?((?i)E-?\d+)?$` . "0", "-1.2", "5E-12" son ejemplos de cadenas que se convierten automáticamente en valores de tipo `Decimal`.
Matriz	`Undefined` .
Objeto	`Undefined` .
Nulo	`Null`.
Sin definir	`Undefined` .

En valor entero

Tipo de argumento	Resultado
`Int`	El valor de origen.
`Decimal`	El valor de origen redondeado al valor `Int` más cercano.

Tipo de argumento	Resultado
`Boolean`	`Undefined` . (Puede utilizar de forma explícita la función cast para transformar true = 1.0, false = 0.0.)
`String`	El motor SQL intenta analizar la cadena como un`Decimal`. AWS IoT intenta analizar las cadenas que coinciden con la expresión regular:`^-?\d+(\.\d+)?((?i)E-?\d +)?$` . «0", «-1.2", «5E-12" son ejemplos de cadenas que se convierten automáticamente en `Decimal` s. AWS IoT Intenta convertir el en a y`Decimal`, `String` a continuación, trunca sus decimales para formar un. `Decimal` `Int`
Matriz	`Undefined` .
Objeto	`Undefined` .
Nulo	`Null`.
Sin definir	`Undefined` .

En valor booleano

Tipo de argumento	Resultado
`Int`	`Undefined` . (Puede utilizar de forma explícita la función `cast` para transformar 0 = False, any_nonzero_value = True.)
`Decimal`	`Undefined` . (Puede utilizar de forma explícita la función cast para transformar 0 = False, any_nonzero_value = True.)
`Boolean`	El valor original.

Tipo de argumento	Resultado
`String`	"true"= True y "false" = False (no distingue entre mayúsculas y minúsculas). Otros valores de string son `Undefined` .
Matriz	`Undefined` .
Objeto	`Undefined` .
Nulo	`Undefined` .
Sin definir	`Undefined` .

En cadena

Tipo de argumento	Resultado
`Int`	Una representación de cadena del valor `Int` en notación estándar.
`Decimal`	Una cadena que representa el valor `Decimal`, posiblemente en notación científica.
`Boolean`	"true" o "false". Todos en minúscula.
`String`	El valor original.
Matriz	El valor `Array` serializado en formato JSON. La cadena obtenida es una lista separada por comas, entre corchetes. Los valores de tipo `String` se indican entre comillas. No así los valores de tipo `Decimal`, `Int`, `Boolean` y `Null`.
Objeto	El objeto serializado al formato JSON. La cadena obtenida es una lista separada por comas de pares clave-valor que comienza y termina con llaves. Los valores de tipo `String`

Tipo de argumento	Resultado
	se indican entre comillas. No así los valores de tipo `Decimal`, `Int`, `Boolean` y `Null`.
Nulo	`Undefined` .
Sin definir	Sin definir.

Operadores

Los operadores siguientes se pueden utilizar en las cláusulas SELECT 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	Output
`Boolean`	`Boolean`	`Boolean`. True si ambos operandos son true. De lo contrario, devuelve 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 *`boolean`* `AND` *`boolean`*.
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	Output
`Boolean`	`Boolean`	`Boolean`. True si uno de los operandos es true. De lo contrario, devuelve false.
`String/Boolean`	`String/Boolean`	Si todas las cadenas son "true" o "false" (no se distingue entre mayúsculas y minúsculas), se convierten en valores booleanos y se procesan normalmente como *`boolean`* `OR` *`boolean`*.
Otro valor	Otro valor	`Undefined` .

Operador NOT

Devuelve un resultado `Boolean`. Realiza una operación NOT lógica. Devuelve true si el operado es false. De lo contrario, devuelve true. Se necesita un operando `Boolean` o un operando de cadena "true" o "false" que no distinga entre mayúsculas y minúsculas.

Sintaxis: `NOT` *`expression`*.

Operador NOT

Operando	Output
`Boolean`	`Boolean`. True si el operando es false. De lo contrario, devuelve true.

Operando	Output
`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	`Undefined` .

Operador IN

Devuelve un resultado `Boolean`. Puede usar el operador IN en una cláusula WHERE para comprobar si un valor coincide con algún valor de una matriz. Devuelve true si hay coincidencia y false en caso contrario.

Sintaxis: *`expression`* `IN` *`expression`*.

Operador IN

Operando izquierdo	Operando derecho	Output
`Int/Decimal/String/A`	`Array`	Verdadero si el elemento `Integer/Decimal/String/Array/Object` está en la raíz. De lo contrario, devuelve false.

Ejemplo:

```
SQL: "select * from 'a/b' where 3 in arr"

JSON: {"arr":[1, 2, 3, "three", 5.7, null]}
```

En este ejemplo, la cláusula de condición `where 3 in arr` se evaluará como verdadera porque el número 3 está presente en la matriz denominada `arr`. Por lo tanto, en la instrucción SQL, se ejecutará `select * from 'a/b'`. Este ejemplo también muestra que la matriz puede ser heterogénea.

Operador EXISTS

Devuelve un resultado `Boolean`. Puede usar el operador EXISTS en una cláusula condicional para comprobar la existencia de elementos en una subconsulta. Devuelve verdadero si la subconsulta devuelve uno o más elementos y falso si no devuelve ningún elemento.

Sintaxis: *`expression`*.

Ejemplo:

```
SQL: "select * from 'a/b' where exists (select * from arr as a where a = 3)"

JSON: {"arr":[1, 2, 3]}
```

En este ejemplo, la cláusula de condición `where exists (select * from arr as a where a = 3)` se evaluará como verdadera porque el número 3 está presente en la matriz denominada `arr`. Por lo tanto, en la instrucción SQL, se ejecutará `select * from 'a/b'`.

Ejemplo:

```
SQL: select * from 'a/b' where exists (select * from e as e where foo = 2)

JSON: {"foo":4,"bar":5,"e":[{"foo":1},{"foo":2}]}
```

En este ejemplo, la cláusula de condición `where exists (select * from e as e where foo = 2)` se evaluará como verdadera porque la matriz e del objeto JSON contiene el objeto `{"foo":2}`. Por lo tanto, en la instrucción SQL, se ejecutará `select * from 'a/b'`.

> 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	Output
`Int/Decimal`	`Int/Decimal`	`Boolean`. Devuelve el valor true si el operando izquierdo es superior al operando derecho. De lo contrario, devuelve false.
`String/Int/Deci`	`String/Int/Deci`	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 false.
Otro valor	`Undefined .`	`Undefined .`

>= 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	Output
`Int/Decimal`	`Int/Decimal`	`Boolean`. Devuelve el valor true si el operando izquierdo es superior o igual al operando derecho. De lo contrario, devuelve false.
`String/Int/Deci`	`String/Int/Deci`	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 false.

Operando izquierdo	Operando derecho	Output
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`*.

Operador <

Operando izquierdo	Operando derecho	Output
`Int/Decimal`	`Int/Decimal`	`Boolean`. Devuelve el valor true si el operando izquierdo es inferior al operando derecho. De lo contrario, devuelve false.
`String/Int/Deci`	`String/Int/Deci`	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 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	Output
`Int/Decimal`	`Int/Decimal`	`Boolean`. Devuelve el valor true si el operando izquierdo es inferior o igual al operando derecho. De lo contrario, devuelve false.
`String/Int/Deci`	`String/Int/Deci`	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 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	Output
`Int`	`Int`	True si el operando izquierdo no es igual al operando derecho. De lo contrario, devuelve false.
`Decimal`	`Decimal`	True si el operando izquierdo no es igual al operando derecho. De lo contrario, devuelve 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, devuelve false.

Operando izquierdo	Operando derecho	Output
Matriz	Matriz	True si los elementos de cada operando no son iguales y no están en el mismo orden. De lo contrario, devuelve false.
Objeto	Objeto	True si las claves y los valores de cada operando no son iguales. De lo contrario, devuelve false. El orden de no keys/values es importante.
Nulo	Nulo	False.
Cualquier valor	`Undefined`	Sin definir.
`Undefined`	Cualquier valor	Sin definir.
Tipo no coincidente	Tipo no coincidente	True.

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	Output
`Int`	`Int`	True si el operando izquierdo es igual al operando derecho. De lo contrario, devuelve false.
`Decimal`	`Decimal`	True si el operando izquierdo es igual al operando derecho. De lo contrario, devuelve false. `Int` se convierte en un valor `Decimal` antes de la comparación.

Operando izquierdo	Operando derecho	Output
`String`	`String`	True si el operando izquierdo es igual al operando derecho. De lo contrario, devuelve false.
Matriz	Matriz	True si los elementos de cada operando son iguales y están en el mismo orden. De lo contrario, devuelve false.
Objeto	Objeto	True si las claves y los valores de cada operando son iguales. De lo contrario, devuelve false. El orden de no keys/values es importante.
Cualquier valor	`Undefined`	`Undefined` .
`Undefined`	Cualquier valor	`Undefined` .
Tipo no coincidente	Tipo no coincidente	False.

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	Output
`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.

Operando izquierdo	Operando derecho	Output
`Int`	`Int`	valor de `Int`. Agrega ambos operandos.
`Int/Decimal`	`Int/Decimal`	valor de `Decimal`. Agrega ambos operandos.
Otro valor	Otro valor	`Undefined` .

Operador -

Resta el operando derecho del operando izquierdo.

Sintaxis: *`expression`* `-` *`expression`*.

Operador -

Operando izquierdo	Operando derecho	Output
`Int`	`Int`	valor de `Int`. Resta el operando derecho del operando izquierdo.
`Int/Decimal`	`Int/Decimal`	valor de `Decimal`. Resta el operando derecho del operando izquierdo.
`String/Int/Deci`	`String/Int/Deci`	Si todas las cadenas se convierten en decimales 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	Output
`Int`	`Int`	valor de `Int`. Multiplica el operando izquierdo por el operando derecho.
`Int/Decimal`	`Int/Decimal`	valor de `Decimal`. Multiplica el operando izquierdo por el operando derecho.
`String/Int/Deci`	`String/Int/Deci`	Si todas las cadenas se convierten en decimales correctamente, se devuelve un valor `Decimal`. Multiplic a 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	Output
`Int`	`Int`	valor de `Int`. Divide el operando izquierdo por el operando derecho.
`Int/Decimal`	`Int/Decimal`	valor de `Decimal`. Divide el operando izquierdo por el operando derecho.
`String/Int/Deci`	`String/Int/Deci`	Si todas las cadenas se convierten en decimales correctamente, se devuelve un valor `Decimal`. Divide el operando izquierdo por el operando derecho. De lo contrario, devuelve `Undefined` .

Operando izquierdo	Operando derecho	Output
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	Output
`Int`	`Int`	valor de `Int`. Devuelve el resto de la división del operando izquierdo por el operando derecho.
`String/Int/Deci`	`String/Int/Deci`	Si todas las cadenas se convierten en decimales 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.

Las siguientes funciones externas se facturan de forma equivalente a la de una acción de regla: `aws_lambda`, `get_dynamodb()get_registry_data()`, y. `get_thing_shadow()` Además, solo se le facturará por la función `decode()` cuando descodifique un mensaje de Protobuf en JSON. Para obtener más detalles, consulte la página de precios de AWS IoT Core.

abs(Decimal)

Devuelve el valor absoluto de un número. Es compatible con la versión 2015-10-08 de SQL y versiones posteriores.

Ejemplo: `abs(-5)` devuelve 5.

Tipo de argumento	Resultado
`Int`	`Int`, el valor absoluto del argumento.
`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` .
Nulo	`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 2015-10-08 de SQL y versiones posteriores.

Ejemplo:

`accountid()` = "123456789012"

acos(Decimal)

Devuelve el coseno inverso de un número en radianes. Los argumentos `Decimal` se redondean con doble precisión antes de la aplicación de la función. Es compatible con la versión 2015-10-08 de SQL 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` .
Nulo	`Undefined` .
Sin definir	`Undefined` .

asin(Decimal)

Devuelve el seno inverso de un número en radianes. Los argumentos `Decimal` se redondean con doble precisión antes de la aplicación de la función. Es compatible con la versión 2015-10-08 de SQL y versiones posteriores.

Ejemplo: `asin(0)` = 0.0

Tipo de argumento	Resultado
`Int`	`Decimal` (con doble precisión), el seno inverso del argumento. Se devuelven resultados imaginarios como `Undefined` .
`Decimal`	`Decimal` (con doble precisión), el seno inverso del argumento. Se devuelven resultados imaginarios como `Undefined` .
`Boolean`	`Undefined` .
`String`	`Decimal` (con doble precisión), el seno inverso del argumento. Si la cadena no se puede convertir, el resultado es `Undefined` . Se devuelven resultados imaginarios como `Undefined` .
Matriz	`Undefined` .
Objeto	`Undefined` .
Nulo	`Undefined` .
Sin definir	`Undefined` .

atan(Decimal)

Devuelve la tangente inversa de un número en radianes. Los argumentos `Decimal` se redondean con doble precisión antes de la aplicación de la función. Es compatible con la versión 2015-10-08 de SQL y versiones posteriores.

Ejemplo: `atan(0)` = 0.0

Tipo de argumento	Resultado
`Int`	`Decimal` (con doble precisión), la tangente inversa del argumento. Se devuelven resultados imaginarios como `Undefined` .
`Decimal`	`Decimal` (con doble precisión), la tangente inversa del argumento. Se devuelven resultados imaginarios como `Undefined` .
`Boolean`	`Undefined` .
`String`	`Decimal`, la tangente inversa del argumento. Si la cadena no se puede convertir, el resultado es `Undefined` . Se devuelven resultados imaginarios como `Undefined` .
Matriz	`Undefined` .
Objeto	`Undefined` .
Nulo	`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 `Decimal` se redondean con doble precisión antes de aplicar la función. Es compatible con la versión 2015-10-08 de SQL y versiones posteriores.

Ejemplo: `atan2(1, 0)` = 1.5707963267948966

Tipo de argumento	Tipo de argumento	Resultado
`Int/Decimal`	`Int/Decimal`	`Decimal` (con doble precisión), entre el eje x y el punto (x, y) es
`Int/Decimal/String`	`Int/Decimal/String`	`Decimal`, la tangente inversa d descrito. Si una cadena no se p convertir, el resultado es `Undef`
Otro valor	Otro valor	`Undefined` .

aws_lambda(functionArn, inputJson)

Llama a la función de Lambda especificada que pasa `inputJson` a la función de Lambda y devuelve el JSON generado por la función de Lambda.

Argumentos

Argumento	Description (Descripción)
`functionArn`	El ARN de la función de Lambda; a la que se llamará. La función de Lambda debe devolver datos JSON.
`inputJson`	La entrada de JSON trasladada a la función de Lambda. Para pasar literales y consultas de objetos anidados, debe usar la versión 2016-03-23 de SQL.

Debe conceder AWS IoT `lambda:InvokeFunction` permisos para invocar la función Lambda especificada. En el siguiente ejemplo, se muestra cómo se puede conceder el permiso `lambda:InvokeFunction` utilizando AWS CLI:

```
aws lambda add-permission --function-name "function_name"
--region "region"
--principal iot.amazonaws.com
--source-arn arn:aws:iot:us-east-1: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 de Lambda. Agrega un nuevo permiso para actualizar la política de recursos de la función.

--region

El Región de AWS de su cuenta.

-- entidad principal

La entidad principal que obtiene el permiso. Esto debería ser `iot.amazonaws.com` para permitir el AWS IoT permiso de llamar a una función Lambda.

--source-arn

El ARN de la regla. Puede usar el get-topic-rule AWS CLI comando para obtener el ARN de una regla.

--source-account

El Cuenta de AWS lugar donde se define la regla.

--statement-id

Un identificador de instrucción único.

--action

La acción Lambda que desea permitir en esta instrucción. Para permitir que AWS IoT invoque una función de Lambda, especifique `lambda:InvokeFunction`.

> **Important**
>
> Si añade un permiso para un AWS IoT principal sin proporcionar el `source-arn` o`source-account`, cualquier permiso Cuenta de AWS que cree una regla con su acción de Lambda puede activar reglas desde las que invocar la función de Lambda. AWS IoT Para obtener más información, consulte Lambda Permission Model.

Dada una carga de mensaje JSON como:

```
{
```

```
    "attribute1": 21,
    "attribute2": "value"
}
```

La función `aws_lambda` se puede utilizar para llamar a la función de Lambda de la siguiente manera:

```
SELECT
aws_lambda("arn:aws:lambda:us-east-1:account_id:function:lambda_function",
 {"payload":attribute1}) as output FROM 'topic-filter'
```

Si desea pasar la carga del mensaje MQTT completa, puede especificar la carga JSON mediante '*', como en el ejemplo siguiente.

```
SELECT
aws_lambda("arn:aws:lambda:us-east-1:account_id:function:lambda_function", *) as output
 FROM 'topic-filter'
```

`payload.inner.element` selecciona datos de mensajes publicados en el tema 'tema/subtema'.

`some.value` selecciona datos de la salida generada por la función de Lambda.

> **Note**
>
> El motor de reglas limita la duración de la ejecución de las funciones de Lambda. Las llamadas a funciones de Lambda desde las reglas deben completarse en 2000 milisegundos.

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 2015-10-08 de SQL y versiones posteriores.

Ejemplo: `bitand(13, 5)` = 5

Tipo de argumento	Tipo de argumento	Resultado
`Int`	`Int`	`Int`, una operación AND bit a b argumentos.

Tipo de argumento	Tipo de argumento	Resultado
`Int/Decimal`	`Int/Decimal`	`Int`, una operación AND bit a b dos argumentos. Todos los núm no son de tipo `Int` se redondea `Int` inferior más cercano. Si alg argumentos no se puede conve valor `Int`, el resultado es `Unde`
`Int/Decimal/String`	`Int/Decimal/String`	`Int`, una operación AND bit a b dos argumentos. Todas las cad convierten en decimales y se re al valor `Int` inferior más cercan produce un error en la conversi resultado obtenido es `Undefin`
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 2015-10-08 de SQL y versiones posteriores.

Ejemplo: `bitor(8, 5)` = 13

Tipo de argumento	Tipo de argumento	Resultado
`Int`	`Int`	`Int`, la operación OR bit a bit d argumentos.
`Int/Decimal`	`Int/Decimal`	`Int`, la operación OR bit a bit d argumentos. Todos los número son de tipo `Int` se redondean a inferior más cercano. Si se prod en la conversión, el resultado ob `Undefined` .

Tipo de argumento	Tipo de argumento	Resultado
`Int/Decimal/String`	`Int/Decimal/String`	`Int`, la operación OR bit a bit en dos argumentos. Todas las cad convierten en decimales y se re al valor `Int` inferior más cercan produce un error en la conversi resultado obtenido es `Undefin`
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 2015-10-08 de SQL y versiones posteriores.

Ejemplo:`bitor(13, 5)` = 8

Tipo de argumento	Tipo de argumento	Resultado
`Int`	`Int`	`Int`, una operación XOR bit a b argumentos.
`Int/Decimal`	`Int/Decimal`	`Int`, una operación XOR bit a b dos argumentos. Los números d `Int` se redondean al valor `Int` cercano.
`Int/Decimal/String`	`Int/Decimal/String`	`Int`, una operación XOR bit a b dos argumentos. Las cadenas s n en decimales y se redondean `Int` más cercano. Si se produc en la conversión, el resultado ob `Undefined` .
Otro valor	Otro valor	`Undefined` .

bitnot(Int)

Ejecuta una operación NOT bit a bit en las representaciones de bits del argumento `Int` (convertido). Es compatible con la versión 2015-10-08 de SQL 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. Las cadenas se convierten en decimales 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 booleanos o viceversa. Si AWS IoT no puede determinar cómo convertir un tipo en otro, el resultado es. `Undefined` Es compatible con la versión 2015-10-08 de SQL y versiones posteriores. Formato: fundido (*`value`*as*`type`*).

Ejemplo:

```
cast(true as Int) = 1
```

Las siguientes palabras clave pueden aparecer después de "as" cuando se llama a `cast`:

Para las versiones 2015-10-08 y 2016-03-23 de SQL

Palabra clave	Resultado
`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`.
Entero	Convierte un valor en `Int`.
Double	Convierte un valor en `Decimal` (con double precision).

Además, para SQL versión 2016-03-23

Palabra clave	Resultado
`Decimal`	Convierte un valor en `Decimal`.
Bool	Convierte un valor en `Boolean`.
`Boolean`	Convierte un valor en `Boolean`.

Reglas de conversión:

Conversión en decimal

Tipo de argumento	Resultado
`Int`	Valor `Decimal` sin separador decimal.
`Decimal`	El valor de origen.

Tipo de argumento	Resultado
	Note Con SQL V2 (2016-03-23), los valores numéricos que son números enteros, como `10.0`, devuelven un valor `Int` (`10`) en lugar del valor `Decimal` esperado (`10.0`). Para convertir de forma fiable los valores numéricos de números enteros como valores `Decimal`, utilice SQL V1 (2015-10-08) como instrucción de consulta de regla.
`Boolean`	true = 1.0, false = 0.0.
`String`	Intenta analizar la cadena como un valor `Decimal`. AWS IoT intenta analizar las cadenas que coincidan con la expresión regular: ^-?\d+(\.\d+)?((?i)E-?\d+)?$. "0", "-1.2", "5E-12" son ejemplos de cadenas que se convierten automáticamente en valores de tipo Decimal.
Matriz	`Undefined` .
Objeto	`Undefined` .
Nulo	`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`	Intenta analizar la cadena como un valor `Decimal`. AWS IoT intenta analizar las cadenas que coincidan con la expresión regular: ^-?\d+(\.\d+)?((?i)E-?\d+)?$. "0", "-1.2", "5E-12" son ejemplos de cadenas que se convierten automáticamente en valores de tipo Decimal. AWS IoT inten ta convertir la cadena en un valor de tipo `Decimal` y redondearlo al valor de tipo `Int` inferior más cercano.
Matriz	`Undefined` .
Objeto	`Undefined` .
Nulo	`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 = `Undefined` .
Matriz	`Undefined` .
Objeto	`Undefined` .
Nulo	`Undefined` .
Sin definir	`Undefined` .

Conversión en cadenas

Tipo de argumento	Resultado
`Int`	Una representación de cadena del valor `Int`, en notación estándar.
`Decimal`	Una cadena que representa el valor `Decimal`, posiblemente en notación científica.
`Boolean`	"true" o "false", todo en minúsculas.
`String`	El valor de origen.
Matriz	La matriz serializada en formato JSON. La cadena obtenida es una lista separada por comas, entre corchetes. Los valores `String` se indican entre comillas. Los valores `Decimal`, `Int` y `Boolean` no se indican entre comillas.
Objeto	El objeto serializado al formato JSON. La cadena JSON es una lista separada por comas de pares clave-valor que comienza

Tipo de argumento	Resultado
	y termina con llaves. Los valores `String` se indican entre comillas. Los valores `Decimal`, `Int`, `Boolean` y `Null` no se indican entre comillas.
Nulo	`Undefined` .
Sin definir	`Undefined` .

ceil(Decimal)

Redondea el valor `Decimal` indicado al valor `Int` superior más cercano. Es compatible con la versión 2015-10-08 de SQL y versiones posteriores.

Ejemplos:

`ceil(1.2)` = 2

`ceil(-1.2)` = -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 2015-10-08 de SQL 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` .
Nulo	`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 2015-10-08 de SQL 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 2015-10-08 de SQL 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	`Undefined` .
1	El argumento se devuelve sin modificar.
2+	Si alguno de los argumentos es un valor `Array`, el resultado es una matriz única

Número de argumentos	Resultado
	que contiene todos los argumentos. Si no hay argumentos de tipo Array y al menos un argumento es un valor `String`, el resultado es la concatenación de las representaciones de `String` de todos los argumentos. Los argumentos se convierte n en cadenas mediante las conversiones estándar indicadas arriba.

cos(Decimal)

Devuelve el coseno de un número en radianes. Los argumentos `Decimal` se redondean con doble precisión antes de la aplicación de la función. Es compatible con la versión 2015-10-08 de SQL y versiones posteriores.

Ejemplo:

`cos(0)` = 1.

Tipo de argumento	Resultado
`Int`	`Decimal` (con doble precisión), el coseno del argumento. Se devuelven resultados imaginarios como `Undefined` .
`Decimal`	`Decimal` (con doble precisión), el coseno del argumento. Se devuelven resultados imaginarios como `Undefined` .
`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` .

Tipo de argumento	Resultado
Matriz	`Undefined` .
Objeto	`Undefined` .
Nulo	`Undefined` .
Sin definir	`Undefined` .

cosh(Decimal)

Devuelve el coseno hiperbólico de un número en radianes. Los argumentos `Decimal` se redondean con doble precisión antes de la aplicación de la función. Es compatible con la versión 2015-10-08 de SQL 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` .

Tipo de argumento	Resultado
Nulo	`Undefined` .
Sin definir	`Undefined` .

decode(value, decodingScheme)

Utilice la función `decode` para descodificar un valor codificado. Si la cadena descodificada es un documento JSON, se devuelve un objeto direccionable. De lo contrario, la cadena descodificada se devuelve como una cadena. La función devuelve NULL si la cadena no se puede descodificar. Esta función admite la descodificación de cadenas codificadas en base64 y el formato de mensaje de búferes de protocolo (protobuf).

Es compatible con la versión SQL 2016-03-23 y versiones posteriores.

valor

Un valor de cadena o cualquiera de las expresiones válidas, tal como se define en AWS IoT Referencia de SQL, que devuelven una cadena.

decodingScheme

Una cadena literal que representa el esquema utilizado para descodificar el valor. Actualmente solo se admiten `'base64'` y `'proto'`.

Descodificar cadenas codificadas en base64

En este ejemplo, la carga del mensaje incluye un valor codificado.

```
{
    encoded_temp: "eyAidGVtcGVyYXR1cmUiOiAzMyB9Cg=="
}
```

La función `decode` de esta instrucción SQL descodifica el valor de la carga del mensaje.

```
SELECT decode(encoded_temp,"base64").temperature AS temp from 'topic/subtopic'
```

Al descodificar el valor `encoded_temp`, se obtiene el siguiente documento JSON válido, que permite que la instrucción SELECT lea el valor de temperatura.

```
{ "temperature": 33 }
```

El resultado de la instrucción SELECT de este ejemplo se muestra aquí.

```
{ "temp": 33 }
```

Si el valor descodificado no era un documento JSON válido, el valor descodificado se devolvería en forma de cadena.

Descodificación de la carga de un mensaje de protobuf

Puede utilizar la función de decodificación de SQL para configurar una regla que pueda descodificar la carga de sus mensajes de protobuf. Para obtener más información, consulte Descodificar cargas de mensajes de protobuf.

> **Important**
>
> Si omites `source-arn` o `source-account` al configurar los permisos para un AWS IoT director, cualquiera Cuenta de AWS puede invocar tu función de decodificación mediante otras reglas. AWS IoT Para proteger la función, consulte Políticas de bucket en la Guía del usuario de Amazon Simple Storage Service.

La firma de la clase tiene el siguiente aspecto:

```
decode(<ENCODED DATA>, 'proto', '<S3 BUCKET NAME>', '<S3 OBJECT KEY>', '<PROTO NAME>',
 '<MESSAGE TYPE>')
```

`ENCODED DATA`

Especifica los datos codificados por protobuf que se van a descodificar. Si todo el mensaje enviado a la regla son datos codificados por protobuf, puede hacer referencia a la carga binaria entrante sin procesar utilizando *. De lo contrario, este campo debe ser una cadena JSON codificada en base64 y se puede transferir directamente una referencia a la cadena.

1) Para descodificar una carga binaria entrante de protobuf sin procesar:

```
decode(*, 'proto', ...)
```

2) Para descodificar un mensaje codificado en protobuf representado por una cadena codificada en base64 'a.b':

```
decode(a.b, 'proto', ...)
```

`proto`

Especifica los datos que se van a descodificar en un formato de mensaje protobuf. Si especifica `base64` en lugar de `proto`, esta función descodificará las cadenas codificadas en base64 como JSON.

`S3 BUCKET NAME`

El nombre del bucket de Amazon S3 donde cargó el archivo `FileDescriptorSet`.

`S3 OBJECT KEY`

La clave de objeto que especifica el archivo `FileDescriptorSet` dentro del bucket de Amazon S3.

`PROTO NAME`

El nombre del archivo `.proto` (excluida la extensión) a partir del cual se generó el archivo `FileDescriptorSet`.

`MESSAGE TYPE`

El nombre de la estructura de mensajes protobuf del archivo `FileDescriptorSet`, a la que deben ajustarse los datos que se van a descodificar.

Un ejemplo de expresión SQL que utilice la función de descodificación de SQL puede tener el siguiente aspecto:

```
SELECT VALUE decode(*, 'proto', 's3-bucket', 'messageformat.desc', 'myproto',
 'messagetype') FROM 'some/topic'
```

- `*`

  Representa una carga binaria entrante, que se ajusta al tipo de mensaje protobuf llamado `mymessagetype`.

- `messageformat.desc`

El archivo `FileDescriptorSet` almacenado en un bucket de Amazon S3 llamado `s3-bucket`.

- `myproto`

  El archivo `.proto` original utilizado para generar el archivo `FileDescriptorSet` llamado `myproto.proto`.

- `messagetype`

  El tipo de mensaje llamado `messagetype` (junto con cualquier dependencia importada) tal y como se define en `myproto.proto`.

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.

valor

Cualquiera de las expresiones válidas, tal y como se define en AWS IoT Referencia de SQL. 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 convierte 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 `Boolean` 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 2015-10-08 de SQL y versiones posteriores.

Ejemplo: `endswith("cat","at")` = true.

Tipo de argumento 1	Tipo de argumento 2	Resultado
`String`	`String`	True si el primer argumento ter el segundo argumento. De lo co devuelve false.
Otro valor	Otro valor	Ambos argumentos se convierte cadenas con las reglas de conv estándar. True si el primer argu termina en el segundo argumen contrario, devuelve false. Si alg argumentos es `Null` o `Undefi` resultado es `Undefined` .

exp(Decimal)

Devuelve e elevado al argumento `Decimal`. Los argumentos `Decimal` se redondean con doble precisión antes de la aplicación de la función. Es compatible con la versión 2015-10-08 de SQL 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` .

floor(Decimal)

Redondea a la baja el valor `Decimal` indicado al valor `Int` más cercano. Es compatible con la versión 2015-10-08 de SQL y versiones posteriores.

Ejemplos:

`floor(1.2)` = 1

`floor(-1.2)` = -2

Tipo de argumento	Resultado
`Int`	`Int`, el valor del argumento.
`Decimal`	`Int`, valor `Decimal` redondeado a la baja al valor `Int` más próximo.
`String`	`Int`. La cadena se convierte en un valor `Decimal` y se redondea a la baja al valor `Int` más cercano. Si la cadena no se puede convertir en un valor `Decimal`, el resultado es `Undefined` .
Otro valor	`Undefined` .

introducción

Extrae un valor de un tipo de recopilación (matriz, cadena, objeto). No se aplica 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 2015-10-08 de SQL y versiones posteriores.

Ejemplos:

`get(["a", "b", "c"], 1)` = "b"

`get({"a":"b"}, "a")` = "b"

`get("abc", 0)` = “a”

Tipo de argumento 1	Tipo de argumento 2	Resultado
Matriz	Cualquier tipo (convertido en `Int`)	El elemento en el índice basado del valor `Array` proporcionado segundo argumento (convertido Si la conversión es incorrecta, e obtenido es `Undefined` . Si el está fuera de los límites del valo (negativo o >= array.length), el r `Undefined` .
Cadena	Cualquier tipo (convertido en `Int`)	El carácter en el índice basado la cadena proporcionada por el argumento (convertido en un va Si la conversión es incorrecta, e obtenido es `Undefined` . Si el encuentra fuera de los límites d (negativo o >= string.length), el `Undefined` .
Objeto	`String` (no se aplica conversión)	El valor almacenado en el objet argumento correspondiente a la la cadena proporcionada como argumento.
Otro valor	Cualquier valor	`Undefined` .

get_dynamodb (Nombre de tabla,,,,, ROLearn partitionKeyName) partitionKeyValue sortKeyName sortKeyValue

Recupera datos de una tabla DymanoDB. `get_dynamodb()` permite consultar una tabla DynamoDB mientras se evalúa una regla. Puede filtrar o aumentar las cargas útiles de mensajes utilizando los datos recuperados de DynamoDB. Es compatible con la versión SQL 2016-03-23 y versiones posteriores.

`get_dynamodb()` utiliza los parámetros siguientes:

tableName

Nombre de la tabla de DynamoDB donde efectuar la consulta.

partitionKeyName

Nombre de la tabla de particiones. Para obtener más información, consulte Claves de DynamoDB.

partitionKeyValue

Valor de la clave de partición utilizada para identificar un registro. Para obtener más información, consulte Claves de DynamoDB.

sortKeyName

(Opcional) Nombre de la clave de clasificación. Este parámetro solo es necesario si la tabla DynamoDB consultada utiliza una clave compuesta. Para obtener más información, consulte Claves de DynamoDB.

sortKeyValue

(Opcional) Valor de la clave de clasificación. Este parámetro solo es necesario si la tabla DynamoDB consultada utiliza una clave compuesta. Para obtener más información, consulte Claves de DynamoDB.

roleArn

El ARN de un rol de IAM que concede acceso a la tabla DynamoDB. El motor de reglas asume este rol para acceder a la tabla DynamoDB en su nombre. Evite usar un rol excesivamente permisivo. Otorgue al rol solo los permisos requeridos por la regla. A continuación se muestra un ejemplo de política que concede acceso a una tabla DynamoDB.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "dynamodb:GetItem",
            "Resource": "arn:aws:dynamodb:us-east-1:123456789012:table/table-
name"
        }
    ]
```

```
}
```

Como ejemplo de cómo usar `get_dynamodb()`, supongamos que tiene una tabla DynamoDB que contiene el ID de dispositivo e información de ubicación para todos los dispositivos conectados a AWS IoT. La siguiente instrucción SELECT utiliza la función `get_dynamodb()` para recuperar la ubicación del ID de dispositivo especificado:

```
SELECT *, get_dynamodb("InServiceDevices", "deviceId", id,
"arn:aws:iam::12345678910:role/getdynamo").location AS location FROM 'some/
topic'
```

> **Note**
>
> - Puede llamar a `get_dynamodb()` un máximo de una vez por instrucción SQL. Llamar a `get_dynamodb()` varias veces en una sola instrucción SQL hace que la regla termine sin invocar ninguna acción.

get_mqtt_property(name)

Hace referencia a cualquiera de los siguientes MQTT5 encabezados:,, y. `contentType` `payLoadFormatIndicator` `responseTopic` `correlationData` Esta función toma como argumento cualquiera de las siguientes cadenas literales: `content_type`, `format_indicator`, `response_topic` y `correlation_data`. Para obtener más información, consulte la siguiente table de argumentos de la función.

contentType

Cadena: una cadena codificada en UTF-8 que describe el contenido del mensaje de publicación.

payLoadFormatIndicador

Cadena: un valor de cadena Enum que indica si la carga tiene formato UTF-8. Los valores válidos son `UNSPECIFIED_BYTES` y `UTF8_DATA`.

responseTopic

Cadena: una cadena codificada en UTF-8 que se utiliza como nombre de tema para un mensaje de respuesta. El tema de respuesta se utiliza para describir el tema en el que el receptor debe

publicar como parte del flujo de solicitud-respuesta. El tema no debe contener caracteres comodín.

correlationData

Cadena: los datos binarios codificados en base64 que utiliza el remitente del mensaje de solicitud para identificar a qué solicitud corresponde el mensaje de respuesta cuando se recibe.

En la siguiente tabla se muestran los argumentos de la función aceptables y los tipos de retorno asociados a la función `get_mqtt_property`:

Argumentos de la función

SQL	Tipo de datos devuelto (si están presentes)	Tipo de datos devuelto (si no están presentes)
`get_mqtt_property( "format_indicator")`	Cadena (UNSPECIF IED_BYTES o _DATA) UTF8	Cadena (UNSPECIF IED_BYTES)
`get_mqtt_property( "content_type")`	Cadena	Sin definir
`get_mqtt_property( "response_topic")`	Cadena	Sin definir
`get_mqtt_property( "correlation_data")`	Cadena codificada en base64	Sin definir
`get_mqtt_property( "some_invalid_name ")`	Sin definir	Sin definir

El siguiente ejemplo de Rules SQL hace referencia a cualquiera de los siguientes MQTT5 encabezados:,, y. `contentType` `payLoadFormatIndicator` `responseTopic` `correlationData`

```
SELECT *, get_mqtt_property('content_type') as contentType,
          get_mqtt_property('format_indicator') as payloadFormatIndicator,
          get_mqtt_property('response_topic') as responseTopic,
          get_mqtt_property('correlation_data') as correlationData
```

```
FROM 'some/topic'
```

get_or_default (expresión, DefaultValue)

Devuelve el valor predeterminado del segundo parámetro si se especifica o, si no, devuelve un valor indefinido cuando la expresión del primer parámetro devuelve un valor nulo, indefinido o produce un error. Es compatible con la versión SQL 2016-03-23 y versiones posteriores.

> **Important**
>
> `get_or_default`no admite cargas útiles que no sean de JSON directamente tal cual. Si utilizas una carga útil que no es de JSON, usa las funciones o. `encode` `decode`

`get_or_default()` utiliza los parámetros siguientes:

expresión

Cualquier expresión válida que contengaTipos de datos, Funciones LiteralesConsultas de objetos anidados, Variables o. Extensiones JSON

defaultValue

(Opcional) Cualquier expresión válida que contenga Tipos de datos FuncionesLiterales,Consultas de objetos anidados, Variables oExtensiones JSON. Este es el valor que se devolverá siempre que el primer argumento devuelva un valor nulo, indefinido o falle.

> **Note**
>
> Las funciones que obtienen datos de los recursos propiedad del cliente, como get_secret, get_dynamodb, aws_lambda, get_thing_shadow, decode-protobuf y machinelearning_predict, no están permitidas para el parámetro DefaultValue.

En la siguiente tabla se muestran los argumentos de función aceptables para cada argumento y sus resultados asociados:

Primer argumento	Segundo argumento	Output
Evaluación exitosa	Cualquier valor o no especificado	El valor del primer argumento.

Primer argumento	Segundo argumento	Output
Indefinido, nulo o erróneo	Cualquier valor, incluidos indefinido o nulo	El valor del segundo argumento
Indefinido, nulo o erróneo	no especificada	`Undefined`

Ejemplos:

Ejemplo 1:

El siguiente ejemplo proporciona un valor DefaultValue si se produce un error en una tabla o consulta de DynamoDB:

```
SELECT
    device_id,
    get_or_default(
        get_dynamodb("DeviceConfig", "deviceId", nonExistentId,
 "arn:aws:iam::123456789012:role/ROLE_NAME"),
        {"mode": "standard", "timeout": 30, "enabled": true }
    ) as config
FROM 'device/telemetry'
```

Ejemplo 2:

El siguiente ejemplo proporciona el valor predeterminado seguro «UNKNOWN» si el estado no está definido:

```
SELECT
  get_or_default( CASE status
    WHEN 'active' THEN 'GOOD'
    WHEN 'inactive' THEN 'BAD'/
    ELSE 'UNKNOWN'
  END, 'UNKNOWN') as status_category
FROM 'topic/subtopic'
```

Ejemplo 3:

En el siguiente ejemplo, se muestra cómo se puede utilizar también get_or_default con un único parámetro. Esto resulta útil en situaciones en las que es posible que no tenga un valor predeterminado claro, pero no desee que se produzca un error en la ejecución de la regla.

```
SELECT
  get_dynamodb("DeviceConfig", "deviceId", nonExistentId,
 "arn:aws:iam::123456789012:role/ROLE_NAME") as config
FROM 'device/telemetry'
```

Si se produce un error en la búsqueda de DynamoDB, se producirá un error en la ejecución de la regla y no se ejecutará ninguna acción. Si en su lugar se utiliza el siguiente SQL:

```
SELECT
  get_or_default(get_dynamodb("DeviceConfig", "deviceId", nonExistentId,
 "arn:aws:iam::123456789012:role/ROLE_NAME")) as config
FROM 'device/telemetry'
```

La sentencia get_or_default se evaluará como`Undefined`, por lo que en este ejemplo, la sentencia SELECT en general se evaluará como resultado `{}` y se intentará realizar cualquier acción de regla.

> **Important**
>
> Recomendamos seguir estas prácticas recomendadas para mantener la seguridad al utilizar esta función:
>
> - Evite el uso de secretos codificados en las definiciones de reglas, incluidos los valores predeterminados
> - Úselo AWS Secrets Manager para administrar información confidencial

get_registry_data (API de registro, ThingName, ROLearn)

Recupera los datos AWS IoT de registro de cosas de una AWS IoT regla. Puede leer los datos del registro (como los atributos, el tipo de cosa y los grupos de cosas a los que pertenece un dispositivo) y usar esta información para filtrar, enriquecer o enrutar los mensajes de forma dinámica. Es compatible con la versión SQL 2016-03-23 y versiones posteriores.

`get_registry_data()` utiliza los parámetros siguientes:

API de registro

La API de registro a la que se está llamando. Los valores válidos son `DescribeThing` y `ListThingGroupsForThing`. Estos valores deben ser cadenas constantes.

thingName

Cadena: el nombre de la cosa cuyos datos de registro desea recuperar.

roleArn

Cadena: un ARN de rol con `iot:DescribeThing` permiso de and/or `iot:ListThingGroupsForThing` permiso basado en la API a la que se llama.

El formato de respuesta de la `get_registry_data` función es el mismo que el de la API de registro a la que se ha llamado. Para obtener más información, consulte DescribeThing y ListThingGroupsForThing APIs.

Ejemplo:

Puede recuperar la información del tipo de cosa para poder filtrar los mensajes de eventos AWS IoT Core del ciclo de vida de las cosas (cuyo nombre coincida con el identificador del cliente de MQTT) donde esté `testenv` el tipo de cosa.

```
SELECT *
FROM '$aws/events/lifecycle/+'
WHERE
    get_registry_data("DescribeThing",clientId,[roleArn]).thingTypeName='testenv'
```

Ejemplo:

Puede recuperar los atributos de un dispositivo con el nombre de la cosa `sensor1` para todos los mensajes enviados por su dispositivo `gateway1` de puerta de enlace.

```
SELECT *, get_registry_data("DescribeThing","sensor1",
[roleArn]).attributes.temperature_threhold AS device1_tempthreshold
FROM home1/gateway1/sensor1/#
```

> **Note**
>
> - Puede llamar `get_registry_data()` como máximo una vez por sentencia SQL y plantillas de sustitución para las acciones y las acciones de error.

get_secret(secretId, secretType, key, roleArn)

Recupera el valor del campo `SecretString` o `SecretBinary` cifrado de la versión actual de un secreto en AWS Secrets Manager. Para obtener más información sobre la creación y el mantenimiento de secretos CreateSecret, consulte UpdateSecret, y PutSecretValue.

`get_secret()` utiliza los parámetros siguientes:

secretId

Cadena: el nombre de recurso de Amazon (ARN) o el nombre fácil de recordar del secreto que se va a recuperar.

secretType

Cadena: el tipo de secreto. Valores válidos: `SecretString` | `SecretBinary`.

SecretString

- Para ver los secretos que se crean como objetos JSON mediante la APIs AWS CLI, la o la AWS Secrets Manager consola:
  - Si especifica un valor para el parámetro `key`, esta función devuelve el valor de la clave especificada.
  - Si no especifica un valor para el parámetro `key`, esta función devuelve el objeto JSON completo.
- Para los secretos que se crean como objetos que no son de JSON mediante APIs o: AWS CLI
  - Si especifica un valor para el parámetro `key`, esta función produce una excepción.
  - Si no especifica un valor para el parámetro `key`, esta función devuelve el contenido del secreto.

SecretBinary

- Si especifica un valor para el parámetro `key`, esta función produce una excepción.

- Si no especificas un valor para el parámetro `key`, esta función devuelve el valor secreto como una cadena UTF-8 codificada en base64.

clave

(Opcional) Cadena: el nombre de la clave dentro de un objeto JSON almacenado en el campo `SecretString` de un secreto. Use este valor cuando desee recuperar solo el valor de una clave almacenada en un secreto en lugar de recuperar todo el objeto JSON.

Si especifica un valor para este parámetro y el secreto no contiene un objeto JSON dentro de su campo `SecretString`, esta función produce una excepción.

roleArn

Cadena: un ARN de rol con permisos `secretsmanager:GetSecretValue` y `secretsmanager:DescribeSecret`.

> **Note**
>
> Esta función siempre devuelve la versión actual del secreto (la versión con la etiqueta `AWSCURRENT`). El motor de AWS IoT reglas guarda en caché cada secreto durante un máximo de 15 minutos. Como resultado, el motor de reglas puede tardar hasta 15 minutos en actualizar un secreto. Esto significa que si recuperas un secreto hasta 15 minutos después de una actualización AWS Secrets Manager, es posible que esta función devuelva la versión anterior.
> Esta función no está sujeta a taxímetro, pero se aplican AWS Secrets Manager cargos. Debido al mecanismo de almacenamiento en caché de secretos, el motor de reglas llama en ocasiones a AWS Secrets Manager. Como el motor de reglas es un servicio totalmente distribuido, es posible que vea varias llamadas a la API de Secrets Manager desde el motor de reglas durante el período de almacenamiento en caché de 15 minutos.

Ejemplos:

Puede usar la función `get_secret` en un encabezado de autenticación en una acción de regla HTTPS, como en el siguiente ejemplo de autenticación con clave de API.

```
"API_KEY": "${get_secret('API_KEY', 'SecretString', 'API_KEY_VALUE',
 'arn:aws:iam::12345678910:role/getsecret')}"
```

Para obtener más información sobre la acción de regla HTTPS, consulte the section called “HTTP”.

get_thing_shadow (ThingName, ShadowName, ROLearn)

Devuelve la sombra especificada de la cosa especificado. Es compatible con la versión SQL 2016-03-23 y versiones posteriores.

thingName

String: el nombre del elemento cuya sombra desea recuperar.

shadowName

(Opcional) Cadena: el nombre de la sombra. Este parámetro solo es necesario cuando se hace referencia a sombras con nombre.

roleArn

String: un ARN de rol con permiso `iot:GetThingShadow`.

Ejemplos:

Cuando se utilice con una sombra con nombre, proporcione el parámetro `shadowName`.

```
SELECT * from 'topic/subtopic'
WHERE
    get_thing_shadow("MyThing","MyThingShadow","arn:aws:iam::123456789012:role/
AllowsThingShadowAccess")
    .state.reported.alarm = 'ON'
```

Cuando se utiliza con una sombra sin nombre, omita el parámetro `shadowName`.

```
SELECT * from 'topic/subtopic'
WHERE
    get_thing_shadow("MyThing","arn:aws:iam::123456789012:role/
AllowsThingShadowAccess")
    .state.reported.alarm = 'ON'
```

get_user_properties () userPropertyKey

Hace referencia a las propiedades del usuario, que es un tipo de encabezados de propiedades que se admiten. MQTT5

userProperty

Cadena: una propiedad de usuario es un par clave-valor. Esta función toma la clave como argumento y devuelve una matriz de todos los valores que coinciden con la clave asociada.

Argumentos de la función

Para las siguientes propiedades de usuario en los encabezados de mensaje:

Clave	Valor
alguna clave	algún valor
una clave diferente	un valor diferente
alguna clave	valor con clave duplicada

La siguiente table muestra el comportamiento de SQL esperado:

SQL	Tipo de datos devueltos	Valor de datos devueltos
get_user_properties('some key')	Matriz de cadena	`['some value', 'value with duplicate key']`
get_user_properties('other key')	Matriz de cadena	`['a different value']`
get_user_properties( )	Matriz de objetos de par clave-valor	`[{'"some key": "some value"'}, {"other key": "a different value"}, {"some key": "value with duplicate key"}]`

SQL	Tipo de datos devueltos	Valor de datos devueltos
get_user_properties('non-existent key')	Sin definir	

El siguiente ejemplo de reglas SQL hace referencia a las propiedades del usuario (un tipo de encabezado de MQTT5 propiedad) en la carga útil:

```
SELECT *, get_user_properties('user defined property key') as userProperty
FROM 'some/topic'
```

Funciones de hash

AWS IoT proporciona 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 la versión 2015-10-08 de SQL 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 2016-03-23 y versiones posteriores.

Ejemplos:

`indexof("abcd", "bc")` = 1

isNull()

Devuelve verdadero si el argumento es el valor `Null`. Es compatible con la versión SQL 2016-03-23 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()

Devuelve verdadero si el argumento tiene el valor `Undefined`. Es compatible con la versión SQL 2016-03-23 y versiones posteriores.

Ejemplos:

`isUndefined(5)` = false.

`isUndefined(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 2016-03-23 y versiones posteriores.

Ejemplos:

`length("hi")` = 2

`length(false)` = 5

ln(Decimal)

Devuelve el logaritmo natural del argumento. Los argumentos `Decimal` se redondean con doble precisión antes de la aplicación de la función. Es compatible con la versión 2015-10-08 de SQL 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.
`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` .
Nulo	`Undefined` .
Sin definir	`Undefined` .

log(Decimal)

Devuelve el logaritmo decimal del argumento. Los argumentos `Decimal` se redondean con doble precisión antes de la aplicación de la función. Es compatible con la versión 2015-10-08 de SQL 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` .
Nulo	`Undefined` .
Sin definir	`Undefined` .

lower(String)

Muestra la versión en minúsculas del valor `String` indicado. Los argumentos que no son cadenas se convierten en cadenas con las reglas de conversión estándar. Es compatible con la versión 2015-10-08 de SQL 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 establece en el valor válido más cercano (0 o 1000). Es compatible con la versión 2015-10-08 de SQL y versiones posteriores.

Ejemplos:

`lpad("hello", 2) = "  hello"`.

`lpad(1, 3) = "   1"`

Tipo de argumento 1	Tipo de argumento 2	Resultado
`String`	`Int`	`String`, el valor `String` propo rellenado en el lado izquierdo co número de espacios igual al val proporcionado.
`String`	`Decimal`	El argumento `Decimal` se redo valor `Int` inferior más cercano y `String` se rellena en el lado izq el número de espacios especific
`String`	`String`	El segundo argumento se convi un valor `Decimal`, que se redon valor `Int` inferior más cercano, `String` se rellena con el núme espacios especificado en la izqu Si el segundo argumento no se convertir en un valor `Int`, el res `Undefined` .
Otro valor	`Int`/`Decimal`/`String`	El primer valor se convierte en u de tipo `String` mediante las co nes estándar y, a continuación, la función LPAD a dicho valor S Si no se puede convertir, el resu `Undefined` .
Cualquier valor	Otro valor	`Undefined` .

ltrim(String)

Elimina todos los espacios en blanco del principio (tabuladores y espacios) del valor `String` proporcionado. Es compatible con la versión 2015-10-08 de SQL y versiones posteriores.

Ejemplo:

`Ltrim(" h i ")` = "hi ".

Tipo de argumento	Resultado
`Int`	La representación `String` de `Int` con todos los espacios en blanco del principio suprimidos.
`Decimal`	La representación `String` de `Decimal` con todos los espacios en blanco del principio suprimidos.
`Boolean`	La representación `String` del valor booleano (“true” o “false”) con todos los espacios en blanco del principio suprimido s.
`String`	El argumento con todos los espacios en blanco del principio suprimidos.
Matriz	La representación `String` de `Array` (mediante las reglas de conversión estándar) con todos los espacios en blanco del principio suprimidos.
Objeto	La representación `String` de la cosa (mediante las reglas de conversión estándar) con todos los espacios en blanco del principio suprimidos.
Nulo	`Undefined` .
Sin definir	`Undefined` .

machinelearning_predict(modelId, roleArn, record)

Utilice la `machinelearning_predict` función para realizar predicciones con los datos de un mensaje MQTT basado en un modelo de Amazon SageMaker AI. Es compatible con la versión 2015-10-08 de SQL 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 conexión 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 se van a transferir a la API SageMaker AI Predict. Debe representarse como un objeto JSON de capa única. Si el registro es un objeto JSON de varias capas, el registro se aplana 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.

Detalles

Contiene los atributos siguientes:

PredictiveModelType

El tipo de modelo. Los valores válidos son REGRESSION, BINARY, MULTICLASS.

Algoritmo

El algoritmo utilizado por la SageMaker IA para hacer predicciones. El valor debe ser SGD.

predictedScores

Contiene la puntuación de clasificación bruta correspondiente a cada etiqueta.

predictedValue

El valor pronosticado por la SageMaker IA.

mod(Decimal, Decimal)

Devuelve el resto de la división del primer argumento por el segundo argumento. Es igual que remainder(Decimal, Decimal). También puede utilizar "%" como un operador infijo para la misma funcionalidad modulo. Es compatible con la versión 2015-10-08 de SQL y versiones posteriores.

Ejemplo: `mod(8, 3)` = 2.

Operando izquierdo	Operando derecho	Output
`Int`	`Int`	`Int`, el primer y el segundo arg los que quiere ejecutar la funció
`Int/Decimal`	`Int/Decimal`	`Decimal`, el primer argumento segundo operando para los que ejecutar la función modulo.
`String/Int/Decimal`	`String/Int/Decimal`	Si todas las cadenas se convier decimales, la función modulo se el primer y el segundo argumen contrario, `Undefined` .
Otro valor	Otro valor	`Undefined` .

nanval (,) AnyValue AnyValue

Devuelve el primer argumento si es un `Decimal` válido. De lo contrario, se devuelve el segundo argumento. Es compatible con la versión 2015-10-08 de SQL y versiones posteriores.

Ejemplo: `Nanvl(8, 3)` = 8.

Tipo de argumento 1	Tipo de argumento 2	Output
Sin definir	Cualquier valor	El segundo argumento.
Nulo	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 2015-10-08 de SQL 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 2016-03-23 y versiones posteriores.

Ejemplos:

`numbytes("hi")` = 2

`numbytes("€")` = 3

parse_time(String, Long[, String])

Use la `parse_time` función para formatear una marca de tiempo en un formato legible por humanos. date/time Es compatible con la versión SQL 2016-03-23 y versiones posteriores. Para convertir una cadena de marca temporal en milisegundos, consulte time_to_epoch(String, String).

La función `parse_time` espera los argumentos siguientes:

pattern

(Cadena) date/time Patrón que sigue los 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().

timezone

(String) La zona horaria de la fecha/hora formateada. El valor predeterminado es "UTC". La función admite zonas horarias Joda-Time Este argumento es opcional.

Ejemplos:

Cuando este mensaje se publica en el tema "A/B", la carga `{"ts": "1970.01.01 AD at 21:46:40 CST"}` se envía al bucket de S3:

```
{
    "ruleArn": "arn:aws:iot:us-east-2:ACCOUNT_ID:rule/RULE_NAME",
    "topicRulePayload": {
        "sql": "SELECT parse_time(\"yyyy.MM.dd G 'at' HH:mm:ss z\", 100000000,
 'America/Belize' ) as ts FROM 'A/B'",

        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "s3": {
                    "roleArn": "arn:aws:iam::ACCOUNT_ID:rule:role/ROLE_NAME",
                    "bucketName": "BUCKET_NAME",
                    "key": "KEY_NAME"
                }
            }
        ],
        "ruleName": "RULE_NAME"
    }
}
```

Cuando este mensaje se publica 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 envía al bucket de S3:

```
{
    "ruleArn": "arn:aws:iot:us-east-2:ACCOUNT_ID:rule/RULE_NAME",
    "topicRulePayload": {
        "sql": "SELECT parse_time(\"yyyy.MM.dd G 'at' HH:mm:ss z\", timestamp() ) as ts
 FROM 'A/B'",
        "awsIotSqlVersion": "2016-03-23",
        "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 publica en el tema "A/B", la carga se envía al bucket de S3 con la clave = "2017":

```
{
    "ruleArn": "arn:aws:iot:us-east-2:ACCOUNT_ID:rule/RULE_NAME",
    "topicRulePayload": {
        "sql": "SELECT * FROM 'A/B'",
        "awsIotSqlVersion": "2016-03-23",
        "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 al segundo argumento. Los argumentos `Decimal` se redondean con doble precisión antes de la aplicación de la función. Es compatible con la versión 2015-10-08 de SQL y versiones posteriores. Es compatible con la versión 2015-10-08 de SQL y versiones posteriores.

Ejemplo: `power(2, 5)` = 32.0.

Tipo de argumento 1	Tipo de argumento 2	Output
`Int/Decimal`	`Int/Decimal`	Un `Decimal` (con doble precisi primer argumento elevado a la segundo argumento.
`Int/Decimal/String`	`Int/Decimal/String`	Un `Decimal` (con doble precisi primer argumento elevado a la segundo argumento. Todas las se convierten en decimales. Si `String` no se convierte en un `Decimal`, el resultado es Unde
Otro valor	Otro valor	`Undefined` .

principal()

Devuelve la entidad principal que el dispositivo utiliza para la autenticación, en función de cómo se publicó el mensaje de activación. En la siguiente tabla se describe la entidad principal devuelta para cada método y protocolo de publicación.

Cómo se publica el mensaje	Protocolo	Tipo de credenciales
Cliente MQTT	MQTT	Certificado de dispositivo X.509
AWS IoT cliente MQTT de consola	MQTT	Usuario o rol de IAM

Cómo se publica el mensaje	Protocolo	Tipo de credenciales
AWS CLI	HTTP	Usuario o rol de IAM
AWS IoT SDK de dispositivo	MQTT	Certificado de dispositivo X.509
AWS IoT SDK de dispositivo	MQTT ha terminado WebSocket	Usuario o rol de IAM

En los siguientes ejemplos se muestran los distintos tipos de valores que `principal()` puede devolver:

- Huella digital del certificado X.509: `ba67293af50bf2506f5f93469686da660c7c844e7b3950bfb16813e0d31e9373`
- ID de rol de IAM y nombre de sesión: `ABCD1EFG3HIJK2LMNOP5:my-session-name`
- Devuelve un ID de usuario: `ABCD1EFG3HIJK2LMNOP5`

rand()

Devuelve un valor pseudoaleatorio, distribuido de forma uniforme entre 0,0 y 1,0. Es compatible con la versión 2015-10-08 de SQL y versiones posteriores.

Ejemplo:

`rand()` = 0.8231909191640703

regexp_matches(String, String)

Devuelve verdadero si la cadena (primer argumento) contiene una coincidencia para la expresión regular (segundo argumento). Si usa `|` en la expresión regular, úselo con `()`.

Ejemplos:

`regexp_matches("aaaa", "a{2,}")` = true.

`regexp_matches("aaaa", "b")` = false.

`regexp_matches("aaa", "(aaa|bbb)")` = true.

`regexp_matches("bbb", "(aaa|bbb)")` = true.

`regexp_matches("ccc", "(aaa|bbb)")` = 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`	La `String`.
Matriz	La representación `String` del valor `Array` (mediante reglas de conversión estándar).
Objeto	La representación `String` de la cosa (mediante reglas de conversión estándar).
Nulo	`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, es posible que la cadena obtenida no sea una expresión regular válida. Si el argumento (convertido) no es un regex válido, 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 2015-10-08 de SQL y versiones posteriores.

Ejemplo:

`regexp_replace("abcd", "bc", "x")` = "axd".

`regexp_replace("abcd", "b(.*)d", "$1")` = "ac".

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 de origen.
Matriz	La representación `String` del valor `Array` (mediante reglas de conversión estándar).
Objeto	La representación `String` de la cosa (mediante reglas de conversión estándar).
Nulo	`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, es posible que la cadena obtenida no sea 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 son cadenas se convierten 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 2015-10-08 de SQL 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` de la cosa (mediante reglas de conversión estándar).
Nulo	`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, es posible que la cadena obtenida no sea una expresión regular válida. Si el argumento (convertido) no es una expresión regex válida, el resultado es `Undefined`.

remainder(Decimal, Decimal)

Devuelve el resto de la división del primer argumento por el segundo argumento. Es igual que mod(Decimal, Decimal). También puede utilizar "%" como un operador infijo para la misma funcionalidad modulo. Es compatible con la versión 2015-10-08 de SQL y versiones posteriores.

Ejemplo: `remainder(8, 3)` = 2.

Operando izquierdo	Operando derecho	Output
`Int`	`Int`	`Int`, el primer y el segundo arg los que quiere ejecutar la funció
`Int/Decimal`	`Int/Decimal`	`Decimal`, el primer argumento segundo operando para los que ejecutar la función modulo.
`String/Int/Decimal`	`String/Int/Decimal`	Si todas las cadenas se convier decimales, la función modulo se el primer y el segundo argumen contrario, `Undefined` .
Otro valor	Otro valor	`Undefined` .

replace(Cadena, Cadena, Cadena)

Reemplaza todas las instancias del segundo argumento que hay en el primer argumento por el tercer argumento. Es compatible con la versión 2015-10-08 de SQL y versiones posteriores.

Ejemplo:

`replace("abcd", "bc", "x")` = `"axd"`.

`replace("abcdabcd", "b", "x")` = `"axcdaxcd"`.

Todos los argumentos

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 de origen.
Matriz	La representación `String` del valor `Array` (mediante reglas de conversión estándar).
Objeto	La representación `String` de la cosa (mediante reglas de conversión estándar).
Nulo	`Undefined` .
Sin definir	`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 establece en el valor válido más cercano (0 o 1000). Es compatible con la versión 2015-10-08 de SQL 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` proporcio nado.
`String`	`String`	El segundo argumento se convierte en un valor `Decimal` que se redondea 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 las conversio nes estándar y, a continuación, se aplica la función rpad a dicho

Tipo de argumento 1	Tipo de argumento 2	Resultado
		valor `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`, (p. ej., 0.5), el valor `Decimal` se redondea al valor superior. Es compatible con la versión 2015-10-08 de SQL 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
`Int`	El argumento.
`Decimal`	El valor `Decimal` se redondea al valor `Int` inferior más cercano.
`String`	El valor `Decimal` se redondea al valor `Int` inferior más cercano. Si la cadena no se puede convertir en un valor `Decimal`, el resultado es `Undefined` .
Otro valor	`Undefined` .

rtrim(String)

Elimina todos los espacios en blanco del final (tabuladores y espacios) del valor `String` proporcionado. Es compatible con la versión 2015-10-08 de SQL y versiones posteriores.

Ejemplos:

`rtrim(" h i ")` = " h i"

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”).
Matriz	La representación `String` del valor `Array` (mediante reglas de conversión estándar).
Objeto	La representación `String` de la cosa (mediante reglas de conversión estándar).
Nulo	`Undefined` .
Sin definir	`Undefined`

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 2015-10-08 de SQL 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 2015-10-08 de SQL y versiones posterior es.
Otro valor	`Undefined` .

sin(Decimal)

Devuelve el seno de un número en radianes. Los argumentos `Decimal` se redondean con doble precisión antes de la aplicación de la función. Es compatible con la versión 2015-10-08 de SQL 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

Tipo de argumento	Resultado
	convertir en un valor `Decimal`, el resultado es `Undefined` .
Matriz	`Undefined` .
Objeto	`Undefined` .
Nulo	`Undefined` .
`Undefined`	`Undefined` .

sinh(Decimal)

Devuelve el seno hiperbólico de un número. Los valores `Decimal` se redondean con doble precisión antes de la aplicación de la función. El resultado es un valor `Decimal` de doble precisión. Es compatible con la versión 2015-10-08 de SQL y versiones posteriores.

Ejemplo: `sinh(2.3)` = 4.936961805545957

Tipo de argumento	Resultado
`Int`	`Decimal` (con doble precisión); el seno hiperbólico del argumento.
`Decimal`	`Decimal` (con doble precisión); el seno hiperbólico del argumento.
`Boolean`	`Undefined` .
`String`	`Decimal` (con doble precisión); el seno hiperbólico del argumento. Si la cadena no se puede convertir en un valor `Decimal`, el resultado es `Undefined` .
Matriz	`Undefined` .
Objeto	`Undefined` .

Tipo de argumento	Resultado
Nulo	`Undefined .`
Sin definir	`Undefined .`

sourceip()

Recupera la dirección IP de un dispositivo o del enrutador que se conecta a él. Si el dispositivo está conectado directamente a internet, la función devolverá la dirección IP de origen del dispositivo. Si el dispositivo está conectado a un enrutador que se conecta a internet, la función devolverá la dirección IP de origen del enrutador. Es compatible con la versión SQL 2016-03-23. `sourceip()` no toma ningún parámetro.

> **Important**
>
> La dirección IP de origen público de un dispositivo suele ser la dirección IP de la última puerta de enlace de traducción de direcciones de red (NAT), como el enrutador o el módem de cable del proveedor de servicios de internet.

Ejemplos:

```
sourceip()="192.158.1.38"
```

```
sourceip()="1.102.103.104"
```

```
sourceip()="2001:db8:ff00::12ab:34cd"
```

Ejemplo de SQL:

```
SELECT *, sourceip() as deviceIp FROM 'some/topic'
```

Ejemplos de cómo utilizar la función sourceip () en las acciones de las AWS IoT Core reglas:

Ejemplo 1

El siguiente ejemplo muestra cómo llamar a la función () como plantilla de sustitución en una acción de DynamoDB.

```
{
```

```
 "topicRulePayload": {
  "sql": "SELECT * AS message FROM 'some/topic'",
  "ruleDisabled": false,
  "awsIotSqlVersion": "2016-03-23",
  "actions": [
   {
    "dynamoDB": {
     "tableName": "my_ddb_table",
     "hashKeyField": "key",
     "hashKeyValue": "${sourceip()}",
     "rangeKeyField": "timestamp",
     "rangeKeyValue": "${timestamp()}",
     "roleArn": "arn:aws:iam::123456789012:role/aws_iot_dynamoDB"
    }
   }
  ]
 }
}
```

Ejemplo 2

El siguiente ejemplo muestra cómo añadir la función sourceip() como propiedad de usuario de MQTT mediante plantillas de sustitución.

```
{
 "topicRulePayload": {
  "sql": "SELECT * FROM 'some/topic'",
  "ruleDisabled": false,
  "awsIotSqlVersion": "2016-03-23",
  "actions": [
   {
    "republish": {
     "topic": "${topic()}/republish",
     "roleArn": "arn:aws:iam::123456789012:role/aws_iot_republish",
     "headers": {
      "payloadFormatIndicator": "UTF8_DATA",
      "contentType": "rule/contentType",
      "correlationData": "cnVsZSBjb3JyZWxhdGlvbiBkYXRh",
      "userProperties": [
       {
        "key": "ruleKey1",
        "value": "ruleValue1"
       },
```

```
        {
         "key": "sourceip",
         "value": "${sourceip()}"
        }
       ]
      }
     }
    }
   ]
  }
}
```

Puede recuperar la dirección IP de origen de los mensajes que se transmiten a AWS IoT Core las reglas desde las rutas Message Broker y Basic Ingest. También puede recuperar la IP de origen de ambos IPv4 IPv6 mensajes. La IP de origen se mostrará de la siguiente manera:

IPv6: `yyyy:yyyy:yyyy::yyyy:yyyy`

IPv4: `xxx.xxx.xxx.xxx`

> **Note**
>
> La IP de origen original no se transferirá a través de la acción Republicar.

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`*) o (*`String`*,*`Int`*,*`Int`*), las conversiones estándar se aplican a los argumentos para intentar convertirlos en los tipos correctos. Si no es posible convertirlos, el resultado de la función será `Undefined`. Es compatible con la versión 2015-10-08 de SQL 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)` = "".

sql_version()

Devuelve la versión de SQL especificada en esta regla. Es compatible con la versión 2015-10-08 de SQL y versiones posteriores.

Ejemplo:

`sql_version()` = "2016-03-23"

sqrt(Decimal)

Devuelve la raíz cuadrada de un número. Los argumentos `Decimal` se redondean con doble precisión antes de la aplicación de la función. Es compatible con la versión 2015-10-08 de SQL y versiones posteriores.

Ejemplo: `sqrt(9)` = 3,0.

Tipo de argumento	Resultado
`Int`	La raíz cuadrada del argumento.
`Decimal`	La raíz cuadrada del argumento.

Tipo de argumento	Resultado
`Boolean`	`Undefined` .
`String`	La raíz cuadrada del argumento. Si la cadena no se puede convertir en un valor `Decimal`, el resultado es `Undefined` .
Matriz	`Undefined` .
Objeto	`Undefined` .
Nulo	`Undefined` .
Sin definir	`Undefined` .

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 2015-10-08 de SQL y versiones posteriores.

Ejemplo:

`startswith("ranger","ran")` = true

Tipo de argumento 1	Tipo de argumento 2	Resultado
`String`	`String`	Si la primera cadena comienza segunda cadena.
Otro valor	Otro valor	Ambos argumentos se convierte cadenas con las reglas de conv estándar. Devuelve verdadero s cadena comienza con la segund Si alguno de los argumentos es `Undefined` , el resultado es U .

tan(Decimal)

Devuelve la tangente de un número en radianes. Los valores `Decimal` se redondean con doble precisión antes de la aplicación de la función. Es compatible con la versión 2015-10-08 de SQL y versiones posteriores.

Ejemplo: `tan(3)` = -0.1425465430742778

Tipo de argumento	Resultado
`Int`	`Decimal` (con doble precisión), la tangente del argumento.
`Decimal`	`Decimal` (con doble precisión), la tangente del argumento.
`Boolean`	`Undefined` .
`String`	`Decimal` (con doble precisión), la tangente del argumento. Si la cadena no se puede convertir en un valor `Decimal`, el resultado es `Undefined` .
Matriz	`Undefined` .
Objeto	`Undefined` .
Nulo	`Undefined` .
Sin definir	`Undefined` .

tanh(Decimal)

Devuelve la tangente hiperbólica de un número en radianes. Los valores `Decimal` se redondean con doble precisión antes de la aplicación de la función. Es compatible con la versión 2015-10-08 de SQL y versiones posteriores.

Ejemplo: `tanh(2.3)` = 0.9800963962661914

Tipo de argumento	Resultado
`Int`	`Decimal` (con doble precisión), la tangente hiperbólica del argumento.
`Decimal`	`Decimal` (con doble precisión), la tangente hiperbólica del argumento.
`Boolean`	`Undefined` .
`String`	`Decimal` (con doble precisión), la tangente hiperbólica del argumento. Si la cadena no se puede convertir en un valor `Decimal`, el resultado es `Undefined` .
Matriz	`Undefined` .
Objeto	`Undefined` .
Nulo	`Undefined` .
Sin definir	`Undefined` .

time_to_epoch(String, String)

Utilice la función `time_to_epoch` para convertir una cadena de marca temporal en un número de milisegundos en el formato de tiempo Unix. Es compatible con la versión SQL 2016-03-23 y versiones posteriores. Para convertir milisegundos en una cadena de marca temporal formateada, consulte parse_time(String, Long[, String]).

La función `time_to_epoch` espera los argumentos siguientes:

timestamp

(String) La cadena de marca temporal que se va a convertir a milisegundos desde tiempo Unix. Si la cadena de marca temporal no especifica una zona horaria, la función utiliza la zona horaria UTC.

pattern

(Cadena) Un date/time patrón que sigue los formatos de JDK11 tiempo.

Ejemplos:

`time_to_epoch("2020-04-03 09:45:18 UTC+01:00", "yyyy-MM-dd HH:mm:ss VV")= 1585903518000`

`time_to_epoch("18 December 2015", "dd MMMM yyyy")= 1450396800000`

`time_to_epoch("2007-12-03 10:15:30.592 America/Los_Angeles", "yyyy-MM-dd HH:mm:ss.SSS z")= 1196705730592`

timestamp()

Devuelve la marca de tiempo actual en milisegundos a partir de las 00:00:00 (hora universal coordinada) del jueves 1 de enero de 1970, según lo observado por el motor de reglas. AWS IoT Es compatible con la versión 2015-10-08 de SQL 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 concreto. 1 designa el primer segmento. Para el tema `foo/bar/baz`, topic(1) devuelve `foo`, topic(2) devuelve `bar` y así sucesivamente. Es compatible con la versión 2015-10-08 de SQL y versiones posteriores.

Ejemplos:

`topic()` = "things/myThings/thingOne"

`topic(1)` = "things"

Cuando se usa Basic Ingest, el prefijo inicial del tema (`$aws/rules/`*`rule-name`*) no está disponible para la función topic(). Por ejemplo, con el tema:

`$aws/rules/BuildingManager/Buildings/Building5/Floor2/Room201/Lights`

`topic()` = "Buildings/Building5/Floor2/Room201/Lights"

`topic(3)` = "Floor2"

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 2015-10-08 de SQL y versiones posteriores.

Ejemplo:

`traceid()` = "12345678-1234-1234-1234-123456789012"

transform(String, Object, Array)

Devuelve una matriz de objetos que contiene el resultado de la transformación especificada del parámetro `Object` en el parámetro `Array`.

Es compatible con la versión SQL 2016-03-23 y versiones posteriores.

Cadena

El modo de transformación que se debe utilizar. Consulte la siguiente tabla para ver los modos de transformación compatibles y cómo crean el `Result` a partir de los parámetros `Object` y `Array`.

Objeto

Un objeto que contiene los atributos que se van a aplicar a cada elemento de la `Array`.

Matriz

Una matriz de objetos a los que se aplican los atributos de `Object`.

Cada objeto de esta matriz corresponde a un objeto de la respuesta de la función. Cada objeto de la respuesta de la función contiene los atributos presentes en el objeto original y los atributos proporcionados por `Object` con arreglo al modo de transformación especificado en `String`.

Parámetro **`String`**	Parámetro **`Object`**	Parámetro **`Array`**	Resultado
`enrichArray`	Objeto	Matriz de objetos	Una matriz de objetos en la que cada objeto contiene los atributos de un elemento del parámetro `Array`

Parámetro **`String`**	Parámetro **`Object`**	Parámetro **`Array`**	Resultado
			y los atributos del parámetro `Object`.
Cualquier otro valor	Cualquier valor	Cualquier valor	Sin definir

> **Note**
>
> La matriz devuelta por esta función está limitada a 128 KiB.

Ejemplo 1 de la función de transformación

En este ejemplo se muestra cómo la función transform() produce una matriz única de objetos a partir de un objeto de datos y una matriz.

En este ejemplo, se publica el siguiente mensaje en el tema `A/B` de MQTT.

```
{
    "attributes": {
        "data1": 1,
        "data2": 2
    },
    "values": [
        {
            "a": 3
        },
        {
            "b": 4
        },
        {
            "c": 5
        }
    ]
}
```

Esta instrucción SQL para una acción de regla temática utiliza la función transform() con un valor de `String` de `enrichArray`. En este ejemplo, `Object` es la propiedad `attributes` de la carga del mensaje y `Array` es la matriz `values`, que contiene tres objetos.

```
select value transform("enrichArray", attributes, values) from 'A/B'
```

Al recibir la carga del mensaje, la instrucción SQL se evalúa en la siguiente respuesta.

```
[
  {
    "a": 3,
    "data1": 1,
    "data2": 2
  },
  {
    "b": 4,
    "data1": 1,
    "data2": 2
  },
  {
    "c": 5,
    "data1": 1,
    "data2": 2
  }
]
```

Ejemplo 2 de la función de transformación

En este ejemplo se muestra cómo la función transform() puede usar valores literales para incluir y cambiar el nombre de los atributos individuales de la carga del mensaje.

En este ejemplo, se publica el siguiente mensaje en el tema `A/B` de MQTT. Es el mismo mensaje que se usó en the section called “Ejemplo 1 de la función de transformación”.

```
{
    "attributes": {
        "data1": 1,
        "data2": 2
    },
    "values": [
        {
            "a": 3
        },
        {
            "b": 4
```

```
        },
        {
            "c": 5
        }
    ]
}
```

Esta instrucción SQL para una acción de regla temática utiliza la función transform() con un valor de `String` de `enrichArray`. El `Object` de la función transform() tiene un único atributo llamado `key` con el valor de `attributes.data1` en la carga del mensaje y `Array` es la matriz `values`, que contiene los mismos tres objetos utilizados en el ejemplo anterior.

```
select value transform("enrichArray", {"key": attributes.data1}, values) from 'A/B'
```

Al recibir la carga del mensaje, esta instrucción SQL se evalúa en la siguiente respuesta. Observe cómo la propiedad `data1` se nombra `key` en la respuesta.

```
[
  {
    "a": 3,
    "key": 1
  },
  {
    "b": 4,
    "key": 1
  },
  {
    "c": 5,
    "key": 1
  }
]
```

Ejemplo 3 de la función de transformación

En este ejemplo se muestra cómo se puede utilizar la función transform() en las cláusulas SELECT anidadas para seleccionar varios atributos y crear nuevos objetos para su posterior procesamiento.

En este ejemplo, se publica el siguiente mensaje en el tema `A/B` de MQTT.

```
{
```

```
  "data1": "example",
  "data2": {
    "a": "first attribute",
    "b": "second attribute",
    "c": [
      {
        "x": {
          "someInt": 5,
          "someString": "hello"
        },
        "y": true
      },
      {
        "x": {
          "someInt": 10,
          "someString": "world"
        },
        "y": false
      }
    ]
  }
}
```

El `Object` para esta función de transformación es el objeto devuelto por la instrucción SELECT, que contiene los elementos `a` y `b` de la cosa `data2` del mensaje. El parámetro `Array` consta de los dos objetos de la matriz `data2.c` del mensaje original.

```
select value transform('enrichArray', (select a, b from data2), (select value c from
 data2)) from 'A/B'
```

Con el mensaje anterior, la instrucción SQL se evalúa en la siguiente respuesta.

```
[
  {
    "x": {
      "someInt": 5,
      "someString": "hello"
    },
    "y": true,
    "a": "first attribute",
    "b": "second attribute"
  },
```

```
  {
    "x": {
      "someInt": 10,
      "someString": "world"
    },
    "y": false,
    "a": "first attribute",
    "b": "second attribute"
  }
]
```

La matriz devuelta en esta respuesta podría usarse con las acciones de la regla temática compatibles con `batchMode`.

trim(String)

Elimina todos los espacios en blanco del principio y del final del valor `String` proporcionado. Es compatible con la versión 2015-10-08 de SQL y versiones posteriores.

Ejemplo:

`Trim(" hi ") =` "hi"

Tipo de argumento	Resultado
`Int`	La representación `String` de `Int` con todos los espacios en blanco del principio y del final suprimidos.
`Decimal`	La representación `String` de `Decimal` con todos los espacios en blanco del principio y del final suprimidos.
`Boolean`	La representación `String` de `Boolean` ("true" o "false") con todos los espacios en blanco del principio y del final suprimidos.
`String`	El argumento `String` con todos los espacios en blanco del principio y del final suprimidos.

Tipo de argumento	Resultado
Matriz	La representación `String` del valor `Array` mediante reglas de conversión estándar.
Objeto	La representación `String` de la cosa mediante reglas de conversión estándar.
Nulo	`Undefined` .
Sin definir	`Undefined` .

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 establece en cero. Si el segundo argumento es superior a 34, se establece en 34. Los ceros del final se eliminan del resultado. Es compatible con la versión 2015-10-08 de SQL y versiones posteriores.

Ejemplos:

`trunc(2.3, 0)` = 2.

`trunc(2.3123, 2)` = 2.31.

`trunc(2.888, 2)` = 2,88.

`trunc(2.00, 5)` = 2.

Tipo de argumento 1	Tipo de argumento 2	Resultado
`Int`	`Int`	El valor de origen.
`Int/Decimal`	`Int/Decimal`	El primer argumento se trunca e longitud indicada por el segundo . Si el segundo argumento no es `Int`, se redondea al valor `Int` i cercano.

Tipo de argumento 1	Tipo de argumento 2	Resultado
`Int/Decimal/String`	`Int/Decimal`	El primer argumento se trunca e longitud indicada por el segundo . Si el segundo argumento no es `Int`, se redondea al valor `Int` i cercano. Los valores de tipo `St` convierten en valores de tipo `De` Si se produce un error en la con n de cadena, el resultado obten `Undefined` .
Otro valor		`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 2015-10-08 de SQL y versiones posteriores.

Ejemplos:

`upper("hello")` = "HELLO"

`upper(["hello"])` = "[\"HELLO\"]"

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 versión 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 `topic/subtopic`: `{"lat_long": [47.606,-122.332]}`

Instrucción SQL: `SELECT {'latitude': get(lat_long, 0),'longitude':get(lat_long, 1)} as lat_long FROM 'topic/subtopic'`

La carga de salida obtenida sería: `{"lat_long": {"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 Array). Por ejemplo:

Carga de entrada publicada en el tema `topic/subtopic`: `{"lat": 47.696, "long": -122.332}`

Instrucción SQL: `SELECT [lat,long] as lat_long FROM 'topic/subtopic'`

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.

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 el valor *t[i]* de todas las cláusulas `WHEN`. Si se encuentra una coincidencia, la expresión *r[i]* correspondiente se convierte en el resultado de la instrucción `CASE`. Las cláusulas `WHEN` se evalúan en orden, de modo que si hay más de una cláusula coincidente, el resultado de la primera cláusula coincidente se convierte en el resultado de la instrucción `CASE`. Si no hay coincidencias, el resultado es *r[e]* de la cláusula `ELSE`. Si no hay ninguna coincidencia ni cláusula `ELSE`, el resultado es `Undefined`.

Las instrucciones `CASE` necesitan como mínimo una cláusula `WHEN`. Una cláusula `ELSE` es opcional.

Por ejemplo:

Carga de entrada publicada en el tema `topic/subtopic`:

```
{
    "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 'topic/subtopic'
```

La carga de salida obtenida sería:

```
{
    "instructions":"caution"
}
```

> **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.

Operador “.”

Este operador accede a los miembros de los objetos JSON incrustados y funciona de forma idéntica a ANSI SQL y. JavaScript Por ejemplo:

```
SELECT foo.bar AS bar.baz FROM 'topic/subtopic'
```

selecciona el valor de la propiedad `bar` de la cosa `foo` de la carga del siguiente mensaje enviado al tema `topic/subtopic`.

```
{
  "foo": {
    "bar": "RED",
    "bar1": "GREEN",
    "bar2": "BLUE"
  }
}
```

Si el nombre de una propiedad JSON incluye un guion o caracteres numéricos, la notación “punto” no funcionará. En su lugar, debe utilizar la función get para extraer el valor de la propiedad.

En este ejemplo, se envía el siguiente mensaje al tema `iot/rules`.

```
{
  "mydata": {
    "item2": {
      "0": {
        "my-key": "myValue"
      }
    }
  }
}
```

Normalmente, el valor de `my-key` se identificaría como en esta consulta.

```
SELECT * from iot/rules WHERE mydata.item2.0.my-key= "myValue"
```

Sin embargo, dado que el nombre de la propiedad `my-key` contiene un guion y `item2` un carácter numérico, se debe utilizar la función get, tal como se muestra en la siguiente consulta.

```
SELECT * from 'iot/rules' WHERE get(get(get(mydata,"item2"),"0"),"my-key") = "myValue"
```

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 'topic/subtopic'
```

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

Puedes usar una plantilla de sustitución para aumentar los datos JSON devueltos 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 admitida AWS IoT en las cláusulas SELECT, WHERE yAWS IoT acciones de reglas. Esta expresión se puede conectar a un campo de acción de una regla, lo que le permite configurar dinámicamente una acción. En efecto, esta función sustituye a una parte de información de una acción. Esto incluye funciones, operadores e información presente en la carga del mensaje original.

> **Important**
>
> Dado que las expresiones en plantillas de sustitución se evalúan por separado de la declaración “SELECT...”, no se puede hacer referencia a un alias creado con la cláusula

AS. Solo puede hacer referencia a la información presente en la carga, las funciones y los operadores originales.

Para obtener más información acerca de las expresiones admitidas, consulte AWS IoT Referencia de SQL.

Las siguientes acciones de las reglas admiten plantillas de sustitución. Cada acción admite diferentes campos que se pueden sustituir.

- Apache Kafka
- CloudWatch alarmas
- CloudWatch Registros
- CloudWatch métricas
- DynamoDB
- Dinamo DBv2
- Elasticsearch
- HTTP
- AWS IoT Events
- AWS IoT SiteWise
- Kinesis Data Streams
- Firehose
- Lambda
- Ubicación
- OpenSearch
- Volver a publicar
- S3
- SNS
- SQS
- Step Functions
- Timestream

Las plantillas de sustitución aparecen en los parámetros de acción dentro de una regla:

```
{
    "sql": "SELECT *, timestamp() AS timestamp 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 siguiente JSON publicado en `my/iot/topic`:

```
{
    "deviceid": "iot123",
    "temp": 54.98,
    "humidity": 32.43,
    "coords": {
        "latitude": 47.615694,
        "longitude": -122.3359976
    }
}
```

A continuación, esta regla publica el siguiente JSON en`my/iot/topic/republish`, que AWS IoT sustituye a`${topic()}/republish`:

```
{
    "deviceid": "iot123",
    "temp": 54.98,
    "humidity": 32.43,
    "coords": {
        "latitude": 47.615694,
        "longitude": -122.3359976
    },
    "timestamp": 1579637878451
}
```

Consultas de objetos anidados

Puede utilizar cláusulas SELECT anidadas para consultar atributos dentro de matrices y objetos JSON internos. Es compatible con la versión SQL 2016-03-23 y versiones posteriores.

Considere el siguiente mensaje MQTT:

```
{
    "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 }
    ]
}
```

Example

Puede convertir valores en una nueva matriz con la siguiente regla.

```
SELECT (SELECT VALUE n FROM e) as sensors FROM 'my/topic'
```

La regla generará la salida siguiente.

```
{
    "sensors": [
        "temperature",
        "light",
        "acidity"
    ]
}
```

Example

Usando el mismo mensaje MQTT, también puede consultar un valor específico dentro de un objeto anidado con la siguiente regla.

```
SELECT (SELECT v FROM e WHERE n = 'temperature') as temperature FROM 'my/topic'
```

La regla generará la salida siguiente.

```
{
    "temperature": [
        {
            "v": 22.5
        }
```

```
    ]
}
```

Example

También puede aplanar la salida con una regla más complicada.

```
SELECT get((SELECT v FROM e WHERE n = 'temperature'), 0).v as temperature FROM 'topic'
```

La regla generará la salida siguiente.

```
{
    "temperature": 22.5
}
```

Uso de las cargas binarias

Para tratar la carga del mensaje 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.

En este tema:

- Ejemplos de carga binaria
- Descodificación de cargas de mensajes de protobuf

Ejemplos de carga binaria

Si utiliza * para hacer referencia a la carga del mensaje como datos binarios sin procesar, puede añadir datos a la regla. Si tiene una carga vacía o JSON, a la carga resultante se le pueden agregar datos mediante la regla. A continuación se muestran ejemplos de cláusulas `SELECT` admitidas.

- Puede usar las siguientes cláusulas `SELECT` con solo un * para cargas binarias.
  - ```
    SELECT * FROM 'topic/subtopic'
    ```
  - ```
    SELECT * FROM 'topic/subtopic' WHERE timestamp() % 12 = 0
    ```
- También puede agregar datos y utilizar las siguientes cláusulas `SELECT`.
  - ```
    SELECT *, principal() as principal, timestamp() as time FROM 'topic/subtopic'
    ```
    ```

- ```
  SELECT encode(*, 'base64') AS data, timestamp() AS ts FROM 'topic/subtopic'
  ```

- También puede usar estas cláusulas `SELECT` con cargas binarias.
  - Lo siguiente hace referencia a `device_type` en la cláusula WHERE.

    ```
    SELECT * FROM 'topic/subtopic' WHERE device_type = 'thermostat'
    ```

  - También se admite lo siguiente.

    ```
    {
     "sql": "SELECT * FROM 'topic/subtopic'",
     "actions": [
      {
       "republish": {
        "topic": "device/${device_id}"
       }
      }
     ]
    }
    ```

Las siguientes acciones de regla no admiten cargas binarias, por lo que debe descodificarlas.

- Algunas acciones de regla no admiten la entrada de carga binaria, como la acción Lambda, por lo que debe descodificar las cargas binarias. La acción de regla Lambda puede recibir datos binarios si está codificada en base64 y en una carga JSON. Puede hacer esto cambiando la regla a lo siguiente.

  ```
  SELECT encode(*, 'base64') AS data FROM 'my_topic'
  ```

- La instrucción SQL no admite cadenas como entrada. Para convertir una entrada de cadena en JSON, puede ejecutar el siguiente comando.

  ```
  SELECT decode(encode(*, 'base64'), 'base64') AS payload FROM 'topic'
  ```

Descodificación de cargas de mensajes de protobuf

Los búferes de protocolo (protobuf) son un formato de datos de código abierto que se utiliza para serializar datos estructurados en un formato binario compacto. Se utiliza para transmitir
  ```

datos a través de redes o para almacenarlos en archivos. Protobuf le permite enviar datos en paquetes pequeños y a un ritmo más rápido que otros formatos de mensajería. AWS IoT Core Las reglas admiten protobuf al proporcionar la función SQL decode (value, decodingScheme), que permite decodificar las cargas útiles de mensajes codificadas por protobuf en formato JSON y enrutarlas a los servicios descendentes. En esta sección se detalla el proceso de configuración de la decodificación protobuf en Rules. step-by-step AWS IoT Core

En esta sección:

- Requisitos previos
- Crear archivos descriptores
- Cargar los archivos descriptores en un bucket de S3
- Configurar la descodificación protobuf en las reglas
- Limitaciones
- Prácticas recomendadas

Requisitos previos

- Conocimientos básicos de los búferes de protocolo (protobuf)
- Los archivos `.proto` que definen los tipos de mensajes y las dependencias relacionadas
- Instalación del compilador Protobuf (protoc) en su sistema

Crear archivos descriptores

Si ya dispone de los archivos descriptores, puede omitir este paso. Un archivo descriptor (`.desc`) es una versión compilada de un archivo `.proto`, que es un archivo de texto que define las estructuras de datos y los tipos de mensajes que se utilizarán en una serialización de protobuf. Para generar un archivo descriptor, debe definir un archivo `.proto` y usar el compilador protoc para compilarlo.

1. Crear archivos `.proto` que definan los tipos de mensajes. Un ejemplo de archivo `.proto` sería el siguiente:

```
syntax = "proto3";

message Person {
  optional string name = 1;
  optional int32 id = 2;
```

```
  optional string email = 3;
}
```

En este archivo `.proto` de ejemplo, se utiliza la sintaxis proto3 y se define el tipo de mensaje `Person`. La definición del mensaje `Person` especifica tres campos (nombre, identificador y correo electrónico). Para obtener más información sobre los formatos de los mensajes de los archivos `.proto`, consulte la Guía de lenguaje (proto3).

2. Utilice el compilador protoc para compilar los archivos `.proto` y generar un archivo descriptor. Un ejemplo de comando para crear un archivo descriptor (`.desc`) puede ser el siguiente:

```
protoc --descriptor_set_out=<FILENAME>.desc \
    --proto_path=<PATH_TO_IMPORTS_DIRECTORY> \
    --include_imports \
    <PROTO_FILENAME>.proto
```

Este comando de ejemplo genera un archivo `<FILENAME>.desc` descriptor que AWS IoT Core Rules puede usar para decodificar las cargas útiles de protobuf que se ajusten a la estructura de datos definida en. `<PROTO_FILENAME>.proto`

- `--descriptor_set_out`

  Especifica el nombre del archivo descriptor (`<FILENAME>.desc`) que se debe generar.

- `--proto_path`

  Especifica las ubicaciones de los archivos `.proto` importados a los que hace referencia el archivo que se está compilando. Puede especificar la marca varias veces si tiene varios archivos `.proto` importados con ubicaciones diferentes.

- `--include_imports`

  Especifica que todos los archivos `.proto` importados también se deben compilar e incluir en el archivo descriptor `<FILENAME>.desc`.

- `<PROTO_FILENAME>.proto`

  Especifica el nombre del archivo `.proto` que desea compilar.

Para obtener más información sobre la referencia protoc, consulte la Referencia de la API.

Cargar los archivos descriptores en un bucket de S3

Tras crear los archivos descriptores`<FILENAME>.desc`, cárguelos en un bucket de Amazon S3 mediante la AWS API, el AWS SDK o el. `<FILENAME>.desc` Consola de administración de AWS

Consideraciones importantes

- Asegúrese de cargar los archivos descriptores en un bucket de Amazon S3 en el mismo Región de AWS lugar Cuenta de AWS en el que pretende configurar sus reglas.
- Asegúrese de conceder AWS IoT Core acceso para leer el contenido `FileDescriptorSet` de S3. Si su bucket de S3 tiene el cifrado del servidor (SSE) desactivado o si el bucket de S3 está cifrado con claves administradas por Amazon S3 (SSE-S3), no se requieren configuraciones de políticas adicionales. Esto se puede lograr con la política de bucket de ejemplo:

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Sid": "Statement1",
   "Effect": "Allow",
   "Principal": {
    "Service": "iot.amazonaws.com"
   },
   "Action": "s3:Get*",
                     "Resource": "arn:aws:s3:::<BUCKET NAME>/<FILENAME>.desc"
  }
 ]
}
```

- Si su bucket de S3 está cifrado con una AWS Key Management Service clave (SSE-KMS), asegúrese de conceder AWS IoT Core permiso para usar la clave al acceder a su bucket de S3. Para ello, agregue esta instrucción a su política de claves:

```
{
 "Sid": "Statement1",
 "Effect": "Allow",
 "Principal": {
  "Service": "iot.amazonaws.com"
 },
 "Action": [
  "kms:Decrypt",
```

```
  "kms:GenerateDataKey*",
  "kms:DescribeKey"
 ],
        "Resource": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"

}
```

Configurar la descodificación protobuf en las reglas

Tras cargar los archivos descriptores en su bucket de Amazon S3, configure una regla que pueda descodificar el formato de carga del mensaje protobuf mediante la función decode(value, decodingScheme) de SQL. Puede encontrar una firma y un ejemplo detallados de la función en la función decode(value, DecodingScheme) de SQL de la referencia de la SQL de AWS IoT .

A continuación se muestra un ejemplo de expresión SQL que utiliza la función decode(value, decodingScheme):

```
SELECT VALUE decode(*, 'proto', '<BUCKET NAME>', '<FILENAME>.desc', '<PROTO_FILENAME>',
 '<PROTO_MESSAGE_TYPE>') FROM '<MY_TOPIC>'
```

En este ejemplo de expresión:

- Utiliza la función decode(value, decodingScheme) de SQL para descodificar la carga del mensaje binario a la que hace referencia `*`. Puede ser una carga binaria codificada por protobuf o una cadena JSON que representa una carga protobuf codificada en base64.
- La carga del mensaje proporcionada se codifica con el tipo de mensaje `Person` definido en `PROTO_FILENAME.proto`.
- El bucket de Amazon S3 llamado `BUCKET NAME` contiene el `FILENAME.desc` generado desde `PROTO_FILENAME.proto`.

Tras completar la configuración, publique un mensaje AWS IoT Core sobre el tema al que está suscrita la regla.

Limitaciones

AWS IoT Core Las reglas admiten protobuf con las siguientes limitaciones:

- No se admite la descodificación de cargas de mensajes de protobuf en plantillas de sustitución.

- Al descodificar las cargas de los mensajes de protobuf, puede utilizar la función de descodificación de SQL en una sola expresión SQL hasta dos veces.
- El tamaño máximo de la carga de entrada es de 128 KiB (1 KiB = 1024 bytes), el tamaño máximo de la carga de salida es de 128 KiB y el tamaño máximo de un objeto `FileDescriptorSet` almacenado en un bucket de Amazon S3 es de 32 KiB.
- No se admiten los buckets de Amazon S3 cifrados con cifrado SSE-C.

Prácticas recomendadas

Estas son algunas prácticas recomendadas y consejos de solución de problemas.

- Haga una copia de seguridad de sus archivos proto en el bucket de Amazon S3.

  Se recomienda hacer copias de seguridad de los archivos proto en caso de que algo salga mal. Por ejemplo, si modifica incorrectamente los archivos proto sin copias de seguridad al ejecutar protoc, esto puede provocar problemas en su pila de producción. Existen varias formas de hacer copias de seguridad de los archivos en un bucket de Amazon S3. Por ejemplo, puede utilizar el control de versiones en buckets de S3. Para obtener más información sobre cómo hacer copias de seguridad de los archivos en los buckets de Amazon S3, consulte la Guía para desarrolladores de Amazon S3.
- Configure el AWS IoT registro para ver las entradas del registro.

  Se recomienda configurar el AWS IoT registro para que puedas comprobar AWS IoT los registros de tu cuenta CloudWatch. Cuando la consulta SQL de una regla llama a una función externa, AWS IoT Core Rules genera una entrada de registro con un `eventType` de`FunctionExecution`, que contiene el campo del motivo, que le ayudará a solucionar errores. Los posibles errores incluyen un objeto de Amazon S3 no encontrado o un descriptor de archivo protobuf no válido. Para obtener más información sobre cómo configurar el registro de AWS IoT y ver las entradas de registro, consulte Configurar el registro de AWS IoT y Entradas del registro del motor de reglas.
- Actualice `FileDescriptorSet` con una clave de objeto nueva y actualice la clave de objeto en su regla.

  Puede actualizar `FileDescriptorSet` cargando un archivo descriptor actualizado en su bucket de Amazon S3. Las actualizaciones de `FileDescriptorSet` pueden tardar hasta 15 minutos en reflejarse. Para evitar este retraso, se recomienda cargar las actualizaciones de `FileDescriptorSet` con una clave de objeto nueva y actualizar la clave de objeto en la regla.

Versiones de SQL

El motor de AWS IoT reglas utiliza una sintaxis similar a la de 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. La `awsIotSqlVersion` propiedad le permite especificar qué versión del motor de reglas AWS IoT SQL 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"
        }
    }]
}
```

AWS IoT actualmente admite las siguientes versiones de SQL:

- `2016-03-23`: la versión de SQL creada el 23/03/2016 (recomendada).
- `2015-10-08`: la versión de SQL original creada el 08/10/2015.
- `beta`: la versión beta de SQL más reciente. 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 dentro de objetos. Para obtener más información, consulte Consultas de objetos anidados.
- Compatibilidad con la generación de una matriz como objeto de nivel superior.
- Adición de la función `encode(value, encodingScheme)`, que se puede aplicar en datos con formato JSON y no JSON. Para obtener más información, consulte la función de codificación.

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]
```

AWS IoT Servicio Device Shadow

El servicio AWS IoT Device Shadow añade sombras a los objetos de las AWS IoT cosas. Las sombras pueden hacer que el estado de un dispositivo esté disponible para las aplicaciones y otros servicios, independientemente de que el dispositivo esté conectado AWS IoT o no. AWS IoT los objetos de las cosas pueden tener múltiples sombras con nombre para que su solución de IoT tenga más opciones para conectar sus dispositivos a otras aplicaciones y servicios.

AWS IoT los objetos tipo cosa no tienen sombras hasta que se crean de forma explícita. Las sombras se pueden crear, actualizar y eliminar mediante la AWS IoT consola. Los dispositivos, otros clientes web y los servicios pueden crear, actualizar y eliminar sombras mediante MQTT, y los temas reservados de MQTT, HTTP, con la API de REST de sombra de dispositivo, y AWS CLI para AWS IoT. Como las sombras se almacenan AWS en la nube, pueden recopilar y generar informes sobre el estado del dispositivo desde aplicaciones y otros servicios en la nube, independientemente de que el dispositivo esté conectado o no.

Uso de sombras

Las sombras proporcionan un almacén de datos de confianza para dispositivos, aplicaciones y otros servicios en la nube para compartir datos. Permiten que dispositivos, aplicaciones y otros servicios en la nube se conecten y desconecten sin perder el estado de un dispositivo.

Mientras los dispositivos, las aplicaciones y otros servicios en la nube estén conectados AWS IoT, estos pueden acceder al estado actual de un dispositivo y controlarlo a través de sus sombras. Por ejemplo, una aplicación puede solicitar un cambio en el estado de un dispositivo actualizando una sombra. AWS IoT publica un mensaje que indica el cambio en el dispositivo. El dispositivo recibe este mensaje, actualiza su estado para que coincida y publica un mensaje con su estado actualizado. El servicio Device Shadow refleja este estado actualizado en la sombra correspondiente. La aplicación puede suscribirse a la actualización de la sombra o puede consultar la sombra para conocer su estado actual.

Cuando un dispositivo se desconecta, la aplicación puede seguir comunicándose con el dispositivo AWS IoT y con las sombras de éste. Cuando el dispositivo se vuelve a conectar, recibe el estado actual de sus sombras para que pueda actualizar su estado para que coincida con el de sus sombras y, a continuación, publicar un mensaje con su estado actualizado. Del mismo modo, cuando una aplicación se desconecta y el estado del dispositivo cambia mientras está fuera de línea, el

dispositivo mantiene la sombra actualizada para que la aplicación pueda consultar las sombras para conocer su estado actual cuando se vuelva a conectar.

Si sus dispositivos están desconectados con frecuencia y quiere configurarlos para que reciban mensajes delta después de que se vuelvan a conectar, puede usar la característica de sesión persistente. Para obtener más información sobre el periodo de caducidad de la sesión persistente, consulte el Persistent session expiry period.

Elegir utilizar sombras con nombre o sin nombre

El servicio sombra de dispositivo admite sombras con nombre y sin nombre (o clásicas). Un objeto puede tener varias sombras con nombre, pero no más de una sombra sin nombre. El objeto también puede tener una sombra reservada con nombre, que funciona de forma similar a una sombra con nombre, con la diferencia de que no se puede actualizar el nombre. Para obtener más información, consulte Sombra con nombre reservado.

Un objeto puede tener sombras con nombre y sin nombre al mismo tiempo; sin embargo, la API utilizada para acceder a cada una es ligeramente diferente, por lo que podría ser más eficiente decidir qué tipo de sombra funcionaría mejor para su solución y usar solo dicho tipo. Para obtener más información acerca de la API para acceder a las sombras, consulte Temas de sombra.

Mediante las sombras con nombre, puede crear distintas vistas del estado de un objeto. Por ejemplo, podría dividir un objeto con muchas propiedades en sombras con grupos lógicos de propiedades, cada una identificada por su nombre de sombra. También puede limitar el acceso a las propiedades agrupándolas en distintas sombras y utilizando políticas para controlar el acceso. Para obtener más información sobre las políticas que se pueden usar con las sombras de dispositivo, consulte Acciones, recursos y claves de condición para AWS IoT y las políticas de AWS IoT Core.

Las sombras clásicas sin nombre son más sencillas, pero algo más limitadas que las sombras con nombre. Cada AWS IoT objeto puede tener solo una sombra sin nombre. Si espera que la solución de IoT tenga una necesidad limitada de datos de sombra, puede que así sea como desee comenzar a usar sombras. Sin embargo, si cree que es posible que desee agregar sombras adicionales en el futuro, plantéese la posibilidad de utilizar sombras con nombre desde el principio.

La indexación de flota admite de forma distinta las sombras sin nombre y con nombre. Para obtener más información, consulte Manage fleet indexing.

Acceso a sombras

Cada sombra tiene un tema de MQTT reservado y una URL HTTP que admite las acciones `get`, `update` y `delete` en la sombra.

Las sombras utilizan documentos de sombra JSON para almacenar y recuperar datos. Un documento de sombra contiene una propiedad de estado que describe estos aspectos del estado del dispositivo:

- `desired`

  Las aplicaciones especifican los estados deseados de las propiedades del dispositivo actualizando el objeto `desired`.
- `reported`

  Los dispositivos notifican su estado actual en el objeto `reported`.
- `delta`

  AWS IoT informa de las diferencias entre el estado deseado y el registrado en el `delta` objeto.

Los datos almacenados en una sombra están determinados por la propiedad de estado del cuerpo del mensaje de la acción de actualización. Las acciones de actualización posteriores pueden modificar los valores de un objeto de datos existente y también agregar y eliminar claves y otros elementos del objeto de estado de la sombra. Para obtener más información sobre cómo acceder a las sombras, consulte Uso de sombras en dispositivos y Uso de sombras en aplicaciones y servicios.

> **Important**
>
> El permiso para realizar solicitudes de actualización debe limitarse a aplicaciones y dispositivos de confianza. Esto evita que la propiedad de estado de la sombra se cambie de forma inesperada; de lo contrario, los dispositivos y aplicaciones que usan la sombra deben diseñarse para esperar que cambien las claves de la propiedad de estado.

Uso de sombras en dispositivos, aplicaciones y otros servicios en la nube

El uso de sombras en dispositivos, aplicaciones y otros servicios en la nube requiere coherencia y coordinación entre todos ellos. El servicio AWS IoT Device Shadow almacena el estado de sombra,

envía mensajes cuando el estado de sombra cambia y responde a los mensajes que cambian de estado. Los dispositivos, las aplicaciones y otros servicios en la nube de la solución IoT deben administrar su estado y mantenerlo coherente con el estado de la sombra del dispositivo.

Los datos de estado de sombra son dinámicos y los pueden modificar los dispositivos, las aplicaciones y otros servicios en la nube con permiso para acceder a la sombra. Por esta razón, es importante considerar cómo interactuarán con la sombra cada dispositivo, aplicación y otro servicio en la nube. Por ejemplo:

- Los dispositivos deben escribir solo en la propiedad `reported` del estado de la sombra al comunicar datos de estado a la sombra.
- Las aplicaciones y otros servicios en la nube deben escribir solo en la propiedad `desired` al comunicar solicitudes de cambio de estado al dispositivo a través de la sombra.

> **Important**
>
> Los datos contenidos en un objeto de datos de sombra son independientes de los de otras sombras y otras propiedades de objetos, como los atributos de un objeto y el contenido de los mensajes MQTT que el dispositivo de un objeto podría publicar. Sin embargo, un dispositivo puede notificar los mismos datos en diferentes temas y sombras de MQTT si es necesario.
>
> Un dispositivo que admita varias sombras debe mantener la coherencia de los datos que notifica en las distintas sombras.

Orden de los mensajes

No se garantiza que los mensajes del AWS IoT servicio lleguen al dispositivo en un orden específico. La siguiente situación muestra lo que sucede en este caso.

Documento de estado inicial:

```
{
  "state": {
    "reported": {
      "color": "blue"
    }
  },
```

```
  "version": 9,
  "timestamp": 123456776
}
```

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.

Recorte de mensajes de sombra

Para reducir el tamaño de los mensajes de sombra que se envían al dispositivo, defina una regla que seleccione solo los campos que necesita el dispositivo y que después vuelva a publicar el mensaje en un tema MQTT al que el dispositivo esté escuchando.

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(3)}/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 el comodín "+" para seleccionar todos los nombres de sombra. 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(3)` toma el valor del nombre de objeto del tema original. Para obtener más información sobre la creación de reglas, consulte Reglas para AWS IoT.

Uso de sombras en dispositivos

En esta sección se describen las comunicaciones de los dispositivos con las sombras mediante mensajes MQTT, el método preferido para que los dispositivos se comuniquen con el servicio AWS IoT Device Shadow.

Las comunicaciones en sombra emulan un request/response modelo utilizando el modelo de comunicación de publicación/suscripción de MQTT. Cada acción de sombra consta de un tema de solicitud, un tema de respuesta correcta (`accepted`) y un tema de respuesta de error (`rejected`).

Si desea que las aplicaciones y servicios puedan determinar si un dispositivo está conectado, consulte Detección de un dispositivo conectado.

> **Important**
>
> Como MQTT utiliza un modelo de publish/subscribe comunicación, debe suscribirse a los temas de respuesta antes de publicar un tema de solicitud. Si no lo hace, es posible que no reciba la respuesta a la solicitud que publique.
> Si utiliza un servicio SDK para dispositivos con AWS IoTpara llamar al servicio Device Shadow APIs, se gestionará por usted.

Los ejemplos de esta sección utilizan una forma abreviada del tema, en la que *`ShadowTopicPrefix`* pueden hacer referencia a una sombra con nombre o sin nombre, tal y como se describe en esta tabla.

Las sombras pueden ser con nombre o sin nombre (clásico). Los temas utilizados por cada uno solo difieren en el prefijo del tema. Esta tabla muestra el prefijo de tema utilizado por cada tipo de sombra.

Valor de *ShadowTopicPrefix*	Tipo de sombra
`$aws/things/ thingName /shadow`	Sombra sin nombre (clásica)
`$aws/things/ thingName /shadow/name/ shadowName`	Sombra con nombre

> **Important**
>
> Asegúrese de que el uso de las sombras por parte de la aplicación o servicio sea coherente y compatible con las implementaciones correspondientes en los dispositivos. Por ejemplo, tenga en cuenta cómo se crean, actualizan y eliminan las sombras. Tenga en cuenta también cómo se tratan las actualizaciones en el dispositivo y en las aplicaciones o servicios que acceden al dispositivo a través de una sombra. El diseño debe ser claro respecto a cómo se actualiza y notifica el estado del dispositivo y cómo interactúan las aplicaciones y los servicios con el dispositivo y sus sombras.

Para crear un tema completo, seleccione el *ShadowTopicPrefix* para el tipo de sombra al que desea hacer referencia, reemplace *thingName* y *shadowName* si procede, con sus valores correspondientes y, a continuación, anexe el código auxiliar del tema como se muestra en la tabla siguiente. Recuerde que los temas distinguen entre mayúsculas y minúsculas.

Consulte Temas de sombra para obtener más información acerca de los temas reservados para las sombras.

Inicializar el dispositivo en la primera conexión a AWS IoT

Una vez que un dispositivo se registre en él AWS IoT, debería suscribirse a estos mensajes MQTT para las sombras que admite.

Topic	Significado	Acción que debe realizar un dispositivo cuando se recibe este tema
ShadowTopicPrefix / delete/accepted	Se aceptó la `delete` solicitud y AWS IoT se eliminó la sombra.	Las acciones necesarias para incorporar la sombra eliminada, como detener la publicación de actualiza ciones.
ShadowTopicPrefix / delete/rejected	La `delete` solicitud fue rechazada por AWS IoT y la sombra no se eliminó. El cuerpo del mensaje contiene la información de error.	Responda al mensaje de error en el cuerpo del mensaje.
ShadowTopicPrefix / get/accepted	La `get` solicitud fue aceptada por AWS IoT y el cuerpo del mensaje contiene el documento alternativo actual.	Las acciones necesarias para procesar el documento de estado en el cuerpo del mensaje.
ShadowTopicPrefix / get/rejected	La `get` solicitud fue rechazada por AWS IoT y el cuerpo del mensaje contiene la informaci ón del error.	Responda al mensaje de error en el cuerpo del mensaje.
ShadowTopicPrefix / update/accepted	La `update` solicitud fue aceptada por AWS IoT y el cuerpo del mensaje contiene el documento alternativo actual.	Confirme que los datos actualizados en el cuerpo del mensaje coinciden con el estado del dispositivo.
ShadowTopicPrefix / update/rejected	La `update` solicitud fue rechazada por AWS IoT y el cuerpo del mensaje contiene la información del error.	Responda al mensaje de error en el cuerpo del mensaje.

Topic	Significado	Acción que debe realizar un dispositivo cuando se recibe este tema
ShadowTopicPrefix /update/delta	El documento alternativo se actualizó mediante una solicitud dirigida a AWS IoT, y el cuerpo del mensaje contiene los cambios solicitados.	Actualice el estado del dispositivo para que coincida con el estado deseado en el cuerpo del mensaje.
ShadowTopicPrefix /update/documents	Recientemente se completó una actualización de la sombra y el cuerpo del mensaje contiene el documento de sombra actual.	Confirme que el estado actualizado en el cuerpo del mensaje coincide con el estado del dispositivo.

Después de suscribirse a los mensajes de la tabla anterior para cada sombra, el dispositivo debe probar si las sombras que admite ya se han creado publicando un tema `/get` en cada sombra. Si se recibe un mensaje `/get/accepted`, el cuerpo del mensaje contiene el documento de sombra, que el dispositivo puede utilizar para inicializar su estado. Si se recibe un mensaje `/get/rejected`, la sombra debe crearse publicando un mensaje `/update` con el estado actual del dispositivo.

Por ejemplo, supongamos que tiene un objeto `My_IoT_Thing` sin sombras, ni clásicas ni con nombre. Si ahora publica una solicitud `/get` en el tema reservado `$aws/things/My_IoT_Thing/shadow/get`, se devolverá un error sobre el tema `$aws/things/My_IoT_Thing/shadow/get/rejected`, ya que el objeto no tiene sombras. Para resolver este error, publique primero un mensaje `/update` utilizando el tema `$aws/things/My_IoT_Thing/shadow/update` con el estado actual del dispositivo, como la siguiente carga.

```
{
 "state": {
  "reported": {
   "welcome": "aws-iot",
   "color": "yellow"
  }
 }
```

```
}
```

Ahora se creará una sombra clásica para el objeto y el mensaje se publicará en el tema `$aws/things/My_IoT_Thing/shadow/update/accepted`. Si publica en el tema `$aws/things/My_IoT_Thing/shadow/get`, se devuelve una respuesta al tema `$aws/things/My_IoT_Thing/shadow/get/accepted` con el estado del dispositivo.

En el caso de las sombras con nombre, debe crear primero la sombra con nombre o publicar una actualización con el nombre de la sombra antes de utilizar la solicitud get. Por ejemplo, para crear una sombra con nombre `namedShadow1`, publique primero la información del estado del dispositivo en el tema `$aws/things/My_IoT_Thing/shadow/name/namedShadow1/update`. Para recuperar la información de estado, use la solicitud `/get` para la sombra con nombre (`$aws/things/My_IoT_Thing/shadow/name/namedShadow1/get`).

Procesar los mensajes mientras el dispositivo está conectado a AWS IoT

Mientras un dispositivo está conectado AWS IoT, puede recibir mensajes /update/delta y debe mantener el estado del dispositivo adaptado a los cambios que se producen en sus sombras de la siguiente manera:

1. Leer todos los mensajes /update/delta recibidos y sincronizar el estado del dispositivo para que coincida.
2. Publicando un mensaje /update con un cuerpo de mensaje `reported` que tenga el estado actual del dispositivo, siempre que cambie el estado del dispositivo.

Mientras un dispositivo está conectado, debe publicar estos mensajes cuando se indique.

Indicación	Topic	Carga útil
El estado del dispositivo ha cambiado.	*`ShadowTopicPrefix`* `/update`	Un documento de sombra con la propiedad `reported`.
Es posible que el dispositivo no esté sincronizado con la sombra.	*`ShadowTopicPrefix`* `/get`	(empty)
Una acción en el dispositivo indica que el dispositivo ya	*`ShadowTopicPrefix`* `/delete`	(empty)

Indicación	Topic	Carga útil
no admite una sombra; por ejemplo, cuando se quita o se reemplaza el dispositivo.		

Procesar los mensajes cuando el dispositivo se vuelve a conectar a AWS IoT

Cuando se conecta un dispositivo con una o más sombras AWS IoT, debe sincronizar su estado con el de todas las sombras que admite de la siguiente manera:

1. Leer todos los mensajes /update/delta recibidos y sincronizar el estado del dispositivo para que coincida.
2. Publicando un mensaje /update con un cuerpo de mensaje `reported` que tenga el estado actual del dispositivo.

Uso de sombras en aplicaciones y servicios

En esta sección se describe cómo interactúa una aplicación o un servicio con el servicio AWS IoT Device Shadow. En este ejemplo se supone que la aplicación o el servicio solo interactúan con la sombra y, a través de la sombra, con el dispositivo. Este ejemplo no incluye ninguna acción de administración, como la creación o eliminación de sombras.

En este ejemplo, se utiliza la API REST del servicio AWS IoT Device Shadow para interactuar con las sombras. A diferencia del ejemplo utilizado enUso de sombras en dispositivos, que utiliza un modelo de publish/subscribe communications model, this example uses the request/response comunicaciones de la API REST. Esto significa que la aplicación o el servicio deben realizar una solicitud antes de poder recibir una respuesta AWS IoT. Una desventaja de este modelo, sin embargo, es que no admite notificaciones. Si tu aplicación o servicio requieren notificaciones puntuales de los cambios de estado del dispositivo, considera usar los protocolos MQTT o MQTT sobre WSS, que admiten el modelo de publish/subscribe comunicación, tal y como se describe en. Uso de sombras en dispositivos

> **Important**
>
> Asegúrese de que el uso de las sombras por parte de la aplicación o servicio sea coherente y compatible con las implementaciones correspondientes en los dispositivos. Tenga en cuenta, por ejemplo, cómo se crean, actualizan y eliminan las sombras, y cómo se tratan las actualizaciones en el dispositivo y en las aplicaciones o servicios que acceden a la sombra. El diseño debe especificar claramente cómo se actualiza y notifica el estado del dispositivo, y cómo interactúan las aplicaciones y los servicios con el dispositivo y sus sombras.

La URL de la API REST para sombras con nombre es:

```
https://endpoint/things/thingName/shadow?name=shadowName
```

y para una sombra sin nombre:

```
https://endpoint/things/thingName/shadow
```

donde:

punto de conexión

El punto de enlace que devuelve el comando de la CLI:

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

thingName

El nombre del objeto al que pertenece la sombra

shadowName

El nombre de la sombra con nombre. Este parámetro no se utiliza con sombras sin nombre.

Inicializar la aplicación o el servicio al conectarse a AWS IoT

Cuando la aplicación se conecte por primera vez AWS IoT, debería enviar una solicitud HTTP GET a URLs las sombras que utiliza para obtener el estado actual de las sombras que utiliza. Esto le permite sincronizar la aplicación o el servicio con la sombra.

El estado del procesamiento cambia mientras la aplicación o el servicio están conectados a AWS IoT

Mientras la aplicación o el servicio están conectados AWS IoT, pueden consultar el estado actual de forma periódica enviando una solicitud HTTP GET URLs desde las sombras que utiliza.

Cuando un usuario final interactúa con la aplicación o el servicio para cambiar el estado del dispositivo, la aplicación o el servicio puede enviar una solicitud HTTP POST a una URLs de las sombras que utiliza para actualizar el `desired` estado de la sombra. Esta solicitud devuelve el cambio que se aceptó, pero es posible que tenga que sondear la sombra realizando solicitudes HTTP GET hasta que el dispositivo haya actualizado la sombra con su nuevo estado.

Detección de un dispositivo conectado

Para determinar si un dispositivo está conectado actualmente, incluya una propiedad `connected` en el documento de sombra y use un mensaje MQTT Last Will and Testament (LWT) para establecer la propiedad `connected` en `false` si un dispositivo está desconectado debido a un error.

> **Note**
>
> El servicio AWS IoT Device Shadow ignora los mensajes LWT de MQTT enviados a temas AWS IoT reservados (temas que comienzan por $). Sin embargo, los procesan los clientes suscritos y el motor de AWS IoT reglas, por lo que tendrá que crear un mensaje LWT que se envíe a un tema no reservado y una regla que vuelva a publicar el mensaje LWT de MQTT como un mensaje de actualización paralelo para el tema de actualización reservado de la sombra. *`ShadowTopicPrefix`*`/update`

Para enviar al servicio Device Shadow un mensaje LWT

1. Cree una regla que vuelva a publicar el mensaje MQTT LWT en el tema reservado. El siguiente ejemplo es una regla que escucha mensajes sobre el tema `my/things/myLightBulb/update` y lo vuelve 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"
            }
        }
     ]
   }
}
```

2. Cuando el dispositivo se conecta a AWS IoT, registra un mensaje de LWT en un tema no reservado para que lo reconozca la regla de republicación. En este ejemplo, ese tema es `my/things/myLightBulb/update` y establece la propiedad conectada en `false`.

```
{
    "state": {
        "reported": {
            "connected":"false"
        }
    }
}
```

3. Después de conectarse, el dispositivo publica un mensaje en su tema de actualización de sombra `$aws/things/myLightBulb/shadow/update`, para notificar su estado actual, que incluye establecer su propiedad `connected` en `true`.

```
{
     "state": {
        "reported": {
            "connected":"true"
        }
    }
}
```

4. Antes de que el dispositivo se desconecte correctamente, publica un mensaje en su tema de actualización de sombras `$aws/things/myLightBulb/shadow/update`, para notificar su estado más reciente, que incluye establecer su propiedad `connected` en `false`.

```
{
    "state": {
        "reported": {
```

```
            "connected":"false"
        }
    }
}
```

5. Si el dispositivo se desconecta debido a un error, el agente de mensajes publica el AWS IoT mensaje LWT del dispositivo en nombre del dispositivo. La regla de republicación detecta este mensaje y publica el mensaje de actualización de sombra para actualizar la propiedad `connected` de la sombra del dispositivo.

> **Note**
>
> Dada la naturaleza asíncrona del procesamiento de desconexiones, no se garantiza que los mensajes de LWT se envíen en orden durante la reconexión. Le recomendamos que utilice los eventos del ciclo de vida para mejorar la precisión de la detección del estado de la conectividad, ya que estos eventos proporcionan atributos para out-of-order gestionarlos.

Simulación de comunicaciones del servicio Device Shadow

En este tema se muestra cómo el servicio Device Shadow actúa como intermediario y permite que los dispositivos y aplicaciones utilicen una sombra para actualizar, almacenar y recuperar el estado de un dispositivo.

Para demostrar la interacción descrita en este tema y explorarla más a fondo, necesitará un Cuenta de AWS sistema en el que pueda ejecutarla AWS CLI. Si no dispone de ello, aún puede ver la interacción en los ejemplos de código.

En este ejemplo, la AWS IoT consola representa el dispositivo. AWS CLI Representa la aplicación o el servicio que accede al dispositivo a través de la sombra. La AWS CLI interfaz es muy similar a la API con la que una aplicación puede comunicarse AWS IoT. El dispositivo de este ejemplo es una bombilla inteligente y la aplicación muestra el estado de la bombilla y puede cambiar el estado de la bombilla.

Configuración de la simulación

Estos procedimientos inicializan la simulación abriendo la consola de AWS IoT, que simula su dispositivo, y la ventana de línea de comandos que simula su aplicación.

Para configurar el entorno de simulación

1. Necesitarás ejecutar Cuenta de AWS los ejemplos de este tema por tu cuenta. Si no tiene uno Cuenta de AWS, cree uno, tal y como se describe enConfigurar Cuenta de AWS.
2. Abra la consola de AWS IoT y, en el menú de la izquierda, elija Probar para abrir el cliente MQTT.
3. En otra ventana, abra una ventana de terminal en un sistema que tenga instalada la AWS CLI .

Debería tener dos ventanas abiertas: una con la AWS IoT consola en la página de pruebas y otra con una línea de comandos.

Inicializar el dispositivo

En esta simulación, trabajaremos con un objeto llamado mySimulatedThing, y su sombra llamada SimShadow1.

Creación de un objeto y de su política de IoT

Para crear un objeto, haga lo siguiente en la consola AWS IoT :

1. Seleccione Administrar y Objetos.
2. Haga clic en el botón Crear si hay cosas en la lista; de lo contrario, haga clic en Registrar una sola cosa para crear una sola AWS IoT cosa.
3. Introduzca el nombre `mySimulatedThing`, deje los demás ajustes con los valores predeterminados y haga clic en Siguiente.
4. Utilice la creación de certificados en un clic para generar los certificados que autenticarán la conexión del dispositivo a AWS IoT. Haga clic en Activar para activar el certificado.
5. Puede asociar la política `My_IoT_Policy` que le daría permiso al dispositivo para publicar en los temas reservados de MQTT y suscribirse a estos. Para obtener pasos más detallados sobre cómo crear una AWS IoT cosa y cómo crear esta política, consulteCrear un objeto.

Creación de una sombra con nombre para el objeto

Puede crear una sombra con nombre para un objeto publicando una solicitud de actualización en el tema `$aws/things/mySimulatedThing/shadow/name/simShadow1/update`, tal y como se describe a continuación.

O para crear una sombra con nombre:

1. En la consola AWS IoT , seleccione el objeto que desee de la lista de objetos y seleccione Sombras.
2. Seleccione Agregar una sombra, introduzca el nombre `simShadow1` y seleccione Crear para agregar la sombra con nombre.

Suscripción a temas reservados de MQTT y publicación en estos

En la consola, suscríbase a los temas de sombras MQTT reservados. Estos temas son las respuestas a las acciones `get`, `update` y `delete` para que el dispositivo esté listo para recibir las respuestas después de publicar una acción.

Para suscribirse a un tema MQTT en el cliente MQTT

1. En el Cliente de MQTT, elija Suscribirse a un tema.
2. Introduzca los temas `get`, `update` y `delete` a los que desea suscribirse. Copie un tema a la vez de la siguiente lista, péguelo en el campo Filtro de temas y haga clic en Suscribirse. Los temas deberían aparecer en Suscripciones.

   - `$aws/things/mySimulatedThing/shadow/name/simShadow1/delete/accepted`
   - `$aws/things/mySimulatedThing/shadow/name/simShadow1/delete/rejected`
   - `$aws/things/mySimulatedThing/shadow/name/simShadow1/get/accepted`
   - `$aws/things/mySimulatedThing/shadow/name/simShadow1/get/rejected`
   - `$aws/things/mySimulatedThing/shadow/name/simShadow1/update/accepted`
   - `$aws/things/mySimulatedThing/shadow/name/simShadow1/update/rejected`
   - `$aws/things/mySimulatedThing/shadow/name/simShadow1/update/delta`
   - `$aws/things/mySimulatedThing/shadow/name/simShadow1/update/documents`

   En este punto, el dispositivo simulado está listo para recibir los temas a medida que los publica AWS IoT.

Para publicar en un tema MQTT, en el Cliente de MQTT

Después de que un dispositivo se haya inicializado y suscrito a los temas de respuesta, debe consultar las sombras que admite. Esta simulación solo admite una sombra, la sombra que soporta un objeto llamado SimShadow1. mySimulatedThing

Para obtener el estado actual de la sombra desde el Cliente MQTT

1. En el cliente MQTT, elija Publicar en un tema.
2. En Publicar, introduzca el siguiente tema y elimine cualquier contenido de la ventana del cuerpo del mensaje, debajo del lugar en el que ha introducido el tema que desea obtener. A continuación, seleccione Publicar en tema para publicar la solicitud `$aws/things/mySimulatedThing/shadow/name/simShadow1/get`.

   Si no ha creado la sombra con nombre `simShadow1`, recibirá un mensaje en el tema `$aws/things/mySimulatedThing/shadow/name/simShadow1/get/rejected` y el `code` será `404`, como se ve en este ejemplo en el que no se ha creado una sombra; a continuación, vamos a crearla.

```
{
  "code": 404,
  "message": "No shadow exists with name: 'simShadow1'"
}
```

Para crear una sombra con el estado actual del dispositivo

1. En Cliente de MQTT, elija Publicar en tema e introduzca este tema:

```
$aws/things/mySimulatedThing/shadow/name/simShadow1/update
```

2. En la ventana del cuerpo del mensaje debajo donde introdujo el tema, escriba este documento de sombra para mostrar que el dispositivo está notificando su ID y su color actual en valores RGB. Seleccione Publicar para publicar la solicitud.

```
{
  "state": {
    "reported": {
      "ID": "SmartLamp21",
      "ColorRGB": [
        128,
        128,
        128
      ]
    }
  },
  "clientToken": "426bfd96-e720-46d3-95cd-014e3ef12bb6"
```

```
}
```

A continuación, se describe el significado de los mensajes recibidos en el tema.

- `$aws/things/mySimulatedThing/shadow/name/simShadow1/update/accepted`: significa que la sombra se ha creado y que el cuerpo del mensaje contiene el documento de sombra actual.
- `$aws/things/mySimulatedThing/shadow/name/simShadow1/update/rejected`: revise el error en el cuerpo del mensaje.
- `$aws/things/mySimulatedThing/shadow/name/simShadow1/get/accepted`: la sombra ya existe y el cuerpo del mensaje tiene el estado de sombra actual, como en este ejemplo. Con esto, puedes configurar su dispositivo o confirmar que coincide con el estado de sombra.

```
{
  "state": {
    "reported": {
      "ID": "SmartLamp21",
      "ColorRGB": [
        128,
        128,
        128
      ]
    }
  },
  "metadata": {
    "reported": {
      "ID": {
        "timestamp": 1591140517
      },
      "ColorRGB": [
        {
          "timestamp": 1591140517
        },
        {
          "timestamp": 1591140517
        },
        {
          "timestamp": 1591140517
        }
      ]
```

```
    }
  },
  "version": 3,
  "timestamp": 1591140517,
  "clientToken": "426bfd96-e720-46d3-95cd-014e3ef12bb6"
}
```

Enviar una actualización desde la aplicación

En esta sección se utiliza AWS CLI para demostrar cómo una aplicación puede interactuar con una sombra.

Para obtener el estado actual de la sombra, utilice el AWS CLI

Desde la línea de comandos, escriba este comando:

```
aws iot-data get-thing-shadow --thing-name mySimulatedThing --shadow-name simShadow1 /
dev/stdout
```

En las plataformas Windows, puede utilizar `con` en lugar de `/dev/stdout`.

```
aws iot-data get-thing-shadow --thing-name mySimulatedThing --shadow-name simShadow1
 con
```

Dado que la sombra existe y la ha inicializado el dispositivo para reflejar su estado actual, debe devolver el siguiente documento de sombra.

```
{
  "state": {
    "reported": {
      "ID": "SmartLamp21",
      "ColorRGB": [
        128,
        128,
        128
      ]
    }
  },
  "metadata": {
    "reported": {
```

```
      "ID": {
        "timestamp": 1591140517
      },
      "ColorRGB": [
        {
          "timestamp": 1591140517
        },
        {
          "timestamp": 1591140517
        },
        {
          "timestamp": 1591140517
        }
      ]
    }
  },
  "version": 3,
  "timestamp": 1591141111
}
```

La aplicación puede usar esta respuesta para inicializar su representación del estado del dispositivo.

Si la aplicación actualiza el estado, como cuando un usuario final cambia el color de nuestra bombilla inteligente a amarillo, la aplicación enviará un comando update-thing-shadow. Este comando corresponde a la API REST `UpdateThingShadow`.

Para actualizar una sombra desde una aplicación

Desde la línea de comandos, escriba este comando:

AWS CLI v2.x

```
aws iot-data update-thing-shadow --thing-name mySimulatedThing --shadow-name
 simShadow1 \
    --cli-binary-format raw-in-base64-out \
    --payload '{"state":{"desired":{"ColorRGB":
[255,255,0]}},"clientToken":"21b21b21-bfd2-4279-8c65-e2f697ff4fab"}' /dev/stdout
```

AWS CLI v1.x

```
aws iot-data update-thing-shadow --thing-name mySimulatedThing --shadow-name
 simShadow1 \
```

```
    --payload '{"state":{"desired":{"ColorRGB":
[255,255,0]}},"clientToken":"21b21b21-bfd2-4279-8c65-e2f697ff4fab"}' /dev/stdout
```

Si tiene éxito, este comando debe devolver el siguiente documento de sombra.

```
{
  "state": {
    "desired": {
      "ColorRGB": [
        255,
        255,
        0
      ]
    }
  },
  "metadata": {
    "desired": {
      "ColorRGB": [
        {
          "timestamp": 1591141596
        },
        {
          "timestamp": 1591141596
        },
        {
          "timestamp": 1591141596
        }
      ]
    }
  },
  "version": 4,
  "timestamp": 1591141596,
  "clientToken": "21b21b21-bfd2-4279-8c65-e2f697ff4fab"
}
```

Responder a la actualización en el dispositivo

Volviendo al cliente MQTT de la AWS consola, debería ver los mensajes AWS IoT publicados que reflejan el comando de actualización emitido en la sección anterior.

Para ver los mensajes de actualización en el Cliente MQTT

En el cliente MQTT, elija $ aws/things/mySimulatedThing/shadow/name/simShadow1/update/delta en la columna Suscripciones. Si el nombre del tema está truncado, puede ponerlo en pausa para ver el tema completo. En el registro de temas para este tema, debería ver un mensaje `/delta` similar a este.

```
{
  "version": 4,
  "timestamp": 1591141596,
  "state": {
    "ColorRGB": [
      255,
      255,
      0
    ]
  },
  "metadata": {
    "ColorRGB": [
      {
        "timestamp": 1591141596
      },
      {
        "timestamp": 1591141596
      },
      {
        "timestamp": 1591141596
      }
    ]
  },
  "clientToken": "21b21b21-bfd2-4279-8c65-e2f697ff4fab"
}
```

El dispositivo procesaría el contenido de este mensaje para establecer el estado del dispositivo para que coincida con el estado `desired` del mensaje.

Una vez que el dispositivo actualice el estado para que coincida con el `desired` estado del mensaje, debe devolver el nuevo estado informado AWS IoT mediante la publicación de un mensaje de actualización. Este procedimiento simula esto en el cliente MQTT.

Para actualizar la sombra desde el dispositivo

1. En el cliente MQTT, elija Publicar en un tema.

2. En el campo de temas situado encima de la ventana del cuerpo del mensaje, introduzca el tema de la sombra seguido de la acción `/update` `$aws/things/mySimulatedThing/shadow/name/simShadow1/update`; en el cuerpo del mensaje, introduzca este documento de sombra actualizado, que describe el estado actual del dispositivo. Elija Publicar para publicar el estado actualizado del dispositivo.

```
{
  "state": {
    "reported": {
      "ColorRGB": [255,255,0]
      }
  },
  "clientToken": "a4dc2227-9213-4c6a-a6a5-053304f60258"
}
```

Si el mensaje se ha recibido correctamente AWS IoT, debería aparecer una nueva respuesta en el registro de aws/things/mySimulatedThing/shadow/name/simShadow1/update/accepted mensajes $ del cliente MQTT con el estado de sombra actual, como en este ejemplo.

```
{
  "state": {
    "reported": {
      "ColorRGB": [
        255,
        255,
        0
      ]
    }
  },
  "metadata": {
    "reported": {
      "ColorRGB": [
        {
          "timestamp": 1591142747
        },
        {
          "timestamp": 1591142747
        },
        {
          "timestamp": 1591142747
        }
```

```
      ]
    }
  },
  "version": 5,
  "timestamp": 1591142747,
  "clientToken": "a4dc2227-9213-4c6a-a6a5-053304f60258"
}
```

Si se actualiza correctamente el estado registrado del dispositivo, también se AWS IoT envía una descripción completa del estado de sombra en un mensaje dedicado al `update/documents` tema, como el cuerpo del mensaje resultante de la actualización oculta realizada por el dispositivo en el procedimiento anterior.

```
{
  "previous": {
    "state": {
      "desired": {
        "ColorRGB": [
          255,
          255,
          0
        ]
      },
      "reported": {
        "ID": "SmartLamp21",
        "ColorRGB": [
          128,
          128,
          128
        ]
      }
    },
    "metadata": {
      "desired": {
        "ColorRGB": [
          {
            "timestamp": 1591141596
          },
          {
            "timestamp": 1591141596
          },
          {
```

```
            "timestamp": 1591141596
          }
        ]
      },
      "reported": {
        "ID": {
          "timestamp": 1591140517
        },
        "ColorRGB": [
          {
            "timestamp": 1591140517
          },
          {
            "timestamp": 1591140517
          },
          {
            "timestamp": 1591140517
          }
        ]
      }
    },
    "version": 4
  },
  "current": {
    "state": {
      "desired": {
        "ColorRGB": [
          255,
          255,
          0
        ]
      },
      "reported": {
        "ID": "SmartLamp21",
        "ColorRGB": [
          255,
          255,
          0
        ]
      }
    },
    "metadata": {
      "desired": {
        "ColorRGB": [
```

```
          {
            "timestamp": 1591141596
          },
          {
            "timestamp": 1591141596
          },
          {
            "timestamp": 1591141596
          }
        ]
      },
      "reported": {
        "ID": {
          "timestamp": 1591140517
        },
        "ColorRGB": [
          {
            "timestamp": 1591142747
          },
          {
            "timestamp": 1591142747
          },
          {
            "timestamp": 1591142747
          }
        ]
      }
    },
    "version": 5
  },
  "timestamp": 1591142747,
  "clientToken": "a4dc2227-9213-4c6a-a6a5-053304f60258"
}
```

Ver la actualización en la aplicación

La aplicación ahora puede consultar la sombra del estado actual según haya notificado el dispositivo.

Para obtener el estado actual de la sombra, utilice el AWS CLI

1. Desde la línea de comandos, escriba este comando:

```
aws iot-data get-thing-shadow --thing-name mySimulatedThing --shadow-name
 simShadow1 /dev/stdout
```

En las plataformas Windows, puede utilizar `con` en lugar de `/dev/stdout`.

```
aws iot-data get-thing-shadow --thing-name mySimulatedThing --shadow-name
 simShadow1 con
```

2. Dado que el dispositivo acaba de actualizar la sombra para reflejar su estado actual, debe devolver el siguiente documento de sombra.

```
{
  "state": {
    "desired": {
      "ColorRGB": [
        255,
        255,
        0
      ]
    },
    "reported": {
      "ID": "SmartLamp21",
      "ColorRGB": [
        255,
        255,
        0
      ]
    }
  },
  "metadata": {
    "desired": {
      "ColorRGB": [
        {
          "timestamp": 1591141596
        },
        {
          "timestamp": 1591141596
        },
        {
          "timestamp": 1591141596
        }
```

```
      ]
    },
    "reported": {
      "ID": {
        "timestamp": 1591140517
      },
      "ColorRGB": [
        {
          "timestamp": 1591142747
        },
        {
          "timestamp": 1591142747
        },
        {
          "timestamp": 1591142747
        }
      ]
    }
  },
  "version": 5,
  "timestamp": 1591143269
}
```

Más allá de la simulación

Experimente con la interacción entre la AWS CLI (que representa a la aplicación) y la consola (que representa al dispositivo) a fin de modelar su solución de IoT.

Interacción con sombras

En este tema se describen los mensajes asociados a cada uno de los tres métodos que proporciona AWS IoT para trabajar con sombras. Estos métodos incluyen lo siguiente:

`UPDATE`

Crea una sombra si no existe o actualiza el contenido de una sombra existente con la información de estado proporcionada en el cuerpo del mensaje. AWS IoT registra una marca temporal con cada actualización para indicar cuándo se actualizó el estado por última vez. Cuando el estado de la sombra cambia, AWS IoT envía `/delta` mensajes a todos los suscriptores de MQTT con la diferencia entre los estados `desired` y los `reported` estados. Los dispositivos o aplicaciones

que reciben un mensaje `/delta` pueden realizar acciones en función de la diferencia. 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 un documento de sombra actual que contiene el estado completo de la sombra, incluidos los metadatos.

`DELETE`

Elimina la sombra de dispositivo y todo su contenido.

No puede restaurar un documento sombra de dispositivo eliminado, pero puede crear un nuevo documento con el nombre de un documento sombra de dispositivo eliminado. Si crea un documento de sombra de dispositivo con el mismo nombre que uno que se haya eliminado en las últimas 48 horas, el número de versión del nuevo documento de sombra de dispositivo será el mismo que el del documento eliminado. Si se ha eliminado un documento de sombra de dispositivo desde hace más de 48 horas, el número de versión de un nuevo documento de sombra de dispositivo con el mismo nombre será 0.

Compatibilidad del protocolo

AWS IoT admite MQTT y una API REST a través de protocolos HTTPS para interactuar con las sombras. AWS IoT proporciona un conjunto de temas de solicitud y respuesta reservados para las acciones de publicación y suscripción de MQTT. Los dispositivos y las aplicaciones deben suscribirse a los temas de respuesta antes de publicarlos en un tema de solicitud para obtener información sobre cómo AWS IoT se gestionó la solicitud. Para obtener más información, consulte Temas MQTT de sombra de dispositivo y API REST de sombra de dispositivo.

Estado de solicitud y notificación

Al diseñar su solución de IoT utilizando AWS IoT y sombras, debe determinar las aplicaciones o dispositivos que solicitarán cambios y los que los implementarán. Normalmente, un dispositivo implementa y notifica los cambios a la sombra y las aplicaciones y los servicios responden y solicitan cambios en la sombra. Su solución podría ser diferente, pero en los ejemplos de este tema se supone que la aplicación cliente o el servicio solicita cambios en la sombra y el dispositivo realiza los cambios y los notifica de nuevo a la sombra.

Actualización de la sombra

La aplicación o el servicio pueden actualizar el estado de una sombra mediante la API UpdateThingShadow o publicando en el tema /update. Las actualizaciones afectan únicamente a los campos especificados en la solicitud.

Actualización de una sombra cuando un cliente solicita un cambio de estado

Cuando un cliente solicita un cambio de estado en una sombra mediante el protocolo MQTT

1. El cliente debe tener un documento de sombra actual para que pueda identificar las propiedades que se van a cambiar. Consulte la acción /get para ver cómo obtener el documento de sombra actual.
2. El cliente se suscribe a los siguientes temas de MQTT:
   - `$aws/things/thingName/shadow/name/shadowName/update/accepted`
   - `$aws/things/thingName/shadow/name/shadowName/update/rejected`
   - `$aws/things/thingName/shadow/name/shadowName/update/delta`
   - `$aws/things/thingName/shadow/name/shadowName/update/documents`
3. El cliente publica un tema de solicitud de `$aws/things/thingName/shadow/name/shadowName/update` con un documento de estado que contiene el estado deseado de la sombra. Solo las propiedades que se van a cambiar deben incluirse en el documento. Este es un ejemplo de un documento con el estado deseado.

   ```
   {
     "state": {
       "desired": {
         "color": {
           "r": 10
         },
         "engine": "ON"
       }
     }
   }
   ```

4. Si la solicitud de actualización es válida, AWS IoT actualiza el estado deseado de forma oculta y publica mensajes sobre los siguientes temas:
   - `$aws/things/thingName/shadow/name/shadowName/update/accepted`

- `$aws/things/thingName/shadow/name/shadowName/update/delta`

El mensaje `/update/accepted` incluye un documento de sombra Documento de estado de la respuesta /aceptado y el mensaje `/update/delta` incluye un documento de sombra Documento de estado de la respuesta /delta.

5. Si la solicitud de actualización no es válida, AWS IoT publica un mensaje con el `$aws/things/thingName/shadow/name/shadowName/update/rejected` tema junto con un documento Documento de respuesta de error paralelo en el que se describe el error.

Cuando un cliente solicita un cambio de estado en una sombra mediante el uso de la API

1. El cliente llama a la API `UpdateThingShadow` con un documento de estado Documento de estado de la solicitud como cuerpo del mensaje.
2. Si la solicitud era válida, AWS IoT devuelve un código HTTP de respuesta correcta y un documento Documento de estado de la respuesta /aceptado alternativo como cuerpo del mensaje de respuesta.

   AWS IoT también publicará un mensaje MQTT `$aws/things/thingName/shadow/name/shadowName/update/delta` sobre el tema con un documento Documento de estado de la respuesta /delta paralelo para todos los dispositivos o clientes que estén suscritos a él.
3. Si la solicitud no era válida, AWS IoT devuelve un código de respuesta a un error HTTP Documento de respuesta de error como cuerpo del mensaje de respuesta.

Cuando el dispositivo recibe el estado `/desired` en el tema `/update/delta`, realiza los cambios deseados en el dispositivo. A continuación, envía un mensaje al tema `/update` para notificar su estado actual a la sombra.

Actualización de una sombra cuando un dispositivo notifica su estado actual

Cuando un dispositivo notifica su estado actual a la sombra mediante el protocolo MQTT

1. El dispositivo debe suscribirse a estos temas de MQTT antes de actualizar la sombra:

   - `$aws/things/thingName/shadow/name/shadowName/update/accepted`
   - `$aws/things/thingName/shadow/name/shadowName/update/rejected`
   - `$aws/things/thingName/shadow/name/shadowName/update/delta`

- `$aws/things/thingName/shadow/name/shadowName/update/documents`

2. El dispositivo notifica su estado actual publicando un mensaje en el tema `$aws/things/thingName/shadow/name/shadowName/update` que notifica estado actual, como en este ejemplo.

```
{
    "state": {
        "reported" : {
            "color" : { "r" : 10 },
            "engine" : "ON"
        }
    }
}
```

3. Si AWS IoT acepta la actualización, publica un mensaje `$aws/things/thingName/shadow/name/shadowName/update/accepted` sobre los temas con un documento Documento de estado de la respuesta /aceptado paralelo.
4. Si la solicitud de actualización no es válida, AWS IoT publica un mensaje con el `$aws/things/thingName/shadow/name/shadowName/update/rejected` tema junto con un documento Documento de respuesta de error alternativo que describa el error.

Cuando un dispositivo notifica su estado actual a la sombra mediante la API

1. El dispositivo llama a la API `UpdateThingShadow` con un documento de estado Documento de estado de la solicitud como cuerpo del mensaje.
2. Si la solicitud era válida, AWS IoT actualiza la sombra y devuelve un código HTTP de respuesta correcta con un documento Documento de estado de la respuesta /aceptado paralelo como cuerpo del mensaje de respuesta.

   AWS IoT también publicará un mensaje MQTT `$aws/things/thingName/shadow/name/shadowName/update/delta` sobre el tema junto con un documento Documento de estado de la respuesta /delta alternativo para todos los dispositivos o clientes que estén suscritos a él.
3. Si la solicitud no era válida, AWS IoT devuelve un código de respuesta a un error HTTP Documento de respuesta de error como cuerpo del mensaje de respuesta.

Bloqueo optimista

Puede utilizar la versión del documento de estado para asegurarse de que actualiza la versión más reciente del documento de sombra de un dispositivo. 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. El código de respuesta de conflicto también puede aparecer en cualquier API que modifique a `ThingShadow`, lo que incluye `DeleteThingShadow`.

Por ejemplo:

Documento inicial:

```
{
  "state": {
    "desired": {
      "colors": [
        "RED",
        "GREEN",
        "BLUE"
      ]
    }
  },
  "version": 10
}
```

Actualización: (la versión no coincide; se rechazará esta solicitud)

```
{
  "state": {
    "desired": {
      "colors": [
        "BLUE"
      ]
    }
  },
  "version": 9
}
```

Resultado:

```
{
```

```
  "code": 409,
  "message": "Version conflict",
  "clientToken": "426bfd96-e720-46d3-95cd-014e3ef12bb6"
}
```

Actualización: (la versión coincide; esta solicitud se aceptará)

```
{
  "state": {
    "desired": {
      "colors": [
        "BLUE"
      ]
    }
  },
  "version": 10
}
```

Estado final:

```
{
  "state": {
    "desired": {
      "colors": [
        "BLUE"
      ]
    }
  },
  "version": 11
}
```

Recuperación de un documento de sombra

Puede recuperar un documento de sombra utilizando la API GetThingShadow o suscribiéndose y publicando en el tema /get. Esto recupera un documento de sombra completo, incluido cualquier delta entre los estados `desired` y `reported`. El procedimiento para esta tarea es el mismo si el dispositivo o un cliente está realizando la solicitud.

Para recuperar un documento de sombra mediante el protocolo MQTT

1. El dispositivo o cliente debe suscribirse a estos temas de MQTT antes de actualizar la sombra:

- `$aws/things/thingName/shadow/name/shadowName/get/accepted`
- `$aws/things/thingName/shadow/name/shadowName/get/rejected`

2. El dispositivo o cliente publica un mensaje en el tema `$aws/things/thingName/shadow/name/shadowName/get` con un cuerpo de mensaje vacío.
3. Si la solicitud se realiza correctamente, AWS IoT publica un mensaje en el `$aws/things/thingName/shadow/name/shadowName/get/accepted` tema con una Documento de estado de la respuesta /aceptado en el cuerpo del mensaje.
4. Si la solicitud no era válida, AWS IoT publica un mensaje en el `$aws/things/thingName/shadow/name/shadowName/get/rejected` tema con una Documento de respuesta de error en el cuerpo del mensaje.

Para recuperar un documento de sombra mediante una API REST

1. El dispositivo o cliente llama a la API `GetThingShadow` con un cuerpo de mensaje vacío.
2. Si la solicitud es válida, AWS IoT devuelve un código HTTP de respuesta correcta con un documento Documento de estado de la respuesta /aceptado paralelo como cuerpo del mensaje de respuesta.
3. Si la solicitud no es válida, AWS IoT devuelve un código de respuesta de error HTTP an Documento de respuesta de error como cuerpo del mensaje de respuesta.

Eliminación de datos de sombra

Hay dos formas de eliminar datos de sombra: puede eliminar propiedades específicas en el documento de sombra y puede eliminar la sombra por completo.

- Para eliminar propiedades específicas de una sombra, actualice la sombra; sin embargo, establezca el valor de las propiedades que desea eliminar en `null`. Los campos con un valor `null` se quitan del documento de sombra.
- Para eliminar toda la sombra, use la API DeleteThingShadow o publique en el tema /delete.

> **Note**
>
> Al eliminar una sombra, no se restablece su número de versión a cero de forma inmediata. Se restablecerá a cero después de 48 horas.

Eliminación de una propiedad de un documento de sombra

Para eliminar una propiedad de una sombra mediante el protocolo MQTT

1. El dispositivo o cliente debe tener un documento de sombra actual para que pueda identificar las propiedades que se van a cambiar. Consulte Recuperación de un documento de sombra para obtener más información sobre cómo obtener el documento de sombra actual.
2. El dispositivo o cliente se suscribe a estos temas de MQTT:
   - `$aws/things/thingName/shadow/name/shadowName/update/accepted`
   - `$aws/things/thingName/shadow/name/shadowName/update/rejected`
3. El dispositivo o cliente publica un tema de solicitud `$aws/things/thingName/shadow/name/shadowName/update` con un documento de estado que asigna valores `null` a las propiedades de la sombra que se va a eliminar. Solo las propiedades que se van a cambiar deben incluirse en el documento. Este es un ejemplo de un documento que elimina la propiedad `engine`.

   ```
   {
     "state": {
       "desired": {
         "engine": null
       }
     }
   }
   ```
4. Si la solicitud de actualización es válida, AWS IoT elimina las propiedades especificadas en la sombra y publica un mensaje con el `$aws/things/thingName/shadow/name/shadowName/update/accepted` tema con un documento Documento de estado de la respuesta /aceptado paralelo en el cuerpo del mensaje.
5. Si la solicitud de actualización no es válida, AWS IoT publique un mensaje con el `$aws/things/thingName/shadow/name/shadowName/update/rejected` tema junto con un documento Documento de respuesta de error alternativo que describa el error.

Para eliminar una propiedad de una sombra mediante la API REST

1. El dispositivo o cliente llama a la API `UpdateThingShadow` con un Documento de estado de la solicitud que asigna valores `null` a las propiedades de la sombra para eliminar. Incluya solo las propiedades que desee eliminar en el documento. Este es un ejemplo de un documento que elimina la propiedad `engine`.

   ```
   {
     "state": {
       "desired": {
         "engine": null
       }
     }
   }
   ```

2. Si la solicitud era válida, AWS IoT devuelve un código HTTP de respuesta correcta y un documento Documento de estado de la respuesta /aceptado alternativo como cuerpo del mensaje de respuesta.
3. Si la solicitud no era válida, AWS IoT devuelve un código de respuesta de error HTTP an Documento de respuesta de error como cuerpo del mensaje de respuesta.

Eliminación de una sombra

Estos son algunos factores que debemos tener en cuenta al eliminar la sombra de un dispositivo.

- Al establecer el estado de sombra del dispositivo en `null` no se elimina la sombra. La versión de sombra se incrementará en la próxima actualización.
- La eliminación de la sombra de un dispositivo no elimina el objeto. La eliminación de un objeto no elimina la sombra del dispositivo correspondiente.
- Al eliminar una sombra, no se restablece su número de versión a cero de forma inmediata. Se restablecerá a cero después de 48 horas.

Para eliminar una sombra mediante el protocolo MQTT

1. El dispositivo o cliente se suscribe a estos temas de MQTT:
   - `$aws/things/thingName/shadow/name/shadowName/delete/accepted`
   - `$aws/things/thingName/shadow/name/shadowName/delete/rejected`

2. El dispositivo o cliente publica un `$aws/things/thingName/shadow/name/shadowName/delete` con un búfer de mensaje vacío.
3. Si la solicitud de eliminación es válida, AWS IoT elimina la sombra y publica un mensaje con el `$aws/things/thingName/shadow/name/shadowName/delete/accepted` tema y un documento Documento de estado de la respuesta /aceptado paralelo abreviado en el cuerpo del mensaje. Este es un ejemplo del mensaje de eliminación aceptado:

```
{
  "version": 4,
  "timestamp": 1591057529
}
```

4. Si la solicitud de actualización no es válida, AWS IoT publique un mensaje con el `$aws/things/thingName/shadow/name/shadowName/delete/rejected` tema junto con un documento Documento de respuesta de error alternativo que describa el error.

Para eliminar una sombra mediante la API REST

1. El dispositivo o cliente llama a la API `DeleteThingShadow` con un búfer de mensaje vacío.
2. Si la solicitud era válida, AWS IoT devuelve un código HTTP de respuesta correcta Documento de estado de la respuesta /aceptado y un documento Documento de estado de la respuesta / aceptado alternativo abreviado en el cuerpo del mensaje. Este es un ejemplo del mensaje de eliminación aceptado:

```
{
  "version": 4,
  "timestamp": 1591057529
}
```

3. Si la solicitud no era válida, AWS IoT devuelve un código de respuesta de error HTTP an Documento de respuesta de error como cuerpo del mensaje de respuesta.

API REST de sombra de dispositivo

Una sombra expone el siguiente URI para actualizar la información de estado:

```
https://account-specific-prefix-ats.iot.region.amazonaws.com/things/thingName/shadow
```

El punto final es específico de su Cuenta de AWS. Para buscar el punto de conexión, puede hacer lo siguiente:

- Utilice el comando describe-endpoint desde la AWS CLI.
- Utilice la configuración AWS IoT de la consola. En Configuración, el punto de conexión aparece en Punto de conexión personalizado
- Usa la página de detalles de la AWS IoT consola. En la consola de :
  1. Abra Administrar y, en Administrar, seleccione Objetos.
  2. En la lista de objetos, seleccione el objeto para el que desea obtener la URI del punto de conexión.
  3. Seleccione la pestaña Sombras de dispositivo y seleccione la sombra. Puede ver el URI del punto de conexión en la sección URL de la sombra de dispositivo de la página Detalles de la sombra de dispositivo.

El formato del punto de enlace es el siguiente:

```
identifier.iot.region.amazonaws.com
```

La API REST oculta sigue las mismas protocols/port asignaciones HTTPS que se describen en. Protocolos de comunicación de dispositivos

> **Note**
>
> Para utilizarla APIs, debe utilizarla `iotdevicegateway` como nombre de servicio para la autenticación. Para obtener más información, consulte Io TData Plane.

Acciones de API de

- GetThingShadow
- UpdateThingShadow
- DeleteThingShadow
- ListNamedShadowsForThing

También puede usar la API para crear una sombra con nombre proporcionándole `name=shadowName` como parte del parámetro de consulta de la API.

GetThingShadow

Obtiene la sombra de 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?name=shadowName
Request body: (none)
```

El parámetro de consulta `name` no es necesario para sombras sin nombre (clásicas).

Respuesta

En caso de éxito, la respuesta incluye encabezados HTTP estándar, así como el código y el cuerpo siguientes:

```
HTTP 200
Response Body: response state document
```

Para obtener más información, consulte Ejemplo de documento de estado de respuesta.

Autorización

Para recuperar una sombra, se necesita una política que permita al intermediario ejecutar la acción `iot:GetThingShadow`. El servicio de sombra de dispositivo acepta dos formas de autenticación: Signature Version 4 con credenciales IAM o autenticación mutua TLS con un certificado de cliente.

A continuación, se muestra una política de ejemplo que permite a un intermediario recuperar la sombra de un dispositivo:

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

```
                "arn:aws:iot:us-east-1:123456789012: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 el valor `null` se eliminarán de la sombra del dispositivo.

Solicitud

La solicitud incluye los encabezados HTTP estándar, así como el URI y el cuerpo siguientes:

```
HTTP POST https://endpoint/things/thingName/shadow?name=shadowName
Request body: request state document
```

El parámetro de consulta `name` no es necesario para sombras sin nombre (clásicas).

Para obtener más información, consulte Ejemplo de documento de estado de solicitud.

Respuesta

En caso de éxito, la respuesta incluye encabezados HTTP estándar, así como el código y el cuerpo siguientes:

```
HTTP 200
Response body: response state document
```

Para obtener más información, consulte Ejemplo de documento de estado de respuesta.

Autorización

Para actualizar una sombra se necesita una política que permita al intermediario ejecutar la acción `iot:UpdateThingShadow`. El servicio de sombra de dispositivo acepta dos formas de autenticación: Signature Version 4 con credenciales IAM o autenticación mutua TLS con un certificado de cliente.

A continuación, se muestra una política de ejemplo que permite a un intermediario actualizar la sombra de un dispositivo:

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

DeleteThingShadow

Elimina la sombra de objeto especificado.

Solicitud

La solicitud incluye los encabezados HTTP estándar y el URI siguiente:

```
HTTP DELETE https://endpoint/things/thingName/shadow?name=shadowName
Request body: (none)
```

El parámetro de consulta `name` no es necesario para sombras sin nombre (clásicas).

Respuesta

En caso de éxito, la respuesta incluye encabezados HTTP estándar, así como el código y el cuerpo siguientes:

```
HTTP 200
Response body: Empty response state document
```

Tenga en cuenta que, al eliminar una sombra, no se restablece su número de versión a 0.

Autorización

Para eliminar la sombra de un dispositivo se necesita una política que permita al intermediario ejecutar la acción `iot:DeleteThingShadow`. El servicio de sombra de dispositivo acepta dos formas de autenticación: Signature Version 4 con credenciales IAM o autenticación mutua TLS con un certificado de cliente.

A continuación, se muestra una política de ejemplo que permite a un intermediario eliminar la sombra de un dispositivo:

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

ListNamedShadowsForThing

Muestra las sombras del objeto especificado.

Solicitud

La solicitud incluye los encabezados HTTP estándar y el URI siguiente:

```
HTTP GET /api/things/shadow/ListNamedShadowsForThing/thingName?
nextToken=nextToken&pageSize=pageSize
Request body: (none)
```

nextToken

El token para recuperar el siguiente grupo de resultados.

Este valor se devuelve en los resultados paginados y se utiliza en la llamada que devuelve la página siguiente.

pageSize

El número de nombres de sombra que devolver en cada llamada. Véase también `nextToken`.

thingName

El nombre del objeto para el que mostrar las sombras con nombre.

Respuesta

En caso de éxito, la respuesta incluye encabezados HTTP estándar, así como el código de respuesta siguiente y un Documento de respuesta de lista de nombres de sombra.

> **Note**
>
> La sombra sin nombre (clásica) no aparece en esta lista. La respuesta es una lista vacía si solo tiene una sombra clásica o si el `thingName` que ha especificado no existe.

```
HTTP 200
Response body: Shadow name list document
```

Autorización

Para incluir la sombra de un dispositivo, se necesita una política que permita que el intermediario ejecute la acción `iot:ListNamedShadowsForThing`. El servicio de sombra de dispositivo acepta dos formas de autenticación: Signature Version 4 con credenciales IAM o autenticación mutua TLS con un certificado de cliente.

A continuación, se muestra una política de ejemplo que permite a un intermediario mostrar las sombras con nombre de un objeto:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iot:ListNamedShadowsForThing",
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:thing/thing"
```

```
            ]
        }
    ]
}
```

Temas MQTT de sombra de dispositivo

El servicio Device Shadow utiliza temas MQTT reservados para permitir a los dispositivos y las aplicaciones obtener, actualizar o eliminar la información de estado de un dispositivo (sombra).

La publicación y suscripción de temas de sombra requiere una autorización basada en temas. AWS IoT se reserva el derecho a añadir 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, `$aws/things/thingName/shadow/#` ya que el número de temas que coinciden con este filtro podría aumentar a medida que se AWS IoT introduzcan nuevos temas ocultos. Para consultar ejemplos de mensajes publicados en estos temas vaya a Interacción con sombras.

Las sombras pueden ser con nombre o sin nombre (clásico). Los temas utilizados por cada uno solo difieren en el prefijo del tema. Esta tabla muestra el prefijo de tema utilizado por cada tipo de sombra.

Valor de ***ShadowTopicPrefix***	Tipo de sombra
`$aws/things/` *`thingName`* `/shadow`	Sombra sin nombre (clásica)
`$aws/things/` *`thingName`* `/shadow/name/` *`shadowName`*	Sombra con nombre

Para crear un tema completo, seleccione el *`ShadowTopicPrefix`* para el tipo de sombra al que desea hacer referencia, reemplace *`thingName`* y *`shadowName`* si procede, por sus valores correspondientes y, a continuación, anéxelo al código auxiliar del tema como se muestra en las secciones siguientes.

A continuación se muestran los temas MQTT utilizados para interactuar con las sombras.

Temas

- /get

- /get/accepted
- /get/rejected
- /update
- /update/delta
- /update/accepted
- /update/documents
- /update/rejected
- /delete
- /delete/accepted
- /delete/rejected

/get

Publique un mensaje vacío en este tema para obtener la sombra de objeto:

```
ShadowTopicPrefix/get
```

AWS IoT responde publicando en uno de los dos/get/accepted. /get/rejected

Ejemplo de política de

A continuación, mostramos un ejemplo de la política requerida:

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

```
    ]
}
```

/get/accepted

AWS IoT publica un documento paralelo de respuesta sobre este tema al devolver la sombra del dispositivo:

```
ShadowTopicPrefix/get/accepted
```

Para obtener más información, consulte Documentos de estado de la respuesta.

Ejemplo de política de

A continuación, mostramos un ejemplo de la política requerida:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/
things/thingName/shadow/get/accepted"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/thingName/
shadow/get/accepted"
            ]
        }
    ]
```

```
}
```

/get/rejected

AWS IoT publica un documento de respuesta a errores sobre este tema cuando no puede mostrar la sombra del dispositivo:

```
ShadowTopicPrefix/get/rejected
```

Para obtener más información, consulte Documento de respuesta de error.

Ejemplo de política de

A continuación, mostramos un ejemplo de la política requerida:

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Subscribe"
      ],
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/things/thingName/
shadow/get/rejected"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/thingName/shadow/get/
rejected"
      ]
    }
  ]
}
```

/update

Publique un documento de estado de solicitud en este tema para actualizar la sombra del dispositivo:

```
ShadowTopicPrefix/update
```

El cuerpo del mensaje contiene un documento de estado de solicitud parcial.

Un cliente que intente actualizar el estado de un dispositivo enviaría un documento de estado de solicitud JSON con la propiedad `desired` como esta:

```
{
  "state": {
    "desired": {
      "color": "red",
      "power": "on"
    }
  }
}
```

Un dispositivo que actualice su sombra enviaría un documento de estado de solicitud JSON con la propiedad `reported`, como esta:

```
{
  "state": {
    "reported": {
      "color": "red",
      "power": "on"
    }
  }
}
```

AWS IoT responde publicando en una de las /update/accepted dos/update/rejected.

Ejemplo de política de

A continuación, mostramos un ejemplo de la política requerida:

```
{
```

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

/update/delta

AWS IoT publica un documento de estado de respuesta sobre este tema cuando acepta un cambio en la sombra del dispositivo, y el documento de estado de respuesta contiene valores `desired` y `reported` estados diferentes:

```
ShadowTopicPrefix/update/delta
```

El búfer del mensaje contiene un Documento de estado de la respuesta /delta.

Detalles del cuerpo del mensaje

- 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 de

A continuación, mostramos un ejemplo de la política requerida:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/
things/thingName/shadow/update/delta"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/thingName/
shadow/update/delta"
            ]
        }
    ]
}
```

/update/accepted

AWS IoT publica un documento de estado de respuesta sobre este tema cuando acepta un cambio en la sombra del dispositivo:

```
ShadowTopicPrefix/update/accepted
```

El búfer del mensaje contiene un Documento de estado de la respuesta /aceptado.

Ejemplo de política de

A continuación, mostramos un ejemplo de la política requerida:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/
things/thingName/shadow/update/accepted"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/thingName/
shadow/update/accepted"
            ]
        }
    ]
}
```

/update/documents

AWS IoT publica un documento de estado sobre este tema cada vez que se realiza correctamente una actualización de la sombra:

```
ShadowTopicPrefix/update/documents
```

El cuerpo del mensaje contiene un Documento de estado de respuesta /documentos.

Ejemplo de política de

A continuación, mostramos un ejemplo de la política requerida:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/
things/thingName/shadow/update/documents"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/thingName/
shadow/update/documents"
            ]
        }
    ]
}
```

/update/rejected

AWS IoT publica un documento de respuesta a errores sobre este tema cuando rechaza un cambio en la sombra del dispositivo:

```
ShadowTopicPrefix/update/rejected
```

El cuerpo del mensaje contiene un Documento de respuesta de error.

Ejemplo de política de

A continuación, mostramos un ejemplo de la política requerida:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/
things/thingName/shadow/update/rejected"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/thingName/
shadow/update/rejected"
            ]
        }
    ]
}
```

/delete

Para eliminar la sombra de un dispositivo, publique un mensaje vacío para eliminar el tema:

```
ShadowTopicPrefix/delete
```

El contenido del mensaje no se tiene en cuenta.

Tenga en cuenta que, al eliminar una sombra, no se restablece su número de versión a 0.

AWS IoT responde publicando en una de las /delete/accepted dos/delete/rejected.

Ejemplo de política de

A continuación, mostramos un ejemplo de la política requerida:

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

/delete/accepted

AWS IoT publica un mensaje sobre este tema cuando se elimina la sombra de un dispositivo:

```
ShadowTopicPrefix/delete/accepted
```

Ejemplo de política de

A continuación, mostramos un ejemplo de la política requerida:

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

```
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/
things/thingName/shadow/delete/accepted"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/thingName/
shadow/delete/accepted"
            ]
        }
    ]
}
```

/delete/rejected

AWS IoT publica un documento de respuesta a errores sobre este tema cuando no puede eliminar la sombra del dispositivo:

```
ShadowTopicPrefix/delete/rejected
```

El cuerpo del mensaje contiene un Documento de respuesta de error.

Ejemplo de política de

A continuación, mostramos un ejemplo de la política requerida:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/
things/thingName/shadow/delete/rejected"
```

```
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/thingName/
shadow/delete/rejected"
            ]
        }
    ]
}
```

Documentos del servicio Device Shadow

El servicio Device Shadow respeta todas las reglas de la especificación JSON. Los valores, objetos y matrices se almacenan en el documento de sombra del dispositivo.

Contenido

- Ejemplos de documento de sombra
- Propiedades del documento
- Estado delta
- Control de versiones de documentos de sombra
- Tokens de cliente en documentos de sombra
- Propiedades de documento de sombra vacío
- Valores de matriz en documentos sombra

Ejemplos de documento de sombra

El servicio Device Shadow utiliza estos documentos en las operaciones UPDATE, GET y DELETE mediante la API REST o los Pub/Sub mensajes MQTT.

Ejemplos

- Documento de estado de la solicitud

- Documentos de estado de la respuesta
- Documento de respuesta de error
- Documento de respuesta de lista de nombres de sombra

Documento de estado de la solicitud

Un documento de estado de solicitud tiene el siguiente formato:

```
{
    "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. Normalmente, utilizará la propiedad `desired` o la propiedad `reported`, pero no ambas en la misma solicitud.
  - `desired`: las propiedades y los valores de estado que estamos solicitando actualizar en el dispositivo.
  - `reported`: las propiedades y los valores de estado informados por el dispositivo.
- `clientToken`: si se usa, puede encontrar la correspondencia entre la solicitud y la respuesta mediante el token del cliente.
- `version`: si se utiliza, el servicio Device Shadow 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 respuesta tienen el siguiente formato en función del tipo de respuesta.

Documento de estado de la respuesta /aceptado

```
{
    "state": {
        "desired": {
            "attribute1": integer2,
            "attribute2": "string2",
            ...
            "attributeN": boolean2
        }
    },
    "metadata": {
        "desired": {
            "attribute1": {
                "timestamp": timestamp
            },
            "attribute2": {
                "timestamp": timestamp
            },
            ...
            "attributeN": {
                "timestamp": timestamp
            }
        }
    },
    "timestamp": timestamp,
    "clientToken": "token",
    "version": version
}
```

Documento de estado de la respuesta /delta

```
{
    "state": {
        "attribute1": integer2,
        "attribute2": "string2",
        ...
        "attributeN": boolean2
    },
```

```
    "metadata": {
        "attribute1": {
            "timestamp": timestamp
        },
        "attribute2": {
            "timestamp": timestamp
        },
        ...
        "attributeN": {
            "timestamp": timestamp
        }
    },
    "timestamp": timestamp,
    "clientToken": "token",
    "version": version
}
```

Documento de estado de respuesta /documentos

```
{
  "previous" : {
    "state": {
        "desired": {
            "attribute1": integer2,
            "attribute2": "string2",
            ...
            "attributeN": boolean2
        },
        "reported": {
            "attribute1": integer1,
            "attribute2": "string1",
            ...
            "attributeN": boolean1
        }
    },
    "metadata": {
        "desired": {
            "attribute1": {
                "timestamp": timestamp
            },
            "attribute2": {
                "timestamp": timestamp
            },
```

```
            ...
            "attributeN": {
                "timestamp": timestamp
            }
        },
        "reported": {
            "attribute1": {
                "timestamp": timestamp
            },
            "attribute2": {
                "timestamp": timestamp
            },
            ...
            "attributeN": {
                "timestamp": timestamp
            }
        }
    },
    "version": version-1
},
"current": {
    "state": {
        "desired": {
            "attribute1": integer2,
            "attribute2": "string2",
            ...
            "attributeN": boolean2
        },
        "reported": {
            "attribute1": integer2,
            "attribute2": "string2",
            ...
            "attributeN": boolean2
        }
    },
    "metadata": {
        "desired": {
            "attribute1": {
                "timestamp": timestamp
            },
            "attribute2": {
                "timestamp": timestamp
            },
            ...
```

```
            "attributeN": {
                "timestamp": timestamp
            }
        },
        "reported": {
            "attribute1": {
                "timestamp": timestamp
            },
            "attribute2": {
                "timestamp": timestamp
            },
            ...
            "attributeN": {
                "timestamp": timestamp
            }
        }
    },
    "version": version
  },
  "timestamp": timestamp,
  "clientToken": "token"
}
```

Propiedades del documento de estado de la respuesta

- `previous`: tras una actualización correcta, contiene el `state` del objeto antes de la actualización.
- `current`: tras una actualización correcta, contiene el `state` del objeto después de la actualización.
- `state`
  - `reported`: solo ocurre 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 ocurre si un dispositivo 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 temporales de cada atributo de las secciones `desired` y `reported` para que se pueda determinar cuándo se actualizó el estado.
- `timestamp`— La época, fecha y hora en que se generó la respuesta. AWS IoT
- `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 del dispositivo compartido en AWS IoT. Se aumenta en una unidad con relación a la versión anterior del documento.

Documento de respuesta de error

Un documento de respuesta de error tiene 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`— La fecha y la hora en que se generó la respuesta. AWS IoT Esta propiedad no está presente en todos los documentos de respuesta de error.
- `clientToken`: solo ocurre si se ha utilizado un token de cliente en el mensaje publicado.

Para obtener más información, consulte Mensajes de error de Device Shadow.

Documento de respuesta de lista de nombres de sombra

Un documento de respuesta de lista de nombre de sombra tiene el siguiente formato:

```
{
    "results": [
        "shadowName-1",
        "shadowName-2",
        "shadowName-3",
        "shadowName-n"
    ],
    "nextToken": "nextToken",
    "timestamp": timestamp
}
```

- `results`: la matriz de nombres de sombras.

- `nextToken`: el valor del token que se utilizará en las solicitudes paginadas para obtener la siguiente página de la secuencia. Esta propiedad no está presente cuando no hay más nombres de sombra que devolver.
- `timestamp`— La fecha y la hora en que se generó la respuesta AWS IoT.

Propiedades del documento

El documento de sombra de un dispositivo tiene las propiedades siguientes:

`state`

`desired`

El estado deseado del dispositivo. Las aplicaciones pueden escribir en esta parte del documento para actualizar el estado de un dispositivo directamente, sin tener que conectarse al mismo.

`reported`

El estado notificado del dispositivo. Los dispositivos escriben en esta parte del documento para notificar su nuevo estado. Las aplicaciones leen esta parte del documento para determinar el último estado notificado del dispositivo.

`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.

> **Note**
>
> Los metadatos no contribuyen al tamaño de documento para establecer los límites del servicio o el precio. Para obtener más información, consulte AWS IoT Límites de servicio.

`timestamp`

Indica cuándo se envió el mensaje AWS IoT. Mediante el uso de la marca temporal en el mensaje y las marcas temporales para atributos individuales en la sección `desired` o `reported`, un dispositivo puede determinar la antigüedad de una propiedad, incluso si el dispositivo no tiene 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 Ejemplos de documento de sombra.

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",
      "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 Device Shadow calcula la diferencia iterando por cada campo con el 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.

Control de versiones de documentos de sombra

El servicio Device Shadow admite el control de versiones en cada mensaje de actualización, tanto de solicitud como de respuesta. Esto significa que con cada actualización de una sombra, se incrementa la versión del documento JSON. 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 la sombra de un dispositivo.
- Un cliente puede decidir omitir un mensaje recibido si su versión es anterior a la que tiene almacenada.

Un cliente puede omitir la coincidencia de versiones al no incluir una versión en el documento de sombra.

Tokens de cliente en documentos de sombra

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.

> **Note**
>
> El token de cliente no puede ser superior a 64 bytes. Un token de cliente que sea superior a 64 bytes provocará una respuesta 400 (solicitud errónea) y un mensaje de error clientToken no válido.

Propiedades de documento de sombra vacío

Las propiedades `reported` y `desired` de un documento de sombra pueden estar vacías u omitidas cuando no se aplican al estado de sombra actual. Por ejemplo, un documento de sombra contiene una propiedad `desired` solo si tiene un estado deseado. A continuación se muestra un ejemplo válido de un documento de estado sin propiedad `desired`:

```
{
    "reported" : { "temp": 55 }
}
```

La propiedad `reported` también puede estar vacía, por ejemplo, si el dispositivo no ha actualizado la sombra:

```
{
    "desired" : { "color" : "RED" }
}
```

Si una actualización hace que las propiedades `desired` o `reported` se conviertan en nulas, se quita del documento. A continuación se muestra cómo quitar la propiedad `desired` estableciéndola en `null`. Puede hacerlo cuando un dispositivo actualice su estado, por ejemplo.

```
{
    "state": {
        "reported": {
```

```
            "color": "red"
        },
        "desired": null
    }
}
```

Un documento de sombra también puede no tener ni la propiedad `desired` ni `reported`, lo que hace que el documento de sombra esté vacío. Este es un ejemplo de un documento de sombra vacío pero válido.

```
{
}
```

Valores de matriz en documentos sombra

Las sombras son compatibles con las matrices, pero las tratan como valores normales, en el sentido de que la actualización de una matriz sustituye toda la matriz. No es posible actualizar parte de una matriz.

Estado inicial:

```
{
    "desired" : { "colors" : ["RED", "GREEN", "BLUE" ] }
}
```

Actualizar:

```
{
    "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" ]
    }
}
```

Mensajes de error de Device Shadow

El servicio Device Shadow 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)	• JSON no válido • Falta un nodo necesario: estado • El nodo de estado debe ser un objeto • El nodo deseado debe ser un objeto • El nodo notificado debe ser un objeto • Versión no válida • ClientToken no válido **Note** Un token de cliente que sea mayor de 64 bytes provocará esta respuesta. • JSON contiene demasiados niveles de anidamiento; el máximo permitido es 6 • El estado contiene un nodo no válido
401 (sin autorización)	• No autorizado
403 (prohibido)	• Prohibido

Código de error HTTP	Mensajes de error
404 (no encontrado)	• Objeto no encontrado • No existe ninguna sombra con el nombre: *shadowName*
409 (conflicto)	• Conflicto de versiones
413 (carga demasiado grande)	• La carga supera el tamaño máximo permitido
415 (tipo de medio incompatible)	• Codificación documentada incompatible, se admite la codificación UTF-8
429 (demasiadas solicitudes)	• El servicio de sombra de dispositivo generará este mensaje de error cuando haya más de diez solicitudes en tránsito en una sola conexión. Una solicitud en tránsito es una solicitud en curso que se ha iniciado, pero no se ha completado.
500 (error de servidor interno)	• Error de servicio interno

AWS IoT Device Management Catálogo de paquetes de software

Con AWS IoT Device Management Software Package Catalog, puede mantener un inventario de los paquetes de software y sus versiones. Puede asociar las versiones de los paquetes a elementos individuales y a grupos de elementos AWS IoT dinámicos e implementarlos mediante procesos o AWS IoT trabajos internos.

El paquete de software contiene una o más versiones de paquete, que es una colección de archivos que se pueden implementar como una sola unidad. Las versiones del paquete pueden contener firmware, actualizaciones del sistema operativo, aplicaciones de dispositivos, configuraciones y parches de seguridad. A medida que el software evoluciona con el tiempo, puede crear una nueva versión del paquete e implementarla en su flota.

El centro AWS IoT de paquetes de software se encuentra dentro AWS IoT Core. Puede utilizar el hub para registrar y mantener de forma centralizada el inventario y los metadatos de sus paquetes de software, lo que crea un catálogo de paquetes de software y sus versiones. Puede optar por agrupar los dispositivos en función de los paquetes de software y las versiones de paquetes implementadas en el dispositivo. Esta característica brinda la oportunidad de mantener el inventario de paquetes del lado del dispositivo como una sombra, asociar y agrupar los dispositivos en función de las versiones, y visualizar la distribución de las versiones de los paquetes en toda la flota mediante las métricas de la flota.

Si ha establecido un sistema interno de implementación de software, puede seguir utilizando ese proceso para implementar las versiones de sus paquetes. Si no tiene un proceso de implementación establecido o si lo prefiere, le recomendamos que utilice trabajos de AWS IoT para usar las características del Catálogo de Paquetes de Software. Para obtener más información, consulte Preparación AWS IoT de trabajos.

El capítulo contiene las siguientes secciones:

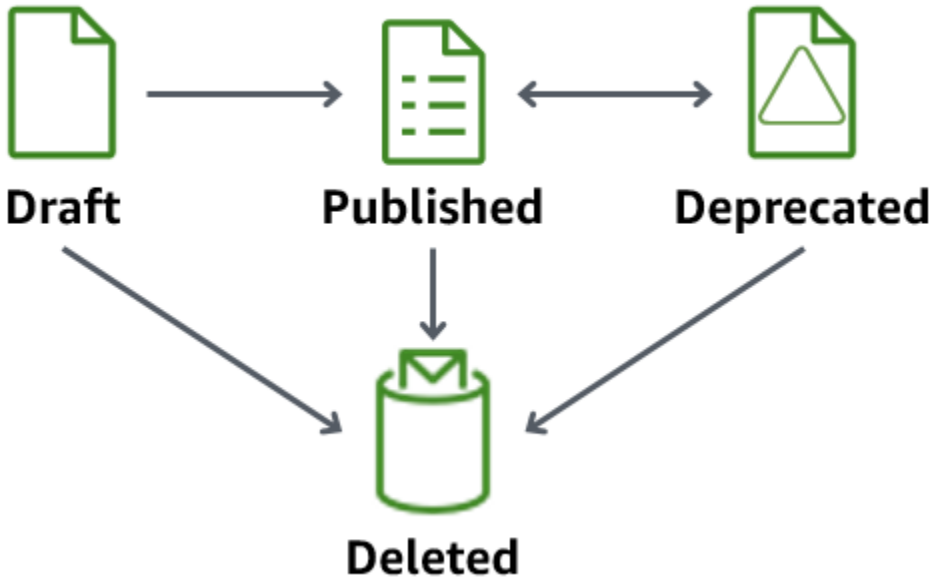
- Preparación para utilizar el Catálogo de paquetes de software
- Preparación de la seguridad
- Preparación para la indexación de flotas
- Preparando trabajos AWS IoT
- Introducción al Catálogo de paquetes de software

Preparación para utilizar el Catálogo de paquetes de software

La siguiente sección proporciona una descripción general del ciclo de vida de la versión del paquete e información para usar AWS IoT Device Management Software Package Catalog.

Ciclo de vida de la versión de paquete

La versión de un paquete puede evolucionar a través de los siguientes estados del ciclo de vida: `draft`, `published` y`deprecated`. También puede ser `deleted`.



- Borrador

  Cuando se crea una versión de paquete, este tiene el estado `draft`. Este estado indica que el paquete de software se está preparando o está incompleto.

  Mientras la versión del paquete tenga este estado, no se podrá implementar. Puede editar la descripción, los atributos y las etiquetas de la versión del paquete.

  Puede realizar la transición de una versión de paquete que esté en `draft` estado `published` o en estado `deleted` utilizando la consola UpdatePackageVersiono realizando operaciones de DeletePackageVersionAPI.

- Publicado

  Cuando la versión del paquete esté lista para implementarse, pase la versión del paquete al estado `published`. Mientras se encuentre en este estado, puede elegir identificar la versión

del paquete como la versión predeterminada editando el paquete de software en la consola o mediante la operación de la UpdatePackageAPI. En este estado, solo puede editar la descripción y las etiquetas.

Puedes hacer la transición de una versión de paquete que esté `deleted` en ese `published` estado a `deprecated` la consola o realizando las operaciones de DeletePackageVersionAPI UpdatePackageVersiono las mismas.

- Obsoleto

  Si hay disponible una nueva versión del paquete, puede realizar la transición a una versión anterior del paquete a `deprecated`. De todos modos, puede seguir implementando trabajos con una versión de paquete obsoleta. También puede nombrar una versión de paquete obsoleta como la versión predeterminada y editar solo la descripción y las etiquetas.

  Considere la posibilidad de hacer la transición de una versión de paquete a `deprecated` cuando la versión esté desactualizada pero todavía usa dispositivos que tienen la versión anterior o deba mantenerla debido a la dependencia del tiempo de ejecución.

  Puedes realizar la transición de una versión de paquete que se `deleted` encuentre en ese `deprecated` estado a `published` o a través de la consola o ejecutando las UpdatePackageVersionoperaciones de DeletePackageVersionAPI.

- Eliminado

  Si ya no quieres usar una versión de paquete, puedes eliminarla desde la consola o mediante la operación de DeletePackageVersionAPI.

  > **Note**
  >
  > Si elimina una versión de paquete mientras hay trabajos pendientes que hacen referencia a ella, recibirá un mensaje de error cuando el trabajo finalice con éxito e intente actualizar la sombra con nombre reservado.
  > Si la versión del paquete de software que desea eliminar aparece como la versión del paquete por defecto, primero debe actualizar el paquete para nombrar otra versión como predeterminada o dejar el campo sin nombre. Para ello, puede utilizar la consola o la operación UpdatePackageVersionde API. (Para eliminar cualquier versión de paquete con nombre predeterminado, defina el unsetDefaultVersionparámetro en true cuando ejecute la operación de UpdatePackageAPI).

Si elimina un paquete de software a través de la consola, se eliminan todas las versiones del paquete asociadas a ese paquete, a menos que una de ellas se nombre como versión por defecto.

Convenciones de nomenclatura de versiones de paquetes

Al asignar nombres a las versiones de los paquetes, es importante planificar y aplicar una estrategia de nomenclatura lógica para que usted y los demás puedan identificar fácilmente la última versión del paquete y su progresión. Debe proporcionar un nombre de versión al crear la versión del paquete, pero la estrategia y el formato dependen en gran medida de su modelo de negocio.

Como práctica recomendada, recomendamos utilizar el formato de control de versiones SemVersemántico. Por ejemplo, `1.2.3` donde `1` es la versión principal para los cambios funcionalmente incompatibles, `2` la versión principal para los cambios compatibles desde el punto de vista funcional y `3` la versión de parche (para las correcciones de errores) Para obtener más información, consulte Semantic Versioning 2.0.0. Para obtener más información sobre los requisitos de nombre de la versión del paquete, consulta versionName en la guía de referencia de la AWS IoT API.

Versión predeterminada

Establecer una versión como predeterminada es opcional. Puede agregar o eliminar versiones de paquete predeterminadas. También puede implementar una versión de paquete que no figure como la versión predeterminada.

Al crear una versión de paquete, se coloca en un estado `draft` y no se le puede asignar el nombre de versión predeterminada hasta que se haga la transición de la versión del paquete a publicada. El catálogo de paquetes de software no selecciona automáticamente una versión como predeterminada ni actualiza una versión de paquete más reciente como predeterminada. Debe asignar un nombre intencionado a la versión del paquete que elija a través de la consola o mediante la operación de UpdatePackageVersionAPI.

Atributos de la versión

Los atributos de versión y sus valores contienen información importante sobre las versiones de los paquetes. Se recomienda definir los atributos de uso general para un paquete o una versión del paquete. Por ejemplo, puede crear un par nombre-valor para la plataforma, la arquitectura, el sistema operativo, la fecha de lanzamiento, el autor o la URL de Amazon S3.

Al crear un AWS IoT trabajo con un documento de trabajo, también puede optar por utilizar una variable de sustitución (`$parameter`) que haga referencia al valor de un atributo. Para obtener más información, consulte Preparación AWS IoT de trabajos.

Los atributos de versión que se utilizan en las versiones de paquetes no se añadirán automáticamente a la sombra reservada denominada y no se podrán indexar ni consultar directamente a través de la indexación de flotas. Para indexar o consultar los atributos de la versión del paquete mediante la indexación de flotas, puede rellenar el atributo de la versión en la sombra reservada con nombre.

Se recomienda que el parámetro del atributo de versión de la reserva se llame Shadow Capture las propiedades informadas por el dispositivo, como el sistema operativo y el tiempo de instalación. También se pueden indexar y consultar mediante la indexación de flotas.

No es necesario que los atributos de la versión sigan una convención de nomenclatura específica. Puede crear pares de nombre-valor para satisfacer las necesidades de su empresa. El tamaño combinado de todos los atributos de una versión de paquete está limitado a 3 KB. Para obtener más información, consulte Límites de paquetes de software y versiones de paquetes del Catálogo de paquetes de software.

Uso de todos los atributos de un documento de trabajo

Puede hacer que todos los atributos de la versión del paquete se añadan automáticamente a la implementación de su trabajo para los dispositivos seleccionados. Para utilizar automáticamente todos los atributos de la versión del paquete mediante programación en un comando de la API o la CLI, consulte el siguiente ejemplo de documento de trabajo:

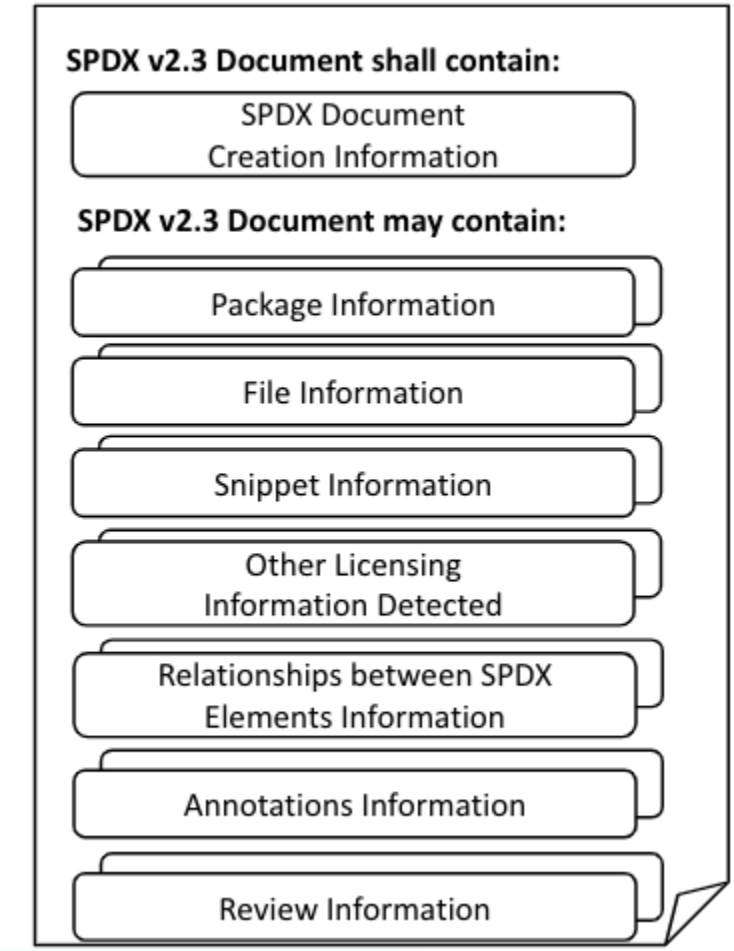
```
"TestPackage": "${aws:iot:package:TestPackage:version:PackageVersion:attributes}"
```

Lista de materiales de software

La lista de materiales de software (SBOM) proporciona un repositorio central para todos los aspectos del paquete de software. Además de almacenar los paquetes de software y las versiones de los paquetes, puede almacenar la lista de materiales del software (SBOM) asociada a cada versión del AWS IoT Device Management paquete en Software Package Catalog. El paquete de software contiene una o más versiones de paquete y cada versión de paquete consta de uno o más componentes. Cada uno de esos componentes que sirven de base para componer una versión de paquete específica se puede describir y catalogar mediante una lista de materiales de software. Los estándares del sector para la lista de materiales de software admitidos son SPDX y CyclonedX.

Cuando se crea una SBOM por primera vez, se somete a validación según el formato estándar del sector SPDX y CyclonedX. Para obtener más información sobre SPDX, consulte System Package Data Exchange. Para obtener más información sobre CyclonedX, consulte CycloneDX.

La lista de materiales de software describe todos los aspectos de los componentes de una versión de paquete específica, como la información del paquete, la información del archivo y otros metadatos pertinentes. Consulte el siguiente ejemplo de la estructura de un documento de una lista de materiales de software en formato SPDX:



Ventajas de la lista de materiales de software

Una de las principales ventajas de añadir la lista de materiales de software para una versión de paquete en el Catálogo de paquetes de software es la administración de vulnerabilidades.

Administración de vulnerabilidades

Evaluar y mitigar su vulnerabilidad ante los aparentes riesgos de seguridad de los componentes de software sigue siendo fundamental para proteger la integridad de la flota de dispositivos. Al

añadir la lista de materiales de software almacenada en el Catálogo de paquetes de software para cada versión de paquete, puede exponer de forma proactiva las brechas de seguridad al saber qué dispositivos están en riesgo en función de su versión de paquete y de la SBOM usando su propia solución interna de administración de vulnerabilidades. Puede implementar correcciones en los dispositivos afectados y proteger su flota de dispositivos.

Almacenamiento de la lista de materiales de software

La lista de materiales de software (SBOM) de cada versión del paquete de software se almacena en un bucket de Amazon S3 con la característica de control de versiones de Amazon S3. El bucket de Amazon S3 que almacena la SBOM debe estar ubicado en la misma región en la que se creó la versión del paquete. El bucket de Amazon S3 que utiliza la característica de control de versiones mantiene diversas variantes de un objeto en el mismo bucket. Para obtener más información sobre el uso del control de versiones en un bucket de Amazon S3, consulte Uso de control de versiones en buckets de Amazon S3.

> **Note**
>
> Cada versión del paquete de software puede tener varios archivos SBOM asociados a él, pero los archivos SBOM deben almacenarse en un único archivo zip.

La clave de Amazon S3 y el ID de versión específicos de su bucket se utilizan para identificar de forma exclusiva cada versión de una lista de materiales de software para una versión de paquete.

> **Note**
>
> Para una versión de paquete con un único archivo SBOM, puede almacenar ese archivo SBOM en su bucket de Amazon S3 como archivo zip.
> Para una versión de paquete con varios archivos SBOM, debe colocar todos los archivos SBOM en un único archivo zip y, a continuación, almacenar ese archivo zip en su bucket de Amazon S3.
> Todos los archivos SBOM almacenados en un único archivo zip en ambos escenarios tienen el formato de archivo .json de SPDX o CyclonedX.

Política de permisos

Para poder AWS IoT actuar como el principal especificado para acceder a los archivos zip de la SBOM almacenados en el bucket de Amazon S3, necesita una política de permisos basada en los recursos. Consulte el siguiente ejemplo para conocer la política de permisos basada en recursos correcta:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "iot.amazonaws.com"
                ]
        },
            "Action": "s3:*",
            "Resource": "arn:aws:s3:::bucketName/*"
        }
    ]
}
```

Para obtener más información sobre las políticas de permisos basadas en recursos, consulte AWS IoT políticas basadas en recursos.

Actualización de la SBOM

Puede actualizar la lista de materiales del software tantas veces como sea necesario para proteger y mejorar su flota de dispositivos. Cada vez que se actualiza la lista de materiales de software en el bucket de Amazon S3, el ID de versión cambia y usted debe asociar la nueva URL del bucket de Amazon S3 a la versión del paquete de software correspondiente. Verá el nuevo ID de versión en la columna ID de versión del objeto de Amazon S3 en la página de la versión del paquete en la Consola de administración de AWS. Además, puede usar la operación de la API `GetPackageVersion` o el comando de la CLI `get-package-version` para ver el nuevo ID de versión.

> **Note**
>
> La actualización de la lista de materiales de software, que generará un nuevo ID de versión, no provocará la creación de una nueva versión del paquete.

Para obtener más información acerca de las claves de objeto de Amazon S3, consulte Creación de nombres de clave de objeto.

AWS IoT Habilitar la indexación de la flota

Habilitar AWS IoT la indexación de la flota es un requisito para utilizar AWS IoT Device Management Software Package Catalog. Para aprovechar la indexación de flotas de AWS IoT con el Catálogo de paquetes de software, establezca la sombra con nombre reservada (`$package`) como el origen de datos para cada dispositivo que desee indexar y del que recopilar métricas. Para obtener más información sobre las sombras con nombre reservadas, consulte Sombra con nombre reservado.

La indexación de flotas proporciona un soporte que permite agrupar los AWS IoT elementos mediante grupos de elementos dinámicos que se filtran por versión del paquete de software. Por ejemplo, la indexación de flotas puede identificar elementos que tienen o no una versión de paquete específica instalada, que no tienen ninguna versión de paquete instalada o que coinciden con pares de nombre y valor específicos. Por último, la indexación de la flota proporciona métricas estándar y personalizadas que puede utilizar para obtener más información sobre el estado de su flota de dispositivos. Para obtener más información, consulte Preparación para la indexación de flotas.

> **Note**
>
> Habilitar la indexación de flotas para el Catálogo de Paquetes de Software incurre en costes de servicio estándar. Para obtener más información, consulte Precios de AWS IoT Device Management.

Sombra con nombre reservado

El nombre reservado shadow, `$package`, refleja el estado de los paquetes de software y las versiones de los paquetes instalados en el dispositivo. La indexación de flotas utiliza la sombra con nombre reservado como origen de datos para crear métricas estándar y personalizadas que

le permitan consultar el estado de su flota. Para más información, consulte Preparación de la indexación de flotas.

Una sombra con nombre reservado es similar a una sombra con nombre, con la excepción de que su nombre está predefinido y no se puede cambiar. Además, la sombra reservada con nombre asignado no se actualiza con los metadatos y solo usa las palabras clave `version` y `attributes`.

Las solicitudes de actualización que incluyan otras palabras clave, por ejemplo `description`, recibirán una respuesta de error en relación con el tema `rejected`. Para más información, consulte Mensajes de error de Device Shadow.

Se puede crear cuando se crea AWS IoT algo a través de la consola, cuando un AWS IoT trabajo se completa correctamente y actualiza la sombra y si se ejecuta la operación de `UpdateThingShadow`API. Para obtener más información, consulta UpdateThingShadowla guía para AWS IoT Core desarrolladores.

> **Note**
>
> La indexación de la sombra con nombre reservada no cuenta para el número de sombras con nombre que la indexación de flotas puede indexar. Para más información, consulte Límites y cuotas de indexación de flotas AWS IoT Device Management. Además, si decides que los AWS IoT trabajos actualicen la sombra con el nombre reservado cuando un trabajo se complete correctamente, la llamada a la API se tendrá en cuenta para tus operaciones de registro y Device Shadow, y puede suponer un coste. Para obtener más información, consulta los límites y cuotas de los AWS IoT Device Management trabajos y el tipo de datos de la IndexingFilterAPI.

Estructura de la sombra **`$package`**

La sombra reservada con el nombre contiene lo siguiente:

```
{
    "state": {
        "reported": {
            "<packageName>": {
                "version": "",
                "attributes": {
                }
            }
```

```
        }
    },
    "version" : 1
    "timestamp" : 1672531201
}
```

Las propiedades de la sombra se actualizan con la siguiente información:

- `<packageName>`: el nombre del paquete de software instalado, que se actualiza con el parámetro packageName.
- `version`: El nombre de la versión del paquete instalado, que se actualiza con el parámetro versionName.
- `attributes`: metadatos opcionales almacenados por el dispositivo e indexados mediante la indexación de flotas. Esto permite a los clientes consultar sus índices en función de los datos almacenados.
- `version`: El número de versión de la sombra. Se incrementa automáticamente cada vez que se actualiza la sombra y comienza en 1.
- `timestamp`: Indica cuándo se actualizó por última vez la sombra y se registra en tiempo Unix.

Para obtener más información sobre el formato y el comportamiento de una sombra con nombre, consulte Orden de los mensajes Servicio de sombra de dispositivo de AWS IoT.

Eliminar un paquete de software y sus versiones

Antes de eliminar un paquete de software, haga lo siguiente:

- Confirme que el paquete y sus versiones no se estén implementando activamente.
- Elimine primero todas las versiones asociadas. Si una de las versiones está designada como la versión predeterminada, debe eliminar la versión predeterminada nombrada del paquete. Como la designación de una versión predeterminada es opcional, no hay ningún conflicto al eliminarla. Para eliminar la versión predeterminada del paquete de software, edítelo a través de la consola o utilice la operación de UpdatePackageVersionAPI.

Mientras no haya una versión de paquete predeterminada con nombre, puede usar la consola para eliminar un paquete de software y también se eliminarán todas sus versiones del paquete. Si utiliza una llamada a la API para eliminar paquetes de software, primero debe eliminar las versiones del paquete y, a continuación, el paquete de software.

Preparación de la seguridad

En esta sección se analizan los principales requisitos de seguridad para AWS IoT Device Management Software Package Catalog.

Autenticación basada en recursos

El Catálogo de paquetes de software utiliza la autorización basada en recursos para proporcionar una mayor seguridad al actualizar el software de su flota. Esto significa que debe crear una política AWS Identity and Access Management (IAM) que conceda derechos de ejecución `create` `read` `updatedelete`, y `list` acciones para los paquetes de software y las versiones de los paquetes, y hacer referencia a los paquetes de software y las versiones de paquetes específicos que desee implementar en la `Resources` sección. También necesita estos derechos para poder actualizar la sombra con nombre reservado. Para hacer referencia a los paquetes de software y las versiones de paquete, debe incluir un nombre de recurso de Amazon (ARN) para cada entidad.

> **Note**
>
> Si su intención es conceder derechos para las llamadas a la API en versiones de paquetes (como CreatePackageVersionUpdatePackageVersion,, DeletePackageVersion), debe incluir tanto el paquete de software como la versión del paquete ARNs en la política. Si pretende que la política conceda derechos para las llamadas a la API de paquetes de software (como CreatePackageUpdatePackage, y DeletePackage), debe incluir únicamente el ARN del paquete de software en la política.

Estructure el paquete de software y la versión del paquete de la ARNs siguiente manera:

- Paquete de software: `arn:aws:iot:<region>:<accountID>:package/<packageName>/package`
- Versión de paquete: `arn:aws:iot:<region>:<accountID>:package/<packageName>/version/<versionName>`

> **Note**
>
> Existen otros derechos relacionados que podría incluir en esta política. Por ejemplo, puede incluir un ARN para `job`, `thinggroup` y `jobtemplate`. Para obtener más información y

una lista completa de las opciones de políticas, consulte Proteger usuarios y dispositivos con AWS IoT Jobs.

Por ejemplo, si tiene un paquete de software y una versión de paquete con el siguiente nombre:

- AWS IoT cosa: `myThing`
- Nombre del paquete: `samplePackage`
- Versión: `1.0.0`

La política podría parecerse al siguiente ejemplo:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:createPackage",
                "iot:createPackageVersion",
                "iot:updatePackage",
                "iot:updatePackageVersion"
            ],
            "Resource": [
               "arn:aws:iot:us-east-1:111122223333:package/samplePackage",
               "arn:aws:iot:us-east-1:111122223333:package/samplePackage/
version/1.0.0"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:GetThingShadow",
                "iot:UpdateThingShadow"
            ],
            "Resource": "arn:aws:iot:us-east-1:111122223333:thing/myThing/$package"
        }
    ]
}
```

AWS IoT Job: derechos para implementar versiones de paquetes

Por motivos de seguridad, es importante que conceda derechos para desplegar paquetes y versiones de paquetes, y que nombre los paquetes y versiones de paquetes específicos que están autorizados a desplegar. Para ello, debe crear un rol y una política de IAM que concedan permiso para implementar trabajos con versiones de paquetes. La política debe especificar las versiones del paquete de destino como recurso.

Política de IAM

La política de IAM otorga el derecho a crear un trabajo que incluya el paquete y la versión que se indican en la sección `Resource`.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:CreateJob",
                "iot:CreateJobTemplate"
            ],
            "Resource":[
                "arn:aws:iot:*:111122223333:job/<jobId>",
                "arn:aws:iot:*:111122223333:thing/<thingName>/$package",
                "arn:aws:iot:*:111122223333:thinggroup/<thingGroupName>",
                "arn:aws:iot:*:111122223333:jobtemplate/<jobTemplateName>",
                "arn:aws:iot:*:111122223333:package/<packageName>/
version/<versionName>"
            ]
        }
    ]
}
```

> **Note**
>
> Si desea implementar un trabajo que desinstale un paquete de software y una versión del paquete, debe autorizar un ARN donde esté la versión del paquete `$null`, como en los siguientes casos:

```
arn:aws:iot:<regionCode>:111122223333:package/<packageName>/version/$null
```

AWS IoT Job: derechos para actualizar la sombra reservada con nombre

Para permitir que los trabajos actualicen la sombra del nombre reservado de la cosa cuando el trabajo se complete correctamente, debe crear un rol y una política de IAM. Hay dos formas de realizar esta operación en la consola de AWS IoT . La primera es al crear un paquete de software en la consola. Si ve el cuadro de diálogo Habilitar las dependencias para la administración de paquetes, puede elegir usar un rol existente o crear uno nuevo. O bien, en la consola de AWS IoT , seleccione Configuración, seleccione Administrar la indexación y, a continuación, Administrar la indexación de los paquetes y las versiones de los dispositivos.

> **Note**
>
> Si decides que el servicio AWS IoT Job actualice la sombra con el nombre reservado cuando una tarea se complete correctamente, la llamada a la API se tendrá en cuenta para tus operaciones de Device Shadow y registro, y puede suponer un coste. Para obtener más información, consulte Precios de AWS IoT Core.

Cuando se utiliza la opción Crear rol, el nombre del rol generado comienza por `aws-iot-role-update-shadows` y contiene las siguientes políticas:

Configuración un rol

Permisos

La política de permisos otorga los derechos para consultar y actualizar la sombra de la cosa. El parámetro `$package` del ARN del recurso apunta a la sombra con nombre reservado.

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

```
            "Action": [
                "iot:GetThingShadow",
                "iot:UpdateThingShadow"
            ],
            "Resource": [
            "arn:aws:iot:us-east-1:111122223333:thing/<thingName>/$package"
            ]
        }
    ]
}
```

Relación de confianza

Además de la política de permisos, el rol requiere una relación de confianza con AWS IoT Core para que la entidad pueda asumir el rol y actualizar la sombra nominal reservada.

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

Configurar una política de usuario

iam: permiso PassRole

Por último, debes tener el permiso para transferir el rol AWS IoT Core cuando llames a la operación de la UpdatePackageConfigurationAPI.

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

```
            "Action": [
                "iam:PassRole",
                "iot:UpdatePackageConfiguration"
            ],
            "Resource": "arn:aws:iam::111122223333:role/<roleName>"
        }
    ]
}
```

AWS IoT Permisos de tareas para descargar desde Amazon S3

El documento de trabajo se guarda en Amazon S3. Hace referencia a este archivo cuando realiza envíos a través de AWS IoT Jobs. Debes proporcionar a AWS IoT Jobs los derechos para descargar el archivo (`s3:GetObject`). También debe establecer una relación de confianza entre Amazon S3 y AWS IoT Jobs. Para obtener instrucciones sobre cómo crear estas políticas, consulte Presigned URLs in Managing Jobs.

Permisos para actualizar la lista de materiales de software para una versión de paquete

Para actualizar la lista de materiales del software de una versión de paquete en los `Draft` estados de `Deprecated` ciclo de vida o ciclo de vida, necesita una AWS Identity and Access Management función y políticas para localizar la nueva lista de materiales del software en Amazon S3 y actualizar la versión del paquete en AWS IoT Core. `Published`

En primer lugar, colocará la lista de materiales del software actualizada en su bucket de Amazon S3 con control de versiones y llamará a la operación de la API `UpdatePackageVersion` con el parámetro `sboms` incluido. A continuación, su director autorizado asumirá la función de IAM que ha creado, localizará la lista de materiales de software actualizada en Amazon S3 y actualizará la versión del paquete en el Catálogo de AWS IoT Core Paquetes de Software.

Se requieren las siguientes políticas para realizar esta actualización:

Políticas

- Política de confianza: política que establece una relación de confianza con la entidad principal autorizada que asume el rol de IAM para que pueda localizar la lista de materiales de software actualizada de su bucket con control de versiones en Amazon S3 y actualizar la versión del paquete en AWS IoT Core.

- ```
  {
      "Version":"2012-10-17",
      "Statement": [
          {
              "Effect": "Allow",
              "Principal": {
                  "Service": "s3.amazonaws.com"
          },
              "Action": "sts:AssumeRole"
          }
      ]
  }
  ```

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

- Política de permisos: política para acceder al bucket con control de versiones de Amazon S3 donde se almacena la lista de materiales del software para una versión de paquete y actualizar la versión del paquete en AWS IoT Core.

  - ```
    {
        "Version":"2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": [
                    "s3:GetObject"
                ],
                "Resource": [
                    "arn:aws:s3:::awsexamplebucket1"
                ]
    ```
    ```

```
        }
    ]
}
```

-

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:UpdatePackageVersion"
            ],
            "Resource": [
                "arn:aws:iot:*:111122223333:package/<packageName>/
version/<versionName>"
            ]
        }
    ]
}
```

- Permisos de rol de transferencia: política que otorga permisos para transferir el rol de IAM a Amazon S3 y AWS IoT Core cuando se llama a la operación de UpdatePackageVersion API.

-

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:PassRole",
        "s3:GetObject"
      ],
      "Resource": [
        "arn:aws:s3:::awsexamplebucket1"
      ]
    }
  ]
}
```

-

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

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole",
                "iot:UpdatePackageVersion"
            ],
            "Resource": "arn:aws:iam::111122223333:role/<roleName>"
        }
    ]
}
```

> **Note**
>
> No puede actualizar la lista de materiales de software en una versión de paquete que haya pasado al estado del ciclo de vida `Deleted`.

Para obtener más información sobre la creación de un rol de IAM para un AWS servicio, consulte Crear un rol para delegar permisos a un AWS servicio.

Para obtener más información sobre la creación de un bucket de Amazon S3 y cómo cargar objetos en él, consulte Crear un bucket y Carga de objetos.

Preparación para la indexación de flotas

Con la indexación de AWS IoT flotas, puede buscar y agregar datos mediante el nombre reservado shadow ()`$package`. También puede agrupar AWS IoT cosas consultando los grupos de cosas dinámicos Sombra con nombre reservado y los grupos de cosas dinámicos. Por ejemplo, puede encontrar información sobre qué AWS IoT elementos utilizan una versión de paquete específica, qué no tienen instalada una versión de paquete específica o no tienen ninguna versión de paquete instalada. Puede obtener más información mediante la combinación de atributos. Por ejemplo, identificar cosas que tienen una versión específica y que son de un tipo de cosa específico (como la versión 1.0.0 y el tipo de cosa de pump_sensor). Para obtener más información, consulte Indexación de flotas.

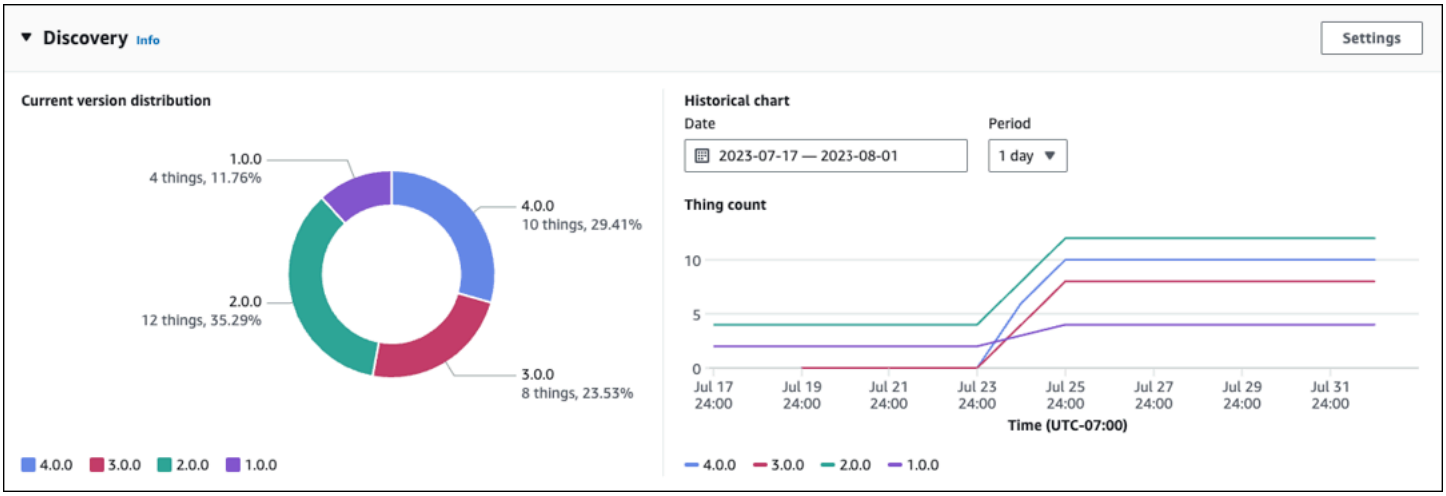
Establecer la sombra **$package** como origen de datos

Para utilizar la indexación de flotas con el Catálogo de paquetes de software, debe habilitar la indexación de flotas, establecer la sombra con nombre como origen de datos y definir `$package` como filtro de sombra con nombre. Si no ha activado la indexación de flotas, puede habilitarla en este proceso. Desde AWS IoT Core en la consola, abra Configuración, elija Gestionar indexación y, a continuación, Añadir sombras con nombre, Añadir paquetes y versiones de software del dispositivo y Actualizar. Para obtener más información, consulte Gestionar la indexación de flotas.

Alternativamente, puede activar la indexación de flotas cuando cree su primer paquete. Cuando aparezca el cuadro de diálogo Habilitar las dependencias para la gestión de paquetes, elija la opción de añadir paquetes y versiones del software del dispositivo como fuentes de datos para la indexación de la flota. Al seleccionar esta opción, también habilita la indexación de la flota.

> **Note**
>
> Habilitar la indexación de flotas para el Catálogo de Paquetes de Software incurre en costes de servicio estándar. Para obtener más información, consulte Precios de AWS IoT Device Management.

Las métricas se muestran en la consola



En la página de detalles del paquete de software de la AWS IoT consola, el panel Discovery muestra las métricas estándar recopiladas de forma `$package` oculta.

- El gráfico de distribución de la versión actual muestra el número de dispositivos y el porcentaje de las 10 versiones más recientes del paquete que están asociadas a un elemento AWS IoT de entre todos los dispositivos asociados a este paquete de software. Nota: Si el paquete de software tiene más versiones de paquete que las indicadas en el gráfico, puede encontrarlas agrupadas en Otras.
- El gráfico histórico muestra la cantidad de dispositivos asociados a las versiones de paquetes seleccionadas durante un período de tiempo específico. Al principio, el gráfico está vacío hasta que seleccione hasta 5 versiones del paquete y defina el rango de fechas y el intervalo de tiempo. Para seleccionar los parámetros del gráfico, elija Configuración. Los datos que se muestran en el gráfico histórico pueden ser diferentes a los del gráfico de distribución de la versión actual debido a la diferencia en el número de versiones de paquetes que se muestran y también a que puede elegir qué versiones de paquetes analizar en el gráfico histórico. Nota: Cuando selecciona una versión de paquete para visualizarla, se tiene en cuenta para el número máximo de límites de las métricas de la flota. Para más información, consulte Límites y cuotas de indexación de flotas.

Para ver otro método para obtener información sobre la recopilación de la distribución de versiones de paquetes, consulte Recopilación de la distribución de versiones de paquetes mediante `getBucketsAggregation`.

Patrones de consulta

La indexación de flotas con el Catálogo de paquetes de software utiliza la mayoría de las características compatibles estándar (como, por ejemplo, términos, frases y campos de búsqueda). Sin embargo, los operadores de comparación (por ejemplo, menor que < y mayor que >) y las consultas de `range` no están disponibles para la clave `version` de sombra con nombre reservado (`$package`). Estas consultas están disponibles para la clave `attributes`. Para obtener más información, consulte Sintaxis de consultas.

Datos de ejemplo

Nota: para obtener información sobre la sombra con nombre reservado y su estructura, consulte Sombra con nombre reservado.

En este ejemplo, se nombra un primer dispositivo `AnyThing` y tiene instalados los siguientes paquetes:

- Paquete de software: `SamplePackage`

  Versión de paquete: `1.0.0`

ID del paquete. 1111

La sombra tiene el siguiente aspecto:

```
{
    "state": {
        "reported": {
            "SamplePackage": {
                "version": "1.0.0",
                "attributes": {
                    "s3UrlForSamplePackage": "https://EXAMPIEBUCKET.s3.us-
west-2.amazonaws.com/exampleCodeFile1",
                    "packageID": "1111"
                    }
            }
        }
    }
}
```

Se nombra un segundo dispositivo `AnotherThing` y tiene instalado el siguiente paquete:

- Paquete de software: `SamplePackage`

  Versión de paquete: `1.0.0`

  ID del paquete. 1111
- Paquete de software: `OtherPackage`

  Versión de paquete: `1.2.5`

  ID del paquete. 2222

La sombra tiene el siguiente aspecto:

```
{
    "state": {
        "reported": {
            "SamplePackage": {
                "version": "1.0.0",
                "attributes": {
```

```
                    "s3UrlForSamplePackage": "https://EXAMPIEBUCKET.s3.us-
west-2.amazonaws.com/exampleCodeFile1",
                    "packageID": "1111"
                }
            },
            "OtherPackage": {
                "version": "1.2.5",
                "attributes": {
                    "s3UrlForOtherPackage": "https://EXAMPIEBUCKET.s3.us-
west-2.amazonaws.com/exampleCodeFile2",
                    "packageID": "2222"
                    }
            },
        }
    }
}
```

Consultas de ejemplo

En la siguiente tabla se muestran ejemplos de consultas basadas en las sombras de los dispositivos de ejemplo para `AnyThing` y `AnotherThing`. Para más información, consulte Ejemplo de consultas sobre cosas.

Última versión de AWS IoT Device Tester for FreeRTOS

Información solicitada	Query	Resultado
Cosas que tienen instalada una versión de paquete específica	`shadow.name.$package.reported.SamplePackage.version:1.0.0`	`AnyThing`, `OtherThing`
Cosas que tienen instalada una versión de paquete específica	`NOT shadow.name.$package.reported.OtherPackage.version:1.2.5`	`AnyThing`
Cualquier dispositivo que utilice una versión de paquete cuyo identificador de paquete sea superior a 1500	`shadow.name.$package.reported.*.attributes.packageID>1500"`	`OtherThing`

Información solicitada	Query	Resultado
Cosas que tienen un paquete específico instalado y tienen más de un paquete instalado	`shadow.name.$package.reported.SamplePackage.version:1.0.0 AND shadow.name.$package.reported.totalCount:2`	`OtherThing`

Recopilación de la distribución de las versiones del paquete mediante **`getBucketsAggregation`**

Además del panel Discovery de la AWS IoT consola, también puede obtener información sobre la distribución de las versiones del paquete mediante la operación de `GetBucketsAggregation`API. Para obtener la información de distribución de la versión del paquete, necesita lo siguiente:

- Defina un campo personalizado en la indexación de flotas para cada paquete de software. Nota: La creación de campos personalizados cuenta para las cuotas de servicio de indexación de la flota AWS IoT.
- Formatee el campo personalizado de la siguiente manera:

  `shadow.name.$package.reported.<packageName>.version`

Para obtener más información, consulta la sección Campos personalizados de la indexación de AWS IoT flotas.

Preparando trabajos AWS IoT

AWS IoT Device Management Software Package Catalog amplía las AWS IoT tareas mediante parámetros de sustitución e integración con la indexación de AWS IoT flotas, los grupos de cosas dinámicos y la AWS IoT cosa reservada llamada shadow.

> **Note**
>
> Para utilizar todas las funciones que ofrece Software Package Catalog, debe crear estas funciones y políticas AWS Identity and Access Management (de IAM): derechos de AWS

IoT trabajo para implementar versiones de paquetes y derechos de AWS IoT trabajo para actualizar la sombra reservada denominada. Para más información, consulte Preparación de la seguridad.

Parámetros de sustitución de trabajos AWS IoT

Puede utilizar los parámetros de sustitución como marcador de posición en el documento de AWS IoT trabajo. Cuando el servicio de trabajos encuentra un parámetro de sustitución, dirige el trabajo al atributo de una versión de software determinada para el valor del parámetro. Puede utilizar este proceso para crear un único documento de trabajo y pasar los metadatos al trabajo mediante atributos de uso general. Por ejemplo, puede pasar una URL de Amazon Simple Storage Service (Amazon S3), un paquete de Nombre de recurso de Amazon (ARN) o una firma en el documento de trabajo mediante los atributos de la versión del paquete.

Los parámetros de sustitución debe tener el siguiente formato en el documento de trabajo:

- Nombre del paquete de software y versión del paquete
  - La cadena vacía entre `package::version` representa el parámetro de sustitución del nombre del paquete de software. La cadena vacía entre `version::attribute` representa el parámetro de sustitución de la versión del paquete de software. Consulte el siguiente ejemplo para utilizar el nombre del paquete y los parámetros de sustitución de la versión de paquete en un documento de trabajo: `${aws:iot:package::version::attributes:<attributekey>}`.
  - El documento de trabajo rellenará automáticamente estos parámetros de sustitución con el ARN de la versión de los detalles de la versión del paquete. Si va a crear un trabajo o una plantilla de trabajo para una implementación de paquete único mediante un comando de la API o la CLI, el ARN de la versión de una versión de paquete se representa con el parámetro `destinationPackageVersions` en `CreateJob` y `DescribeJob`.
- Todos los atributos para una versión de paquete de software
  - Consulte el siguiente ejemplo para utilizar todos los atributos de un parámetro de sustitución de la versión de un paquete de software en un documento de trabajo: `${aws:iot:package:<packageName>:version:<versionName>:attributes}`.

> **Note**
>
> El nombre del paquete, la versión del paquete y todos los parámetros de sustitución de los atributos se pueden utilizar juntos. Consulte el siguiente ejemplo para utilizar los tres parámetros de sustitución en un documento de trabajo: `${aws:iot:package::version::attributes}`.

En el siguiente ejemplo, hay un paquete de software denominado `samplePackage` que tiene una versión de paquete denominada `2.1.5` con los siguientes atributos:

- Nombre: `s3URL`, valor: `https://EXAMPIEBUCKET.s3.us-west-2.amazonaws.com/exampleCodeFile`
  - Este atributo identifica la ubicación del archivo de código almacenado en Amazon S3.
- Nombre: `signature`, valor: `aaaaabbbbbcccccdddddeeeeefffffgggggghhhhhiiiiijjjj`
  - Este atributo proporciona un valor de firma de código que el dispositivo requiere como medida de seguridad. Para obtener más información, consulte Firma de código para trabajos. Nota: Este atributo es un ejemplo y no es obligatorio como parte del catálogo de paquetes de software ni de los trabajos.

En el caso de `s3URL`, el parámetro del documento de trabajo se escribe de la siguiente manera:

```
{
"samplePackage": "${aws:iot:package:samplePackage1:version:2.1.5:attributes:s3URL}"
}
```

En el caso de `signature`, el parámetro del documento de trabajo se escribe de la siguiente manera:

```
{
"samplePackage": "${aws:iot:package:samplePackage1:version:2.1.5:attributes:signature}"
}
```

El documento de trabajo completo está redactado de la siguiente manera:

```
{
  ...
  "Steps": {
    "uninstall": ["samplePackage"],
```

```
    "download": [
      {
        "samplePackage":
 "${aws:iot:package:samplePackage1:version:2.1.5:attributes:s3URL}"
      },
    ],
    "signature": [
      "samplePackage" :
 "${aws:iot:package:samplePackage1:version:2.1.5:attributes:signature}"
    ]
  }
}
```

Una vez realizada la sustitución, se implementa el siguiente documento de trabajo en los dispositivos:

```
{
  ...
  "Steps": {
    "uninstall": ["samplePackage"],
    "download": [
      {
        "samplePackage": "https://EXAMPIEBUCKET.s3.us-west-2.amazonaws.com/
exampleCodeFile"
      },
    ],
    "signature": [
      "samplePackage" : "aaaaabbbbbcccccdddddeeeeeffffffgggggghhhhhiiiiijjjj"
    ]
  }
}
```

Parámetros de sustitución (vista antes y después)

Los parámetros de sustitución simplifican la creación de un documento de trabajo con varios indicadores, como `$default` para la versión de paquete predeterminada. Esto elimina la necesidad de introducir manualmente los metadatos de la versión específica del paquete para cada implementación de trabajo, ya que esos indicadores se rellenan automáticamente con los metadatos a los que se hace referencia en la versión específica del paquete. Para obtener más información sobre los atributos de la versión del paquete, como `$default` para la versión del paquete predeterminada, consulte Preparación del documento de trabajo y la versión del paquete para su implementación.

En él Consola de administración de AWS, active el botón Vista previa de la sustitución de la ventana del editor de archivos de instrucciones de despliegue durante el despliegue de un trabajo por una versión de paquete para ver el documento de trabajo con y sin los parámetros de sustitución.

Si utiliza el parámetro «antes de la sustitución» que aparece en `DescribeJob` y `GetJobDocument` APIs, podrá ver la respuesta de la API antes y después de eliminar los parámetros de sustitución. Consulta los siguientes ejemplos con las letras y: `DescribeJob` `GetJobDocument` APIs

- `DescribeJob`
  - Vista predeterminada

```
{
    "jobId": "<jobId>",
    "description": "<description>",
    "destinationPackageVersions": ["arn:aws:iot:us-west-2:123456789012:package/
TestPackage/version/1.0.2"]
}
```

  - Antes de la vista de sustitución

```
{
    "jobId": "<jobId>",
    "description": "<description>",
    "destinationPackageVersions": ["arn:aws:iot:us-west-2:123456789012:package/
TestPackage/version/$default"]
}
```

- `GetJobDocument`
  - Vista predeterminada

```
{
    "attributes": {
        "location": "prod-artifacts.s3.us-east-1.amazonaws.com/mqtt-core",
        "signature": "IQoJb3JpZ2luX2VjEIrwEaCXVzLWVhc3QtMSJHMEUCIAofPNPpZ9cI",
        "streamName": "mqtt-core",
        "fileId": "0"
    },
}
```

  - Antes de la vista de sustitución

```
{
    "attributes": "${aws:iot:package:TestPackage:version:$default:attributes}",
}
```

Para obtener más información sobre los AWS IoT trabajos, la creación de documentos de trabajo y la implementación de trabajos, consulte Jobs.

Preparación del documento de trabajo y la versión del paquete para su implementación

Cuando se crea una versión de paquete, tiene el estado de `draft`, lo que indica que se está preparando para su implementación. Para preparar la versión del paquete para la implementación, debe crear un documento de trabajo, guardar el documento en una ubicación a la que pueda acceder el trabajo (como Amazon S3) y confirmar que la versión del paquete tiene los valores de atributo que desea que utilice el documento de trabajo. (Nota: Puede actualizar los atributos de una versión del paquete mientras tenga el estado `draft`).

Al crear un AWS IoT trabajo o una plantilla de trabajo para una implementación de paquete único, tiene las siguientes opciones para personalizar su documento de trabajo:

Archivo de instrucciones de implementación (**`recipe`**)

- El archivo de instrucciones de implementación de una versión de paquete contiene las instrucciones de implementación, incluido un documento de trabajo en línea, para implementar una versión de paquete en varios dispositivos. El archivo asocia instrucciones de implementación específicas a una versión de paquete para una implementación rápida y eficiente del trabajo.

  En ella Consola de administración de AWS, puede crear el archivo en la ventana de vista previa del archivo de instrucciones de despliegue de la pestaña Configuraciones de despliegue de versiones del flujo de trabajo de creación de un paquete nuevo. Puede aprovechar AWS IoT para generar automáticamente un archivo de instrucciones a partir de los atributos de la versión de su paquete mediante la opción Comenzar desde el archivo AWS IoT recomendado o utilizar el documento de trabajo existente almacenado en un bucket de Amazon S3 mediante la opción Use your own deployment instruction file.

> **Note**
>
> Si utiliza su propio documento de trabajo, puede actualizarlo directamente en la ventana de Vista previa del archivo de instrucciones de implementación, pero no actualizará automáticamente el documento de trabajo original almacenado en su bucket de Amazon S3.

Cuando utilice el comando AWS CLI o un comando de la API, por ejemplo `CreatePackageVersionGetPackageVersion`, o`UpdatePackageVersion`, `recipe` represente el archivo de instrucciones de despliegue, que incluye un documento de trabajo en línea.

Para obtener más información sobre lo que es un documento de trabajo, consulte Conceptos básicos.

Consulte el siguiente ejemplo para ver el archivo de instrucciones de implementación tal como lo representa `recipe`:

```
{
    "packageName": "sample-package-name",
    "versionName": "sample-package-version",
    ...
    "recipe": "{...}"
}
```

> **Note**
>
> El archivo de instrucciones de implementación tal como lo representa `recipe` se puede actualizar cuando una versión de paquete tiene el estado `published`, ya que es independiente de los metadatos de la versión del paquete. Se vuelve inmutable durante la implementación del trabajo.

Atributo de la versión de **`Artifact`**

- Con el atributo de la version de `artifact` de la versión de su paquete de software, puede añadir la ubicación de Amazon S3 para los artefactos de la versión del

paquete. Cuando se desencadena la implementación de un trabajo para la versión de su paquete con AWS IoT Jobs, el marcador de posición de URL prefirmada `${aws:iot:package:<packageName>:version:<versionName>:artifact-location:s3-presigned-url}` del documento de trabajo se actualizará con el bucket de Amazon S3, la clave del bucket y la versión del archivo almacenado en el bucket de Amazon S3. El bucket de Amazon S3 que almacena los artefactos de la versión de paquete debe estar ubicado en la misma región en la que se creó la versión del paquete.

> **Note**
>
> Para almacenar varias versiones de objetos del mismo archivo en su bucket de Amazon S3, debe activar el control de versiones en su bucket. Para obtener más información, consulte Habilitar el control de versiones en buckets.

Para acceder a los artefactos de la versión del paquete en el bucket de Amazon S3 al utilizar la operación de la API `CreatePackageVersion` o `UpdatePackageVersion`, debe tener los siguientes permisos:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetObjectVersion",
            "Resource": "arn:aws:s3:::bucket-name/key-name"
        }
    ]
}
```

Para obtener más información sobre el atributo de versión `artifact` en las operaciones de la `UpdatePackageVersion` API `CreatePackageVersion` y, consulte CreatePackageVersiony UpdatePackageVersion.

Consulte el siguiente ejemplo, que muestra el atributo de versión `artifact` que admite la ubicación del artefacto en Amazon S3 al crear una nueva versión del paquete:

```
{
    "packageName": "sample package name",
```

```
    "versionName": "1.0",
    "artifact": {
        "s3Location": {
            "bucket": "firmware",
            "key": "image.bin",
            "version": "12345"
        }
    }
}
```

> **Note**
>
> Cuando la versión de paquete se actualiza de un estado `draft` a otro estado `published`, los atributos de la versión del paquete y la ubicación del artefacto se vuelven inmutables. Para actualizar esta información, debe crear una nueva versión del paquete y realizar esas actualizaciones mientras tenga el estado `draft`.

Versión de paquete

- Se puede indicar una versión de paquete de software predeterminada en las versiones disponibles del paquete de software, lo que proporciona una versión de paquete segura y estable. Sirve como versión básica del paquete de software al implementar la versión predeterminada del paquete en su flota de dispositivos mediante AWS IoT Jobs. Al crear un trabajo para implementar la versión de paquete `$default` para un paquete de software, la versión del paquete del documento de trabajo y la de la nueva implementación del trabajo deben coincidir como `$default`. La versión del paquete en la implementación del trabajo está representada por `destinationPackageVersions` para los comandos de la API y la CLI y `VersionARN` en Consola de administración de AWS. La versión del paquete en el documento de trabajo se representa mediante el siguiente marcador de posición del documento de trabajo que se muestra a continuación:

  ```
  arn:aws:iot:<regionCode>:111122223333:package/<packageName>/version/$default
  ```

  Para crear un trabajo o una plantilla de trabajo con la versión de paquete predeterminada, utilice la marca `$default` en el comando de la API `CreateJob` o `CreateJobTemplate`, tal como se muestra a continuación:

```
"$ aws iot create-job \
    --destination-package-versions "arn:aws:iot:us-west-2:123456789012:package/
TestPackage/version/$default"
    --document file://jobdoc.json
```

> **Note**
>
> El atributo de versión de paquete `$default` que hace referencia a la versión predeterminada es un atributo opcional que solo se requiere cuando se hace referencia a la versión del paquete predeterminada para la implementación de un trabajo mediante AWS IoT Jobs.

Cuando esté satisfecho con la versión del paquete, publíquela en la página de detalles del paquete de software de la AWS IoT consola o ejecute la operación de UpdatePackageVersionAPI. A continuación, puede hacer referencia a la versión del paquete al crear el trabajo a través de la AWS IoT consola o emitiendo la operación de CreateJobAPI.

Asignar un nombre a los paquetes y las versiones al desplegarlos

Para implementar una versión de paquete de software en un dispositivo, confirme que el paquete de software y la versión del paquete a los que se hace referencia en el documento de trabajo coincidan con el paquete de software y la versión del paquete indicados en el parámetro `destinationPackageVersions` de la operación de la API `CreateJob`. Si no coinciden, recibirá un mensaje de error en el que se le pedirá que haga que ambas referencias coincidan. Para obtener más información sobre los mensajes de error del Catálogo de paquetes de software, consulte Mensajes de error de la resolución de problemas general.

Además de los paquetes de software y las versiones de paquete a los que se hace referencia en el documento de trabajo, puede incluir paquetes de software y versiones de paquetes adicionales en el parámetro `destinationPackageVersions` de la operación de la API `CreateJob` a los que no se hace referencia en el documento de trabajo. Asegúrese de incluir la información de instalación necesaria en el documento de trabajo para que los dispositivos instalen correctamente las versiones adicionales del paquete de software. Para obtener más información sobre el funcionamiento `CreateJob` de la API, consulte CreateJob.

Segmentar los trabajos mediante grupos de cosas AWS IoT dinámicos

El Catálogo de paquetes de software funciona con la indexación de flotas, los trabajos de AWS IoT y los grupos de objetos dinámicos de AWS IoT para filtrar y dirigir los dispositivos de la flota para seleccionar qué versión de paquete implementar en los dispositivos. Puedes ejecutar una consulta de indexación de la flota en función de la información actual del paquete de tu dispositivo y segmentar esas cosas para un AWS IoT trabajo. También puede lanzar actualizaciones de software, pero sólo a los dispositivos de destino elegibles. Por ejemplo, puede especificar que desea implementar una configuración solo en los dispositivos en los que actualmente se ejecuta `iot-device-client 1.5.09`. Para más información, consulte Crear un grupo de cosas dinámico.

Versiones reservadas de paquetes y sombras con nombre

Si está configurado, AWS IoT Jobs puede actualizar la reserva de una cosa llamada shadow (`$package`) cuando la tarea se complete correctamente. Si lo hace, no necesita asociar manualmente la versión de un paquete a la versión reservada de una cosa llamada shadow.

Puede optar por asociar o actualizar manualmente una versión de paquete a la sombra de nombre reservado de la cosa en las siguientes situaciones:

- Se registra algo AWS IoT Core sin asociar la versión del paquete instalado.
- AWS IoT Jobs no está configurado para actualizar la cosa reservada llamada shadow.
- Utilizas un proceso interno para enviar las versiones de los paquetes a tu flota y ese proceso no AWS IoT Core se actualiza cuando se completa.

> **Note**
>
> Te recomendamos que utilices AWS IoT Jobs para actualizar la versión del paquete en la sección reservada denominada shadow (`$package`). Si se actualiza el parámetro de versión en la `$package` sombra mediante otros procesos (por ejemplo, llamadas a la API manuales o programáticas) y AWS IoT Jobs también está configurado para actualizar la sombra, se pueden producir incoherencias entre la versión real del dispositivo y la versión reportada a la versión reservada denominada shadow.

Puede añadir o actualizar la versión de un paquete a una versión reservada llamada sombra (`$package`) mediante la consola o la operación de la API `UpdateThingShadow`. Para obtener más información, consulte Asociar una versión de paquete a una cosa. AWS IoT

> **Note**
>
> Al asociar una versión de paquete a una AWS IoT cosa, no se actualiza directamente el software del dispositivo. Debe implementar la versión del paquete en el dispositivo para actualizar el software del dispositivo.

Desinstalación de un paquete de software y de su versión

`$null`es un marcador de posición reservado que solicita al servicio AWS IoT Jobs que elimine el paquete de software y la versión del paquete existentes de la sombra reservada denominada del dispositivo. `$package` Para obtener más información, consulte Sombra con nombre reservado.

Para utilizar esta función, sustituya el nombre de la versión al final del destinationPackageVersionAmazon Resource Name (ARN) por. `$null` Después, debe indicar a su servicio que elimine el software del dispositivo.

El ARN autorizado utiliza el siguiente formato:

```
arn:aws:iot:<regionCode>:111122223333:package/<packageName>/version/$null
```

Por ejemplo:

```
$ aws iot create-job \
    ... \
    --destinationPackageVersions ["arn:aws:iot:us-east-1:111122223333:package/
samplePackage/version/$null"]
```

Introducción al Catálogo de paquetes de software

Puede crear y mantener el catálogo de paquetes de AWS IoT Device Management software mediante Consola de administración de AWS las operaciones de AWS IoT Core API y AWS Command Line Interface (AWS CLI).

> **Note**
>
> Habilitar AWS IoT la indexación de la flota es un requisito para utilizar Software Package Catalog. Las operaciones básicas, como la creación de una versión de un paquete de software en el comando API Consola de administración de AWS y el uso del comando `CreatePackage` API, fallarán si no está habilitada la indexación de la AWS IoT flota. Para obtener más información sobre el uso de la indexación de AWS IoT flotas con Software Package Catalog, consultePreparación para la indexación de flotas.

Uso de la consola

Para utilizarla Consola de administración de AWS, inicie sesión en su AWS cuenta y navegue hasta AWS IoT Core. En el panel de navegación, seleccione Paquetes de software. A continuación, puede crear y gestionar los paquetes y sus versiones desde esta sección.

Uso de operaciones de API o CLI

Puede usar las operaciones de la AWS IoT Core API para crear y administrar las funciones del Catálogo de Paquetes de Software. Para obtener más información, consulte la referencia AWS SDKs y los kits de herramientas de las AWS IoT API. Los AWS CLI comandos también administran su catálogo. Para obtener más información, consulte la Referencia de comandos de CLI AWS IoT.

El capítulo contiene las siguientes secciones:

- Crear un paquete de software y una versión del paquete
- Implementación de una versión de paquete mediante trabajos AWS IoT
- Asociar una versión de paquete a cualquier cosa AWS IoT

Crear un paquete de software y una versión del paquete

Puede seguir los siguientes pasos para crear un paquete y una versión inicial mediante la Consola de administración de AWS.

Para crear un paquete de software

1. Inicie sesión en su AWS cuenta y vaya a la AWS IoT consola.
2. En el panel de navegación, seleccione Paquetes de software.

3. En la página del paquete de software AWS IoT , elija Crear paquete. Aparece el cuadro de diálogo Habilitar dependencias para la administración de paquetes.
4. En Indexación de flotas, seleccione Agregar paquetes y versiones de software del dispositivo. Esto es obligatorio para el catálogo de paquetes de software y proporciona indexación de la flota y métricas sobre su flota.
5. [Opcional] Si desea que los AWS IoT trabajos actualicen la sombra con el nombre reservado cuando los trabajos se completen correctamente, seleccione Actualizar automáticamente las sombras de los trabajos. Si no desea que los AWS IoT trabajos realicen esta actualización, deje esta casilla de verificación sin seleccionar.
6. [Opcional] Para conceder a los AWS IoT trabajos los derechos de actualización de la sombra reservada denominada, en Seleccionar función, seleccione Crear función. Si no quieres que AWS IoT Jobs realice esta actualización, este rol no es obligatorio.
7. Crear o seleccionar un rol.

   a. Si no tiene un rol para este propósito: cuando aparezca el cuadro de diálogo Crear rol, introduzca el nombre del rol y, a continuación, elija Crear.

   b. Si tiene un rol para este propósito: en Seleccionar rol, elija su rol y, a continuación, asegúrese de que la casilla asociar política al rol de IAM esté seleccionada.
8. Elija Confirmar. Aparece la página Crear nuevo paquete.
9. En Detalles del paquete, introduzca un nombre de paquete.
10. En la Descripción del paquete, introduzca información que le ayude a identificar y gestionar este paquete.
11. [Opcional] Puede utilizar etiquetas como ayuda para clasificar y administrar este paquete. Para añadir etiquetas, expanda Etiquetas, seleccione Añadir etiqueta e introduzca un par clave-valor. Puede añadir hasta 50 etiquetas. Para obtener más información, consulte Etiquetar AWS IoT los recursos.

Para añadir una versión de paquete al crear un paquete nuevo

1. En Versión inicial, introduzca un Nombre de la versión.

   Recomendamos utilizar el SemVer formato (por ejemplo,`1.0.0.0`) para identificar de forma exclusiva la versión del paquete. También puede utilizar una estrategia de formato diferente que se adapte mejor a su caso de uso. Para obtener más información, consulte Ciclo de vida de la versión de paquete.

2. En la Descripción del paquete, introduzca información que le ayude a identificar y gestionar esta versión del paquete.

> **Note**
>
> La casilla de verificación Versión predeterminada está desactivada porque las versiones de los paquetes se crean en un estado `draft`. Puede asignar un nombre a la versión predeterminada después de crear la versión del paquete y al cambiar el estado a `published`. Para obtener más información, consulte Ciclo de vida de la versión de paquete.

3. [Opcional] Para ayudarle a gestionar esta versión o para comunicar información a sus dispositivos, introduzca uno o más pares nombre-valor para los atributos de Versión. Seleccione Añadir atributo para cada par de nombre-valor que introduzca. Para obtener más información, consulte Atributos de la versión.
4. [Opcional] Puede utilizar etiquetas como ayuda para clasificar y administrar este paquete. Para añadir etiquetas, expanda Etiquetas, seleccione Añadir etiqueta e introduzca un par clave-valor. Puede añadir hasta 50 etiquetas. Para obtener más información, consulte Etiquetar AWS IoT los recursos.
5. Elija Siguiente.

Asociación de la lista de materiales de software con una versión de paquete (opcional)

1. En el paso 3: Versión SBOMs (opcional) de la ventana de configuraciones de la SBOM, elija el formato de archivo de la SBOM y el modo de validación predeterminados que se utilizan para validar la lista de materiales del software antes de asociarla a la versión del paquete.
2. En la ventana Agregar archivo SBOM, introduzca el nombre de recurso de Amazon (ARN) que representa su bucket de Amazon S3 con control de versiones y el formato de archivo SBOM preferido si el tipo predeterminado no funciona.

> **Note**
>
> Puede añadir un único archivo SBOM o un único archivo zip que contenga varios SBOMs si tiene más de una lista de materiales de software para la versión del paquete.

3. En la ventana del archivo Archivo SBOM agregado, puede ver el archivo SBOM que ha agregado para la versión del paquete.

4. Seleccione Crear paquete y versión. Aparece la página de la versión del paquete donde puede ver el estado de validación del archivo SBOM en la ventana Archivo SBOM agregado. El estado inicial será `In progress` cuando el archivo SBOM se someta a validación.

> **Note**
>
> Los estados de validación del archivo SBOM son `Invalid file`, `Not started`, `In progress`, `Validated (SPDX)`, `Validated (CycloneDX)` y los motivos del error de validación.

Implementación de una versión de paquete mediante trabajos AWS IoT

Puede utilizar los siguientes pasos para desplegar una versión del paquete a través de la Consola de administración de AWS.

Requisitos previos:

Antes de comenzar, haga lo siguiente:

- Registra AWS IoT cosas con AWS IoT Core. Para obtener instrucciones sobre cómo añadir tus dispositivos AWS IoT Core, consulta Crear un objeto de objeto.
- [Opcional] Cree un grupo de AWS IoT cosas o un grupo de cosas dinámico para dirigirse a los dispositivos en los que va a implementar la versión del paquete. Para obtener instrucciones sobre cómo crear un grupo de cosas, consulte Crear un grupo de cosas estático. Para obtener instrucciones sobre cómo crear un grupo de cosas dinámico, consulte Crear un grupo de cosas dinámico.
- Cree un paquete de software y una versión del paquete. Para obtener más información, consulte Crear un paquete de software y una versión del paquete.
- Creación de un documento de trabajo Para más información, consulte Preparar el documento de trabajo y la versión del paquete para su implementación.

Para implementar un AWS IoT trabajo

1. En la consola de AWS IoT, elija Paquetes de software.
2. Elija el paquete de software que desee implementar. Aparecerá la página de detalles del paquete de software.

3. Elija la versión del paquete que desea implementar, en Versiones, y elija Implementar la versión del trabajo.
4. Si es la primera vez que implementa un trabajo a través de este portal, aparecerá un cuadro de diálogo en el que se describen los requisitos. Revise la información y seleccione Reconocer.
5. Introduzca un nombre para la implementación o deje el nombre generado automáticamente en el campo Nombre.
6. [Opcional] En el campo Descripción, introduzca una descripción que identifique el propósito o el contenido de la implementación, o deje la información generada automáticamente.

   Nota: Le recomendamos que no utilice información personal identificable en los campos Nombre del puesto y Descripción.
7. [Opcional] Añada las etiquetas que desee asociar a este trabajo.
8. Elija Siguiente.
9. En Objetivos del trabajo, elija las cosas o grupos de cosas que deben recibir el trabajo.
10. En el campo Archivo de trabajo, especifique el archivo JSON del documento de trabajo.
11. Integración de Open Jobs con el servicio Catálogo de paquetes.
12. Seleccione los paquetes y las versiones que se especifican en su documento de trabajo.

> **Note**
>
> Debe elegir los mismos paquetes y versiones de paquetes que se especifican en el documento de trabajo. Puede incluir más, pero el trabajo solo emitirá instrucciones para los paquetes y las versiones incluidos en el documento de trabajo. Para obtener más información, consulte Asignar un nombre a los paquetes y las versiones al implementarlos.

13. Elija Siguiente.
14. En la página Configuración del trabajo, seleccione uno de los siguientes tipos de trabajo en el cuadro de diálogo Configuración del trabajo:

    - Trabajo de instantánea: un trabajo de instantánea está completo cuando termina de ejecutarse en los dispositivos y grupos de destino.
    - Trabajo continuo: un trabajo continuo se aplica a grupos de objetos y se ejecuta en cualquier dispositivo que luego se añada a un grupo de destino específico.

15. En el cuadro de diálogo Configuraciones adicionales (opcional), revise las siguientes configuraciones de trabajo opcionales y seleccione las opciones correspondientes: Para obtener más información, consulte Configuraciones de despliegue, programación y cancelación de trabajos y Configuraciones de tiempo de espera y reintento de ejecución de trabajos.

    - Configuración de despliegue
    - Configuración de programación
    - Configuración del tiempo de espera de ejecuciones de trabajo
    - Configuración de reintento de ejecución de trabajos
    - Configuración de anulación

16. Revise las selecciones de trabajos y, a continuación, elija Enviar.

Después de crear el trabajo, la consola genera una firma JSON y la coloca en su documento del trabajo. Puede usar la AWS IoT consola para ver el estado de un trabajo o cancelar o eliminar un trabajo. Para gestionar los trabajos, vaya al Centro de trabajos de la consola.

Asociar una versión de paquete a cualquier cosa AWS IoT

Después de instalar el software en el dispositivo, puedes asociar una versión de paquete a una AWS IoT cosa reservada llamada shadow. Si se ha configurado AWS IoT jobs para actualizar el objeto reservado denominado shadow después de que el trabajo se haya desplegado y finalizado correctamente, no es necesario que complete este procedimiento. Para obtener más información, consulte Sombra con nombre reservado.

Requisitos previos:

Antes de comenzar, haga lo siguiente:

- Cree una o AWS IoT varias cosas y establezca la telemetría mediante ella. AWS IoT Core Para obtener más información, consulte Cómo empezar con. AWS IoT Core
- Cree un paquete de software y una versión del paquete. Para obtener más información, consulte Crear un paquete de software y una versión del paquete.
- Instale el software de la versión empaquetada en el dispositivo.

> **Note**
>
> Al asociar una versión de paquete a AWS IoT algo, no se actualiza ni instala el software en el dispositivo físico. La versión del paquete debe implementarse en el dispositivo.

Para asociar una versión de paquete a algo AWS IoT

1. En el panel de navegación de la consola de AWS IoT, expanda el menú Todos los dispositivos y seleccione Cosas.
2. Identifique AWS IoT el elemento que desea actualizar en la lista y elija el nombre del elemento para mostrar su página de detalles.
3. En la sección Detalles, seleccione Paquetes y versiones.
4. Seleccione Añadir al paquete y a la versión.
5. En Elegir un paquete de dispositivos, elija el paquete de software que desee.
6. En Elija una versión, elija la versión de software que desee.
7. Seleccione Añadir paquete de dispositivo.

   El paquete y la versión aparecen en la lista de paquetes y versiones seleccionados.
8. Repita estos pasos para cada paquete y versión que desee asociar a esta cosa.
9. Cuando haya terminado, elija Agregar detalles del paquete y la versión. Se abre la página de detalles de la cosa y podrá ver el nuevo paquete y la nueva versión en la lista.

AWS IoT Empleos

Use AWS IoT Jobs para definir un conjunto de operaciones remotas que se pueden enviar a uno o más dispositivos conectados y ejecutarse en ellos AWS IoT. Por ejemplo, puede definir un trabajo que indique a un conjunto de dispositivos descargar e instalar aplicaciones, ejecutar actualizaciones de firmware, reiniciar, rotar certificados o realizar operaciones remotas de solución de problemas.

Acceder a AWS IoT los trabajos

Puede empezar a utilizar AWS IoT Jobs mediante la consola o la AWS IoT Core API.

Uso de la consola

Inicie sesión en la Consola de administración de AWS AWS IoT consola y vaya a ella. En el panel de navegación, elija Administrar y, a continuación, Trabajos. Puede crear y administrar los trabajos desde esta sección. Si desea crear y administrar plantillas de trabajo, en el panel de navegación, elija Plantillas de trabajo. Para obtener más información, consulte Cree y gestione trabajos mediante el Consola de administración de AWS.

Uso de la API o la CLI

Para empezar, utilice las operaciones de la AWS IoT Core API. Para obtener más información, consulte Referencia de la API de AWS IoT. La AWS IoT Core API en la que se basa AWS IoT Jobs es compatible con el AWS SDK. Para obtener más información, consulte AWS SDKs los kits de herramientas.

Puede usar los comandos AWS CLI para ejecutar y crear y administrar trabajos y plantillas de trabajos. Para obtener más información, consulte la Referencia de la CLI de AWS IoT.

AWS IoT Puestos de trabajo, regiones y puntos finales

AWS IoT Jobs admite los puntos finales de la API del plano de control y del plano de datos que son específicos para usted. Región de AWS Los puntos finales de la API del plano de datos son específicos para usted Cuenta de AWS y. Región de AWSPara obtener más información sobre los puntos finales de AWS IoT Jobs, consulte AWS IoT Device Management : puntos finales de datos de trabajos en la AWS Referencia general.

¿Qué es una operación remota?

> **Note**
>
> Over-the-air Las actualizaciones (OTA) no están disponibles en las siguientes regiones:
>
> - Europa (España)
> - Asia-Pacífico (Malasia)

Una operación remota es cualquier actualización o acción que se puede realizar en un dispositivo físico, virtual o punto de conexión y que se puede realizar de forma remota sin necesidad de la presencia física de un operario o un técnico. La operación remota se realiza mediante una actualización over-the-air (OTA) para que los dispositivos no tengan que estar presentes físicamente. Administrar su flota de dispositivos en el Nube de AWS le permite realizar operaciones remotas en sus dispositivos cuando están registrados en ellos AWS IoT Core.

AWS IoT Device Management Jobs ofrece un enfoque escalable para realizar acciones remotas en los dispositivos en los que esté registrado AWS IoT Core. Se crea un trabajo Nube de AWS y se envía a todos los dispositivos de destino mediante una actualización OTA a través del protocolo MQTT o HTTP.

AWS IoT Device Management Los trabajos le permiten realizar operaciones remotas, como el restablecimiento de los valores de fábrica, el reinicio de los dispositivos y las actualizaciones OTA del software, de una manera segura, escalable y más rentable.

Para obtener más información al respecto AWS IoT Core, consulte. ¿Qué es AWS IoT?

Para obtener más información acerca de AWS IoT Device Management Jobs, consulte ¿Qué es Jobs? AWS IoT.

Ventajas de usar AWS IoT Device Management Jobs para operaciones remotas

El uso AWS IoT Device Management de Jobs para realizar sus operaciones remotas agiliza la administración de su flota de dispositivos. En la siguiente lista se destacan algunos de las ventajas clave de usar AWS IoT Device Management Jobs para realizar operaciones remotas:

- Integración perfecta con otros Servicios de AWS

- AWS IoT Device Management Jobs se integra estrechamente con las siguientes características y valor añadido: Servicios de AWS
  - Amazon S3: almacene sus instrucciones de operación remota en un bucket de Amazon S3 seguro desde donde controlar los permisos de acceso a ese contenido. El uso de un bucket de Amazon S3 proporciona una solución de almacenamiento escalable y duradera que se integra de forma nativa con el AWS IoT Device Management Software Package Catalog, lo que permite a AWS IoT Device Management Jobs hacer referencia a las instrucciones de actualización y sustituirlas en ellas. Para obtener más información, consulte ¿Qué es Amazon S3?.
  - Amazon CloudWatch: Supervise y registre el estado de implementación de la operación remota de la ejecución del trabajo para cada dispositivo, además de la actividad de otros dispositivos, para rastrear y analizar el desempeño general del trabajo en AWS IoT Device Management Jobs. Para obtener más información, consulta ¿Qué es Amazon CloudWatch? Supervise los registros de trabajos y capture datos históricos para solucionar problemas. Vea cómo funciona con los trabajos.
  - Servicio de sombra de dispositivo de AWS IoT: Mantén una representación digital de tu dispositivo AWS IoT a través de AWS IoT Device Management Jobs para que las aplicaciones y otros servicios puedan acceder al estado de tu dispositivo, independientemente de la conectividad del dispositivo. Para obtener más información, consulte Servicio de sombra de dispositivo de AWS IoT.
- Prácticas recomendadas de seguridad
  - Control de permisos: controle los permisos de acceso a sus instrucciones de operación remotas mediante Amazon S3 y determine qué usuarios de IAM pueden implementar sus instrucciones de operación remotas en su flota de dispositivos mediante AWS IoT políticas y roles de usuario de IAM.
    - Para obtener más información sobre AWS IoT las políticas, consulte. Cree una AWS IoT política
    - Para obtener más información sobre los roles de IAM, consulte Gestión de identidad y acceso para AWS IoT.
- Escalabilidad
  - Implementación de trabajo específica: controle qué dispositivos reciben el documento de trabajo de un trabajo mediante una implementación de trabajo específica utilizando los criterios de agrupación de dispositivos específicos indicados en el documento de trabajo al crear el trabajo. Al crear AWS IoT un elemento para cada dispositivo y almacenar esa información en el AWS

IoT registro, podrá realizar búsquedas específicas mediante la indexación de flotas. Puede crear grupos personalizados en función de los resultados de búsqueda de la indexación de flotas para respaldar la implementación de trabajo específica. Para obtener más información, consulte Administrar dispositivos con AWS IoT. Utilice los trabajos para realizar trabajos instantáneos en lugar de trabajos continuos.

- Estado del trabajo: realice un seguimiento del estado de la implementación del documento de trabajo en su flota de dispositivos y del estado general del trabajo en el nivel de la flota de dispositivos, además del estado de implementación individual del documento de trabajo en cada dispositivo. Para obtener más información, consulte Trabajos y estados de ejecución de los trabajos.
- Nueva escalabilidad de dispositivos: implemente fácilmente su documento de trabajo en un nuevo dispositivo añadiéndolo a un grupo personalizado existente creado mediante la indexación de flotas en un trabajo continuo. Esto le ahorrará tiempo pues no tendrá que implementar el documento de trabajo en cada dispositivo nuevo por separado. También puede utilizar un enfoque más específico con una instantánea. Para ello, debe implementar un documento de trabajo en un grupo predeterminado de dispositivos una vez finalizado el trabajo.

- Flexibilidad
  - Configuraciones de trabajo: personalice su trabajo y su documento de trabajo con el despliegue, la programación, la cancelación, el tiempo de espera y el reintento de las configuraciones de trabajo opcionales para satisfacer sus necesidades específicas. Para obtener más información, consulte Configuraciones de trabajos.
- Rentabilidad
  - Introduce una estructura de costes más eficiente para el mantenimiento de tu flota de dispositivos aprovechando AWS IoT Device Management Jobs para implementar actualizaciones críticas y realizar tareas de mantenimiento rutinarias. Una solución do-it-yourself (hecha por usted mismo) para mantener su flota de dispositivos incluye costos recurrentes y variables, como la infraestructura necesaria para alojar y administrar la solución de bricolaje, los costos de mano de obra para desarrollar, mantener y escalar la solución de bricolaje y los costos de transmisión de datos. Al aprovechar la estructura transparente y de costes fijos de AWS IoT Device Management Jobs, sabrá exactamente cuánto costará la ejecución de cada trabajo para un dispositivo, además de los costes de transmisión de datos necesarios para facilitar la distribución de los documentos de trabajo en su flota de dispositivos y el seguimiento del estado de ejecución de los trabajos de cada dispositivo. Para obtener más información, consulte Precios de AWS IoT Core.

¿Qué es Jobs? AWS IoT

Use AWS IoT Jobs para definir un conjunto de operaciones remotas que se pueden enviar a uno o más dispositivos conectados y ejecutarse en ellos AWS IoT.

Para crear trabajos, defina primero un documento de trabajo que contenga una lista de instrucciones que describan las operaciones que el dispositivo debe realizar de forma remota. Para realizar estas operaciones, especifique una lista de destinos que sean objetos individuales, grupos de objetos o ambos. El documento de trabajo y los destinos constituyen juntos una implementación.

Cada implementación puede tener configuraciones adicionales:

- Despliegue: esta configuración define cuántos dispositivos reciben el documento de trabajo cada minuto.
- Anular: si un número determinado de dispositivos no reciben la notificación de trabajo, utilice esta configuración para cancelar el trabajo. Esto evita enviar una actualización incorrecta a toda la flota.
- Tiempo de espera: si no se recibe una respuesta de los destinos de trabajo en un plazo determinado, el trabajo puede fallar. Puede realizar un seguimiento del trabajo que se está ejecutando en estos dispositivos.
- Reintentar: si un dispositivo informa de un fallo o se agota el tiempo de espera de un trabajo, puede usar AWS IoT Jobs para volver a enviar el documento de trabajo al dispositivo automáticamente.
- Programación: esta configuración le permite programar un trabajo para una fecha y hora futuras. También le permite crear períodos de mantenimiento periódicos que actualizan los dispositivos durante períodos predefinidos de poco tráfico.

AWS IoT Jobs envía un mensaje para informar a los destinatarios de que hay un trabajo disponible. El objetivo inicia la ejecución del trabajo descargando el documento del trabajo, realizando las operaciones que especifique e informando sobre su progreso AWS IoT. Puede realizar un seguimiento del progreso de un trabajo para un objetivo específico o para todos los objetivos ejecutando los comandos proporcionados por AWS IoT Jobs. Cuando se inicia un trabajo, tiene el estado En curso. A continuación, los dispositivos notifican las actualizaciones incrementales y muestran este estado hasta que el trabajo se complete correctamente, se produzca un error o se agote el tiempo de espera.

En los temas siguientes se describen algunos conceptos clave de los trabajos y del ciclo de vida de estos y de su ejecución.

Temas

- Conceptos clave de trabajos
- Trabajos y estados de ejecución de los trabajos

Conceptos clave de trabajos

Los siguientes conceptos proporcionan detalles sobre los AWS IoT trabajos y sobre cómo crear e implementar trabajos para ejecutar operaciones remotas en sus dispositivos.

Conceptos básicos

Los siguientes son conceptos básicos que debe conocer al usar AWS IoT Jobs.

Trabajo

Un trabajo es una operación remota que se envía a uno o varios dispositivos conectados al AWS IoT y que se ejecuta en ellos. Por ejemplo, puede definir un trabajo que indique a un conjunto de dispositivos descargar e instalar una aplicación o ejecutar actualizaciones de firmware, reiniciar, rotar certificados o realizar operaciones remotas de solución de problemas.

Documento de trabajo

Para crear un trabajo, primero debe crear un documento de trabajo, que es una descripción de las operaciones remotas que deben realizar los dispositivos.

Los documentos de trabajo son documentos JSON con codificación UTF-8 y contienen la información que necesitan los dispositivos para realizar un trabajo. Un documento de trabajo contiene uno o varios documentos URLs en los que el dispositivo puede descargar una actualización u otros datos. El documento de trabajo puede almacenarse en un bucket de Amazon S3 o insertarse con el comando que crea el trabajo.

> **Note**
>
> En las respuestas de MQTT, el campo `jobDocument` es un objeto JSON. En las respuestas de HTTP, es una representación de cadena del objeto JSON.

Destino

Cuando cree un trabajo, debe especificar una lista de destinos que son los dispositivos que deben realizar las operaciones. Los destinos pueden ser objetos, grupos de objetos o ambos. El servicio de AWS IoT trabajos envía un mensaje a cada destino para informarle de que hay un trabajo disponible.

Implementación

Tras crear un trabajo proporcionando el documento de trabajo y especificando la lista de destinos, el documento de trabajo se implementa en los dispositivos de destino remotos para los que desee realizar la actualización. En el caso de los trabajos instantáneos, el trabajo se completará después de implementarlo en los dispositivos de destino. En el caso de los trabajos continuos, un trabajo se implementa en un grupo de dispositivos a medida que se agregan a los grupos.

Ejecución de trabajo

Una ejecución de trabajo es una instancia de un trabajo en un dispositivo de destino. El destino inicia una ejecución de trabajo descargando el documento de trabajo. A continuación, realiza las operaciones especificadas en el documento e informa de su progreso a AWS IoT. Un número de ejecución es un identificador único de una ejecución de trabajo en un destino específico. El servicio AWS IoT Jobs proporciona comandos para realizar un seguimiento del progreso de la ejecución de un trabajo en un objetivo y del progreso de un trabajo en todos los objetivos.

Conceptos de tipos de trabajo

Los siguientes conceptos pueden ayudarle a comprender mejor los distintos tipos de trabajos que puede crear con AWS IoT Jobs.

Trabajo de instantánea

De forma predeterminada, un trabajo se envía a todos los destinos especificados al crearlo. Después de que esos destinos finalicen el trabajo (o informen de que no pueden hacerlo), el trabajo se completa.

Trabajo continuo

Un trabajo continuo se envía a todos los destinos especificados al crearlo. Sigue ejecutándose y se envía a todos los nuevos dispositivos (objetos) que se añaden al grupo de destino. Por ejemplo, puede usarse un trabajo continuo para incorporar o actualizar dispositivos a medida que se añaden a un grupo. Puede hacer que un trabajo sea continuo estableciendo un parámetro opcional al crearlo.

> **Note**
>
> Cuando se dirija a su flota de IoT mediante grupos de objetos dinámicos, recomendamos que utilice trabajos continuos en lugar de trabajos instantáneos. Al utilizar trabajos continuos, los dispositivos que se unen al grupo reciben la ejecución del trabajo incluso después de que este se haya creado.

Prefirmado URLs

Para acceder de forma segura y por tiempo limitado a los datos que no están incluidos en el documento de trabajo, puede utilizar Amazon S3 prefirmado. URLs Coloque los datos en un bucket de Amazon S3 y añadir un enlace de marcador de posición para los datos en el documento de trabajo. Cuando AWS IoT Jobs recibe una solicitud del documento de trabajo, analiza el documento de trabajo buscando los enlaces de los marcadores de posición y, a continuación, los reemplaza por Amazon S3 prefirmado. URLs

El enlace de marcador de posición tiene el formato siguiente:

```
${aws:iot:s3-presigned-url:https://s3.amazonaws.com/bucket/key}
```

donde *`bucket`* está el nombre del bucket y el objeto del bucket al que se *`key`* está enlazando.

En las regiones de Beijing y Ningxia, los prefirmados solo URLs funcionan si el propietario del recurso tiene una licencia ICP (proveedor de contenido de Internet). Para obtener más información, consulte Amazon Simple Storage Service en la documentación Introducción a AWS los servicios en China.

Conceptos de configuración de trabajo

Los siguientes conceptos pueden ayudarlo a comprender cómo configurar los trabajos.

Despliegues

Puede especificar la rapidez con la que se notifica a los destinos la ejecución de un trabajo pendiente. Esto le permite crear un despliegue por etapas para administrar mejor las actualizaciones, los reinicios y otras operaciones. Puede crear una configuración de despliegue mediante una tasa de despliegue estática o una tasa de despliegue exponencial. Para especificar el número máximo de destinos de trabajo de los que se informará por minuto, utilice una velocidad de despliegue estática.

Para ver ejemplos de cómo establecer las velocidades de despliegue y obtener más información sobre la configuración de los despliegues de trabajos, consulte Configuraciones de despliegue, programación y cancelación de trabajos.

Programación

La programación de trabajos le permite programar el plazo de despliegue de un documento de trabajo en todos los dispositivos del grupo de destino para trabajos continuos e instantáneos. Además, puede crear un periodo de mantenimiento opcional que contenga fechas y horas específicas en las que un trabajo distribuirá el documento de trabajo a todos los dispositivos del grupo de destino. Un periodo de mantenimiento es una instancia recurrente con una frecuencia de fechas y horas diarias, semanales, mensuales o personalizadas seleccionadas durante la creación inicial del trabajo o de la plantilla de trabajo. Solo los trabajos continuos se pueden programar para que se desplieguen durante un periodo de mantenimiento.

La programación de trabajos es específica para su trabajo. No se pueden programar ejecuciones de trabajos individuales. Para obtener más información, consulte Configuraciones de despliegue, programación y cancelación de trabajos.

Anular

Puede crear un conjunto de condiciones para cancelar despliegues una vez se cumplan los criterios que especifique. Para obtener más información, consulte Configuraciones de despliegue, programación y cancelación de trabajos.

Tiempos de espera

Los tiempos espera de trabajos lo notifican cada vez que la implementación de un trabajo se bloquea en el estado `IN_PROGRESS` durante un periodo de tiempo más largo de lo previsto. Existen dos tipos de temporizadores: temporizadores en curso y temporizadores de pasos. Una vez el trabajo está `IN_PROGRESS`, puede monitorizar y realizar un seguimiento del progreso de la implementación del trabajo.

Las configuraciones de implementación y anulación son específicas de su trabajo, mientras que la configuración del tiempo de espera es específica de la implementación de un trabajo. Para obtener más información, consulte Configuraciones de tiempo de espera y reintento de ejecución de trabajos.

Reintentos

Los reintentos de trabajo permiten reintentar la ejecución del trabajo cuando se produce un error en un trabajo, se agota el tiempo de espera o ambos. Puede disponer de hasta 10 reintentos

para ejecutar el trabajo. Puede monitorizar y realizar un seguimiento del progreso del reintento y comprobar si la ejecución del trabajo se ha realizado correctamente.

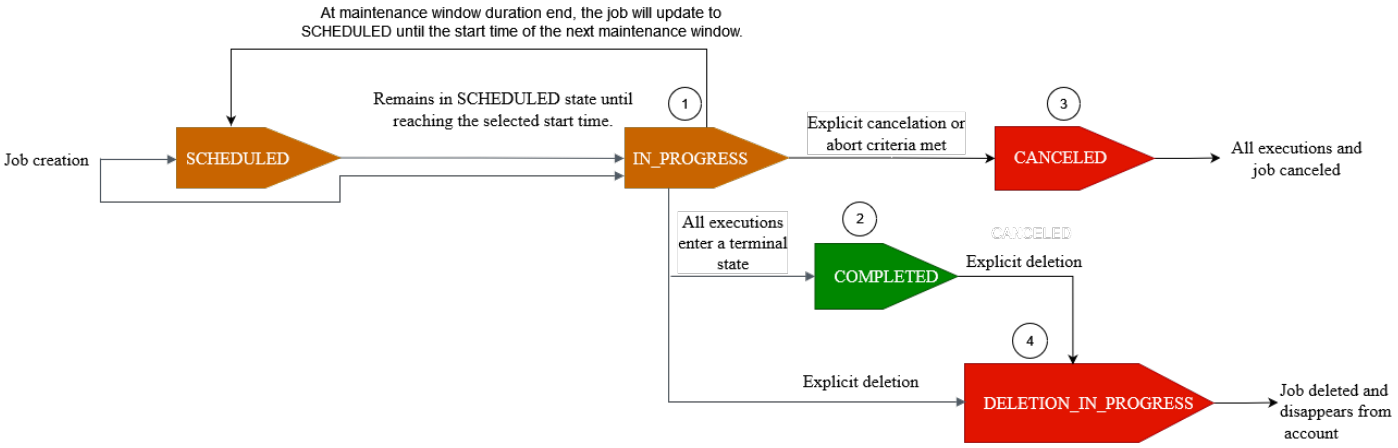
Las configuraciones de implementación y anulación son específicas de su trabajo, mientras que las configuraciones del tiempo de espera y de reintento son específicas de la ejecución de un trabajo. Para obtener más información, consulte Configuraciones de tiempo de espera y reintento de ejecución de trabajos.

Trabajos y estados de ejecución de los trabajos

En las siguientes secciones se describe el ciclo de vida de un AWS IoT trabajo y el ciclo de vida de la ejecución de un trabajo.

Estados de trabajo

El siguiente diagrama muestra los diferentes estados de un AWS IoT trabajo.



Un trabajo que cree con AWS IoT Jobs puede estar en uno de los siguientes estados:

- SCHEDULED

  Durante la creación inicial del trabajo o de la plantilla de trabajo mediante la AWS IoT consola, CreateJobla CreateJobTemplateAPI o la API, puede seleccionar la configuración de programación opcional en la AWS IoT consola o `SchedulingConfig` en la CreateJobAPI o CreateJobTemplateAPI. Al iniciar un trabajo programado que contiene un `startTime`, `endTime` y `endBehavior` específicos y el estado del trabajo se actualiza a `SCHEDULED`. Cuando el trabajo llegue al `startTime` seleccionado o al `startTime` del siguiente periodo de mantenimiento (si seleccionó el despliegue del trabajo durante dicho periodo), el estado se actualizará de

`SCHEDULED` a `IN_PROGRESS` y comenzará a desplegarse el documento de trabajo en todos los dispositivos del grupo de destino.

- IN_PROGRESS

  Al crear un trabajo mediante la AWS IoT consola o la CreateJobAPI, el estado del trabajo se actualiza a`IN_PROGRESS`. Durante la creación del trabajo, Jobs de AWS IoT comienza a desplegar las ejecuciones de trabajo en los dispositivos del grupo de destino. Una vez desplegadas todas las ejecuciones de trabajo, Jobs de AWS IoT espera a que los dispositivos completen la acción remota.

  Para obtener información sobre la simultaneidad y los límites que se aplican a los trabajos en curso, consulte AWS IoT Límites de puestos.

  > **Note**
  >
  > Cuando un trabajo de `IN_PROGRESS` llegue al final del periodo de mantenimiento actual, se detendrá el despliegue del documento del trabajo. El trabajo se actualizará `SCHEDULED` hasta el `startTime` del siguiente periodo de mantenimiento.

- COMPLETED

  Un trabajo continuo se gestiona de una de las siguientes formas:

  - En el caso de un trabajo continuo sin la configuración de programación opcional seleccionada, siempre está en curso y sigue ejecutándose para cualquier dispositivo nuevo que se añada al grupo de destino. Nunca alcanzará el estado `COMPLETED`.
  - En el caso de un trabajo continuo con la configuración de programación opcional seleccionada, se cumple lo siguiente:
    - Si se ha proporcionado un `endTime`, un trabajo continuo alcanzará el estado `COMPLETED` cuando el `endTime` haya pasado y todas las ejecuciones de trabajos hayan alcanzado un estado terminal.
    - Si no se ha proporcionado un `endTime` en la configuración de programación opcional, el trabajo continuo seguirá realizando el despliegue del documento de trabajo.

  En el caso de un trabajo instantáneo, el estado del trabajo cambia a `COMPLETED` cuando todas sus ejecuciones pasan a un estado terminal, como `SUCCEEDED`, `FAILED`, `TIMED_OUT`, `REMOVED` o `CANCELED`.

- CANCELADO

Al cancelar un trabajo mediante la AWS IoT consola, la CancelJobAPI o laConfiguración de anulación del trabajo, el estado del trabajo cambia a`CANCELED`. Durante la cancelación de un trabajo, AWS IoT Jobs comienza a cancelar las ejecuciones de trabajos creados anteriormente.

Para obtener información sobre la simultaneidad y los límites que se aplican a los trabajos que se van a cancelar, consulte AWS IoT Límites de puestos.

- DELETION_IN_PROGRESS

  Al eliminar un trabajo mediante la AWS IoT consola o la DeleteJobAPI, el estado del trabajo cambia a`DELETION_IN_PROGRESS`. Durante la eliminación de un trabajo, AWS IoT Jobs comienza a eliminar las ejecuciones de trabajos previamente creadas. Una vez que se hayan eliminado todas las ejecuciones de trabajos, el trabajo desaparecerá de su AWS cuenta.

Estados de ejecución de trabajos

En la siguiente tabla se muestran los diferentes estados de la ejecución de una AWS IoT tarea y si el cambio de estado lo inicia el dispositivo o AWS IoT Jobs.

Estados y origen de ejecución de trabajos

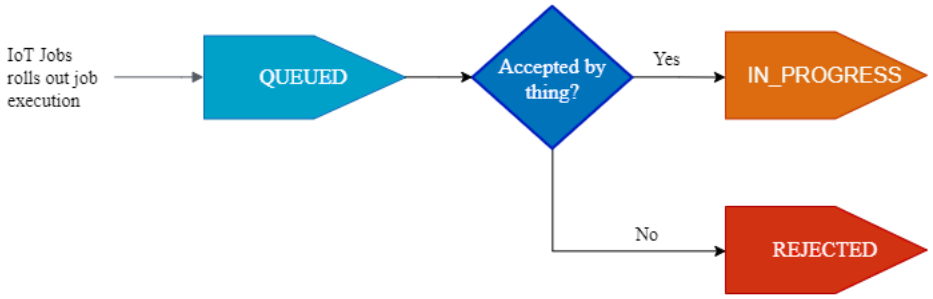
Estado de ejecución de trabajos	¿Iniciado por un dispositivo?	¿Iniciado por AWS IoT Jobs?	¿Estado terminal?	¿Se puede reintentar?
`QUEUED`	No	Sí	No	No aplicable
`IN_PROGRESS`	Sí	No	No	No aplicable
`SUCCEEDED`	Sí	No	Sí	No aplicable
`FAILED`	Sí	No	Sí	Sí
`TIMED_OUT`	No	Sí	Sí	Sí
`REJECTED`	Sí	No	Sí	No
`REMOVED`	No	Sí	Sí	No
`CANCELED`	No	Sí	Sí	No

En la siguiente sección se describe más información sobre los estados de la ejecución de un trabajo que se implementa al crear un trabajo con AWS IoT Jobs.

- QUEUED

  Cuando AWS IoT Jobs despliega la ejecución de una tarea para un dispositivo de destino, el estado de ejecución de la tarea se establece en`QUEUED`. La ejecución del trabajo permanece en el estado `QUEUED` hasta que:

  - El dispositivo recibe la ejecución del trabajo, invoca las operaciones de la API de trabajos e informa del estado como `IN_PROGRESS`.
  - Se cancela el trabajo o la ejecución del trabajo, o cuando se cumplen los criterios de anulación especificados y el estado cambia a `CANCELED`.
  - El dispositivo se elimina del grupo de destino y el estado cambia a `REMOVED`.



- IN_PROGRESS

  Si su dispositivo IoT se suscribe a la reserva Temas de trabajos `$notify` y`$notify-next`, además, invoca la `StartNextPendingJobExecution` API o la `UpdateJobExecution` API con el estado de`IN_PROGRESS`, AWS IoT Jobs establecerá el estado de ejecución del trabajo en. `IN_PROGRESS`

  La API de `UpdateJobExecution` se puede invocar varias veces con un estado `IN_PROGRESS`. Puede especificar detalles adicionales sobre los pasos de ejecución mediante el objeto `statusDetails`.

  > **Note**
  >
  > Si creas varios trabajos para cada dispositivo, AWS IoT Jobs y el protocolo MQTT no garantizan el orden de entrega.

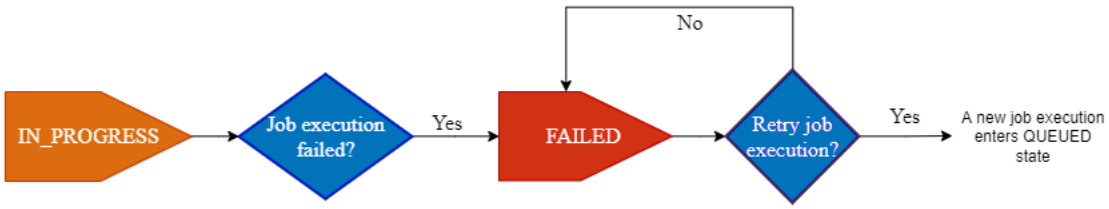
- SUCCEEDED

Cuando el dispositivo complete correctamente la operación remota, deberá invocar la `UpdateJobExecution` API con un estado igual `SUCCEEDED` a para indicar que la ejecución del trabajo se realizó correctamente. AWS IoT A continuación, Jobs actualiza y devuelve el estado de ejecución del trabajo como`SUCCEEDED`.



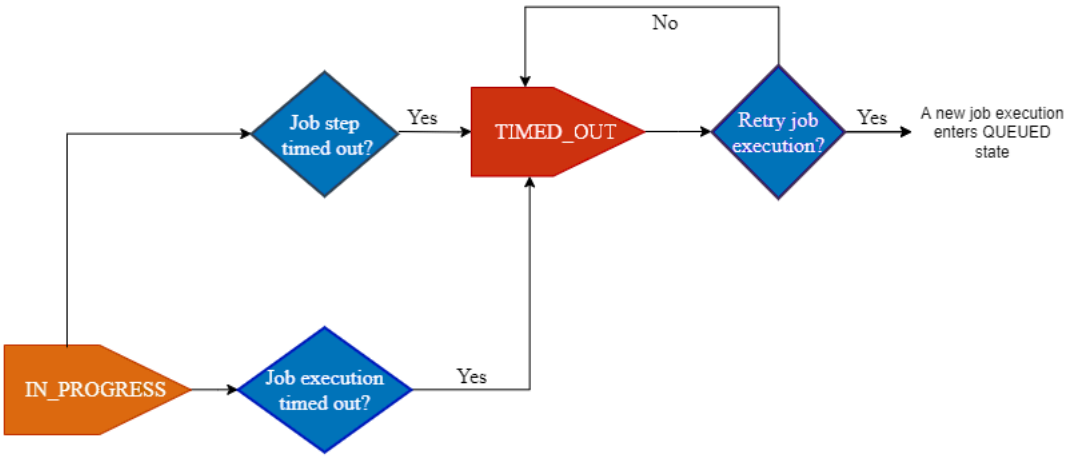
- ERROR

  Si el dispositivo no puede completar la operación remota, debe invocar la `UpdateJobExecution` API con un estado igual `Failed` a para indicar que se ha producido un error en la ejecución de la tarea. AWS IoT A continuación, Jobs actualiza y devuelve el estado de ejecución del trabajo como`Failed`. Mediante la Configuración de reintento de ejecución de trabajos se puede reintentar la ejecución de este trabajo en el dispositivo.



- TIMED_OUT

  Si el dispositivo no puede completar un paso de la tarea en ese estado`IN_PROGRESS`, o si no puede completar la operación remota dentro del tiempo de espera del temporizador en curso, AWS IoT Jobs establece el estado de ejecución de la tarea en. `TIMED_OUT` También se dispone de un temporizador de pasos para cada paso de un trabajo en curso, que solo se aplica a la ejecución del trabajo. La duración del temporizador en curso se especifica mediante la propiedad `inProgressTimeoutInMinutes` de la Configuración del tiempo de espera de las ejecuciones de trabajo. Mediante la Configuración de reintento de ejecución de trabajos se puede reintentar la ejecución de este trabajo en el dispositivo.



- REJECTED

  Cuando el dispositivo recibe una solicitud no válida o incompatible, debe invocar la `UpdateJobExecution` API con el estado de. `REJECTED` AWS IoT A continuación, Jobs actualiza y devuelve el estado de ejecución del trabajo como`REJECTED`.

- REMOVED

  Cuando el dispositivo deja de ser un destino válido para la ejecución del trabajo, por ejemplo, cuando está separado de un grupo de objetos dinámico, Jobs de AWS IoT establece el estado de ejecución del trabajo en `REMOVED`. Puede volver a asociar el dispositivo al grupo de destino y reiniciar la ejecución del trabajo en el dispositivo.

- CANCELADO

  Cuando se cancela un trabajo o se cancela la ejecución de un trabajo mediante la consola `CancelJob` o la `CancelJobExecution` API, o cuando se cumplen los criterios de anulación especificados mediante elConfiguración de anulación del trabajo, AWS IoT Jobs cancela el trabajo y establece el estado de ejecución del trabajo en. `CANCELED`

Administración de trabajos

Utilice los trabajos para notificar a los dispositivos sobre una actualización de software o firmware. Puede utilizar la AWS IoT consola, la Operaciones de la API de administración y control del trabajo AWS Command Line Interface, la o la AWS SDKspara crear y gestionar trabajos.

Firma de código para trabajos

Al enviar código a los dispositivos, para que los dispositivos detecten si el código se ha modificado durante el tránsito, recomendamos firmar el archivo de códigos utilizando la AWS CLI. Para obtener instrucciones, consulte Creación y administración de trabajos mediante la AWS CLI.

Para obtener más información, consulte ¿Para qué sirve la firma de código AWS IoT? .

Documento de trabajo

Antes de crear un trabajo debe crear un documento del trabajo. Si utiliza la firma de código AWS IoT, debe cargar su documento de trabajo en un bucket de Amazon S3 versionado. Para obtener más información acerca de cómo crear un bucket de Amazon S3 y cargar archivos en él, consulte Introducción a Amazon Simple Storage Service en la Guía de introducción a Amazon S3.

> **Tip**
>
> Para ver ejemplos de documentos de trabajo, consulte el ejemplo de jobs-agent.js en el AWS IoT SDK de JavaScript.

Prefirmado URLs

Su documento del trabajo puede contener una URL de Amazon S3 que apunte a su archivo de código (u otro archivo). Los Amazon S3 prefirmados URLs son válidos solo durante un período de tiempo limitado y se generan cuando un dispositivo solicita un documento de trabajo. Dado que la URL prefirmada no se crea cuando se crea el documento del trabajo, en su lugar se usa una URL de marcador de posición en el documento de trabajo. La URL de un marcador de posición tiene un aspecto similar al siguiente:

```
${aws:iot:s3-presigned-url-v2:https://
s3.region.amazonaws.com/<bucket>/<code file>}
```

donde:

- *bucket*es el bucket de Amazon S3 que contiene el archivo de código.
- *code file*es la clave Amazon S3 del archivo de código.

Cuando un dispositivo solicita el documento de trabajo, AWS IoT genera la URL prefirmada y reemplaza la URL del marcador de posición por la URL prefirmada. El documento del trabajo se envía al dispositivo.

Rol de IAM para conceder el permiso de descarga de archivos desde S3

Al crear un trabajo que utiliza Amazon S3 prefirmado URLs, debe proporcionar un rol de IAM. El rol debe conceder permiso para descargar archivos desde el bucket de Amazon S3 en el que se almacenaron los datos o las actualizaciones. El rol debe conceder permiso también para que AWS IoT asuma el rol.

Puede especificar un tiempo de espera opcional para la URL prefirmada. Para obtener más información, consulte CreateJob.

Conceda permiso a AWS IoT Jobs para que asuma su función

1. Vaya al Centro de roles de la consola de IAM y elija su rol.
2. En la pestaña Relaciones de confianza, seleccione Editar la relación de confianza y sustituya el documento de política por el siguiente JSON. Elija Actualizar la política de confianza.

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

3. Para protegerse contra el problema de la sustitución confusa, agregue las claves contextuales de condición global `aws:SourceArn` y `aws:SourceAccount` a la política.

> **Important**
>
> El `aws:SourceArn` debe ajustarse al formato: `arn:aws:iot:`*`region`*`:`*`account-id`*`:*`. Asegúrese de que *`region`* coincida con su AWS IoT región y *`account-id`* con el ID de su cuenta de cliente. Para obtener más información, consulte Prevención de la sustitución confisa entre servicios.

```
{
  "Effect": "Allow",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service":
          "iot.amazonaws.com"
       },
      "Action": "sts:AssumeRole",
      "Condition": {
         "StringEquals": {
            "aws:SourceAccount": "123456789012"
         },
         "ArnLike": {
              "aws:SourceArn": "arn:aws:iot:*:123456789012:job/*"
         }
       }
     }
   ]
}
```

4. Si su trabajo utiliza un documento de trabajo que es un objeto de Amazon S3, elija Permisos y utilice el siguiente JSON. Esto añade una política que concede permiso para descargar archivos de su bucket de Amazon S3:

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

```
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::your_S3_bucket/*"
        }
    ]
}
```

URL prefirmada para la carga de archivos

Si sus dispositivos necesitan cargar archivos en un bucket de Amazon S3 durante la implementación de un trabajo, puede incluir el siguiente marcador de posición de URL prefirmada en su documento de trabajo:

```
${aws:iot:s3-presigned-url-upload:https://s3.region.amazonaws.com/<bucket>/<key>}
```

Puede utilizar un máximo de dos palabras de cada una de las palabras clave reservadas `${thingName}`, `${jobId}` y `${executionNumber}` dentro del atributo `key` de la URL del marcador de carga de archivos que se encuentra en su documento de trabajo. El marcador de posición local que representa esas palabras clave reservadas en el atributo `key` se analizará y reemplazará cuando se cree la ejecución del trabajo. Al usar un marcador de posición local con palabras clave reservadas específicas para cada dispositivo se garantiza que cada archivo cargado desde un dispositivo sea específico de ese dispositivo y no se sobrescriba con un archivo cargado similar desde otro dispositivo al que se dirija la misma implementación de trabajos. Para obtener más información sobre cómo solucionar problemas con los marcadores de posición locales incluidos en un marcador de posición de URL prefirmada para cargar archivos durante la implementación de un trabajo, consulte Mensajes de error de la resolución de problemas general.

> **Note**
>
> El nombre del bucket de Amazon S3 no puede contener el marcador de posición local que representa las palabras clave reservadas para el archivo cargado. El marcador de posición local debe estar ubicado en el atributo `key`.

Este marcador de posición de URL prefirmada se convertirá en una URL de carga prefirmada de Amazon S3 en el documento de trabajo cuando lo reciba un dispositivo. Los dispositivos lo utilizarán para cargar archivos en un bucket de Amazon S3 de destino.

> **Note**
>
> Si el bucket y la clave de Amazon S3 no aparecen en la URL del marcador de posición anterior, AWS IoT Jobs generará automáticamente una clave para cada dispositivo con un máximo de dos de los siguientes valores `${thingName}`, `${jobId}` y `${executionNumber}`.

URL prefirmada mediante el control de versiones de Amazon S3

Proteger la integridad de un archivo almacenado en un bucket de Amazon S3 es fundamental para garantizar una implementación segura de los trabajos con ese archivo en la flota de dispositivos. Al usar el control de versiones de Amazon S3, puede añadir un identificador de versión para cada variante del archivo almacenado en su bucket de Amazon S3 para realizar un seguimiento de cada versión del archivo. Esto proporciona información sobre qué versión del archivo se implementa en su flota de dispositivos mediante AWS IoT Jobs. Para obtener más información sobre el control de versiones en buckets de Amazon S3, consulte Uso de control de versiones en buckets de Amazon S3.

Si el archivo está almacenado en Amazon S3 y el documento de trabajo contiene un marcador de posición de URL prefirmado, AWS IoT Jobs generará una URL prefirmada en el documento de trabajo utilizando el bucket de Amazon S3, la clave del bucket y la versión del archivo almacenado en el bucket de Amazon S3. Esta URL prefirmada generada en el documento de trabajo sustituirá al marcador de posición de URL prefirmada originalmente en el documento de trabajo. Si actualiza el archivo almacenado en su bucket de Amazon S3, se crearán una nueva versión del archivo y versiones posteriores de `versionId` para indicar las actualizaciones realizadas y ofrecer la posibilidad de dirigirse a ese archivo específico en futuras implementaciones de trabajos.

Consulte los siguientes ejemplos para ver antes y durante el Amazon S3 prefirmado URLs en su documento de trabajo mediante: `versionId`

Marcador de posición de URL prefirmada de Amazon S3 (antes de la implementación del trabajo)

```
//Virtual-hosted style URL
${aws:iot:s3-presigned-url-v2:https://bucket-name.s3.region-code.amazonaws.com/key-name%3FversionId%3Dversion-id}

//Path-style URL
```

```
${aws:iot:s3-presigned-url-v2:https://s3.region-code.amazonaws.com/bucket-name/key-name%3FversionId%3Dversion-id}
```

URL prefirmada de Amazon S3 (durante la implementación del trabajo)

```
//Virtual-hosted style URL
${aws:iot:s3-presigned-url-v2:https://sample-bucket-name.s3.us-west-2.amazonaws.com/sample-code-file.png%3FversionId%3Dversion1}

//Path-style
${aws:iot:s3-presigned-url-v2:https://s3.us-west-2.amazonaws.com/sample-bucket-name/sample-code-file.png%3FversionId%3Dversion1}
```

Para obtener más información sobre los objetos de tipo ruta y hospedados virtualmente en Amazon S3 URLs, consulte Virtual-hosted-style solicitudes y solicitudes de tipo Path.

> **Note**
>
> Si quiere añadir un `versionId` a una URL prefirmada de Amazon S3, deberá adaptarse a la codificación de URL compatible con AWS SDK for Java 2.x. Para obtener más información, consulte Cambios en el análisis de Amazon S3 URIs de la versión 1 a la versión 2.

Diferencias de versión de marcador de posición de URL prefirmada de Amazon S3

En la siguiente lista se describen las diferencias entre los marcadores de posición de URL prefirmada de Amazon S3 `${aws:iot:s3-presigned-url-v1` (versión 1) y `${aws:iot:s3-presigned-url-v2` (versión 2):

- El marcador de posición de URL prefirmada de Amazon S3 `${aws:iot:s3-presigned-url-v1` no es compatible con `version-id`.
- El marcador de posición de URL prefirmada de Amazon S3 `${aws:iot:s3-presigned-url-v1` recibe la URL de Amazon S3 sin codificar. El marcador de posición de URL prefirmada de Amazon S3 `${aws:iot:s3-presigned-url-v2` requiere que la URL de Amazon S3 esté codificada para cumplir con el estándar del SDK de Amazon S3.

Temas

- Cree y gestione trabajos mediante el Consola de administración de AWS

- Cree y administre trabajos mediante el AWS CLI

Cree y gestione trabajos mediante el Consola de administración de AWS

En esta sección se describe cómo puede crear y administrar trabajos desde la AWS IoT consola. Después de crear el trabajo, podrá ver información sobre el trabajo en la página de detalles y administrarlo.

> **Note**
>
> Si desea realizar la firma de código para los AWS IoT trabajos, utilice el AWS CLI. Para obtener más información, consulte Crear y administrar trabajos mediante la AWS CLI.

Temas

- Cree y administre trabajos mediante el Consola de administración de AWS
- Vea y gestione los trabajos mediante la Consola de administración de AWS

Cree y administre trabajos mediante el Consola de administración de AWS

Para crear un trabajo, inicie sesión en la AWS IoT consola y vaya al centro de trabajos de la sección Acciones remotas. A continuación, siga estos pasos.

1. En la página Trabajos del cuadro de diálogo Trabajos, elija Crear trabajo.
2. Según el dispositivo que utilice, puede crear un trabajo personalizado, un trabajo de actualización OTA de FreeRTOS o un AWS IoT Greengrass trabajo. En este ejemplo, seleccione Crear un trabajo personalizado. Elija Siguiente.
3. En la página Propiedades de trabajo personalizadas, en el cuadro de diálogo Propiedades del trabajo, introduzca la información de los siguientes campos:
   - Nombre: introduzca un nombre de trabajo alfanumérico único.
   - Descripción (opcional): introduzca una descripción opcional del trabajo.
   - Etiquetas: (opcional):

> **Note**
>
> Le recomendamos que no utilice información de identificación personal en su puesto IDs y en su descripción.

Elija Siguiente.

4. En la página Configuración de archivos del cuadro de diálogo Destinos del trabajo, seleccione las opciones Objetos o Grupos de objetos en los que desee ejecutar este trabajo.

   En el cuadro de diálogo Documento de trabajo, seleccione una de las acciones siguientes:

   - Desde archivo: un archivo de trabajo JSON que cargó anteriormente en un bucket de Amazon S3
     - Firma de código

       En el documento de trabajo ubicado en su URL de Amazon S3, `${aws:iot:code-sign-signature:s3://region.bucket/code-file@code-file-version-id}` es obligatorio como marcador de posición hasta que se sustituya por la ruta del archivo de código firmada mediante su Perfil de firma de código. El nuevo archivo de código firmado aparecerá inicialmente en una carpeta `SignedImages` del bucket de origen de Amazon S3. Se creará un nuevo documento de trabajo con un prefijo `Codesigned_` en el que la ruta del archivo con firma de código sustituirá al marcador de posición del firma de código y se colocará en la URL de Amazon S3 para crear un nuevo trabajo.
     - Recurso previo a la firma URLs

       En el menú desplegable Función de prefirma, elige la función de IAM que creaste en Presigned. URLs La `${aws:iot:s3-presigned-url:` mejor práctica de seguridad URLs para los dispositivos que descargan objetos de Amazon S3 consiste en prefirmar objetos de Amazon S3.

       Si quiere usar prefirmado como marcador de posición URLs de firma de código, utilice la siguiente plantilla de ejemplo:

       ```
       ${aws:iot:s3-presigned-url:${aws:iot:code-sign-signature:<S3 URL>}
       ```

- Desde plantilla: una plantilla de trabajo que contiene un documento de trabajo y las configuraciones del trabajo. La plantilla de trabajo puede ser una plantilla de trabajo personalizada que haya creado o una plantilla AWS gestionada.

  Si va a crear un trabajo para realizar acciones remotas de uso frecuente, como reiniciar el dispositivo, puede utilizar una plantilla AWS gestionada. Estas plantillas ya están preconfiguradas para su uso. Para obtener más información, consulte Creación de una plantilla de trabajo personalizada y Creación de plantillas de trabajo personalizadas a partir de plantillas administradas.

5. En la página Configuración del trabajo del cuadro de diálogo Configuración del trabajo, seleccione uno de los siguientes tipos de trabajo:

   - Trabajo de instantánea: un trabajo de instantánea está completo cuando termina de ejecutarse en los dispositivos y grupos de destino.
   - Trabajo continuo: un trabajo continuo se aplica a grupos de objetos y se ejecuta en cualquier dispositivo que luego se añada a un grupo de destino específico.

6. En el cuadro de diálogo Configuraciones adicionales (opcional), revise las siguientes configuraciones de trabajo opcionales y seleccione las opciones correspondientes:

   - Configuración de despliegue
   - Configuración de programación
   - Configuración del tiempo de espera de ejecuciones de trabajo
   - Configuración de reintentos de ejecuciones de trabajos (nuevo)
   - Configuración de anulación

   Consulte las siguientes secciones para obtener información adicional sobre las configuraciones de los trabajos:

   - Configuraciones de despliegue, programación y cancelación de trabajos
   - Configuraciones de tiempo de espera y reintento de ejecución de trabajos

   Revise todas las selecciones de trabajo y, a continuación, pulse Enviar para crear el trabajo.

Vea y gestione los trabajos mediante la Consola de administración de AWS

Después de crear el trabajo, la consola genera una firma JSON y la coloca en su documento del trabajo. Puede usar la consola de AWS IoT para ver el estado de un trabajo, cancelarlo o eliminarlo.

Si elige el trabajo que ha creado, verá:

- Detalles generales del trabajo, como el nombre del trabajo, la descripción, el tipo, la hora en que se creó, la última actualización y la hora estimada de inicio.
- Cualquier configuración de trabajo que haya especificado y su estado.
- El documento de trabajo.
- Las ejecuciones de los trabajos y cualquier etiqueta opcional que haya especificado.

Para administrar los trabajos, vaya al centro de trabajos de la consola y elija si desea editar, eliminar o cancelar el trabajo.

Cree y administre trabajos mediante el AWS CLI

En esta sección se describe cómo crear y administrar trabajos.

Creación de trabajos

Para crear un AWS IoT trabajo, utilice el CreateJob comando. El trabajo se pone en cola para la ejecución en los destinos (objetos o grupos de objetos) que especifique. Para crear un AWS IoT trabajo, necesita un documento de trabajo que pueda incluirse en el cuerpo de la solicitud o como un enlace a un documento de Amazon S3. Si el trabajo incluye la descarga de archivos mediante Amazon S3 prefirmado URLs, necesitará un rol de IAM: Amazon Resource Name (ARN) que tenga permiso para descargar el archivo y que conceda permiso al servicio AWS IoT Jobs para que asuma el rol.

Para obtener más información sobre la sintaxis al introducir la fecha y la hora mediante un comando de la API o el AWS CLI, consulte Timestamp.

Firma de código con trabajos

Si utiliza la firma de código para AWS IoT, debe iniciar un trabajo de firma de código e incluir el resultado en el documento de trabajo. Esto sustituirá al marcador de posición de firma de código del documento de trabajo, que es obligatorio como marcador de posición hasta que se sustituya por la

ruta del archivo de código firmada mediante su Perfil de firma de código. El marcador de la firma de código tendrá el siguiente aspecto:

```
    ${aws:iot:code-sign-signature:s3://region.bucket/code-file@code-file-version-id}
```

Utilice el start-signing-jobcomando para crear un trabajo de firma de código. `start-signing-job`devuelve un identificador de trabajo. Utilice el comando describe-signing-job para obtener la ubicación de Amazon S3 donde se almacena la firma. Después podrá descargar la firma desde Amazon S3. Para obtener más información sobre trabajos de firma de código, consulte la sección sobre Firma de código para AWS IoT.

El documento de trabajo debe contener un marcador de posición de URL prefirmada para su archivo de código y el resultado de la firma JSON en un bucket de Amazon S3 con el comando start-signing-job:

```
{
    "presign": "${aws:iot:s3-presigned-url:https://s3.region.amazonaws.com/bucket/
image}",
}
```

Creación de un trabajo con un documento de trabajo

El siguiente comando muestra cómo crear un trabajo mediante un documento de trabajo (*job-document.json*) almacenado en un bucket de Amazon S3 (*jobBucket*) y un rol con permiso para descargar archivos de Amazon S3 (*S3DownloadRole*).

```
aws iot create-job  \
      --job-id 010  \
      --targets arn:aws:iot:us-east-1:123456789012:thing/thingOne  \
      --document-source https://s3.amazonaws.com/amzn-s3-demo-bucket/job-document.json
 \
      --timeout-config inProgressTimeoutInMinutes=100 \
      --job-executions-rollout-config "{ \"exponentialRate\": { \"baseRatePerMinute
\": 50, \"incrementFactor\": 2, \"rateIncreaseCriteria\": { \"numberOfNotifiedThings
\": 1000, \"numberOfSucceededThings\": 1000}}, \"maximumPerMinute\": 1000}" \
      --abort-config "{ \"criteriaList\": [ { \"action\": \"CANCEL\", \"failureType
\": \"FAILED\", \"minNumberOfExecutedThings\": 100, \"thresholdPercentage\": 20},
 { \"action\": \"CANCEL\", \"failureType\": \"TIMED_OUT\", \"minNumberOfExecutedThings
\": 200, \"thresholdPercentage\": 50}]}" \
      --presigned-url-config "{\"roleArn\":\"arn:aws:iam::123456789012:role/
S3DownloadRole\", \"expiresInSec\":3600}"
```

El trabajo se ejecuta en *thingOne*.

El parámetro `timeout-config` opcional especifica la cantidad de tiempo que cada dispositivo tiene para finalizar su ejecución del trabajo. El temporizador comienza cuando el estado de ejecución del trabajo se establece en `IN_PROGRESS`. Si el estado de ejecución del trabajo no se establece en otro estado terminal antes de que se cumpla el plazo, se establecerá en `TIMED_OUT`.

El temporizador en curso no se puede actualizar y se aplica a todas las ejecuciones de trabajos para el trabajo. Cuando la ejecución de una tarea permanece en ese `IN_PROGRESS` estado durante más tiempo que este intervalo, se produce un error y pasa al `TIMED_OUT` estado terminal. AWS IoT también publica una notificación MQTT.

Para obtener más información acerca de cómo crear configuraciones sobre despliegues de trabajos y anulaciones, consulte Despliegue de trabajos y configuración de anulaciones.

> **Note**
>
> Los documentos de trabajo que está especificados como archivos de Amazon S3 se recuperan en el momento en el que crea el trabajo. Si cambia el contenido del archivo de Amazon S3 que usó como origen de su documento de trabajo después de haberlo creado, no cambia lo que se envía a los destinos del trabajo.

Actualización de un trabajo

Para actualizar un trabajo se utiliza el comando UpdateJob. Puede actualizar los campos `description`, `presignedUrlConfig`, `jobExecutionsRolloutConfig`, `abortConfig` y `timeoutConfig` para un trabajo.

```
aws iot update-job  \
  --job-id 010  \
  --description "updated description" \
  --timeout-config inProgressTimeoutInMinutes=100 \
  --job-executions-rollout-config "{ \"exponentialRate\": { \"baseRatePerMinute\": 50,
 \"incrementFactor\": 2, \"rateIncreaseCriteria\": { \"numberOfNotifiedThings\": 1000,
 \"numberOfSucceededThings\": 1000}, \"maximumPerMinute\": 1000}}" \
  --abort-config "{ \"criteriaList\": [ { \"action\": \"CANCEL\", \"failureType
\": \"FAILED\", \"minNumberOfExecutedThings\": 100, \"thresholdPercentage\": 20},
 { \"action\": \"CANCEL\", \"failureType\": \"TIMED_OUT\", \"minNumberOfExecutedThings
\": 200, \"thresholdPercentage\": 50}]}" \
```

```
  --presigned-url-config "{\"roleArn\":\"arn:aws:iam::123456789012:role/
S3DownloadRole\", \"expiresInSec\":3600}"
```

Para obtener más información, consulte Despliegue de trabajos y configuración de anulaciones.

Cancelación de un trabajo

Para cancelar un trabajo se utiliza el comando CancelJob. La cancelación de un trabajo AWS IoT impide que se ejecuten nuevos trabajos para el trabajo. También cancela cualquier ejecución de un trabajo que se produzca en un `QUEUED` estado determinado. AWS IoT mantiene intactas las ejecuciones de tareas en estado terminal porque el dispositivo ya ha completado la tarea. Si el estado de la ejecución de un trabajo es `IN_PROGRESS`, también se mantendrá intacta a menos que se use el parámetro opcional `--force`.

El siguiente comando muestra cómo cancelar un trabajo con ID 010.

```
aws iot cancel-job --job-id 010
```

El comando muestra el resultado siguiente:

```
{
    "jobArn": "string",
    "jobId": "string",
    "description": "string"
}
```

Cuando se cancela un trabajo, se cancelan las ejecuciones de trabajos con estado `QUEUED`. Las ejecuciones de trabajos en estado `IN_PROGRESS` se cancelarán, pero solo si especifica el parámetro opcional `--force`. Las ejecuciones de trabajo con un estado terminal no se cancelarán.

> **Warning**
>
> Si se cancela un trabajo en estado `IN_PROGRESS` (al establecer el parámetro `--force`), se cancelarán las ejecuciones de trabajos en curso, y se hará que el dispositivo que está ejecutando el trabajo no pueda actualizar el estado de ejecución del trabajo. Actúe con precaución y asegúrese de que cada dispositivo que esté ejecutando un trabajo cancelado pueda recuperarse a un estado válido.

El estado de un trabajo cancelado o de la ejecución de uno de sus trabajos es, en última instancia, uniforme. AWS IoT deja de programar las ejecuciones de nuevos `QUEUED` trabajos y las ejecuciones de trabajos para ese trabajo en los dispositivos lo antes posible. Cambiar el estado de la ejecución de un trabajo a `CANCELED` puede llevar algo de tiempo, según el número de dispositivos y otros factores.

Si un trabajo se cancela porque cumple los criterios definidos por un objeto `AbortConfig`, el servicio añade valores rellenados automáticamente para los campos `comment` y `reasonCode`. Puede crear sus propios valores `reasonCode` cuando el trabajo se cancela por iniciativa del usuario.

Cancelación de una ejecución de trabajo

Para cancelar la ejecución de un trabajo en un dispositivo, utilice el comando CancelJobExecution. Este cancela la ejecución de un trabajo que se encuentra en estado `QUEUED`. Si desea cancelar la ejecución de un trabajo en curso, debe utilizar el parámetro `--force`.

El siguiente comando muestra cómo cancelar la ejecución de un trabajo del trabajo 010 que se ejecuta en `myThing`.

```
aws iot cancel-job-execution --job-id 010 --thing-name myThing
```

El comando no muestra ninguna salida.

Se cancela la ejecución de un trabajo que se encuentra en estado `QUEUED`. La ejecución de un trabajo en estado `IN_PROGRESS` se cancela, pero solo si especifica el parámetro opcional `--force`. Las ejecuciones de trabajo con un estado terminal no se pueden cancelar.

> **Warning**
>
> Cuando se cancela la ejecución de un trabajo con estado `IN_PROGRESS`, el dispositivo no puede actualizar el estado de ejecución del trabajo. Actúe con precaución y asegúrese de que el dispositivo pueda recuperarse a un estado válido.

Si la ejecución del trabajo se encuentra en estado final, o si la ejecución del trabajo está en estado `IN_PROGRESS` y el parámetro `--force` no está definido en `true`, el comando genera `InvalidStateTransitionException`.

El estado de la ejecución de un trabajo cancelada es a la larga coherente. Cambiar el estado de la ejecución de un trabajo a `CANCELED` puede llevar algo de tiempo, dependiendo de varios factores.

Eliminación de un trabajo

Para eliminar un trabajo y sus ejecuciones, utilice el comando DeleteJob. De forma predeterminada, solo puede eliminar un trabajo en estado final (`SUCCEEDED` o `CANCELED`). En caso contrario, se produce una excepción. Podrá eliminar un trabajo con estado `IN_PROGRESS`, aunque solo si el parámetro `force` está definido en `true`.

Para eliminar un trabajo, ejecute el siguiente comando:

```
aws iot delete-job --job-id 010 --force|--no-force
```

El comando no muestra ninguna salida.

> **Warning**
>
> Cuando se elimina un trabajo que se encuentra en estado `IN_PROGRESS`, el dispositivo que está implementando el trabajo no puede obtener acceso a la información del trabajo ni actualizar el estado de su ejecución. Actúe con precaución y asegúrese de que cada dispositivo que implemente un trabajo eliminado pueda recuperarse a un estado válido.

Eliminar un trabajo podría llevar algún tiempo, en función del número de ejecuciones de trabajos creadas para el trabajo y otros factores. Aunque el trabajo se esté eliminando, como estado del trabajo se muestra `DELETION_IN_PROGRESS`. Si se intenta eliminar o cancelar un trabajo cuyo estado ya es `DELETION_IN_PROGRESS`, se producirá un error.

Solo puede haber 10 trabajos con estado `DELETION_IN_PROGRESS` al mismo tiempo. De lo contrario, se genera `LimitExceededException`.

Obtención de un documento de trabajo

Para recuperar un documento de trabajo para un trabajo, utilice el comando GetJobDocument. Un documento de trabajo es una descripción de las operaciones remotas que deben ejecutar los dispositivos.

Para obtener un documento de trabajo, ejecute el siguiente comando:

```
aws iot get-job-document --job-id 010
```

El comando devuelve el documento de trabajo para el trabajo especificado:

```
{
    "document": "{\n\t\"operation\":\"install\",\n\t\"url\":\"http://amazon.com/
firmWareUpate-01\",\n\t\"data\":\"${aws:iot:s3-presigned-url:https://s3.amazonaws.com/
amzn-s3-demo-bucket/datafile}\"\n}"
}
```

> **Note**
>
> Cuando utilizas este comando para recuperar un documento de trabajo, los marcadores de posición URLs no se sustituyen por Amazon S3 prefirmado. URLs Cuando un dispositivo llama a la operación de la GetPendingJobExecutionsAPI, el marcador de posición URLs se sustituye por Amazon S3 prefirmado URLs en el documento de trabajo.

Enumeración de trabajos

Para obtener una lista de todos los trabajos que tiene Cuenta de AWS, utilice el ListJobs comando. Los datos del trabajo y los datos de ejecución del trabajo se conservan durante un tiempo limitado. Ejecute el siguiente comando para obtener una lista de todos los trabajos de su Cuenta de AWS:

```
aws iot list-jobs
```

El comando devuelve todos los trabajos en su cuenta ordenados por estado del trabajo:

```
{
    "jobs": [
        {
            "status": "IN_PROGRESS",
            "lastUpdatedAt": 1486687079.743,
            "jobArn": "arn:aws:iot:us-east-1:123456789012:job/013",
            "createdAt": 1486687079.743,
            "targetSelection": "SNAPSHOT",
            "jobId": "013"
        },
        {
            "status": "SUCCEEDED",
            "lastUpdatedAt": 1486685868.444,
            "jobArn": "arn:aws:iot:us-east-1:123456789012:job/012",
            "createdAt": 1486685868.444,
            "completedAt": 148668789.690,
```

```
            "targetSelection": "SNAPSHOT",
            "jobId": "012"
        },
        {
            "status": "CANCELED",
            "lastUpdatedAt": 1486678850.575,
            "jobArn": "arn:aws:iot:us-east-1:123456789012:job/011",
            "createdAt": 1486678850.575,
            "targetSelection": "SNAPSHOT",
            "jobId": "011"
        }
    ]
}
```

Descripción de un trabajo

Para obtener el estado de un trabajo, ejecute el comando DescribeJob. El siguiente comando muestra cómo describir un trabajo:

```
$ aws iot describe-job --job-id 010
```

El comando devuelve el estado de un trabajo especificado. Por ejemplo:

```
{
    "documentSource": "https://s3.amazonaws.com/amzn-s3-demo-bucket/job-
document.json",
    "job": {
        "status": "IN_PROGRESS",
        "jobArn": "arn:aws:iot:us-east-1:123456789012:job/010",
        "targets": [
            "arn:aws:iot:us-east-1:123456789012:thing/myThing"
        ],
        "jobProcessDetails": {
            "numberOfCanceledThings": 0,
            "numberOfFailedThings": 0,
            "numberOfInProgressThings": 0,
            "numberOfQueuedThings": 0,
            "numberOfRejectedThings": 0,
            "numberOfRemovedThings": 0,
            "numberOfSucceededThings": 0,
            "numberOfTimedOutThings": 0,
            "processingTargets": [
                arn:aws:iot:us-east-1:123456789012:thing/thingOne,
```

```
                arn:aws:iot:us-east-1:123456789012:thinggroup/thinggroupOne,
                arn:aws:iot:us-east-1:123456789012:thing/thingTwo,
                arn:aws:iot:us-east-1:123456789012:thinggroup/thinggroupTwo
            ]
        },
        "presignedUrlConfig": {
            "expiresInSec": 60,
            "roleArn": "arn:aws:iam::123456789012:role/S3DownloadRole"
        },
        "jobId": "010",
        "lastUpdatedAt": 1486593195.006,
        "createdAt": 1486593195.006,
        "targetSelection": "SNAPSHOT",
        "jobExecutionsRolloutConfig": {
            "exponentialRate": {
                "baseRatePerMinute": integer,
                "incrementFactor": integer,
                "rateIncreaseCriteria": {
                    "numberOfNotifiedThings": integer, // Set one or the other
                    "numberOfSucceededThings": integer // of these two values.
                },
            "maximumPerMinute": integer
         }
        },
        "abortConfig": {
            "criteriaList": [
                {
                    "action": "string",
                    "failureType": "string",
                    "minNumberOfExecutedThings": integer,
                    "thresholdPercentage": integer
                }
            ]
        },
        "timeoutConfig": {
           "inProgressTimeoutInMinutes": number
          }
    }
}
```

Enumeración de ejecuciones para un trabajo

Un trabajo que se ejecuta en un dispositivo específico se representa mediante un objeto de ejecución de trabajo. Ejecute el comando ListJobExecutionsForJob para enumerar todas las ejecuciones de trabajo para un trabajo. A continuación se muestra cómo enumerar las ejecuciones para un trabajo:

```
aws iot list-job-executions-for-job --job-id 010
```

El comando devuelve una lista de ejecuciones de trabajo:

```
{
    "executionSummaries": [
    {
        "thingArn": "arn:aws:iot:us-east-1:123456789012:thing/thingOne",
        "jobExecutionSummary": {
            "status": "QUEUED",
            "lastUpdatedAt": 1486593196.378,
            "queuedAt": 1486593196.378,
            "executionNumber": 1234567890
        }
    },
    {
        "thingArn": "arn:aws:iot:us-east-1:123456789012:thing/thingTwo",
        "jobExecutionSummary": {
            "status": "IN_PROGRESS",
            "lastUpdatedAt": 1486593345.659,
            "queuedAt": 1486593196.378,
            "startedAt": 1486593345.659,
            "executionNumber": 4567890123
        }
    }
    ]
}
```

Enumeración de ejecuciones de trabajo para un objeto

Ejecute el comando ListJobExecutionsForThing para enumerar todas las ejecuciones de trabajo que se ejecutan en un objeto. A continuación se muestra cómo listar las ejecuciones de trabajos para un objeto:

```
aws iot list-job-executions-for-thing --thing-name thingOne
```

El comando devuelve una lista de ejecuciones de trabajo que se ejecutan o se han ejecutado en el objeto especificado:


```
{
    "executionSummaries": [
    {
        "jobExecutionSummary": {
            "status": "QUEUED",
            "lastUpdatedAt": 1486687082.071,
            "queuedAt": 1486687082.071,
            "executionNumber": 9876543210
        },
        "jobId": "013"
    },
    {
        "jobExecutionSummary": {
            "status": "IN_PROGRESS",
            "startAt": 1486685870.729,
            "lastUpdatedAt": 1486685870.729,
            "queuedAt": 1486685870.729,
            "executionNumber": 1357924680
        },
        "jobId": "012"
    },
    {
        "jobExecutionSummary": {
            "status": "SUCCEEDED",
            "startAt": 1486678853.415,
            "lastUpdatedAt": 1486678853.415,
            "queuedAt": 1486678853.415,
            "executionNumber": 4357680912
        },
        "jobId": "011"
    },
    {
        "jobExecutionSummary": {
            "status": "CANCELED",
            "startAt": 1486593196.378,
            "lastUpdatedAt": 1486593196.378,
            "queuedAt": 1486593196.378,
            "executionNumber": 2143174250
        },
        "jobId": "010"
```

```
    }
    ]
}
```

Descripción de una ejecución de trabajo

Ejecute el comando DescribeJobExecution para obtener el estado de la ejecución de un trabajo. Debe especificar un ID de trabajo y el nombre del objeto y, opcionalmente, un número de ejecución para identificar la ejecución del trabajo. A continuación se muestra cómo describir la ejecución de un trabajo:

```
aws iot describe-job-execution --job-id 017 --thing-name thingOne
```

El comando devuelve `JobExecution`. Por ejemplo:

```
{
    "execution": {
        "jobId": "017",
        "executionNumber": 4516820379,
        "thingArn": "arn:aws:iot:us-east-1:123456789012:thing/thingOne",
        "versionNumber": 123,
        "createdAt": 1489084805.285,
        "lastUpdatedAt": 1489086279.937,
        "startedAt": 1489086279.937,
        "status": "IN_PROGRESS",
        "approximateSecondsBeforeTimedOut": 100,
        "statusDetails": {
            "status": "IN_PROGRESS",
            "detailsMap": {
                "percentComplete": "10"
            }
        }
    }
}
```

Eliminación de una ejecución de trabajo

Ejecute el comando DeleteJobExecution para eliminar la ejecución de un trabajo. Debe especificar un ID de trabajo, un nombre de objeto y un número de ejecución para identificar la ejecución del trabajo. A continuación se muestra cómo eliminar la ejecución de un trabajo:

```
aws iot delete-job-execution --job-id 017 --thing-name thingOne --execution-number
 1234567890 --force|--no-force
```

El comando no muestra ninguna salida.

De forma predeterminada, el estado de ejecución de los trabajos debe ser `QUEUED` o un estado final (`SUCCEEDED`, `FAILED`, `REJECTED`, `TIMED_OUT`, `REMOVED` o `CANCELED`). En caso contrario, se produce un error. Para eliminar la ejecución de un trabajo con estado `IN_PROGRESS`, puede establecer el parámetro `force` en `true`.

> **Warning**
>
> Cuando se elimina la ejecución de un trabajo con estado `IN_PROGRESS`, el dispositivo que está ejecutando el trabajo no puede obtener acceso a la información del trabajo ni actualizar el estado de su ejecución. Actúe con precaución y asegúrese de que el dispositivo pueda recuperarse a un estado válido.

Plantillas de tarea

Utilice plantillas de trabajo para preconfigurar los trabajos que puede implementar en varios conjuntos de dispositivos de destino. Para implementar acciones remotas que se realizan con frecuencia en sus dispositivos, como reiniciar o instalar una aplicación, puede usar plantillas para definir configuraciones estándar. Para realizar operaciones como la implementación de parches de seguridad y correcciones de errores, puede crear plantillas a partir de los trabajos existentes.

Al crear una plantilla de trabajo, especifique las siguientes configuraciones y recursos adicionales.

- Propiedades del trabajo
- Documentos y destinos del trabajo
- Criterios de despliegue, programación y cancelación
- Criterios de tiempo de espera y reintento

Plantillas personalizadas y AWS gestionadas

En función de la acción remota que desee realizar, puede crear una plantilla de trabajo personalizada o utilizar una plantilla AWS gestionada. Utilice plantillas de trabajo personalizadas para proporcionar

su propio documento de trabajo personalizado y cree trabajos reutilizables para implementarlos en sus dispositivos. AWS Las plantillas gestionadas son plantillas de trabajo que proporciona AWS IoT Jobs para las acciones que se realizan habitualmente. Estas plantillas tienen un documento de trabajo predefinido para algunas acciones remotas, por lo que no tiene que crear su propio documento de trabajo. Las plantillas administradas le ayudan a crear trabajos reutilizables para lanzarlos más rápido a los dispositivos.

Temas

- Use plantillas AWS administradas para implementar operaciones remotas comunes
- Creación de plantillas de trabajo personalizadas

Use plantillas AWS administradas para implementar operaciones remotas comunes

AWS Las plantillas gestionadas son plantillas de trabajo proporcionadas por AWS. Se utilizan para realizar acciones remotas frecuentes, como reiniciar, descargar un archivo o instalar una aplicación en los dispositivos. Estas plantillas tienen un documento de trabajo predefinido para cada acción remota, por lo que no tiene que crear su propio documento de trabajo.

Puede elegir entre un conjunto de configuraciones predefinidas y crear trabajos con estas plantillas sin necesidad de escribir ningún código adicional. Con las plantillas administradas, puede ver el documento de trabajo implementado en sus flotas. Puede crear un trabajo con estas plantillas y crear una plantilla de trabajo personalizada, que puede reutilizar para sus operaciones remotas.

¿Qué contienen las plantillas administradas?

Cada plantilla AWS gestionada contiene:

- El entorno para ejecutar los comandos del documento de trabajo.
- Un documento de trabajo que especifica el nombre de la operación y sus parámetros. Por ejemplo, si utiliza una plantilla Descargar archivo, el nombre de la operación es Descargar archivo y los parámetros pueden ser:
  - La URL del archivo que desea descargar a su dispositivo. Puede ser un recurso de Internet o una URL pública o prefirmada de Amazon Simple Storage Service (Amazon S3).
  - Una ruta de archivo local en el dispositivo para almacenar el archivo descargado.

Para obtener más información sobre los documentos de trabajo y sus parámetros, consulte Acciones remotas y documentos de trabajo de plantillas administradas.

Requisitos previos

Para que los dispositivos ejecuten las acciones remotas especificadas en el documento de trabajo de la plantilla administrada, debe:

- Instalar el software específico en el dispositivo

  Utilice el software y los gestores de tareas de su propio dispositivo, o bien utilice el cliente del AWS IoT dispositivo. Según el caso empresarial, también puede ejecutar ambos para que desempeñen funciones diferentes.

  - Uso del software y los controladores de trabajos de su propio dispositivo

    Puede escribir su propio código para los dispositivos utilizando el SDK para dispositivos con AWS IoT y su biblioteca de controladores que admiten las operaciones remotas. Para implementar y ejecutar trabajos, compruebe que las bibliotecas de agentes de dispositivo se hayan instalado correctamente y se estén ejecutando en los dispositivos.

    También puede optar por utilizar sus propios controladores que admitan las operaciones remotas. Para obtener más información, consulte Ejemplos de gestores de tareas en el repositorio de AWS IoT Device Client GitHub.
  - Utilice el cliente de AWS IoT dispositivos

    O bien, puede instalar y ejecutar el AWS IoT Device Client en sus dispositivos, ya que, de forma predeterminada, admite el uso de todas las plantillas administradas directamente desde la consola.

    El cliente de dispositivo es un software de código abierto escrito en C++ que puede compilar e instalar en los dispositivos de IoT integrados basados en Linux. El cliente de dispositivo tiene un cliente base y características discretas en el lado del cliente. El cliente base establece la conectividad a AWS IoT través del protocolo MQTT y puede conectarse con las diferentes funciones del lado del cliente.

    Para realizar operaciones remotas en los dispositivos, utilice la característica de Jobs del lado del cliente del cliente de dispositivo. Esta característica contiene un analizador para recibir el documento de trabajo y controladores de trabajos que implementan las acciones remotas

especificadas en el documento de trabajo. Para obtener más información sobre el cliente de dispositivo y sus características, consulte AWS IoT Cliente de dispositivo de .

Cuando se ejecuta en dispositivos, el cliente de dispositivo recibe el documento de trabajo y tiene una implementación específica de la plataforma que utiliza para ejecutar los comandos del documento. Para obtener más información acerca de la configuración del dispositivo de cliente y el uso de la característica Jobs, consulte los Tutoriales de AWS IoT.

- Uso de un entorno compatible

Para cada plantilla administrada, encontrará información sobre el entorno que puede utilizar para ejecutar las acciones remotas. Le recomendamos que use la plantilla con un entorno Linux compatible, tal y como se especifica en la plantilla. Utilice el AWS IoT Device Client para ejecutar las acciones remotas de la plantilla gestionada, ya que es compatible con los microprocesadores y entornos Linux más comunes, como Debian y Ubuntu.

Acciones remotas y documentos de trabajo de plantillas administradas

En la siguiente sección se enumeran las diferentes plantillas AWS gestionadas para AWS IoT Jobs y se describen las acciones remotas que se pueden realizar en los dispositivos. La siguiente sección contiene información sobre el documento de trabajo y una descripción de los parámetros del documento de trabajo para cada acción remota. El software del dispositivo utiliza el nombre de la plantilla y sus parámetros para realizar la acción remota.

AWS Las plantillas gestionadas aceptan parámetros de entrada para los que se especifica un valor al crear un trabajo con la plantilla. Todas las plantillas administradas tienen dos parámetros de entrada opcionales en común: `runAsUser` y `pathToHandler`. A excepción de la plantilla `AWS-Reboot`, las plantillas requieren parámetros de entrada adicionales para los que debe especificar un valor al crear un trabajo con la plantilla. Estos parámetros de entrada obligatorios varían en función de la plantilla que elija. Por ejemplo, si elige la plantilla `AWS-Download-File`, debe especificar una lista de paquetes para instalar y una URL de la que descargar los archivos.

Especifique un valor para los parámetros de entrada cuando utilice la AWS IoT consola o el AWS Command Line Interface (AWS CLI) para crear un trabajo que utilice una plantilla gestionada. Cuando utilice la CLI, proporcione estos valores mediante el objeto `document-parameters`. Para obtener más información, consulte documentParameters.

> **Note**
>
> Use `document-parameters` solo al crear trabajos a partir de plantillas administradas por AWS . Este parámetro no se puede usar con plantillas de trabajos personalizadas ni para crear trabajos a partir de ellas.

A continuación se muestra una descripción de los parámetros de entrada opcionales más comunes. Verá una descripción de los demás parámetros de entrada que requiere cada plantilla administrada en la siguiente sección.

`runAsUser`

Este parámetro especifica si se debe ejecutar el controlador de trabajos como otro usuario. Si no se especifica durante la creación del trabajo, el controlador de trabajos se ejecuta con el mismo usuario que el cliente de dispositivo. Cuando ejecute el controlador de trabajos como otro usuario, especifique un valor de cadena que no supere los 256 caracteres.

`pathToHandler`

La ruta al controlador de trabajos que se ejecuta en el dispositivo. Si no se especifica durante la creación del trabajo, el cliente de dispositivo utiliza el directorio de trabajo actual.

A continuación se muestran las distintas acciones remotas, sus documentos de trabajo y los parámetros que aceptan. Todas estas plantillas son compatibles con el entorno Linux para ejecutar la operación remota en el dispositivo.

AWS–Download–File

Nombre de la plantilla

`AWS-Download-File`

Descripción de la plantilla

Una plantilla gestionada proporcionada por AWS para descargar un archivo.

Parámetros de entrada

Esta plantilla tiene los siguientes parámetros obligatorios. También puede especificar los parámetros opcionales `runAsUser` y `pathToHandler`.

`downloadUrl`

La URL de la que descargar el archivo. Puede ser un recurso de Internet, un objeto de Amazon S3 al que se pueda acceder públicamente o un objeto de Amazon S3 al que solo pueda acceder su dispositivo mediante una URL prefirmada. Para obtener más información sobre el uso URLs y la concesión de permisos prefirmados, consulte[Prefirmado URLs](#).

`filePath`

Una ruta de archivo local que muestra la ubicación en el dispositivo para almacenar el archivo descargado.

Comportamiento del dispositivo

El dispositivo descarga el archivo desde la ubicación especificada, comprueba que la descarga se ha completado y lo almacena localmente.

Documento de trabajo

A continuación se muestra el documento de trabajo y su versión más reciente. La plantilla muestra la ruta al controlador de tareas y al script de shell, `download-file.sh`, que el controlador de trabajos debe ejecutar para descargar el archivo. También muestra los parámetros obligatorios `downloadUrl` y `filePath`.

```
{
  "version": "1.0",
  "steps": [
    {
      "action": {
        "name": "Download-File",
        "type": "runHandler",
        "input": {
          "handler": "download-file.sh",
          "args": [
            "${aws:iot:parameter:downloadUrl}",
            "${aws:iot:parameter:filePath}"
          ],
          "path": "${aws:iot:parameter:pathToHandler}"
        },
```

```
        "runAsUser": "${aws:iot:parameter:runAsUser}"
      }
    }
  ]
}
```

AWS–Install–Application

Nombre de la plantilla

`AWS-Install-Application`

Descripción de la plantilla

Plantilla gestionada proporcionada por AWS para instalar una o más aplicaciones.

Parámetros de entrada

Esta plantilla tiene el siguiente parámetro obligatorio, `packages`. También puede especificar los parámetros opcionales `runAsUser` y `pathToHandler`.

`packages`

Una lista separada por espacios de una o varias aplicaciones que se van a instalar.

Comportamiento del dispositivo

El dispositivo instala las aplicaciones tal y como se especifica en el documento de trabajo.

Documento de trabajo

A continuación se muestra el documento de trabajo y su versión más reciente. La plantilla muestra la ruta al controlador de tareas y al script de shell, `install-packages.sh`, que el controlador de trabajos debe ejecutar para descargar el archivo. También muestra el parámetro obligatorio `packages`.

```
{
  "version": "1.0",
  "steps": [
    {
      "action": {
        "name": "Install-Application",
        "type": "runHandler",
```

```
        "input": {
          "handler": "install-packages.sh",
          "args": [
            "${aws:iot:parameter:packages}"
          ],
          "path": "${aws:iot:parameter:pathToHandler}"
        },
        "runAsUser": "${aws:iot:parameter:runAsUser}"
      }
    }
  ]
}
```

AWS–Reboot

Nombre de la plantilla

`AWS-Reboot`

Descripción de la plantilla

Una plantilla gestionada proporcionada por AWS para reiniciar el dispositivo.

Parámetros de entrada

Esta plantilla no tiene parámetros obligatorios. Puede especificar los parámetros opcionales `runAsUser` y `pathToHandler`.

Comportamiento del dispositivo

El dispositivo se reinicia correctamente.

Documento de trabajo

A continuación se muestra el documento de trabajo y su versión más reciente. La plantilla muestra la ruta al controlador de tareas y al script de shell, `reboot.sh`, que el controlador de trabajos debe ejecutar para reiniciar el dispositivo.

```
{
  "version": "1.0",
  "steps": [
    {
      "action": {
        "name": "Reboot",
```

```
        "type": "runHandler",
        "input": {
          "handler": "reboot.sh",
          "path": "${aws:iot:parameter:pathToHandler}"
        },
        "runAsUser": "${aws:iot:parameter:runAsUser}"
      }
    }
  ]
}
```

AWS–Remove–Application

Nombre de la plantilla

`AWS-Remove-Application`

Descripción de la plantilla

Una plantilla gestionada proporcionada por AWS para desinstalar una o más aplicaciones.

Parámetros de entrada

Esta plantilla tiene el siguiente parámetro obligatorio, `packages`. También puede especificar los parámetros opcionales `runAsUser` y `pathToHandler`.

`packages`

Una lista separada por espacios de una o varias aplicaciones que se van a desinstalar.

Comportamiento del dispositivo

El dispositivo desinstala las aplicaciones tal y como se especifica en el documento de trabajo.

Documento de trabajo

A continuación se muestra el documento de trabajo y su versión más reciente. La plantilla muestra la ruta al controlador de tareas y al script de shell, `remove-packages.sh`, que el controlador de trabajos debe ejecutar para descargar el archivo. También muestra el parámetro obligatorio `packages`.

```
{
  "version": "1.0",
  "steps": [
```

```
    {
      "action": {
        "name": "Remove-Application",
        "type": "runHandler",
        "input": {
          "handler": "remove-packages.sh",
          "args": [
            "${aws:iot:parameter:packages}"
          ],
          "path": "${aws:iot:parameter:pathToHandler}"
        },
        "runAsUser": "${aws:iot:parameter:runAsUser}"
      }
    }
  ]
}
```

AWS–Restart–Application

Nombre de la plantilla

`AWS-Restart-Application`

Descripción de la plantilla

Plantilla administrada proporcionada por AWS para detener y reiniciar uno o más servicios.

Parámetros de entrada

Esta plantilla tiene el siguiente parámetro obligatorio, `services`. También puede especificar los parámetros opcionales `runAsUser` y `pathToHandler`.

Services

Una lista separada por espacios de una o varias aplicaciones que se van a reiniciar.

Comportamiento del dispositivo

Las aplicaciones especificadas se detienen y, a continuación, se reinician en el dispositivo.

Documento de trabajo

A continuación se muestra el documento de trabajo y su versión más reciente. La plantilla muestra la ruta al controlador de tareas y al script de shell, `restart-services.sh`, que el controlador

de trabajos debe ejecutar para reiniciar los servicios del sistema. También muestra el parámetro obligatorio `services`.

```
{
  "version": "1.0",
  "steps": [
    {
      "action": {
        "name": "Restart-Application",
        "type": "runHandler",
        "input": {
          "handler": "restart-services.sh",
          "args": [
            "${aws:iot:parameter:services}"
          ],
          "path": "${aws:iot:parameter:pathToHandler}"
        },
        "runAsUser": "${aws:iot:parameter:runAsUser}"
      }
    }
  ]
}
```

AWS–Start–Application

Nombre de la plantilla

`AWS-Start-Application`

Descripción de la plantilla

Plantilla gestionada proporcionada por AWS para iniciar uno o más servicios.

Parámetros de entrada

Esta plantilla tiene el siguiente parámetro obligatorio, `services`. También puede especificar los parámetros opcionales `runAsUser` y `pathToHandler`.

`services`

Una lista separada por espacios de una o varias aplicaciones que se van a iniciar.

Comportamiento del dispositivo

Las aplicaciones especificadas comienzan a ejecutarse en el dispositivo.

Documento de trabajo

A continuación se muestra el documento de trabajo y su versión más reciente. La plantilla muestra la ruta al controlador de tareas y al script de shell, `start-services.sh`, que el controlador de trabajos debe ejecutar para iniciar los servicios del sistema. También muestra el parámetro obligatorio `services`.

```
{
  "version": "1.0",
  "steps": [
    {
      "action": {
        "name": "Start-Application",
        "type": "runHandler",
        "input": {
          "handler": "start-services.sh",
          "args": [
            "${aws:iot:parameter:services}"
          ],
          "path": "${aws:iot:parameter:pathToHandler}"
        },
        "runAsUser": "${aws:iot:parameter:runAsUser}"
      }
    }
  ]
}
```

AWS–Stop–Application

Nombre de la plantilla

`AWS-Stop-Application`

Descripción de la plantilla

Una plantilla gestionada proporcionada por AWS para detener uno o más servicios.

Parámetros de entrada

Esta plantilla tiene el siguiente parámetro obligatorio, `services`. También puede especificar los parámetros opcionales `runAsUser` y `pathToHandler`.

`services`

Una lista separada por espacios de una o varias aplicaciones que se van a detener.

Comportamiento del dispositivo

Las aplicaciones especificadas dejan de ejecutarse en el dispositivo.

Documento de trabajo

A continuación se muestra el documento de trabajo y su versión más reciente. La plantilla muestra la ruta al controlador de tareas y al script de shell, `stop-services.sh`, que el controlador de trabajos debe ejecutar para detener los servicios del sistema. También muestra el parámetro obligatorio `services`.

```
{
  "version": "1.0",
  "steps": [
    {
      "action": {
        "name": "Stop-Application",
        "type": "runHandler",
        "input": {
          "handler": "stop-services.sh",
          "args": [
            "${aws:iot:parameter:services}"
          ],
          "path": "${aws:iot:parameter:pathToHandler}"
        },
        "runAsUser": "${aws:iot:parameter:runAsUser}"
      }
    }
  ]
}
```

AWS–Run–Command

Nombre de la plantilla

`AWS-Run-Command`

Descripción de la plantilla

Una plantilla gestionada proporcionada por AWS para ejecutar un comando de shell.

Parámetros de entrada

Esta plantilla tiene el siguiente parámetro obligatorio, `command`. También puede especificar el parámetro opcional `runAsUser`.

`command`

Una cadena de comandos separados por comas. Todas las comas contenidas en el propio comando deben ir con una secuencia de escape.

Comportamiento del dispositivo

El dispositivo ejecuta el comando shell tal y como se especifica en el documento de trabajo.

Documento de trabajo

A continuación se muestra el documento de trabajo y su versión más reciente. La plantilla muestra la ruta al comando de trabajo y el comando proporcionado, que ejecutará el dispositivo.

```
{
  "version": "1.0",
  "steps": [
    {
      "action": {
        "name": "Run-Command",
        "type": "runCommand",
        "input": {
          "command": "${aws:iot:parameter:command}"
        },
        "runAsUser": "${aws:iot:parameter:runAsUser}"
      }
    }
  ]
}
```

Temas

- Cree un trabajo a partir de plantillas AWS administradas mediante el Consola de administración de AWS
- Cree un trabajo a partir de plantillas AWS gestionadas mediante el AWS CLI

Cree un trabajo a partir de plantillas AWS administradas mediante el Consola de administración de AWS

Utilice la Consola de administración de AWS para obtener información sobre las plantillas AWS gestionadas y crear un trabajo con estas plantillas. A continuación, puede guardar el trabajo que ha creado como su propia plantilla personalizada.

Obtención de detalles sobre las plantillas administradas

Puede obtener información sobre las diferentes plantillas administradas que están disponibles para su uso desde la AWS IoT consola.

1. Para ver las plantillas gestionadas disponibles, vaya al centro de plantillas de tareas de la AWS IoT consola y seleccione la pestaña Plantillas gestionadas.
2. Para ver los detalles, elija una plantilla administrada.

La página de detalles contiene la siguiente información:

- Nombre, descripción y nombre de recurso de Amazon (ARN) de la plantilla administrada.
- El entorno en el que se pueden realizar las operaciones remotas, como Linux.
- El documento de trabajo de JSON que especifica la ruta al controlador de trabajos y los comandos que se van a ejecutar en el dispositivo. Por ejemplo, a continuación se muestra un ejemplo de documento de trabajo para la plantilla AWS-Reboot. La plantilla muestra la ruta al controlador de tareas y al script de shell, `reboot.sh`, que el controlador de trabajos debe ejecutar para reiniciar el dispositivo.

```
{
  "version": "1.0",
  "steps": [
    {
      "action": {
        "name": "Reboot",
        "type": "runHandler",
        "input": {
          "handler": "reboot.sh",
          "path": "${aws:iot:parameter:pathToHandler}"
        },
        "runAsUser": "${aws:iot:parameter:runAsUser}"
      }
    }
```

```
  ]
}
```

Para obtener más información sobre el documento de trabajo y sus parámetros para diversas acciones remotas, consulte Acciones remotas y documentos de trabajo de plantillas administradas.

- La última versión del documento de trabajo.

Creación de un trabajo mediante plantillas administradas

Puede usar la consola AWS de administración para elegir una plantilla AWS administrada y usarla para crear un trabajo. En esta sección le demostramos cómo.

También puede iniciar el flujo de trabajo de creación de trabajos y, a continuación, elegir la plantilla AWS gestionada que desee utilizar al crear el trabajo. Para obtener más información acerca de este flujo de trabajo, consulte Cree y gestione trabajos mediante el Consola de administración de AWS.

1. Elija su plantilla AWS gestionada

   Vaya al centro de plantillas de trabajos de la AWS IoT consola, seleccione la pestaña Plantillas gestionadas y, a continuación, elija su plantilla.

2. Creación de un trabajo con la plantilla administrada

   1. En la página de detalles de la plantilla, elija Crear trabajo.

      La consola cambia al paso Propiedades de trabajo personalizadas del flujo de trabajo Crear trabajo, en el que se ha agregado la configuración de la plantilla.

   2. Introduzca un nombre de trabajo alfanumérico único y una descripción y etiquetas opcionales y, a continuación, seleccione Siguiente.

   3. Elija los objetos o grupos de objetos como destinos de trabajo que desee ejecutar en este trabajo.

   4. En la sección Documento de trabajo, se muestra la plantilla con sus ajustes de configuración y parámetros de entrada. Introduzca los valores de los parámetros de entrada de la plantilla elegida. Por ejemplo, si eligió la plantilla AWS-Download-File:

      - En downloadUrl, introduzca la URL del archivo que se va a descargar, por ejemplo: `https://example.com/index.html`.
      - En filePath, introduzca la ruta en el dispositivo para almacenar el archivo descargado, por ejemplo: *path/to/file*.

Si lo desea, también puede introducir valores para los parámetros `runAsUser` y `pathToHandler`. Para obtener más información sobre los parámetros de entrada de cada plantilla, consulte Acciones remotas y documentos de trabajo de plantillas administradas.

5. En la página Configuración del trabajo, elija el tipo de trabajo como trabajo continuo o instantáneo. Un trabajo de instantánea está completo cuando termina de ejecutarse en los dispositivos y grupos de destino. Un trabajo continuo se aplica a grupos de objetos y se ejecuta en cualquier dispositivo que se añada a un grupo de destino específico.
6. Siga añadiendo cualquier configuración adicional para el trabajo y, a continuación, revise y cree el trabajo. Para obtener información sobre las configuraciones adicionales, consulte:
   - Configuraciones de despliegue, programación y cancelación de trabajos
   - Configuraciones de tiempo de espera y reintento de ejecución de trabajos

Creación de plantillas de trabajo personalizadas a partir de plantillas administradas

Puede utilizar una plantilla AWS gestionada y un trabajo personalizado como punto de partida para crear su propia plantilla de trabajo personalizada. Para crear una plantilla de trabajo personalizada, cree primero un trabajo a partir de la plantilla AWS gestionada, tal y como se describe en la sección anterior.

A continuación, puede guardar el trabajo personalizado como plantilla para crear su propia plantilla personalizada. Para guardarlo como plantilla:

1. Vaya al centro de tareas de la AWS IoT consola y elija el trabajo que contiene la plantilla gestionada.
2. Elija Guardar como plantilla de trabajo y, a continuación, cree su plantilla de trabajo personalizada. Para obtener más información acerca de la creación de una plantilla de trabajo personalizada, consulte Creación de una plantilla de trabajo a partir de un trabajo existente.

Cree un trabajo a partir de plantillas AWS gestionadas mediante el AWS CLI

Úselo AWS CLI para obtener información sobre las plantillas AWS administradas y crear un trabajo con estas plantillas. A continuación, puede guardar el trabajo como una plantilla y crear su propia plantilla personalizada.

Enumeración de plantillas administradas

El `list-managed-job-templates` AWS CLI comando muestra todas las plantillas de trabajo de su Cuenta de AWS.

```
aws iot list-managed-job-templates
```

De forma predeterminada, al ejecutar este comando, se muestran todas las plantillas AWS gestionadas disponibles y sus detalles.

```
{
    "managedJobTemplates": [
        {
            "templateArn": "arn:aws:iot:region::jobtemplate/AWS-Reboot:1.0",
            "templateName": "AWS-Reboot",
            "description": "A managed job template for rebooting the device.",
            "environments": [
                "LINUX"
            ],
            "templateVersion": "1.0"
        },
        {
            "templateArn": "arn:aws:iot:region::jobtemplate/AWS-Remove-
Application:1.0",
            "templateName": "AWS-Remove-Application",
            "description": "A managed job template for uninstalling one or more
 applications.",
            "environments": [
                "LINUX"
            ],
            "templateVersion": "1.0"
        },
        {
            "templateArn": "arn:aws:iot:region::jobtemplate/AWS-Stop-Application:1.0",
            "templateName": "AWS-Stop-Application",
            "description": "A managed job template for stopping one or more system
 services.",
            "environments": [
                "LINUX"
            ],
            "templateVersion": "1.0"
```

```
        },

        ...

        {
            "templateArn": "arn:aws:iot:us-east-1::jobtemplate/AWS-Restart-
Application:1.0",
            "templateName": "AWS-Restart-Application",
            "description": "A managed job template for restarting one or more system
 services.",
            "environments": [
                "LINUX"
            ],
            "templateVersion": "1.0"
        }
    ]
}
```

Para obtener más información, consulte ListManagedJobTemplates.

Obtención de detalles sobre una plantilla administrada

El `describe-managed-job-template` AWS CLI comando obtiene detalles sobre una plantilla de trabajo específica. Especifique el nombre de la plantilla de trabajo y, opcionalmente, su versión. Si no se especifica la versión de la plantilla, se devuelve la versión predeterminada predefinida. A continuación se muestra un ejemplo de ejecución del comando para obtener los detalles de la plantilla `AWS-Download-File`.

```
aws iot describe-managed-job-template \
    --template-name AWS-Download-File
```

El comando muestra los detalles de la plantilla y el ARN, su documento de trabajo y el parámetro `documentParameters`, que es una lista de pares clave-valor de los parámetros de entrada de la plantilla. Para obtener información sobre las diferentes plantillas y parámetros de entrada, consulte Acciones remotas y documentos de trabajo de plantillas administradas.

> **Note**
>
> El `documentParameters` objeto devuelto al usar esta API solo debe usarse al crear trabajos a partir de plantillas AWS administradas. El objeto no debe usarse para plantillas

de trabajo personalizadas. Para ver un ejemplo que muestra cómo utilizar este parámetro, consulte Creación de un trabajo mediante plantillas administradas.

```
{
    "templateName": "AWS-Download-File",
    "templateArn": "arn:aws:iot:region::jobtemplate/AWS-Download-File:1.0",
    "description": "A managed job template for downloading a file.",
    "templateVersion": "1.0",
    "environments": [
        "LINUX"
    ],
    "documentParameters": [
        {
            "key": "downloadUrl",
            "description": "URL of file to download.",
            "regex": "(.*?)",
            "example": "http://www.example.com/index.html",
            "optional": false
        },
        {
            "key": "filePath",
            "description": "Path on the device where downloaded file is written.",
            "regex": "(.*?)",
            "example": "/path/to/file",
            "optional": false
        },
        {
            "key": "runAsUser",
            "description": "Execute handler as another user. If not specified, then
 handler is executed as the same user as device client.",
            "regex": "(.){0,256}",
            "example": "user1",
            "optional": true
        },
        {
            "key": "pathToHandler",
            "description": "Path to handler on the device. If not specified, then
 device client will use the current working directory.",
            "regex": "(.){0,4096}",
            "example": "/path/to/handler/script",
            "optional": true
        }
```

```
    ],
    "document": "{\"version\":\"1.0\",\"steps\":[{\"action\":{\"name
\":\"Download-File\",\"type\":\"runHandler\",\"input\":{\"handler\":
\"download-file.sh\",\"args\":[\"${aws:iot:parameter:downloadUrl}\",
\"${aws:iot:parameter:filePath}\"],\"path\":\"${aws:iot:parameter:pathToHandler}\"},
\"runAsUser\":\"${aws:iot:parameter:runAsUser}\"}}]}"
}
```

Para obtener más información, consulte DescribeManagedJobTemplate.

Creación de un trabajo mediante plantillas administradas

El `create-job` AWS CLI comando se puede usar para crear un trabajo a partir de una plantilla de trabajo. Se dirige a un dispositivo llamado `thingOne` y especifica el nombre de recurso de Amazon (ARN) de la plantilla administrada que se va a utilizar como base para el trabajo. Puede anular las configuraciones avanzadas, como las configuraciones del tiempo de espera y de cancelación, omitiendo los parámetros correspondientes del comando `create-job`.

En el ejemplo se muestra cómo crear un trabajo que utiliza la plantilla `AWS-Download-File`. También muestra cómo especificar los parámetros de entrada de la plantilla mediante el parámetro `document-parameters`.

> **Note**
>
> Utilice el `document-parameters` objeto únicamente con plantillas AWS gestionadas. Este objeto no debe usarse con plantillas de trabajo personalizadas.

```
aws iot create-job \
   --targets arn:aws:iot:region:account-id:thing/thingOne \
   --job-id "new-managed-template-job" \
   --job-template-arn arn:aws:iot:region::jobtemplate/AWS-Download-File:1.0 \
   --document-parameters downloadUrl=https://example.com/index.html,filePath=path/to/
file
```

donde:

- *region*es el Región de AWS.
- *account-id*es el Cuenta de AWS número único.

- *thingOne* es el nombre del objeto de IoT al que se dirige el trabajo.
- *AWS-Download-File:1.0*es el nombre de la plantilla gestionada.
- `https://`*example.com/index*`.html` es la URL de la que descargar el archivo.
- `https://`*pathto/file/index* es la ruta en el dispositivo para almacenar el archivo descargado.

Ejecute el siguiente comando para crear un trabajo para la plantilla, *AWS-Download-File*.

```
{
    "jobArn": "arn:aws:iot:region:account-id:job/new-managed-template-job",
    "jobId": "new-managed-template-job",
    "description": "A managed job template for downloading a file."
}
```

Creación de una plantilla de trabajo personalizada a partir de plantillas administradas

1. Cree un trabajo mediante una plantilla administrada tal y como se describe en la sección anterior.
2. Cree una plantilla de trabajo personalizada mediante el ARN del trabajo que ha creado. Para obtener más información, consulte Creación de una plantilla de trabajo a partir de un trabajo existente.

Creación de plantillas de trabajo personalizadas

Puede crear plantillas de trabajo mediante la consola AWS CLI y la AWS IoT consola. También puede crear trabajos a partir de plantillas de trabajo mediante la AWS CLI AWS IoT consola y las aplicaciones web Fleet Hub for AWS IoT Device Management. Para obtener más información sobre cómo trabajar con plantillas de trabajo en las aplicaciones de Fleet Hub, consulte Trabajar con plantillas de trabajo en Fleet Hub para la administración de AWS IoT dispositivos.

> **Note**
>
> El número total de patrones de sustitución en un documento de trabajo debe ser menor o igual a diez.

Temas

- Cree plantillas de trabajo personalizadas mediante el Consola de administración de AWS

- Cree plantillas de trabajo personalizadas mediante el AWS CLI

Cree plantillas de trabajo personalizadas mediante el Consola de administración de AWS

En este tema se explica cómo crear, eliminar y ver los detalles de las plantillas de trabajo mediante la AWS IoT consola.

Creación de una plantilla de trabajo personalizada

Puede crear una plantilla de trabajo personalizada original o a partir de un trabajo existente. También puede crear una plantilla de trabajo personalizada a partir de un trabajo existente que se haya creado mediante una plantilla AWS gestionada. Para obtener más información, consulte Creación de plantillas de trabajo personalizadas a partir de plantillas administradas.

Creación de una plantilla de trabajo original

1. Empiece a crear la plantilla de trabajo

   1. Vaya al centro de plantillas de trabajos de la AWS IoT consola y elija la pestaña Plantillas personalizadas.
   2. Elija Crear plantilla de trabajo.

   > **Note**
   >
   > También puede ir a la página Plantillas de trabajo desde la página Servicios relacionados en Fleet Hub.

2. Especifique las propiedades de la plantilla de trabajo

   En la página Crear plantilla de trabajo, introduzca un identificador alfanumérico para el nombre del trabajo y una descripción alfanumérica para proporcionar detalles adicionales sobre la plantilla.

   > **Note**
   >
   > No recomendamos utilizar información de identificación personal en su puesto IDs o en sus descripciones.

3. Proporcione el documento de trabajo

   Proporcione un archivo de trabajo JSON que esté almacenado en un bucket de S3 o como un documento de trabajo en línea que se especifique en el trabajo. Este archivo de trabajo se convertirá en el documento de trabajo cuando cree un trabajo con esta plantilla.

   Si el archivo de trabajo está almacenado en un bucket de S3, introduzca la URL de S3 o elija Examinar S3 y, a continuación, vaya al documento de trabajo y selecciónelo.

   > **Note**
   >
   > Solo puede seleccionar buckets de S3 de su región actual.

4. Siga añadiendo cualquier configuración adicional para el trabajo y, a continuación, revise y cree el trabajo. Para obtener más información acerca de las configuraciones opcionales adicionales, consulte los siguientes enlaces:

   - Configuraciones de despliegue, programación y cancelación de trabajos
   - Configuraciones de tiempo de espera y reintento de ejecución de trabajos

Creación de una plantilla de trabajo a partir de un trabajo existente

1. Elija el trabajo

   1. Ve al centro de tareas de la AWS IoT consola y elige el trabajo que quieres usar como base para tu plantilla de trabajo.
   2. Elija Guardar como plantilla de trabajo.

   > **Note**
   >
   > Si lo desea, puede elegir un documento de trabajo diferente o editar las configuraciones avanzadas del trabajo original y, a continuación, elegir Crear plantilla de trabajo. La nueva plantilla de trabajo aparece en la página Plantillas de trabajo.

2. Especifique las propiedades de la plantilla de trabajo

   En la página Crear plantilla de trabajo, introduzca un identificador alfanumérico para el nombre del trabajo y una descripción alfanumérica para proporcionar detalles adicionales sobre la plantilla.

   > **Note**
   >
   > El documento de trabajo es el archivo de trabajo que especificó al crear la plantilla. Si el documento de trabajo se especifica dentro del trabajo en lugar de en una ubicación de S3, puede ver el documento del trabajo en la página de detalles de este trabajo.

3. Siga añadiendo cualquier configuración adicional para el trabajo y, a continuación, revise y cree el trabajo. Para obtener información sobre las configuraciones adicionales, consulte:

   - Configuraciones de despliegue, programación y cancelación de trabajos
   - Configuraciones de tiempo de espera y reintento de ejecución de trabajos

Creación de un trabajo a partir de una plantilla de trabajo personalizada

Para crear un trabajo a partir de una plantilla de trabajo personalizada, vaya a la página de detalles de la plantilla de trabajo, tal y como se describe en este tema. También puede crear un trabajo eligiendo la plantilla de trabajo que desee usar al ejecutar el flujo de trabajo de creación de trabajos. Para obtener más información, consulte Cree y gestione trabajos mediante el Consola de administración de AWS.

En este tema se muestra cómo crear un trabajo desde la página de detalles de una plantilla de trabajo personalizada. También puede crear un trabajo a partir de una plantilla AWS gestionada. Para obtener más información, consulte Creación de un trabajo mediante plantillas administradas.

1. Elija la plantilla de trabajo personalizada

   Vaya al centro de plantillas de trabajos de la AWS IoT consola, elija la pestaña Plantillas personalizadas y, a continuación, elija su plantilla.

2. Creación de un trabajo con la plantilla personalizada

   Para crear un trabajo:

   1. En la página de detalles de la plantilla, elija Crear trabajo.

La consola cambia al paso Propiedades de trabajo personalizadas del flujo de trabajo Crear trabajo, en el que se ha agregado la configuración de la plantilla.

2. Introduzca un nombre de trabajo alfanumérico único y una descripción y etiquetas opcionales y, a continuación, seleccione Siguiente.
3. Elija los objetos o grupos de objetos como destinos de trabajo que desee ejecutar en este trabajo.

   En la sección Documento de trabajo, se muestra la plantilla con sus ajustes de configuración. Si desea utilizar un documento de trabajo diferente, elija Examinar y seleccione un bucket y un documento diferentes. Elija Siguiente.
4. En la página Configuración del trabajo, elija el tipo de trabajo como trabajo continuo o instantáneo. Un trabajo de instantánea está completo cuando termina de ejecutarse en los dispositivos y grupos de destino. Un trabajo continuo se aplica a grupos de objetos y se ejecuta en cualquier dispositivo que se añada a un grupo de destino específico.
5. Siga añadiendo cualquier configuración adicional para el trabajo y, a continuación, revise y cree el trabajo. Para obtener información sobre las configuraciones adicionales, consulte:
   - Configuraciones de despliegue, programación y cancelación de trabajos
   - Configuraciones de tiempo de espera y reintento de ejecución de trabajos

> **Note**
>
> Cuando un trabajo creado a partir de una plantilla de trabajo actualiza los parámetros existentes proporcionados por la plantilla de trabajo, los parámetros actualizados anularán los existentes proporcionados por la plantilla de trabajo para ese trabajo.

También puede crear trabajos a partir de plantillas de trabajo con las aplicaciones web de Fleet Hub. Para obtener información sobre la creación de trabajos en Fleet Hub, consulte Trabajar con plantillas de trabajo en Fleet Hub para la administración de AWS IoT dispositivos.

Eliminación de una plantilla de trabajo

Para eliminar una plantilla de trabajo, primero vaya al centro de plantillas de trabajo de la AWS IoT consola y elija la pestaña Plantillas personalizadas. A continuación, seleccione la plantilla que desea eliminar y, a continuación, elija Siguiente.

> **Note**
>
> La eliminación es permanente y la plantilla de trabajo ya no aparece en la pestaña Plantillas personalizadas.

Cree plantillas de trabajo personalizadas mediante el AWS CLI

En este tema se explica cómo crear, eliminar y recuperar los detalles sobre las plantillas de trabajo mediante la AWS CLI.

Creación de una plantilla de trabajo desde cero

El siguiente AWS CLI comando muestra cómo crear un trabajo mediante un documento de trabajo (*job-document.json*) almacenado en un bucket de S3 y un rol con permiso para descargar archivos de Amazon S3 (*S3DownloadRole*).

```
aws iot create-job-template  \
      --job-template-id 010  \
      --description "My custom job template for updating the device firmware"
      --document-source https://s3.amazonaws.com/amzn-s3-demo-bucket/job-document.json
 \
      --timeout-config inProgressTimeoutInMinutes=100 \
      --job-executions-rollout-config "{ \"exponentialRate\": { \"baseRatePerMinute\":
 50, \"incrementFactor\": 2, \"rateIncreaseCriteria\": { \"numberOfNotifiedThings\":
 1000, \"numberOfSucceededThings\": 1000}}, \"maximumPerMinute\": 1000}" \
      --abort-config "{ \"criteriaList\": [ { \"action\": \"CANCEL\", \"failureType
\": \"FAILED\", \"minNumberOfExecutedThings\": 100, \"thresholdPercentage\": 20},
 { \"action\": \"CANCEL\", \"failureType\": \"TIMED_OUT\", \"minNumberOfExecutedThings
\": 200, \"thresholdPercentage\": 50}]}" \
      --presigned-url-config "{\"roleArn\":\"arn:aws:iam::123456789012:role/
S3DownloadRole\", \"expiresInSec\":3600}"
```

El parámetro `timeout-config` opcional especifica la cantidad de tiempo que cada dispositivo tiene para finalizar su ejecución del trabajo. El temporizador comienza cuando el estado de ejecución del trabajo se establece en `IN_PROGRESS`. Si el estado de ejecución del trabajo no se establece en otro estado terminal antes de que se cumpla el plazo, se establecerá en `TIMED_OUT`.

El temporizador en curso no se puede actualizar y se aplica a todos los lanzamientos del trabajo. Cuando el lanzamiento de un trabajo permanece en ese `IN_PROGRESS` estado durante más tiempo que este intervalo, se produce un error en el lanzamiento del trabajo y pasa al `TIMED_OUT` estado terminal. AWS IoT también publica una notificación MQTT.

Para obtener más información acerca de cómo crear configuraciones sobre despliegues de trabajos y anulaciones, consulte Despliegue de trabajos y configuración de anulaciones.

> **Note**
>
> Los documentos de trabajo que está especificados como archivos de Amazon S3 se recuperan en el momento en el que crea el trabajo. Si cambia el contenido del archivo de Amazon S3 que usó como origen de su documento de trabajo después de haberlo creado, no cambia lo que se envía a los destinos del trabajo.

Creación de una plantilla de trabajo a partir de un trabajo existente

El siguiente AWS CLI comando crea una plantilla de trabajo especificando el nombre de recurso de Amazon (ARN) de un trabajo existente. La nueva plantilla de trabajo utiliza todas las configuraciones especificadas en el trabajo. Si lo desea, puede cambiar cualquiera de las configuraciones del trabajo existente mediante cualquiera de los parámetros opcionales.

```
aws iot create-job-template  \
      --job-arn arn:aws:iot:region:123456789012:job/job-name  \
      --timeout-config inProgressTimeoutInMinutes=100
```

Obtención de detalles sobre una plantilla de trabajo

El siguiente AWS CLI comando obtiene detalles sobre una plantilla de trabajo específica.

```
aws iot describe-job-template \
      --job-template-id template-id
```

El comando muestra el resultado siguiente.

```
{
   "abortConfig": {
      "criteriaList": [
         {
            "action": "string",
            "failureType": "string",
            "minNumberOfExecutedThings": number,
            "thresholdPercentage": number
         }
      ]
   },
   "createdAt": number,
   "description": "string",
   "document": "string",
   "documentSource": "string",
   "jobExecutionsRolloutConfig": {
      "exponentialRate": {
         "baseRatePerMinute": number,
         "incrementFactor": number,
         "rateIncreaseCriteria": {
            "numberOfNotifiedThings": number,
            "numberOfSucceededThings": number
         }
      },
      "maximumPerMinute": number
   },
   "jobTemplateArn": "string",
   "jobTemplateId": "string",
   "presignedUrlConfig": {
      "expiresInSec": number,
      "roleArn": "string"
   },
   "timeoutConfig": {
      "inProgressTimeoutInMinutes": number
   }
}
```

Obtención de una lista de las plantillas de trabajo

El siguiente AWS CLI comando muestra todas las plantillas de trabajo de su Cuenta de AWS.

```
aws iot list-job-templates
```

El comando muestra el resultado siguiente.

```
{
   "jobTemplates": [
      {
         "createdAt": number,
         "description": "string",
         "jobTemplateArn": "string",
         "jobTemplateId": "string"
      }
   ],
   "nextToken": "string"
}
```

Para recuperar páginas adicionales de resultados, utilice el valor del campo `nextToken`.

Eliminación de una plantilla de trabajo

El siguiente AWS CLI comando elimina una plantilla de trabajo especificada.

```
aws iot delete-job-template \
      --job-template-id template-id
```

El comando no muestra ninguna salida.

Creación de un trabajo a partir de una plantilla de trabajo personalizada

El siguiente AWS CLI comando crea un trabajo a partir de una plantilla de trabajo personalizada. Se dirige a un dispositivo llamado `thing0ne` y especifica el nombre de recurso de Amazon (ARN) de la plantilla de trabajo que se va a utilizar como base para el trabajo. Puede anular las configuraciones avanzadas, como las configuraciones del tiempo de espera y de cancelación, omitiendo los parámetros correspondientes del comando `create-job`.

> **Warning**
>
> El objeto `document-parameters` se debe utilizar con el comando `create-job` únicamente al crear trabajos a partir de plantillas administradas por AWS . Este objeto no debe usarse con plantillas de trabajo personalizadas. Para ver un ejemplo que muestra cómo crear trabajos mediante este parámetro, consulte Creación de un trabajo mediante plantillas administradas.

```
aws iot create-job \
      --targets arn:aws:iot:region:123456789012:thing/thingOne  \
      --job-template-arn arn:aws:iot:region:123456789012:jobtemplate/template-id
```

Configuraciones de trabajos

Puede tener las siguientes configuraciones adicionales para cada trabajo que implemente en los destinos especificados.

- Despliegue: define cuántos dispositivos reciben el documento de trabajo cada minuto.
- Programación: programa un trabajo para una fecha y hora futuras, además de utilizar periodos de mantenimiento periódicos.
- Anular: cancela un trabajo en casos como cuando algunos dispositivos no reciben la notificación de trabajo o cuando los dispositivos notifican un error al ejecutarlo.
- Tiempo de espera: si los destinos de trabajo no responden dentro de un periodo determinado después de que se haya iniciado la ejecución del trabajo, este puede fallar.
- Reintentar: vuelve a intentar la ejecución del trabajo si el dispositivo informa de un error al intentar completarla o si se agota su tiempo de espera para dicha ejecución.

Al utilizar estas configuraciones, se puede monitorizar el estado de la ejecución del trabajo y evitar que se envíe una actualización incorrecta a toda la flota.

Temas

- Cómo funcionan las configuraciones de trabajos
- Especificación de configuraciones adicionales

Cómo funcionan las configuraciones de trabajos

Las configuraciones de despliegue y anulación se utilizan cuando se implementa un trabajo y las configuraciones de tiempo de espera y reintento, para su ejecución. En las siguientes secciones se muestra más información sobre el funcionamiento de estas configuraciones.

Temas

- Configuraciones de despliegue, programación y cancelación de trabajos
- Configuraciones de tiempo de espera y reintento de ejecución de trabajos

Configuraciones de despliegue, programación y cancelación de trabajos

Puede utilizar las configuraciones de despliegue, programación y anulación de tareas para definir cuántos dispositivos reciben el documento de trabajo, programar el despliegue de un trabajo y determinar los criterios para cancelarlo.

Configuración del despliegue de trabajos

Puede especificar la rapidez con la que se notifica a los destinos la ejecución de un trabajo pendiente. También puede crear un despliegue por etapas para administrar las actualizaciones, los reinicios y otras operaciones. Para especificar cómo se notifican los destinos, utilice las velocidades de despliegue de trabajos.

velocidades de despliegue de trabajos

Puede crear una configuración de despliegue mediante una velocidad de despliegue constante o una velocidad de despliegue exponencial. Para especificar el número máximo de destinos de trabajo de los que se informará por minuto, utilice una velocidad de despliegue constante.

AWS IoT los trabajos se pueden implementar utilizando tasas de implementación exponenciales a medida que se cumplan varios criterios y umbrales. Si el número de trabajos fallidos coincide con un conjunto de criterios especificados, puede cancelar el despliegue de los trabajos. Los criterios de la velocidad de despliegue de trabajos se establecen al crear un trabajo mediante el objeto `JobExecutionsRolloutConfig`. Los criterios de anulación de un trabajo también se establecen en el momento de crear un trabajo a través del objeto `AbortConfig`.

El siguiente ejemplo muestra cómo funcionan las velocidades de despliegue. Por ejemplo, el despliegue de un trabajo con una velocidad base de 50 por minuto, un factor de incremento de 2 y

un número de dispositivos notificados y completados con éxito de 1000 cada uno funcionaría de la siguiente manera: el trabajo se iniciará a una velocidad de 50 ejecuciones de trabajos por minuto y continuará a ese ritmo hasta que 1000 objetos hayan recibido notificaciones de ejecución de tareas o se hayan realizado 1000 ejecuciones de trabajos correctas.

En la siguiente tabla se muestra cómo se produciría el despliegue durante los primeros cuatro incrementos.

Velocidad de despliegue por minuto	50	100	200	400
Número de dispositivos notificados o de ejecuciones de tareas realizadas correctamente para justificar un aumento de la velocidad	1 000	2,000	3000	4.000

> **Note**
>
> Si se encuentra en el límite máximo de 500 trabajos simultáneos (`isConcurrent = True`), todos los trabajos activos permanecerán con el estado de `IN-PROGRESS` y no se ejecutará ningún trabajo nuevo hasta que el número de trabajos simultáneos sea igual o inferior a 499 (`isConcurrent = False`)). Esto se aplica a los trabajos instantáneos y continuos.
> Si `isConcurrent = True`, el trabajo actualmente está desplegando las ejecuciones de trabajos en todos los dispositivos del grupo de destino. Si `isConcurrent = False`, el trabajo ha completado el despliegue de todas las ejecuciones de trabajos en todos los dispositivos del grupo de destino. Actualizará su estado una vez que todos los dispositivos del grupo de destino alcancen un estado terminal, o un porcentaje mínimo del grupo de destino si ha seleccionado una configuración de anulación de trabajo. Los estados del nivel de trabajo para `isConcurrent = True` y `isConcurrent = False` son ambos `IN_PROGRESS`.
> Para obtener más información sobre los límites de trabajos activos y simultáneos, consulte Límites de trabajos activos y simultáneos.

Velocidades de despliegue de trabajos continuos que utilizan grupos de objetos dinámicos

Cuando utilizas un trabajo continuo para implementar operaciones remotas en tu flota, AWS IoT Jobs ejecuta tareas para los dispositivos de tu grupo objetivo. En el caso de los nuevos dispositivos que se

añaden al grupo de objetos dinámicos, estas ejecuciones de trabajos se siguen extendiendo a esos dispositivos incluso después de haber creado el trabajo.

La configuración de despliegue puede controlar las velocidades de despliegue solo para los dispositivos que se agreguen al grupo hasta la creación del trabajo. Una vez creado un trabajo, en el caso de los dispositivos nuevos, las ejecuciones del trabajo se crean prácticamente en tiempo real en cuanto los dispositivos se unen al grupo de destino.

Configuración de programación de trabajos

Puede programar un trabajo continuo o instantáneo con hasta un año de antelación utilizando una hora de inicio, una hora de finalización y un comportamiento de finalización predeterminados para determinar lo que ocurrirá con la ejecución de cada trabajo al llegar la hora de finalización. Además, puede crear un periodo de mantenimiento periódico opcional con una frecuencia, hora de inicio y duración flexibles para los trabajos continuos a fin de desplegar un documento de trabajo en todos los dispositivos del grupo de destino.

Configuraciones de programación de trabajos

Hora de inicio

La hora de inicio de un trabajo programado es la fecha y hora futuras en que el trabajo comenzará a desplegar el documento de trabajo en todos los dispositivos del grupo de destino. La hora de inicio de un trabajo programado se aplica a los trabajos continuos y a los trabajos instantáneos. Cuando se crea inicialmente un trabajo programado, mantiene un estado de `SCHEDULED`. Al llegar al `startTime` que ha seleccionado, se actualiza a `IN_PROGRESS` y comienza el despliegue del documento de trabajo. El `startTime` debe ser menor o igual a un año desde la fecha y hora iniciales en que se creó el trabajo programado.

Para obtener más información sobre la sintaxis `startTime` cuando se utiliza un comando de API o el AWS CLI, consulte Timestamp.

En el caso de un trabajo con la configuración de programación opcional que se lleve a cabo durante un periodo de mantenimiento periódico en una ubicación con horario de verano (DST), la hora cambiará una hora al pasar del horario de verano al horario estándar y viceversa.

> **Note**
>
> La zona horaria que se muestra en Consola de administración de AWS es la zona horaria actual de su sistema. Sin embargo, estas zonas horarias se convertirán a UTC en el sistema.

Hora de finalización

La hora de finalización de un trabajo programado es la fecha y hora futuras en las que el trabajo detendrá el despliegue del documento de trabajo en los dispositivos restantes del grupo de destino. La hora de finalización de un trabajo programado se aplica a los trabajos continuos y a los instantáneos. Una vez que un trabajo programado llega al `endTime` seleccionado y todas las ejecuciones del trabajo han alcanzado un estado terminal, actualiza su estado de `IN_PROGRESS` a `COMPLETED`. El `endTime` debe ser menor o igual a dos años desde la fecha y hora iniciales en que se creó el trabajo programado. La duración mínima entre `startTime` y `endTime` es de 30 minutos. Los reintentos de ejecución del trabajo se realizarán hasta que el trabajo alcance el `endTime`, a continuación, el `endBehavior` dictará cómo proceder.

Para obtener más información sobre la sintaxis `endTime` cuando se utiliza un comando de API o el AWS CLI, consulte Timestamp.

En el caso de un trabajo con la configuración de programación opcional que se lleve a cabo durante un periodo de mantenimiento periódico en una ubicación con horario de verano (DST), la hora cambiará una hora al pasar del horario de verano al horario estándar y viceversa.

> **Note**
>
> La zona horaria que se muestra en Consola de administración de AWS es la zona horaria actual de su sistema. Sin embargo, estas zonas horarias se convertirán a UTC en el sistema.

Comportamiento final

El comportamiento final de un trabajo programado determina lo que ocurre con el trabajo y con todas las ejecuciones de trabajos pendientes cuando el trabajo alcanza el `endTime` seleccionado.

A continuación se enumeran los comportamientos finales que se pueden seleccionar al crear el trabajo o la plantilla de trabajo:

- `STOP_ROLLOUT`
  - `STOP_ROLLOUT` detiene el despliegue del documento de trabajo en todos los dispositivos restantes del grupo de destino del trabajo. Además, todas las ejecuciones de trabajos `QUEUED` y `IN_PROGRESS` continuarán hasta que alcancen un estado terminal. Este es el comportamiento final predeterminado a menos que se seleccione `CANCEL` o `FORCE_CANCEL`.
- `CANCEL`

  - `CANCEL` detiene el despliegue del documento de trabajo en todos los dispositivos restantes del grupo de destino del trabajo. Además, todas las ejecuciones de trabajos `QUEUED` se cancelarán, mientras que todas las ejecuciones de trabajos `IN_PROGRESS` continuarán hasta que alcancen un estado terminal.
- `FORCE_CANCEL`
  - `FORCE_CANCEL` detiene el despliegue del documento de trabajo en todos los dispositivos restantes del grupo de destino del trabajo. Además, se cancelarán todas las ejecuciones de trabajos `QUEUED` y `IN_PROGRESS`.

> **Note**
>
> Para poder seleccionar un `endbehavior` debe seleccionar una `endtime`.

Duración máxima

La duración máxima de un trabajo programado debe ser inferior o igual a dos años, independientemente del `startTime` y `endTime`.

En la siguiente tabla se enumeran los escenarios de duración más comunes de un trabajo programado:

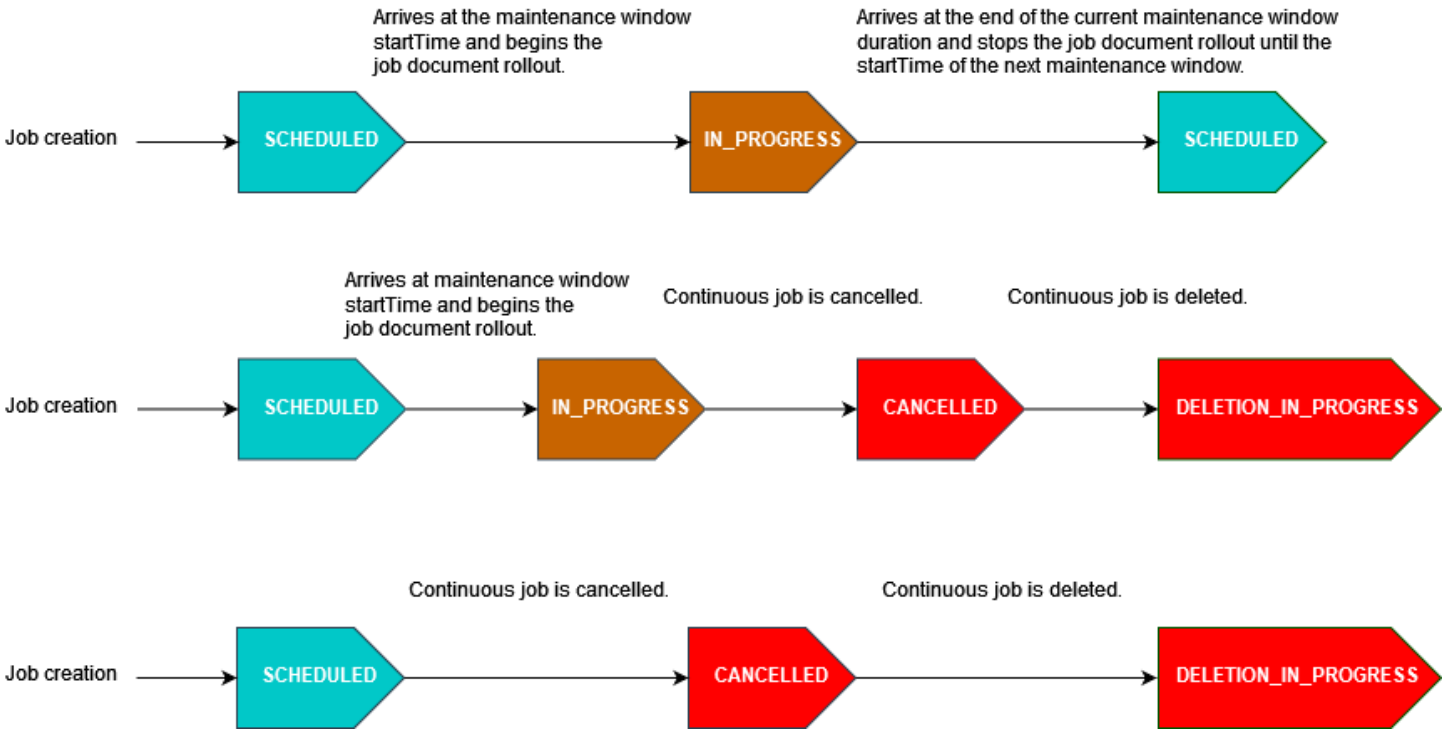
Número de ejemplo de trabajo programado	startTime	endTime	Duración máxima
1	Inmediatamente después de la creación inicial del trabajo.	Un año después de la creación inicial del trabajo.	Un año
2	Un mes después de la creación inicial del trabajo.	13 meses después de la creación inicial del trabajo.	Un año

Número de ejemplo de trabajo programado	startTime	endTime	Duración máxima
3	Un año después de la creación inicial del trabajo.	Dos años después de la creación inicial del trabajo.	Un año
4	Inmediatamente después de la creación inicial del trabajo.	Dos años después de la creación inicial del trabajo.	Dos años

Periodo de mantenimiento periódico

El período de mantenimiento es una configuración opcional dentro de la configuración de programación del Consola de administración de AWS y `SchedulingConfig` dentro del `CreateJob` y `CreateJobTemplate` APIs. Puede configurar un periodo de mantenimiento periódico con una hora de inicio, una duración y una frecuencia predeterminados (diaria, semanal o mensual) en que tendrá lugar. Los periodos de mantenimiento solo se aplican a los trabajos continuos. La duración máxima de un periodo de mantenimiento periódico es de 23 horas y 50 minutos.

El siguiente diagrama ilustra los estados de los trabajos en varios escenarios de trabajos programados con un intervalo de mantenimiento opcional:



Para obtener más información acerca de los estados de un trabajo, consulte Trabajos y estados de ejecución de los trabajos.

> **Note**
>
> Si un trabajo llega al `endTime` durante un periodo de mantenimiento, se actualizará de `IN_PROGRESS` a `COMPLETED`. Además, las ejecuciones de los trabajos restantes seguirán el `endBehavior` del trabajo.

Expresiones cron

En el caso de los trabajos programados que despliegan el documento de trabajo durante un periodo de mantenimiento con una frecuencia personalizada, esta se introduce mediante una expresión cron. Una expresión cron tiene seis campos obligatorios, que están separados por un espacio en blanco.

Sintaxis

```
cron(fields)
```

Campo	Valores	Caracteres comodín
Minutos	0-59	, - * /
Horas	0-23	, - * /
D ay-of-month	1-31	, - * ? / L W
Mes	1-12 o JAN-DEC	, - * /
D ay-of-week	1-7 o SUN-SAT	, - * ? L #
Año	1970-2199	, - * /

Caracteres comodín

- El carácter comodín , (coma) incluye valores adicionales. En el campo Month, JAN, FEB, MAR incluiría enero, febrero y marzo.
- El carácter comodín - (guion) especifica los intervalos. En el campo Day, 1-15 incluiría los días del 1 al 15 del mes especificado.
- El * (asterisco) incluye todos los valores del campo. En el campo Hours, * incluiría cada hora. No puedes usar el asterisco (*) en Day-of-week los campos Day-of-month y. Si lo utiliza en uno, debe utilizar ? en el otro.
- El comodín / (barra inclinada) especifica incrementos. En el campo Minutos, puede escribir 1/10 para especificar cada décimo minuto, empezando desde el primer minuto de la hora (por ejemplo, los minutos 11, 21 y 31, etc.).
- El comodín ? (signo de interrogación) especifica uno u otro. En el Day-of-month campo, puede escribir 7 y, si no le importa qué día de la semana es el 7, ¿puede escribir? en el Day-of-week campo.
- El comodín L en los campos Day-of-month o Day-of-week especifica el último día del mes o de la semana.
- El **`W`** comodín del Day-of-month campo especifica un día de la semana. En el Day-of-month campo, **`3W`** especifica el día de la semana más cercano al tercer día del mes.
- El comodín # del Day-of-week campo especifica una instancia determinada del día de la semana especificado dentro de un mes. Por ejemplo, 3#2 sería el segundo martes del mes: el número

3 hace referencia al martes, ya que es el tercer día de la semana en el calendario anglosajón, mientras que 2 hace referencia al segundo día de ese tipo dentro de un mes.

> **Note**
>
> Si utiliza el carácter «#», solo puede definir una expresión en el day-of-week campo. Por ejemplo, `"3#1,6#3"` no es válido porque se interpreta como dos expresiones.

Restricciones

- No se pueden especificar los campos Day-of-month y Day-of-week en la misma expresión Cron. Si especifica un valor o (o un *) en uno de estos campos, debe utilizar un ? en el otro.

Ejemplos

Consulte los siguientes ejemplos de cadenas cron cuando utilice una expresión cron para el `startTime` de un periodo de mantenimiento periódico.

Minutos	Horas	Día del mes	Mes	Día de la semana	Año	Significado
0	10	*	*	?	*	Ejecutar a las 10:00 h (UTC) todos los días
15	12	*	*	?	*	Ejecutar a las 12:15 h (UTC) todos los días
0	18	?	*	MON-FRI	*	Ejecutar a las 18:00 h (UTC)

Minutos	Horas	Día del mes	Mes	Día de la semana	Año	Significado
						de lunes a viernes
0	8	1	*	?	*	Ejecutar a las 08:00 horas (UTC) todos los primeros de mes

Lógica de finalización de la duración del periodo de mantenimiento periódico

Cuando el despliegue de un trabajo durante un periodo de mantenimiento llegue al final de la duración del periodo de mantenimiento actual, se llevarán a cabo las siguientes acciones:

- El trabajo detendrá todos los despliegues del documento de trabajo a los dispositivos restantes del grupo de destino. Se reanudará en el `startTime` del próximo periodo de mantenimiento.
- Todas las ejecuciones de trabajos con un estado de `QUEUED` permanecerán en `QUEUED` hasta el `startTime` del siguiente periodo de mantenimiento. En él, pueden cambiar a `IN_PROGRESS` cuando el dispositivo esté listo para empezar a realizar las acciones especificadas en el documento de trabajo.
- Todas las ejecuciones de tareas con un estado de `IN_PROGRESS` seguirán realizando las acciones especificadas en el documento de trabajo hasta que alcancen un estado terminal. Cualquier reintento, tal como se especifica en `JobExecutionsRetryConfig`, se realizará en el `startTime` del siguiente periodo de mantenimiento.

Configuración de anulación del trabajo

Esta configuración se utiliza para crear un criterio para cancelar un trabajo cuando un porcentaje umbral de dispositivos lo cumple. Por ejemplo, puede usar esta configuración para cancelar un trabajo en los siguientes casos:

- Cuando un porcentaje mínimo de dispositivos no recibe las notificaciones de ejecución de tareas, por ejemplo, cuando el dispositivo no es compatible con una actualización Over-The-Air (OTA). En este caso, el dispositivo puede informar de un estado `REJECTED`.
- Cuando un porcentaje umbral de dispositivos informa de un error en la ejecución de sus trabajos, como cuando el dispositivo se desconecta al intentar descargar el documento de trabajo desde una URL de Amazon S3. En esos casos, el dispositivo debe estar programado para informar del estado `FAILURE` a AWS IoT.
- Cuando se informa de un estado `TIMED_OUT` porque se agota el tiempo de espera para la ejecución del trabajo en un porcentaje umbral de dispositivos una vez iniciadas las ejecuciones del trabajo.
- Cuando se producen varios errores en los reintentos. Al añadir una configuración de reintentos, cada reintento puede suponer un coste adicional para la Cuenta de AWS. En esos casos, cancelar el trabajo puede cancelar las ejecuciones de trabajos en cola y evitar que se vuelvan a intentar estas ejecuciones. Para obtener más información sobre la configuración de los reintentos y su uso con la configuración de anulación, consulte Configuraciones de tiempo de espera y reintento de ejecución de trabajos.

Puedes configurar una condición de cancelación de tareas mediante la AWS IoT consola o la API de AWS IoT Jobs.

Configuraciones de tiempo de espera y reintento de ejecución de trabajos

Utilice la configuración de tiempo de espera de ejecución de tareas para enviarle Notificaciones de trabajos cuando la ejecución de un trabajo haya estado en curso durante más tiempo del establecido. Utilice la configuración de reintento de ejecución del trabajo para reintentar la ejecución cuando el trabajo falle o se agote el tiempo de espera.

Configuración del tiempo de espera de las ejecuciones de trabajo

Utilice la configuración del tiempo de espera de las ejecuciones de trabajo cada vez que la ejecución de un trabajo se bloquee en el estado `IN_PROGRESS` durante un periodo de tiempo más largo de lo previsto. Una vez que el trabajo esté `IN_PROGRESS`, puede monitorizar el progreso de su ejecución.

Temporizadores para los tiempos de espera de los trabajos

Existen dos tipos de temporizadores: temporizadores en curso y temporizadores de pasos.

Temporizadores en curso

Al crear un trabajo o una plantilla de trabajo, puede especificar un valor para el temporizador en curso que oscile entre 1 minuto y 7 días. Puede actualizar el valor de este temporizador hasta el inicio de la ejecución del trabajo. Una vez iniciado el temporizador, no se puede actualizar y el valor del temporizador se aplica a todas las ejecuciones del trabajo. Cuando la ejecución de un trabajo permanece en ese `IN_PROGRESS` estado durante más tiempo que este intervalo, se produce un error en la ejecución del trabajo y pasa al `TIMED_OUT` estado terminal. AWS IoT también publica una notificación MQTT.

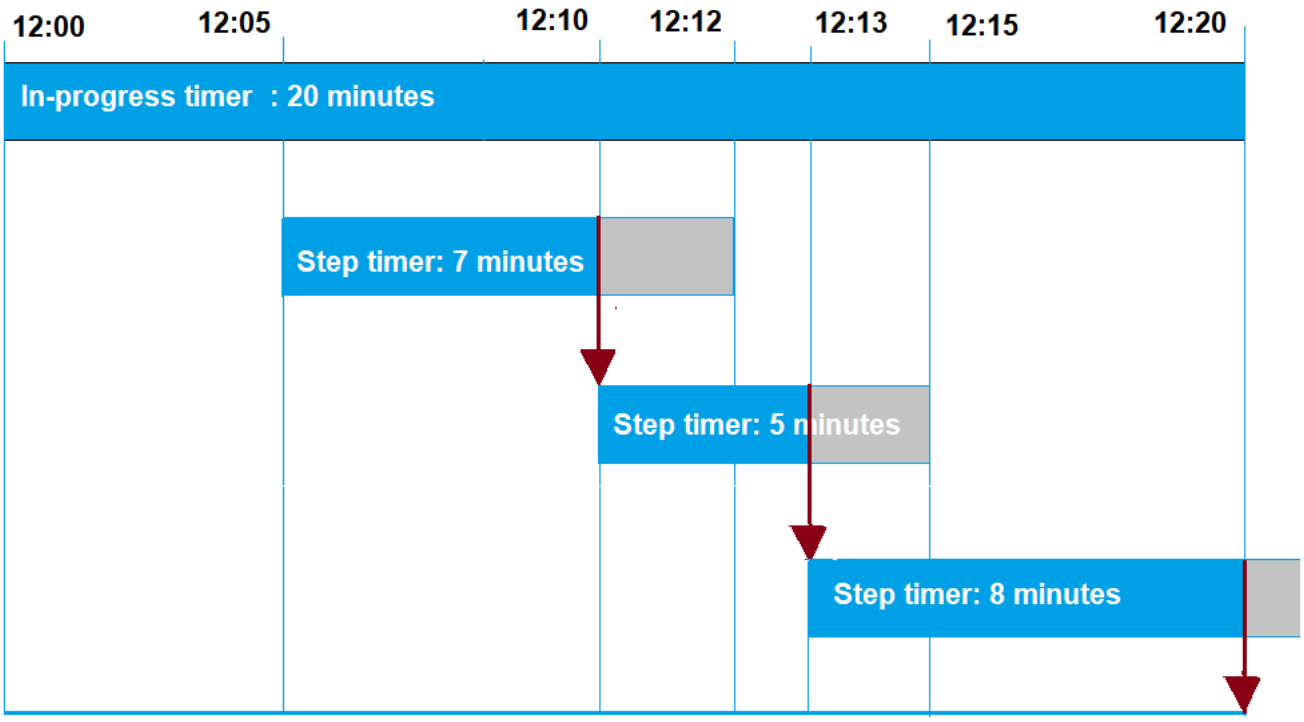
Temporizador de pasos

También puede configurar un temporizador de pasos que se aplique únicamente a la ejecución del trabajo que desee actualizar. Este temporizador no tiene efecto en el temporizador en curso. Puede establecer un nuevo valor para este temporizador cada vez que actualice la ejecución de un trabajo. También puede crear un nuevo temporizador de pasos al iniciar la próxima ejecución de trabajo pendiente para un objeto. Si la ejecución del trabajo permanece en estado `IN_PROGRESS` durante un periodo superior a este intervalo del temporizador de pasos, generará un error y cambiará al estado terminal `TIMED_OUT`.

> **Note**
>
> Puede configurar el temporizador en curso mediante la AWS IoT consola o la API de AWS IoT Jobs. Para especificar el temporizador de pasos, use la API.

Cómo funcionan los temporizadores para los tiempos de espera de los trabajos

A continuación, se ilustran las formas en las que los tiempos de espera en curso y de pasos interactúan entre sí en un periodo de espera de 20 minutos.



Lo siguiente muestra los diferentes pasos:

1. 12:00

   Al crear un trabajo, se crea un nuevo trabajo y se activa un temporizador de veinte minutos en curso. El temporizador en curso comienza a correr y la ejecución del trabajo pasa al estado `IN_PROGRESS`.

2. 12:05

   Se crea un nuevo temporizador de pasos con un valor de 7 minutos. La ejecución del trabajo finalizará ahora a las 12:12.

3. 12:10

   Se crea un nuevo temporizador de pasos con un valor de 5 minutos. Cuando se crea un nuevo temporizador de pasos, se descarta el anterior y la ejecución del trabajo acabará ahora a las 12:15 h.

4. 12:13

   Se crea un nuevo temporizador de pasos con un valor de 9 minutos. El temporizador de pasos anterior se descarta y la ejecución del trabajo acaba ahora a las 12:20 h, ya que el temporizador

en curso finaliza a esa hora. El temporizador de pasos no puede superar el límite absoluto del temporizador en curso.

Configuración de reintento de ejecución de trabajos

Puede utilizar la configuración de reintento para reintentar la ejecución del trabajo cuando se cumpla un determinado conjunto de criterios. El reintento se puede realizar cuando se agote el tiempo de espera de un trabajo o cuando el dispositivo falle. Para reintentar la ejecución debido a un error en el tiempo de espera, debe habilitar la configuración del tiempo de espera.

Cómo utilizar la configuración de reintentos

Siga los pasos siguientes para configurar los reintentos:

1. Determine si se debe utilizar la configuración de reintento para `FAILED`, `TIMED_OUT` o ambos criterios de error. En cuanto al `TIMED_OUT` estado, una vez informado del estado, AWS IoT Jobs vuelve a intentar automáticamente la ejecución del trabajo en el dispositivo.
2. Respecto del estado `FAILED`, compruebe si se puede reintentar el error de ejecución del trabajo. Si se puede reintentar, programe el dispositivo para que informe de un estado `FAILURE` a AWS IoT. En la siguiente sección se describe más información sobre los errores que se pueden reintentar y los que no.
3. Para especificar el número de reintentos que se van a utilizar para cada tipo de error, utilice la información anterior. Para un solo dispositivo, puede especificar hasta 10 reintentos para ambos tipos de error combinados. Los reintentos se detienen automáticamente cuando una ejecución se realiza correctamente o cuando se alcanza el número de intentos especificado.
4. Añada una configuración de cancelación para cancelar el trabajo en caso de que se produzcan errores repetidos en los reintentos, a fin de evitar que se generen cargos adicionales por un gran número de reintentos.

> **Note**
>
> Cuando un trabajo llega al final de un periodo de mantenimiento periódico, todas las ejecuciones de trabajos `IN_PROGRESS` seguirán realizando las acciones identificadas en el documento de trabajo hasta que alcancen un estado terminal. Si la ejecución de un trabajo alcanza un estado terminal `FAILED` o `TIMED_OUT` fuera de un periodo de mantenimiento, se volverá a intentar en el siguiente periodo si no se agotan los intentos. En el `startTime` del

siguiente periodo de mantenimiento, se creará una nueva ejecución de trabajo y pasará a un estado QUEUED hasta que el dispositivo esté listo para empezar.

Configuración de reintentos y anulaciones

Cada reintento conlleva cargos adicionales para usted. Cuenta de AWS Para evitar incurrir en cargos adicionales por errores repetidos en los reintentos, recomendamos añadir una configuración de anulación. Para obtener más información acerca de los precios, consulte Precios de AWS IoT Device Management.

Es posible que se produzcan varios errores en los reintentos cuando un porcentaje elevado de dispositivos agote el tiempo de espera o notifique un error. En este caso, se puede usar la configuración de anulación para cancelar el trabajo y evitar que se ejecute un trabajo en cola o que se vuelva a intentar.

> **Note**
>
> Cuando se cumplen los criterios de anulación para cancelar la ejecución de un trabajo, solo se cancelan las ejecuciones de trabajos QUEUED. No se realizará ningún reintento de dispositivos que estén en cola. Sin embargo, las ejecuciones de trabajos actuales que tengan un estado IN_PROGRESS no se cancelarán.

Antes de reintentar ejecutar un trabajo fallido, también recomendamos comprobar si la ejecución fallida se puede reintentar, tal y como se describe en la siguiente sección.

Reintento del tipo de error **FAILED**

Para realizar reintentos en caso de error FAILED, los dispositivos deben estar programados para informar del estado FAILURE sobre una ejecución fallida de un trabajo a AWS IoT. Defina la configuración de reintentos con los criterios para reintentar ejecutar los trabajos FAILED y especifique el número de reintentos que se van a realizar. Cuando AWS IoT Jobs detecte el FAILURE estado, intentará automáticamente volver a intentar ejecutar el trabajo en el dispositivo. Los reintentos continúan hasta que la ejecución del trabajo se realice correctamente o se alcance el número máximo de reintentos.

Puede realizar un seguimiento de cada reintento y del trabajo que se está ejecutando en estos dispositivos. Al hacer un seguimiento del estado de la ejecución, una vez que se haya intentado

realizar el número especificado de reintentos, puede usar el dispositivo para informar de los errores e iniciar otro reintento.

Errores que se pueden reintentar y que no

Un error en la ejecución del trabajo puede volver a reintentarse o no. Cada reintento puede generar cargos en la Cuenta de AWS. Para evitar incurrir en cargos adicionales por múltiples reintentos, primero considere comprobar si el error en la ejecución del trabajo se puede reintentar. Un ejemplo de error que se puede reintentar puede ser un error de conexión que el dispositivo detecta al intentar descargar el documento de trabajo desde una URL de Amazon S3. Si se puede reintentar ejecutar un trabajo fallido, programe el dispositivo para que notifique un estado `FAILURE` en caso de que la ejecución del trabajo falle. A continuación, defina la configuración de reintentos para reintentar las ejecuciones `FAILED`.

Si no puede reintentar la ejecución, le recomendamos que programe el dispositivo para que informe de un estado `REJECTED` a AWS IoT y así evitar que se efectúen cargos adicionales en su cuenta. Algunos ejemplos de errores que no se pueden reintentar son los casos en los que el dispositivo no puede recibir una actualización del trabajo o cuando se produce un error de memoria al ejecutar un trabajo. En estos casos, AWS IoT Jobs no volverá a intentar la ejecución del trabajo porque solo lo volverá a intentar cuando detecte un `FAILED` estado o. `TIMED_OUT`

Una vez que haya determinado que un error en la ejecución de un trabajo se puede reintentar, si el reintento sigue fallando, considere la posibilidad de comprobar los registros del dispositivo.

> **Note**
>
> Cuando un trabajo con la configuración de programación opcional alcance su `endTime`, el `endBehavior` seleccionado detendrá el despliegue del documento de trabajo en todos los dispositivos restantes del grupo de destino y determinará cómo proceder con las ejecuciones restantes del trabajo. Los reintentos se realizan si se seleccionan mediante la configuración correspondiente.

Reintento del tipo de error **`TIMEOUT`**

Si habilita el tiempo de espera al crear un trabajo, AWS IoT Jobs intentará volver a intentar la ejecución del trabajo en el dispositivo cuando el estado cambie de a. `IN_PROGRESS` `TIMED_OUT` Este cambio de estado puede producirse cuando se agota el tiempo de espera del temporizador en curso o cuando se activa un temporizador de pasos que especificado en `IN_PROGRESS` y este

finaliza. Los reintentos continúan hasta que la ejecución del trabajo se realice correctamente o se alcance el número máximo de reintentos para el tipo de error en cuestión.

Actualizaciones de trabajos continuos y de la suscripción a grupos de objetos

En el caso de los trabajos continuos cuyo estado sea `IN_PROGRESS`, el número de reintentos se restablece a cero cuando se actualiza la suscripción a un grupo. Por ejemplo, imagine que especificó cinco reintentos y ya se han realizado tres. Si ahora se elimina un objeto del grupo de objetos y, a continuación, vuelve a entrar en él, por ejemplo, en el caso de los grupos de objetos dinámicos, el número de reintentos se restablece a cero. Ahora puede realizar cinco reintentos para el grupo de objetos en lugar de los dos que quedaban por hacer. Además, cuando se elimina un objeto del grupo correspondiente, se cancelan los reintentos adicionales.

Especificación de configuraciones adicionales

Al crear un trabajo o una plantilla de trabajo, puede especificar estas configuraciones adicionales. A continuación se muestra cuándo puede especificar estas configuraciones.

- Al crear una plantilla de trabajo personalizada. Los ajustes de configuración adicionales que especifique se guardarán al crear un trabajo a partir de la plantilla.
- Al crear un trabajo personalizado mediante un archivo de trabajo. El archivo de trabajo puede ser un archivo JSON que se carga en un bucket de S3.
- Al crear un trabajo personalizado mediante una plantilla de trabajo personalizada. Si la plantilla ya tiene estos parámetros especificados, puede reutilizarlos o anularlos especificando nuevos ajustes de configuración.
- Al crear un trabajo personalizado mediante una plantilla AWS gestionada.

Temas

- Especifique las configuraciones de los trabajos mediante el Consola de administración de AWS
- Especifique las configuraciones de los trabajos mediante la API AWS IoT de trabajos

Especifique las configuraciones de los trabajos mediante el Consola de administración de AWS

Puede añadir las diferentes configuraciones para su trabajo mediante la AWS IoT consola. Una vez que haya creado un trabajo, podrá ver los detalles de estado de las configuraciones en la página

de detalles del trabajo. Para obtener más información sobre las diferentes configuraciones y cómo funcionan, consulte Cómo funcionan las configuraciones de trabajos.

Agregue las configuraciones del trabajo al crear un trabajo o una plantilla de trabajo.

Al crear una plantilla de trabajo personalizada

Para especificar la configuración de despliegue al crear una plantilla de trabajo personalizada

1. Vaya al centro de plantillas de trabajo de la AWS IoT consola y elija Crear plantilla de trabajo.
2. Especifique las propiedades de la plantilla de trabajo, proporcione el documento de trabajo, expanda la configuración que desee añadir y, a continuación, especifique los parámetros de configuración.

Al crear un trabajo personalizado

Para especificar la configuración de despliegue al crear un trabajo personalizado

1. Ve al centro de tareas de la AWS IoT consola y selecciona Crear trabajo.
2. Elija Crear un trabajo personalizado y especifique las propiedades del trabajo, los destinos y si desea utilizar un archivo de trabajo o una plantilla para el documento de trabajo. Puede utilizar una plantilla personalizada o una plantilla AWS gestionada.
3. Elija la configuración del trabajo y, a continuación, expanda la Configuración de despliegue para especificar si desea utilizar una Velocidad constante o una Velocidad exponencial. A continuación, especifique los parámetros de configuración.

La siguiente sección muestra los parámetros que se pueden especificar para cada configuración.

Configuración de despliegue

Puede especificar si desea utilizar una velocidad de despliegue constante o exponencial.

- Establecimiento de una velocidad de despliegue constante

  Para establecer una velocidad constante para las ejecuciones de los trabajos, seleccione Velocidad constante y, a continuación, especifique el Máximo por minuto como límite superior de la velocidad. Este valor es opcional y va de 1 a 1000. Si no se establece, se usará 1000 como valor predeterminado.

- Establecimiento de una velocidad de despliegue exponencial

  Para establecer una velocidad exponencial, elija Velocidad exponencial y, a continuación, especifique estos parámetros:

  - Velocidad de base por minuto

    La velocidad a la que se ejecutan los trabajos hasta que se alcance el umbral de Número de dispositivos notificados o Número de dispositivos correctos para los Criterios de aumento de velocidad.

  - Factor de incremento

    El factor exponencial por el que la velocidad de despliegue aumenta una vez que se alcanza el umbral de Número de dispositivos notificados o Número de dispositivos correctos para los Criterios de aumento de velocidad.

  - Criterios de aumento de velocidad

    El umbral para el Número de dispositivos notificados o Número de dispositivos correctos.

Configuración de anulación

Seleccione Agregar nueva configuración y especifique los siguientes parámetros para cada configuración:

- Tipo de error

  Especifica los tipos de error que inician la anulación de una tarea. Entre ellos se incluyen FAILED, REJECTED, TIMED_OUT o ALL.

- Factor de incremento

  Especifica el número de ejecuciones de trabajos completadas que deben ocurrir antes de que se cumplan los criterios de anulación del trabajo.

- Porcentaje de umbral

  Especifica el número total de objetos ejecutados que inician la anulación de un trabajo.

Configuración de programación

Cada trabajo puede comenzar inmediatamente después de su creación inicial, programarse para que comience en una fecha y hora posteriores o tener lugar durante un periodo de mantenimiento periódico.

Seleccione Agregar nueva configuración y especifique los siguientes parámetros para cada configuración:

- Inicio del trabajo

  Especifique la fecha y la hora en que se iniciará el trabajo.

- Periodo de mantenimiento periódico

  Un periodo de mantenimiento periódico define la fecha y la hora específicas en las que un trabajo puede desplegar el documento de trabajo en los dispositivos de destino del trabajo. El periodo de mantenimiento puede repetirse de forma diaria, semanal, mensual o con una periodicidad de día y hora personalizada.

- Finalización del trabajo

  Especifique la fecha y la hora en que finalizará el trabajo.

- Comportamiento de finalización del trabajo

  Seleccione un comportamiento final para todas las ejecuciones de trabajos pendientes cuando el trabajo haya terminado.

> **Note**
>
> Cuando un trabajo con la configuración de programación opcional y una hora de finalización seleccionada llega a dicha hora, detiene el despliegue en todos los dispositivos restantes del grupo de destino. También aprovecha el comportamiento final seleccionado para continuar con las ejecuciones de los trabajos restantes y sus reintentos según la configuración al respecto.

Configuración de tiempo de espera

De forma predeterminada, no hay un tiempo de espera y los trabajos se cancelan o se eliminan. Para usar los tiempos de espera, seleccione Habilitar tiempo de espera y, a continuación, especifique un valor de tiempo de espera entre 1 minuto y 7 días.

Configuración de reintentos

> **Note**
>
> Una vez creado un trabajo, no se puede actualizar el número de reintentos. Solo se puede eliminar la configuración de reintentos para todos los tipos de error. Al crear un trabajo, tenga en cuenta el número adecuado de reintentos que desee utilizar en la configuración. Para evitar incurrir en costes excesivos debido a posibles errores en los reintentos, añada una configuración de anulación.

Seleccione Agregar nueva configuración y especifique los siguientes parámetros para cada configuración:

- Tipo de error

  Especifica los tipos de error que deberían desencadenar un reintento de ejecución de un trabajo. Entre ellos se incluyen Fallido, Tiempo de espera y Todos.

- Número de reintentos

  Especifica el número de reintentos para el Tipo de error elegido. Para ambos tipos de error combinados, se pueden realizar hasta 10 reintentos.

Especifique las configuraciones de los trabajos mediante la API AWS IoT de trabajos

Puede usar la API CreateJobo la CreateJobTemplateAPI para especificar las diferentes configuraciones de trabajo. En las siguientes secciones se describe cómo añadir estas configuraciones. Una vez agregadas las configuraciones, puede JobExecutionSummaryutilizarlas y JobExecutionSummaryForJobver su estado.

Para obtener más información sobre las diferentes configuraciones y cómo funcionan, consulte Cómo funcionan las configuraciones de trabajos.

Configuración de despliegue

Puede especificar una velocidad de despliegue constante o exponencial para la configuración de despliegue.

- Establecimiento de una velocidad de despliegue constante

  Para establecer una velocidad de despliegue constante, use el objeto `JobExecutionsRolloutConfig` para añadir el parámetro `maximumPerMinute` a la solicitud `CreateJob`. Este parámetro especifica el límite superior de la velocidad a la cual pueden producirse las ejecuciones de los trabajos. Este valor es opcional y va de 1 a 1000. Si no se establece, se usará 1000 como valor predeterminado.

```
    "jobExecutionsRolloutConfig": {
        "maximumPerMinute": 1000
    }
```

- Establecimiento de una velocidad de despliegue exponencial

  Para establecer una velocidad de despliegue de trabajos variable, utilice el objeto `JobExecutionsRolloutConfig`. Puede configurar la propiedad `ExponentialRolloutRate` al ejecutar la operación `CreateJob` de la API. En el siguiente ejemplo se establece una velocidad de despliegue exponencial mediante el parámetro `exponentialRate`. Para obtener más información sobre los parámetros, consulte `ExponentialRolloutRate`.

```
{
...
  "jobExecutionsRolloutConfig": {
    "exponentialRate": {
      "baseRatePerMinute": 50,
      "incrementFactor": 2,
      "rateIncreaseCriteria": {
        "numberOfNotifiedThings": 1000,
        "numberOfSucceededThings": 1000
      },
      "maximumPerMinute": 1000
    }
  }
...
}
```

Donde el parámetro:

baseRatePerMinuto

Especifica la velocidad a la cual se ejecutan los trabajos hasta llegar al umbral `numberOfNotifiedThings` o `numberOfSucceededThings`.

incrementFactor

Especifica el factor exponencial según el cual aumenta la velocidad de despliegue tras alcanzar el umbral `numberOfNotifiedThings` o `numberOfSucceededThings`.

rateIncreaseCriteria

Especifica el umbral `numberOfNotifiedThings` o `numberOfSucceededThings`.

Configuración de anulación

Para añadir esta configuración mediante la API, especifique el parámetro `AbortConfig` al ejecutar la operación `CreateJob` o `CreateJobTemplate` de la API. En el siguiente ejemplo, se muestra una configuración de anulación para el despliegue de un trabajo que estaba experimentando varias ejecuciones fallidas, tal y como se especificó en la operación `CreateJob` de la API.

> **Note**
>
> Eliminar la ejecución de un trabajo afecta al valor de cálculo del total de ejecuciones realizadas. Cuando se anula un trabajo, el servicio crea valores `comment` y `reasonCode` automáticos para diferenciar una cancelación promovida por un usuario de una de anulación de un trabajo.

```
  "abortConfig": {
     "criteriaList": [
        {
           "action": "CANCEL",
           "failureType": "FAILED",
           "minNumberOfExecutedThings": 100,
           "thresholdPercentage": 20
        },
        {
           "action": "CANCEL",
           "failureType": "TIMED_OUT",
```

```
            "minNumberOfExecutedThings": 200,
            "thresholdPercentage": 50
        }
      ]
    }
```

Donde el parámetro:

action

Especifica la acción que se debe realizar cuando se han cumplido los criterios de anulación. Este parámetro es necesario y `CANCEL` es el único valor válido.

failureType

Especifica qué tipos de error deben iniciar la anulación de un trabajo. Los valores válidos son `FAILED`, `REJECTED`, `TIMED_OUT` y `ALL`.

minNumberOfExecutedThings

Especifica el número de ejecuciones de trabajos completadas que deben ocurrir antes de que se cumplan los criterios de anulación del trabajo. En este ejemplo, AWS IoT no comprueba si se debe anular un trabajo hasta que al menos 100 dispositivos hayan completado ejecuciones de trabajos.

thresholdPercentage

Especifica el número total de objetos para los que se ejecutan trabajos que pueden iniciar la anulación de un trabajo. En este ejemplo, AWS IoT comprueba secuencialmente e inicia una anulación del trabajo si se alcanza el porcentaje límite. Si al menos el 20 % de las ejecuciones completas fallan una vez finalizadas 100 ejecuciones, se cancela el despliegue del trabajo. Si no se cumple este criterio, comprueba si se ha AWS IoT agotado el tiempo de espera de al menos el 50% de las ejecuciones completadas una vez finalizadas 200 ejecuciones. Si este es el caso, se cancela el despliegue del trabajo.

Configuración de programación

Para añadir esta configuración mediante la API, especifique la `SchedulingConfig` opcional al ejecutar la operación `CreateJob` o `CreateJobTemplate` de la API.

```
    "SchedulingConfig": {
      "endBehavior": string
```

```
    "endTime": string
    "maintenanceWindows": string
    "startTime": string
  }
```

Donde el parámetro:

startTime

Especifica la fecha y la hora en que se iniciará el trabajo.

endTime

Especifica la fecha y la hora en que finalizará el trabajo.

maintenanceWindows

Especifica si se ha seleccionado un periodo de mantenimiento opcional para el trabajo programado a fin de desplegar el documento de trabajo en todos los dispositivos del grupo de destino. El formato de cadena `maintenanceWindow` es YYYY/MM/DD para la fecha y hh:mm para la hora.

endBehavior

Especifica el comportamiento de un trabajo programado al llegar al `endTime`.

> **Note**
>
> El elemento opcional `SchedulingConfig` para un trabajo se puede ver en y. `DescribeJobDescribeJobTemplate` APIs

Configuración de tiempo de espera

Para añadir esta configuración mediante la API, especifique el parámetro `TimeoutConfig` al ejecutar la operación `CreateJob` o `CreateJobTemplate` de la API.

Para usar la configuración de tiempo de espera

1. Para configurar el temporizador en curso al crear un trabajo o una plantilla de trabajo, defina un valor para la `inProgressTimeoutInMinutes` propiedad del objeto opcional TimeoutConfig.

```
    "timeoutConfig": {
```

```
      "inProgressTimeoutInMinutes": number
   }
```

2. Para especificar un temporizador gradual para la ejecución de un trabajo, defina un valor para `stepTimeoutInMinutes` cuando llame UpdateJobExecution. El temporizador de pasos se aplica únicamente a la ejecución del trabajo que actualice. Puede establecer un nuevo valor para este temporizador cada vez que actualice la ejecución de un trabajo.

> **Note**
>
> `UpdateJobExecution` puede descartar un temporizador de pasos que ya se ha creado mediante la creación de un nuevo temporizador de pasos con un valor de -1.

```
{
   ...
    "statusDetails": {
      "string" : "string"
   },
   "stepTimeoutInMinutes": number
}
```

3. Para crear un nuevo temporizador por pasos, también puedes llamar a la operación de la StartNextPendingJobExecutionAPI.

Configuración de reintentos

> **Note**
>
> Al crear un trabajo, tenga en cuenta el número adecuado de reintentos que desee utilizar en la configuración. Para evitar incurrir en costes excesivos debido a posibles errores en los reintentos, añada una configuración de anulación. Una vez creado un trabajo, no se puede actualizar el número de reintentos. Solo puedes establecer el número de reintentos en 0 mediante la operación de UpdateJobAPI.

Para añadir esta configuración mediante la API, especifique el parámetro `jobExecutionsRetryConfig` al ejecutar la operación `CreateJob` o `CreateJobTemplate` de la API.

```
{
...
  "jobExecutionsRetryConfig": {
      "criteriaList": [
         {
            "failureType": "string",
            "numberOfRetries": number
         }
      ]
  }
...
}
```

Donde criteriaList es una matriz que especifica la lista de criterios que determina el número de reintentos permitidos para cada tipo de error de un trabajo.

Dispositivos y trabajos

Los dispositivos pueden comunicarse con AWS IoT Jobs mediante MQTT, HTTP Signature versión 4 o HTTP TLS. Para determinar el punto final que se utilizará cuando el dispositivo se comunique con AWS IoT Jobs, ejecute el DescribeEndpoint comando. Por ejemplo, si ejecuta este comando:

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

obtendrá un resultado similar al siguiente:

```
{
    "endpointAddress": "a1b2c3d4e5f6g7-ats.iot.us-west-2.amazonaws.com"
}
```

Uso del protocolo MQTT

Los dispositivos pueden comunicarse con AWS IoT Jobs mediante el protocolo MQTT. Los dispositivos se suscriben a los temas de MQTT para recibir notificaciones de nuevos trabajos y recibir respuestas del servicio de AWS IoT trabajos. Los dispositivos publican en temas MQTT para consultar o actualizar el estado de lanzamiento de un trabajo. Cada dispositivo tiene su propio tema MQTT general. Para obtener más información acerca de cómo publicar en temas de MQTT y suscribirse a ellos, consulte Protocolos de comunicación de dispositivos.

Con este método de comunicación, el dispositivo utiliza el certificado específico del dispositivo y la clave privada para autenticarse con Jobs. AWS IoT

Los dispositivos pueden suscribirse a los siguientes temas. `thing-name` es el nombre del objeto asociado con el dispositivo.

- **`$aws/things/thing-name/jobs/notify`**

  Suscríbase a este tema para recibir notificaciones cuando se añada o elimine un lanzamiento de trabajo de la lista de lanzamientos de trabajos pendientes.

- **`$aws/things/thing-name/jobs/notify-next`**

  Suscríbase a este tema para recibir notificaciones cuando cambie la siguiente ejecución de un trabajo pendiente.

- **`$aws/things/thing-name/jobs/request-name/accepted`**

  El servicio AWS IoT Jobs publica mensajes de éxito y fracaso sobre un tema de MQTT. El tema se forma añadiendo `accepted` o `rejected` al tema utilizado para realizar la solicitud. Aquí, `request-name` es el nombre de una solicitud como `Get` y el tema puede ser:`$aws/things/myThing/jobs/get`. AWS IoTLuego, Jobs publica mensajes de éxito sobre el `$aws/things/myThing/jobs/get/accepted` tema.

- **`$aws/things/thing-name/jobs/request-name/rejected`**

  Aquí, `request-name` es el nombre de una solicitud, como `Get`. Si la solicitud falló, AWS IoT Jobs publicará los mensajes de error sobre el `$aws/things/myThing/jobs/get/rejected` tema.

También puede utilizar las siguientes operaciones de la API HTTPS:

- Actualizar el estado de una ejecución de trabajo llamando a la API `UpdateJobExecution`.
- Consultar el estado de una ejecución de trabajo llamando a la API `DescribeJobExecution`.
- Recuperar una lista de ejecuciones de trabajo pendientes llamando a la API `GetPendingJobExecutions`.
- Recuperar la siguiente ejecución de trabajo pendiente llamando a la API `DescribeJobExecution` con `jobId` como `$next`.
- Obtener e iniciar la ejecución de trabajo pendiente siguiente llamando a la API `StartNextPendingJobExecution`.

Uso de HTTP Signature Version 4

Los dispositivos pueden comunicarse con AWS IoT Jobs mediante la firma HTTP, versión 4, en el puerto 443. Este es el método utilizado por la CLI AWS SDKs y. Para obtener más información sobre estas herramientas, consulte la Referencia de AWS CLI comandos: iot-jobs-data o AWSSDKs y Herramientas y consulte la IotJobsDataPlane sección correspondiente al idioma que prefiera.

Con este método de comunicación, el dispositivo utiliza las credenciales de IAM para autenticarse con AWS IoT Jobs.

Los siguientes comandos están disponibles a través de este método:

- DescribeJobExecution

```
aws iot-jobs-data describe-job-execution ...
```

- GetPendingJobExecutions

```
aws iot-jobs-data get-pending-job-executions ...
```

- StartNextPendingJobExecution

```
aws iot-jobs-data start-next-pending-job-execution ...
```

- UpdateJobExecution

```
aws iot-jobs-data update-job-execution ...
```

Uso de HTTP TLS

Los dispositivos pueden comunicarse con AWS IoT Jobs mediante HTTP TLS en el puerto 8443 mediante un cliente de software de terceros compatible con este protocolo.

Con este método, el dispositivo utiliza la autenticación basada en certificados X.509 (por ejemplo, utilizando su propio certificado y su propia clave privada).

Los siguientes comandos están disponibles a través de este método:

- DescribeJobExecution
- GetPendingJobExecutions
- StartNextPendingJobExecution
- UpdateJobExecution

Programación de dispositivos para trabajar con trabajos

En los ejemplos de esta sección se utiliza MQTT para ilustrar cómo funciona un dispositivo con el servicio Jobs de AWS IoT. También puede usar los comandos de la API o la CLI correspondientes. Para estos ejemplos, supondremos que un dispositivo llamado `MyThing` se suscribe a los siguientes temas de MQTT:

- `$aws/things/MyThing/jobs/notify` (o `$aws/things/MyThing/jobs/notify-next`)
- `$aws/things/MyThing/jobs/get/accepted`
- `$aws/things/MyThing/jobs/get/rejected`
- `$aws/things/MyThing/jobs/jobId/get/accepted`
- `$aws/things/MyThing/jobs/jobId/get/rejected`

Si utilizas la firma de códigoAWS IoT, el código del dispositivo debe verificar la firma del archivo de códigos. La firma se encuentra en el documento de trabajo, en la propiedad `codesign`. Para obtener más información sobre cómo verificar una firma en un archivo de código, consulte el ejemplo de agente de dispositivo.

Temas

- Flujo de trabajo del dispositivo
- Flujo de trabajo
- Notificaciones de trabajos

Flujo de trabajo del dispositivo

Un dispositivo puede gestionar los trabajos que ejecuta mediante una de las siguientes formas.

- Obtención del siguiente trabajo

  1. Cuando un dispositivo está online, debe suscribirse al tema `notify-next` del dispositivo.
  2. Llame a la API de MQTT de DescribeJobExecution con jobId `$next` para obtener el siguiente trabajo, su documento de trabajo y otros detalles, incluido el estado guardado en `statusDetails`. Si el documento de trabajo tiene una forma en archivo de código, debe verificar la firma antes de seguir procesando la solicitud del trabajo.

3. Llame a la API de MQTT de UpdateJobExecution para actualizar el estado del trabajo. También puede combinar este paso y el anterior en una llamada. El dispositivo puede llamar a StartNextPendingJobExecution.
4. Si lo prefiere, puede añadir un temporizador de pasos estableciendo un valor para `stepTimeoutInMinutes` si llama a UpdateJobExecution o StartNextPendingJobExecution.
5. Realice las acciones especificadas por el documento de trabajo con la API de MQTT de UpdateJobExecution para informar del progreso del trabajo.
6. Siga monitorizando la ejecución del trabajo llamando a la API de MQTT DescribeJobExecution con este jobId. Si se elimina la ejecución del trabajo, DescribeJobExecution devuelve una `ResourceNotFoundException`.

   Si la ejecución de trabajo se cancela o se elimina al mismo tiempo que el dispositivo está ejecutando el trabajo, el dispositivo debería ser capaz de recuperarse a un estado válido.
7. Llame a la API de MQTT de UpdateJobExecution cuando termine con el trabajo para actualizar el estado del trabajo e informar del éxito o error.
8. El siguiente trabajo disponible para su ejecución (si lo hubiera) cambiará, puesto que el estado de la ejecución del trabajo ha cambiado a estado final. Se notifica al dispositivo que la siguiente ejecución de trabajo pendiente ha cambiado. En este momento, el dispositivo debe continuar como se describe en el paso 2.

Si el dispositivo permanece en línea, seguirá recibiendo notificaciones de la ejecución del próximo trabajo pendiente. Esto incluye sus datos de ejecución de trabajos, cuando completa un trabajo o se agrega una nueva ejecución de trabajo pendiente. Cuando se produzca esto, el dispositivo seguirá como se describe en el paso 2.

- Selección de los trabajos disponibles

1. Cuando un dispositivo está online, debe suscribirse al tema `notify` del objeto.
2. Llame a la API de MQTT de GetPendingJobExecutions para obtener una lista de ejecuciones de trabajo pendientes.
3. Si la lista contiene una o varias ejecuciones de trabajo, elija una.
4. Llame a la API de MQTT DescribeJobExecution para obtener el documento de trabajo y otros detalles, incluido cualquier estado guardado en `statusDetails`.
5. Llame a la API de MQTT de UpdateJobExecution para actualizar el estado del trabajo. Si el campo `includeJobDocument` está establecido en `true` en este comando, el dispositivo puede omitir el paso anterior y recuperar el documento de trabajo en este momento.

6. Si lo prefiere, puede añadir un temporizador de pasos estableciendo un valor para `stepTimeoutInMinutes` si llama a UpdateJobExecution.
7. Realice las acciones especificadas por el documento de trabajo con la API de MQTT de UpdateJobExecution para informar del progreso del trabajo.
8. Siga monitorizando la ejecución del trabajo llamando a la API de MQTT DescribeJobExecution con este jobId. Si la ejecución de trabajo se cancela o se elimina al mismo tiempo que el dispositivo está ejecutando el trabajo, el dispositivo debería ser capaz de recuperarse a un estado válido.
9. Llame a la API de MQTT de UpdateJobExecution cuando termine con el trabajo para actualizar el estado del trabajo e informar del éxito o error.

Si el dispositivo continúa online, se le notificarán todas las ejecuciones de trabajo pendientes cuando una nueva ejecución de trabajo pendiente esté disponible. Cuando se produzca esto, el dispositivo podrá seguir como se describe en el paso 2.

Si el dispositivo no puede ejecutar el trabajo, debe llamar a la API de MQTT de UpdateJobExecution para actualizar el estado del trabajo a `REJECTED`.

Flujo de trabajo

A continuación se muestran los diferentes pasos del flujo de trabajo, desde el inicio de un nuevo trabajo hasta el informe del estado de finalización de la ejecución de un trabajo.

Inicio de un nuevo trabajo

Cuando se crea un trabajo nuevo, AWS IoT Jobs publica un mensaje sobre el `$aws/things/`*`thing-name`*`/jobs/notify` tema para cada dispositivo de destino.

El mensaje contiene la siguiente información:

```
{
    "timestamp":1476214217017,
    "jobs":{
        "QUEUED":[{
            "jobId":"0001",
            "queuedAt":1476214216981,
            "lastUpdatedAt":1476214216981,
            "versionNumber" : 1
```

```
        }]
    }
}
```

El dispositivo recibe este mensaje en el tema `'$aws/things/thingName/jobs/notify'` cuando la ejecución del trabajo está en la cola.

> **Note**
>
> En el caso de los trabajos con la `SchedulingConfig` opcional, mantendrán el estado inicial `SCHEDULED`. Cuando el trabajo alcance el `startTime` seleccionado, ocurrirá lo siguiente:
>
> - El estado del trabajo se actualizará a `IN_PROGRESS`.
> - El trabajo comenzará el despliegue del documento de trabajo en todos los dispositivos del grupo de destino.

Obtención de información de trabajo

Para obtener más información sobre la ejecución de un trabajo, el dispositivo llama a la API de MQTT DescribeJobExecution con el campo `includeJobDocument` establecido en `true` (el valor predeterminado).

Si la solicitud se realiza correctamente, el servicio AWS IoT Jobs publica un mensaje sobre el `$aws/things/MyThing/jobs/0023/get/accepted` tema:

```
{
    "clientToken" : "client-001",
    "timestamp" : 1489097434407,
    "execution" : {
        "approximateSecondsBeforeTimedOut": number,
        "jobId" : "023",
        "status" : "QUEUED",
        "queuedAt" : 1489097374841,
        "lastUpdatedAt" : 1489097374841,
        "versionNumber" : 1,
        "jobDocument" : {
            < contents of job document >
        }
    }
```

```
}
```

Si se produce un error en la solicitud, el servicio AWS IoT Jobs publica un mensaje sobre el `$aws/things/MyThing/jobs/0023/get/rejected` tema.

El dispositivo ahora tiene el documento de trabajo que puede usar para realizar las operaciones remotas para el trabajo. Si el documento de trabajo contiene una URL prefirmada de Amazon S3, el dispositivo puede usar esa URL para descargar los archivos necesarios para el trabajo.

Informe del estado de la ejecución de trabajo

A medida que el dispositivo ejecute el trabajo, puede llamar a la API de MQTT de UpdateJobExecution para actualizar el estado de la ejecución de trabajo.

Por ejemplo, un dispositivo puede actualizar el estado de ejecución de trabajo a `IN_PROGRESS` mediante la publicación del siguiente mensaje en el tema `$aws/things/MyThing/jobs/0023/update`:

```
{
    "status":"IN_PROGRESS",
    "statusDetails": {
        "progress":"50%"
    },
    "expectedVersion":"1",
    "clientToken":"client001"
}
```

Jobs responde mediante la publicación de un mensaje en el tema `$aws/things/MyThing/jobs/0023/update/accepted` o `$aws/things/MyThing/jobs/0023/update/rejected`:

```
{
    "clientToken":"client001",
    "timestamp":1476289222841
}
```

El dispositivo puede combinar las dos solicitudes anteriores llamando a StartNextPendingJobExecution. Esto obtiene e inicia la ejecución del trabajo pendiente y permite al dispositivo actualizar el estado de ejecución del trabajo. Esta solicitud también devuelve el documento de trabajo cuando hay una ejecución de trabajo pendiente.

Si el trabajo contiene un TimeoutConfig, el temporizador en curso comienza a ejecutarse. También puede configurar un temporizador por pasos para la ejecución de un trabajo estableciendo un valor para `stepTimeoutInMinutes` cuando llame UpdateJobExecution. El temporizador de pasos se aplica únicamente a la ejecución del trabajo que actualice. Puede establecer un nuevo valor para este temporizador cada vez que actualice la ejecución de un trabajo. También puedes crear un temporizador por pasos cuando llames StartNextPendingJobExecution. Si la ejecución del trabajo permanece en estado `IN_PROGRESS` durante un periodo superior a este intervalo del temporizador de pasos, generará un error y cambiará al estado terminal `TIMED_OUT`. El temporizador de pasos no tiene ningún efecto en el temporizador en curso que establezca al crear un trabajo.

El campo `status` se puede definir en `IN_PROGRESS`, `SUCCEEDED` o `FAILED`. No puede actualizar el estado de una ejecución de trabajo que ya está en estado terminal.

Informe de ejecución completada

Cuando el dispositivo ha acabado de ejecutar el trabajo, llama a la API de MQTT de UpdateJobExecution. Si el trabajo se realizó correctamente, establezca `status` en `SUCCEEDED` y, en la carga del mensaje, en `statusDetails`, añada otra información acerca del trabajo, como pares nombre-valor. Los temporizadores en curso y de pasos finalizan cuando se completa la ejecución del trabajo.

Por ejemplo:

```
{
    "status":"SUCCEEDED",
    "statusDetails": {
        "progress":"100%"
    },
    "expectedVersion":"2",
    "clientToken":"client-001"
}
```

Si el trabajo no se realizó correctamente, establezca `status` en `FAILED` y, en `statusDetails`, añada información acerca del error que se ha producido:

```
{
    "status":"FAILED",
    "statusDetails": {
        "errorCode":"101",
        "errorMsg":"Unable to install update"
```

```
    },
    "expectedVersion":"2",
    "clientToken":"client-001"
}
```

> **Note**
>
> El atributo `statusDetails` puede contener cualquier número de pares nombre-valor.

Cuando el servicio AWS IoT Jobs recibe esta actualización, publica un mensaje sobre el `$aws/things/MyThing/jobs/notify` tema para indicar que se ha completado la ejecución del trabajo:

```
{
    "timestamp":1476290692776,
    "jobs":{}
}
```

Trabajos adicionales

Si hay otras ejecuciones de trabajo pendientes para el dispositivo, se incluyen en el mensaje publicado en `$aws/things/MyThing/jobs/notify`.

Por ejemplo:

```
{
    "timestamp":1476290692776,
    "jobs":{
        "QUEUED":[{
            "jobId":"0002",
            "queuedAt":1476290646230,
            "lastUpdatedAt":1476290646230
        }],
        "IN_PROGRESS":[{
            "jobId":"0003",
            "queuedAt":1476290646230,
            "lastUpdatedAt":1476290646230
        }]
    }
}
```

Notificaciones de trabajos

El servicio AWS IoT Jobs publica mensajes MQTT en temas reservados cuando hay trabajos pendientes o cuando cambia la primera ejecución de un trabajo de la lista. Los dispositivos pueden realizar un seguimiento de los trabajos pendientes suscribiéndose a estos temas.

Tipos de notificaciones de trabajo

Las notificaciones de trabajo se publican en temas MQTT como cargas JSON. Existen dos tipos de notificaciones:

ListNotification

Una `ListNotification` contiene una lista de no más de 15 ejecuciones de trabajos pendientes. Se almacenan por estado (ejecuciones de trabajo `IN_PROGRESS` antes que las ejecuciones de trabajo `QUEUED`) y, a continuación, por las veces que se incluyeron en la cola.

Una `ListNotification` se publica siempre que se cumple uno de los siguientes criterios.

- Se pone en cola una nueva ejecución de trabajo o se cambia a un estado no terminal (`IN_PROGRESS` o `QUEUED`).
- Una ejecución de trabajo antigua cambia a un estado terminal (`FAILED`, `SUCCEEDED`, `CANCELED`, `TIMED_OUT`, `REJECTED` o `REMOVED`).

Para obtener más información sobre los límites con y sin la configuración de programación, consulte Límites de ejecución de trabajos.

NextNotification

- `NextNotification` contiene información resumida sobre la ejecución del trabajo siguiente en la cola.

  Se publica una `NextNotification` siempre que cambia la ejecución del primer trabajo de la lista.

  - Se añade a la lista una nueva ejecución del trabajo como `QUEUED` y se coloca el primero en la lista.
  - El estado de la ejecución de un trabajo existente que no estaba en el primer lugar en la lista cambia de `QUEUED` a `IN_PROGRESS` y pasa a colocarse en primer lugar en la lista. (Esto sucede

cuando no hay otras ejecuciones de trabajos de `IN_PROGRESS` en la lista o cuando la ejecución del trabajo cuyo estado cambia de `QUEUED` a `IN_PROGRESS` estaba en la cola antes que cualquier otra ejecución de trabajo de `IN_PROGRESS` de la lista.)

- El estado de la ejecución del trabajo existente que se encuentra en el primer lugar de la lista cambia a un estado terminal y se quita de la lista.

Para obtener más información acerca de cómo publicar en temas de MQTT y suscribirse a ellos, consulte the section called “Protocolos de comunicación de dispositivos”.

> **Note**
>
> Las notificaciones no están disponibles cuando se usa HTTP Signature Version 4 o HTTP TLS para comunicarse con trabajos.

Trabajo pendiente

El servicio AWS IoT Jobs publica un mensaje sobre un tema de MQTT cuando se añade o elimina un trabajo de la lista de ejecuciones pendientes o cuando cambia la primera ejecución de un trabajo de la lista:

- `$aws/things/thingName/jobs/notify`
- `$aws/things/thingName/jobs/notify-next`

Los mensajes contienen las siguientes cargas de ejemplo:

`$aws/things/thingName/jobs/notify`:

```
{
  "timestamp" : 10011,
  "jobs" : {
    "IN_PROGRESS" : [ {
      "jobId" : "other-job",
      "queuedAt" : 10003,
      "lastUpdatedAt" : 10009,
      "executionNumber" : 1,
      "versionNumber" : 1
    } ],
    "QUEUED" : [ {
```

```
      "jobId" : "this-job",
      "queuedAt" : 10011,
      "lastUpdatedAt" : 10011,
      "executionNumber" : 1,
      "versionNumber" : 0
    } ]
  }
}
```

Si la ejecución del trabajo llamado `this-job` se originó en un trabajo con la configuración de programación opcional seleccionada y el despliegue del documento de trabajo está programado para realizarse durante un periodo de mantenimiento, solo aparecerá durante un periodo de mantenimiento periódico. Fuera del periodo de mantenimiento, el trabajo llamado `this-job` quedará excluido de la lista de ejecuciones de trabajos pendientes, como se muestra en el siguiente ejemplo.

```
{
  "timestamp" : 10011,
  "jobs" : {
    "IN_PROGRESS" : [ {
      "jobId" : "other-job",
      "queuedAt" : 10003,
      "lastUpdatedAt" : 10009,
      "executionNumber" : 1,
      "versionNumber" : 1
    } ],
    "QUEUED" : []
  }
}
```

`$aws/things/`*`thingName`*`/jobs/notify-next`:

```
{
  "timestamp" : 10011,
  "execution" : {
    "jobId" : "other-job",
    "status" : "IN_PROGRESS",
    "queuedAt" : 10009,
    "lastUpdatedAt" : 10009,
    "versionNumber" : 1,
    "executionNumber" : 1,
    "jobDocument" : {"c":"d"}
  }
```

```
}
```

Si la ejecución del trabajo llamado `other-job` se originó en un trabajo con la configuración de programación opcional seleccionada y el despliegue del documento de trabajo está programado para realizarse durante un periodo de mantenimiento, solo aparecerá durante un periodo de mantenimiento periódico. Fuera de un periodo de mantenimiento, el trabajo llamado `other-job` no figurará como la siguiente ejecución de trabajo, como se muestra en el siguiente ejemplo.

```
{} //No other pending jobs
```

```
{
  "timestamp" : 10011,
  "execution" : {
      "jobId" : "this-job",
      "queuedAt" : 10011,
      "lastUpdatedAt" : 10011,
      "executionNumber" : 1,
      "versionNumber" : 0,
      "jobDocument" : {"a":"b"}
  }
} // "this-job" is pending next to "other-job"
```

Los valores del estado de ejecución de trabajo posibles son `QUEUED`, `IN_PROGRESS`, `FAILED`, `SUCCEEDED`, `CANCELED`, `TIMED_OUT`, `REJECTED` y `REMOVED`.

La siguiente serie de ejemplos muestra las notificaciones publicadas en cada tema a medida que se crean las ejecuciones de trabajo y cambian de un estado a otro.

En primer lugar se crea un trabajo llamado `job1`. La notificación se publica en el tema `jobs/notify`:

```
{
  "timestamp": 1517016948,
  "jobs": {
    "QUEUED": [
      {
        "jobId": "job1",
        "queuedAt": 1517016947,
        "lastUpdatedAt": 1517016947,
        "executionNumber": 1,
```

```
        "versionNumber": 1
      }
    ]
  }
}
```

La notificación se publica en el tema `jobs/notify-next`:

```
{
  "timestamp": 1517016948,
  "execution": {
    "jobId": "job1",
    "status": "QUEUED",
    "queuedAt": 1517016947,
    "lastUpdatedAt": 1517016947,
    "versionNumber": 1,
    "executionNumber": 1,
    "jobDocument": {
      "operation": "test"
    }
  }
}
```

Cuando se crea otro trabajo (`job2`), esta notificación se publica en el tema `jobs/notify`:

```
{
  "timestamp": 1517017192,
  "jobs": {
    "QUEUED": [
      {
        "jobId": "job1",
        "queuedAt": 1517016947,
        "lastUpdatedAt": 1517016947,
        "executionNumber": 1,
        "versionNumber": 1
      },
      {
        "jobId": "job2",
        "queuedAt": 1517017191,
        "lastUpdatedAt": 1517017191,
        "executionNumber": 1,
        "versionNumber": 1
      }
```

```
    ]
  }
}
```

No se publica una notificación en el tema `jobs/notify-next` porque el siguiente trabajo de la cola (`job1`) no ha cambiado. Cuando `job1` comienza a ejecutarse, su estado cambia a `IN_PROGRESS`. No se publican notificaciones ya que la lista de trabajos y el siguiente trabajo en la cola no han cambiado.

Cuando se añade un tercer trabajo (`job3`), esta notificación se publica en el tema `jobs/notify`:

```
{
  "timestamp": 1517017906,
  "jobs": {
    "IN_PROGRESS": [
      {
        "jobId": "job1",
        "queuedAt": 1517016947,
        "lastUpdatedAt": 1517017472,
        "startedAt": 1517017472,
        "executionNumber": 1,
        "versionNumber": 2
      }
    ],
    "QUEUED": [
      {
        "jobId": "job2",
        "queuedAt": 1517017191,
        "lastUpdatedAt": 1517017191,
        "executionNumber": 1,
        "versionNumber": 1
      },
      {
        "jobId": "job3",
        "queuedAt": 1517017905,
        "lastUpdatedAt": 1517017905,
        "executionNumber": 1,
        "versionNumber": 1
      }
    ]
  }
}
```

No se publica una notificación en el tema `jobs/notify-next` porque el siguiente trabajo en la cola sigue siendo `job1`.

Cuando `job1` finaliza, su estado cambia a `SUCCEEDED` y se publica esta notificación en el tema `jobs/notify`:

```
{
  "timestamp": 1517186269,
  "jobs": {
    "QUEUED": [
      {
        "jobId": "job2",
        "queuedAt": 1517017191,
        "lastUpdatedAt": 1517017191,
        "executionNumber": 1,
        "versionNumber": 1
      },
      {
        "jobId": "job3",
        "queuedAt": 1517017905,
        "lastUpdatedAt": 1517017905,
        "executionNumber": 1,
        "versionNumber": 1
      }
    ]
  }
}
```

En este punto, `job1` se ha eliminado de la cola y el siguiente trabajo que ejecutar es `job2`. La notificación se publica en el tema `jobs/notify-next`:

```
{
  "timestamp": 1517186269,
  "execution": {
    "jobId": "job2",
    "status": "QUEUED",
    "queuedAt": 1517017191,
    "lastUpdatedAt": 1517017191,
    "versionNumber": 1,
    "executionNumber": 1,
    "jobDocument": {
      "operation": "test"
    }
```

```
  }
}
```

Si `job3` tiene que empezar a ejecutarse antes que `job2` (lo cual no se recomienda), el estado de `job3` puede cambiarse a `IN_PROGRESS`. Si esto sucede, `job2` deja de ser el siguiente en la cola y se publica esta notificación en el tema `jobs/notify-next`:

```
{
  "timestamp": 1517186779,
  "execution": {
    "jobId": "job3",
    "status": "IN_PROGRESS",
    "queuedAt": 1517017905,
    "startedAt": 1517186779,
    "lastUpdatedAt": 1517186779,
    "versionNumber": 2,
    "executionNumber": 1,
    "jobDocument": {
      "operation": "test"
    }
  }
}
```

No se publica ninguna notificación en el tema `jobs/notify`, dado que no se ha añadido o eliminado ningún trabajo.

Si el dispositivo rechaza `job2` y actualiza su estado a `REJECTED`, se publica esta notificación en el tema `jobs/notify`:

```
{
  "timestamp": 1517189392,
  "jobs": {
    "IN_PROGRESS": [
      {
        "jobId": "job3",
        "queuedAt": 1517017905,
        "lastUpdatedAt": 1517186779,
        "startedAt": 1517186779,
        "executionNumber": 1,
        "versionNumber": 2
      }
    ]
```

```
  }
}
```

Si `job3` (que aún está en curso) se elimina de forma forzada, esta notificación se publica en el tema `jobs/notify`:

```
{
  "timestamp": 1517189551,
  "jobs": {}
}
```

En este momento, la cola está vacía. La notificación se publica en el tema `jobs/notify-next`:

```
{
  "timestamp": 1517189551
}
```

AWS IoT operaciones de API de trabajos

AWS IoT La API de trabajos se puede utilizar para cualquiera de las siguientes categorías:

- Tareas administrativas, como la gestión y el control de los trabajos. Este es el plano de control.
- Los dispositivos que realizan esos trabajos. Este es el plano de datos, que permite enviar y recibir datos.

La administración y control de los trabajos utilizan una API de protocolo HTTPS. Los dispositivos pueden usar una API de protocolo HTTPS o MQTT. La API del plano de control se ha diseñado para un volumen reducido de llamadas normalmente cuando se crean trabajos y se hace un seguimiento de ellos. Normalmente, abre una conexión para una solicitud única y, a continuación, cierra la conexión después de que se reciba la respuesta. Las API HTTPS y MQTT del plano de datos permiten un sondeo prolongado. Estas operaciones de API se han diseñado para cantidades grandes de tráfico que pueden escalar millones de dispositivos.

Cada API HTTPS de AWS IoT Jobs tiene un comando correspondiente que te permite llamar a la API desde AWS Command Line Interface (AWS CLI). Los comandos están en minúsculas, con guiones entre las palabras que conforman el nombre de la API. Por ejemplo, puede invocar la API `CreateJob` en la CLI escribiendo lo siguiente:

```
aws iot create-job ...
```

Si ocurre un error durante una operación, recibirá una respuesta de error que contiene información acerca de este.

ErrorResponse

Contiene información acerca de un error que se produjo durante una operación del servicio Jobs de AWS IoT .

En el siguiente ejemplo se muestra la sintaxis de esta operación:

```
{
    "code": "ErrorCode",
    "message": "string",
    "clientToken": "string",
    "timestamp": timestamp,
    "executionState": JobExecutionState
}
```

A continuación se muestra una descripción de esta `ErrorResponse`:

`code`

ErrorCode se puede configurar de la siguiente manera:

InvalidTopic

La solicitud se envió a un tema del espacio de nombres de AWS IoT Jobs que no se corresponde con ninguna operación de la API.

InvalidJson

El contenido de la solicitud podría no interpretarse como un formato JSON con codificación UTF-8 válida.

InvalidRequest

El contenido de la solicitud no es válido. Por ejemplo, se devuelve este código cuando una solicitud `UpdateJobExecution` contiene detalles de estado no válido. El mensaje contiene detalles acerca del error.

InvalidStateTransition

Una actualización intentó cambiar la ejecución del trabajo a un estado que no es válido debido al estado actual de dicha ejecución. Por ejemplo, un intento de cambiar una solicitud en el estado SUCCEEDED al estado IN_PROGRESS. En este caso, el cuerpo del mensaje de error también contiene el campo `executionState`.

ResourceNotFound

La `JobExecution` especificada por el tema de la solicitud no existe.

VersionMismatch

La versión esperada especificada en la solicitud no coincide con la versión de la ejecución del trabajo en el servicio AWS IoT Jobs. En este caso, el cuerpo del mensaje de error también contiene el campo `executionState`.

InternalError

Se ha producido un error interno al procesar la solicitud.

RequestThrottled

La solicitud se ha limitado.

TerminalStateReached

Se produce cuando un comando para describir un trabajo se realiza en un trabajo que está en un estado terminal.

`message`

Una cadena de mensajes de error.

`clientToken`

Una cadena arbitraria utilizada para correlacionar una solicitud con su respuesta.

`timestamp`

El tiempo, en segundos, desde la fecha de inicio.

`executionState`

Un objeto `JobExecutionState`. Este campo se incluye solo cuando el campo `code` tiene el valor `InvalidStateTransition` o `VersionMismatch`. Esto hace que no sea necesario en esos casos realizar una solicitud `DescribeJobExecution` independiente para obtener los datos de estado de ejecución de trabajo actuales.

A continuación, se enumeran las operaciones y los tipos de datos de la API de trabajos.

- API de administración y control de trabajos y tipos de datos
- Operaciones y tipos de datos de las API MQTT y HTTPS de dispositivos de Jobs

API de administración y control de trabajos y tipos de datos

Los siguientes comandos están disponibles para la administración y control de trabajos en la CLI a través del protocolo HTTPS.

- Tipos de datos de administración y control de trabajo
- Operaciones de la API de administración y control del trabajo

Para determinar el *endpoint-url* parámetro de los comandos CLI, ejecute este comando.

```
aws iot describe-endpoint --endpoint-type=iot:Jobs
```

Este comando devuelve la siguiente salida.

```
{
"endpointAddress": "account-specific-prefix.jobs.iot.aws-region.amazonaws.com"
}
```

> **Note**
>
> El punto de conexión de Jobs no admite la ALPN `x-amzn-http-ca`.
> Si utiliza puntos finales de doble pila (IPv6 y IPv6), utilice el `iot:Data-ATS` punto final. El `iot:Jobs` punto final solo es compatible. IPv4

Tipos de datos de administración y control de trabajo

Las aplicaciones de administración y control utilizan los siguientes tipos de datos para comunicarse con AWS IoT Jobs.

Trabajo

El objeto `Job` contiene detalles acerca de un trabajo. En el siguiente ejemplo se muestra la sintaxis:

```
{
    "jobArn": "string",
    "jobId": "string",
    "status": "IN_PROGRESS|CANCELED|SUCCEEDED",
    "forceCanceled": boolean,
    "targetSelection": "CONTINUOUS|SNAPSHOT",
    "comment": "string",
    "targets": ["string"],
    "description": "string",
    "createdAt": timestamp,
    "lastUpdatedAt": timestamp,
    "completedAt": timestamp,
    "jobProcessDetails": {
        "processingTargets": ["string"],
        "numberOfCanceledThings": long,
        "numberOfSucceededThings": long,
        "numberOfFailedThings": long,
        "numberOfRejectedThings": long,
        "numberOfQueuedThings": long,
        "numberOfInProgressThings": long,
        "numberOfRemovedThings": long,
        "numberOfTimedOutThings": long
    },
    "presignedUrlConfig": {
        "expiresInSec": number,
        "roleArn": "string"
    },
    "jobExecutionsRolloutConfig": {
        "exponentialRate": {
           "baseRatePerMinute": integer,
           "incrementFactor": integer,
           "rateIncreaseCriteria": {
              "numberOfNotifiedThings": integer, // Set one or the other
              "numberOfSucceededThings": integer // of these two values.
           },
           "maximumPerMinute": integer
      }
    },
    "abortConfig": {
       "criteriaList": [
          {
             "action": "string",
             "failureType": "string",
```

```
            "minNumberOfExecutedThings": integer,
            "thresholdPercentage": integer
         }
      ]
   },
   "SchedulingConfig": {
     "startTime": string
     "endTime": string
     "timeZone": string


     "endTimeBehavior": string

  },
   "timeoutConfig": {
       "inProgressTimeoutInMinutes": long
   }
}
```

Para obtener más información, consulte `Job` o `job`.

JobSummary

El objeto `JobSummary` contiene un resumen de trabajos. En el siguiente ejemplo se muestra la sintaxis:

```
{
    "jobArn": "string",
    "jobId": "string",
    "status": "IN_PROGRESS|CANCELED|SUCCEEDED|SCHEDULED",
    "targetSelection": "CONTINUOUS|SNAPSHOT",
    "thingGroupId": "string",
    "createdAt": timestamp,
    "lastUpdatedAt": timestamp,
    "completedAt": timestamp
}
```

Para obtener más información, consulte `JobSummary` o `job-summary`.

JobExecution

El objeto `JobExecution` representa la ejecución de un trabajo en un dispositivo. En el siguiente ejemplo se muestra la sintaxis:

> **Note**
>
> Cuando se utilizan las operaciones de la API del plano de control, el tipo de datos `JobExecution` no contiene ningún campo `JobDocument`. Para obtener esta información, puede utilizar la operación `GetJobDocument` de la API o el comando `get-job-document` de la CLI.

```
{
    "approximateSecondsBeforeTimedOut": 50,
    "executionNumber": 1234567890,
    "forceCanceled": true|false,
    "jobId": "string",
    "lastUpdatedAt": timestamp,
    "queuedAt": timestamp,
    "startedAt": timestamp,
    "status": "QUEUED|IN_PROGRESS|FAILED|SUCCEEDED|CANCELED|TIMED_OUT|REJECTED|
REMOVED",
    "forceCanceled": boolean,
    "statusDetails": {
        "detailsMap": {
            "string": "string" ...
        },
        "status": "string"
    },
    "thingArn": "string",
    "versionNumber": 123
}
```

Para obtener más información, consulte `JobExecution` o `job-execution`.

JobExecutionSummary

El objeto `JobExecutionSummary` contiene información del resumen de ejecución de trabajo. En el siguiente ejemplo se muestra la sintaxis:

```
{
    "executionNumber": 1234567890,
    "queuedAt": timestamp,
    "lastUpdatedAt": timestamp,
    "startedAt": timestamp,
```

```
    "status": "QUEUED|IN_PROGRESS|FAILED|SUCCEEDED|CANCELED|TIMED_OUT|REJECTED|REMOVED"
}
```

Para obtener más información, consulte JobExecutionSummary o job-execution-summary.

JobExecutionSummaryForJob

El objeto JobExecutionSummaryForJob contiene un resumen de información acerca de las ejecuciones de trabajo para un trabajo específico. En el siguiente ejemplo se muestra la sintaxis:

```
{
    "executionSummaries": [
        {
            "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/MyThing",
            "jobExecutionSummary": {
                "status": "IN_PROGRESS",
                "lastUpdatedAt": 1549395301.389,
                "queuedAt": 1541526002.609,
                "executionNumber": 1
            }
        },
        ...
    ]
}
```

Para obtener más información, consulte JobExecutionSummaryForJob o job-execution-summary-for-job.

JobExecutionSummaryForThing

El JobExecutionSummaryForThing objeto contiene un resumen de la información sobre la ejecución de un trabajo en un elemento específico. FThe El siguiente ejemplo muestra la sintaxis:

```
{
    "executionSummaries": [
        {
            "jobExecutionSummary": {
                "status": "IN_PROGRESS",
                "lastUpdatedAt": 1549395301.389,
                "queuedAt": 1541526002.609,
                "executionNumber": 1
            },
            "jobId": "MyThingJob"
```

```
        },
        ...
    ]
}
```

Para obtener más información, consulte JobExecutionSummaryForThing o job-execution-summary-for-thing.

Operaciones de la API de administración y control del trabajo

Utilice las siguientes operaciones de la API o comandos de la CLI:

AssociateTargetsWithJob

Asocia un grupo a un trabajo continuo. Deben cumplirse los siguientes criterios:

- El trabajo debe haberse creado con el campo `targetSelection` establecido en `CONTINUOUS`.
- El estado del trabajo debe ser actualmente `IN_PROGRESS`.
- El número total de destinos asociados con un trabajo no debe ser superior a 100.

HTTPS request

```
POST /jobs/jobId/targets

{
"targets": [ "string" ],
"comment": "string"
}
```

Para obtener más información, consulte AssociateTargetsWithJob.

CLI syntax

```
aws iot  associate-targets-with-job \
--targets <value> \
--job-id <value> \
[--comment <value>]  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

`cli-input-json` formato:

```
{
"targets": [
"string"
],
"jobId": "string",
"comment": "string"
}
```

Para obtener más información, consulte `associate-targets-with-job`.

CancelJob

Cancela un trabajo.

HTTPS request

```
PUT /jobs/jobId/cancel

{
"force": boolean,
"comment": "string",
"reasonCode": "string"
}
```

Para obtener más información, consulte `CancelJob`.

CLI syntax

```
aws iot cancel-job \
    --job-id <value> \
    [--force <value>]  \
    [--comment <value>]  \
    [--reasonCode <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` formato:

```
{
    "jobId": "string",
    "force": boolean,
    "comment": "string"
```

```
}
```

Para obtener más información, consulte `cancel-job`.

CancelJobExecution

Cancela la ejecución de un trabajo en un dispositivo.

HTTPS request

```
PUT /things/thingName/jobs/jobId/cancel

{
"force": boolean,
"expectedVersion": "string",
"statusDetails": {
    "string": "string"
    ...
}
}
```

Para obtener más información, consulte `CancelJobExecution`.

CLI syntax

```
aws iot cancel-job-execution \
--job-id <value> \
--thing-name <value> \
[--force | --no-force] \
[--expected-version <value>] \
[--status-details <value>]  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

`cli-input-json` formato:

```
{
"jobId": "string",
"thingName": "string",
"force": boolean,
"expectedVersion": long,
"statusDetails": {
```

```
"string": "string"
}
}
```

Para obtener más información, consulte `cancel-job-execution`.

CreateJob

Crea un trabajo. Puede proporcionar el documento de trabajo como enlace a un archivo en un bucket de Amazon S3 (parámetro `documentSource`) o en el cuerpo de la solicitud (parámetro `document`).

Un trabajo puede ser continuo si se establece el parámetro opcional `targetSelection` en `CONTINUOUS` (el predeterminado es `SNAPSHOT`). Un trabajo continuo se puede usar para incorporar o actualizar los dispositivos a medida que se agregan a un grupo, ya que continúa ejecutándose y se lanza cuando se agregan nuevos objetos. Esto puede ocurrir incluso después de que los objetos del grupo en el momento en que se creó el trabajo lo hayan completado.

Un trabajo puede tener una opción TimeoutConfig, que establece el valor del temporizador en curso. El temporizador en curso no se puede actualizar y se aplica a todas las ejecuciones del trabajo.

Se realizan las siguientes validaciones en los argumentos para la API `CreateJob`:

- El `targets` argumento debe ser una lista de cosas o grupos ARNs de cosas válidos. Todas las cosas y grupos de cosas deben estar en su Cuenta de AWS.
- El argumento `documentSource` debe ser una URL de Amazon S3 válida para un documento de trabajo. Amazon S3 URLs tienen el formato:`https://s3.amazonaws.com/`*`bucketName`*`/`*`objectName`*.
- El documento almacenado en la URL especificada por el argumento `documentSource` debe ser un documento JSON con codificación UTF-8.
- El tamaño del documento de trabajo está limitado a 32 KB debido al límite del tamaño de un mensaje MQTT (128 KB) y el cifrado.
- `jobId`Debe ser único en tu Cuenta de AWS.

HTTPS request

```
PUT /jobs/jobId

{
```

```
"targets": [ "string" ],
"document": "string",
"documentSource": "string",
"description": "string",
"jobTemplateArn": "string",
"presignedUrlConfigData": {
    "roleArn": "string",
    "expiresInSec": "integer"
},
"targetSelection": "CONTINUOUS|SNAPSHOT",
"jobExecutionsRolloutConfig": {
    "exponentialRate": {
       "baseRatePerMinute": integer,
       "incrementFactor": integer,
       "rateIncreaseCriteria": {
          "numberOfNotifiedThings": integer, // Set one or the other
          "numberOfSucceededThings": integer // of these two values.
       },
       "maximumPerMinute": integer
  }
},
"abortConfig": {
   "criteriaList": [
      {
         "action": "string",
         "failureType": "string",
         "minNumberOfExecutedThings": integer,
         "thresholdPercentage": integer
      }
   ]
},
"SchedulingConfig": {
    "startTime": string
    "endTime": string
    "timeZone": string


    "endTimeBehavior": string

   }
"timeoutConfig": {
  "inProgressTimeoutInMinutes": long
}
```

```
}
```

Para obtener más información, consulte `CreateJob`.

CLI syntax

```
aws iot create-job \
    --job-id <value> \
    --targets <value> \
    [--document-source <value>] \
    [--document <value>] \
    [--description <value>] \
    [--job-template-arn <value>] \
    [--presigned-url-config <value>] \
    [--target-selection <value>] \
    [--job-executions-rollout-config <value>] \
    [--abort-config <value>] \
    [--timeout-config <value>] \
    [--document-parameters <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` formato:

```
{
    "jobId": "string",
    "targets": [ "string" ],
    "documentSource": "string",
    "document": "string",
    "description": "string",
    "jobTemplateArn": "string",
    "presignedUrlConfig": {
        "roleArn": "string",
        "expiresInSec": long
     },
    "targetSelection": "string",
    "jobExecutionsRolloutConfig": {
          "exponentialRate": {
              "baseRatePerMinute": integer,
              "incrementFactor": integer,
              "rateIncreaseCriteria": {
                 "numberOfNotifiedThings": integer, // Set one or the other
                 "numberOfSucceededThings": integer // of these two values.
```

```
            },
      "maximumPerMinute": integer
      }
    },
    "abortConfig": {
    "criteriaList": [
        {
           "action": "string",
           "failureType": "string",
           "minNumberOfExecutedThings": integer,
           "thresholdPercentage": integer
         }
      ]
    },
    "timeoutConfig": {
          "inProgressTimeoutInMinutes": long
    },
    "documentParameters": {
    "string": "string"
    }
}
```

Para obtener más información, consulte `create-job`.

DeleteJob

Elimina un trabajo y sus ejecuciones de trabajo relacionadas.

La eliminación de un trabajo puede tardar tiempo, en función del número de ejecuciones de trabajo creadas para el trabajo y otros factores diversos. Aunque el trabajo se está eliminando, el estado del trabajo se muestra como "DELETION_IN_PROGRESS". Si se intenta eliminar o cancelar un trabajo cuyo estado ya es DELETION_IN_PROGRESS, se producirá un error.

HTTPS request

```
DELETE /jobs/jobId?force=force
```

Para obtener más información, consulte `DeleteJob`.

CLI syntax

```
aws iot  delete-job \
```

```
--job-id <value> \
[--force | --no-force]  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

`cli-input-json` formato:

```
{
"jobId": "string",
"force": boolean
}
```

Para obtener más información, consulte `delete-job`.

DeleteJobExecution

Elimina una ejecución de trabajo.

HTTPS request

```
DELETE /things/thingName/jobs/jobId/executionNumber/executionNumber?force=force
```

Para obtener más información, consulte `DeleteJobExecution`.

CLI syntax

```
aws iot  delete-job-execution \
--job-id <value> \
--thing-name <value> \
--execution-number <value> \
[--force | --no-force]  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

`cli-input-json` formato:

```
{
"jobId": "string",
"thingName": "string",
"executionNumber": long,
"force": boolean
```

```
}
```

Para obtener más información, consulte `delete-job-execution`.

DescribeJob

Obtiene los detalles de la ejecución de trabajo.

HTTPS request

```
GET /jobs/jobId
```

Para obtener más información, consulte `DescribeJob`.

CLI syntax

```
aws iot describe-job \
--job-id <value>  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

`cli-input-json` formato:

```
{
"jobId": "string"
}
```

Para obtener más información, consulte `describe-job`.

DescribeJobExecution

Obtiene los detalles de una ejecución de trabajo. El estado de la ejecución del trabajo debe ser `SUCCEEDED` o `FAILED`.

HTTPS request

```
GET /things/thingName/jobs/jobId?executionNumber=executionNumber
```

Para obtener más información, consulte `DescribeJobExecution`.

CLI syntax

```
aws iot  describe-job-execution \
--job-id <value> \
--thing-name <value> \
[--execution-number <value>]  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

`cli-input-json` formato:

```
{
"jobId": "string",
"thingName": "string",
"executionNumber": long
}
```

Para obtener más información, consulte `describe-job-execution`.

GetJobDocument

Obtiene el documento de trabajo para un trabajo.

> **Note**
>
> URLs Los marcadores de posición no se sustituyen por Amazon S3 prefirmado URLs en el documento devuelto. Los prefirmados solo URLs se generan cuando el servicio AWS IoT Jobs recibe una solicitud a través de MQTT.

HTTPS request

```
GET /jobs/jobId/job-document
```

Para obtener más información, consulte `GetJobDocument`.

CLI syntax

```
aws iot get-job-document \
--job-id <value>  \
```

```
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

`cli-input-json` formato:

```
{
"jobId": "string"
}
```

Para obtener más información, consulte `get-job-document`.

ListJobExecutionsForJob

Obtiene una lista de ejecuciones de trabajo para un trabajo.

HTTPS request

```
GET /jobs/jobId/things?status=status&maxResults=maxResults&nextToken=nextToken
```

Para obtener más información, consulte `ListJobExecutionsForJob`.

CLI syntax

```
aws iot  list-job-executions-for-job \
--job-id <value> \
[--status <value>] \
[--max-results <value>] \
[--next-token <value>]  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

`cli-input-json` formato:

```
{
"jobId": "string",
"status": "string",
"maxResults": "integer",
"nextToken": "string"
}
```

Para obtener más información, consulte `list-job-executions-for-job`.

ListJobExecutionsForThing

Obtiene una lista de ejecuciones de trabajo para un objeto.

HTTPS request

```
GET /things/thingName/jobs?status=status&maxResults=maxResults&nextToken=nextToken
```

Para obtener más información, consulte ListJobExecutionsForThing.

CLI syntax

```
aws iot list-job-executions-for-thing \
--thing-name <value> \
[--status <value>] \
[--max-results <value>] \
[--next-token <value>]  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

`cli-input-json` formato:

```
{
"thingName": "string",
"status": "string",
"maxResults": "integer",
"nextToken": "string"
}
```

Para obtener más información, consulte list-job-executions-for-thing.

ListJobs

Obtiene una lista de los trabajos que tiene en su. Cuenta de AWS

HTTPS request

```
GET /jobs?
status=status&targetSelection=targetSelection&thingGroupName=thingGroupName&thingGroupId=thi
```

Para obtener más información, consulte ListJobs.

CLI syntax

```
aws iot list-jobs \
[--status <value>] \
[--target-selection <value>] \
[--max-results <value>] \
[--next-token <value>] \
[--thing-group-name <value>] \
[--thing-group-id <value>]  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

`cli-input-json` formato:

```
{
"status": "string",
"targetSelection": "string",
"maxResults": "integer",
"nextToken": "string",
"thingGroupName": "string",
"thingGroupId": "string"
}
```

Para obtener más información, consulte `list-jobs`.

UpdateJob

Actualiza los campos admitidos del trabajo especificado. Los valores actualizados de `timeoutConfig` surten efecto solo en lanzamientos recientemente en curso. Actualmente, los lanzamientos en curso siguen lanzándose con la configuración de tiempo de espera anterior.

HTTPS request

```
PATCH /jobs/jobId
{
"description": "string",
"presignedUrlConfig": {
  "expiresInSec": number,
  "roleArn": "string"
},
"jobExecutionsRolloutConfig": {
```

```
  "exponentialRate": {
     "baseRatePerMinute": number,
     "incrementFactor": number,
     "rateIncreaseCriteria": {
        "numberOfNotifiedThings": number,
        "numberOfSucceededThings": number
     },
  "maximumPerMinute": number
  },
"abortConfig": {
  "criteriaList": [
     {
        "action": "string",
        "failureType": "string",
        "minNumberOfExecutedThings": number,
        "thresholdPercentage": number
     }
  ]
},
"timeoutConfig": {
  "inProgressTimeoutInMinutes": number
}
}
```

Para obtener más información, consulte `UpdateJob`.

CLI syntax

```
aws iot  update-job \
--job-id <value> \
[--description <value>] \
[--presigned-url-config <value>] \
[--job-executions-rollout-config <value>] \
[--abort-config <value>] \
[--timeout-config <value>] \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

`cli-input-json` formato:

```
{
"description": "string",
"presignedUrlConfig": {
```

```
  "expiresInSec": number,
  "roleArn": "string"
},
"jobExecutionsRolloutConfig": {
  "exponentialRate": {
     "baseRatePerMinute": number,
     "incrementFactor": number,
     "rateIncreaseCriteria": {
        "numberOfNotifiedThings": number,
        "numberOfSucceededThings": number
     }
  },
  "maximumPerMinute": number
},
"abortConfig": {
  "criteriaList": [
     {
        "action": "string",
        "failureType": "string",
        "minNumberOfExecutedThings": number,
        "thresholdPercentage": number
     }
  ]
},
"timeoutConfig": {
  "inProgressTimeoutInMinutes": number
}
}
```

Para obtener más información, consulte [update-job](#).

Operaciones y tipos de datos de las API MQTT y HTTPS de dispositivos de Jobs

Los siguientes comandos están disponibles a través de los protocolos MQTT y HTTPS. Utilice estas operaciones de API en el plano de datos para los dispositivos que ejecutan los trabajos.

Tipos de datos de MQTT y HTTPS de dispositivos de Jobs

Los siguientes tipos de datos se utilizan para comunicarse con el servicio AWS IoT Jobs a través de los protocolos MQTT y HTTPS.

JobExecution

El objeto `JobExecution` representa la ejecución de un trabajo en un dispositivo. En el siguiente ejemplo se muestra la sintaxis:

> **Note**
>
> Cuando se utilizan las operaciones de la API del plano de datos HTTP y MQTT, el tipo de datos `JobExecution`contiene un campo `JobDocument`. Sus dispositivos pueden usar esta información para recuperar el documento de trabajo de la ejecución de un trabajo.

```
{
    "jobId" : "string",
    "thingName" : "string",
    "jobDocument" : "string",
    "status": "QUEUED|IN_PROGRESS|FAILED|SUCCEEDED|CANCELED|TIMED_OUT|REJECTED|
REMOVED",
    "statusDetails": {
        "string": "string"
    },
    "queuedAt" : "timestamp",
    "startedAt" : "timestamp",
    "lastUpdatedAt" : "timestamp",
    "versionNumber" : "number",
    "executionNumber": long
}
```

Para obtener más información, consulte `JobExecution` o `job-execution`.

JobExecutionState

El `JobExecutionState` contiene información sobre el estado de la ejecución de un trabajo. En el siguiente ejemplo se muestra la sintaxis:

```
{
    "status": "QUEUED|IN_PROGRESS|FAILED|SUCCEEDED|CANCELED|TIMED_OUT|REJECTED|
REMOVED",
    "statusDetails": {
        "string": "string"
        ...
```

```
    }
    "versionNumber": "number"
}
```

Para obtener más información, consulte JobExecutionState o job-execution-state.

JobExecutionSummary

Contiene una subred de información acerca de una ejecución de trabajo. En el siguiente ejemplo se muestra la sintaxis:

```
{
    "jobId": "string",
    "queuedAt": timestamp,
    "startedAt": timestamp,
    "lastUpdatedAt": timestamp,
    "versionNumber": "number",
    "executionNumber": long
}
```

Para obtener más información, consulte JobExecutionSummary o job-execution-summary.

En las siguientes secciones se puede consultar más información sobre las operaciones de las API MQTT y HTTPS:

- Operaciones de la API MQTT de dispositivos de Jobs
- API HTTP de dispositivos de Jobs

Operaciones de la API MQTT de dispositivos de Jobs

Puede emitir comandos de dispositivos de Jobs publicando los mensajes MQTT en los Temas reservados utilizados para comandos de Jobs.

El cliente del dispositivo debe estar suscrito a los temas de los mensajes de respuesta de estos comandos. Si utiliza el cliente de AWS IoT dispositivo, su dispositivo se suscribirá automáticamente a los temas de respuesta. Esto significa que el agente de mensajes publicará los temas de los mensajes de respuesta en el cliente que publicó el mensaje de comando, independientemente de que su cliente se haya suscrito o no a los temas de los mensajes de respuesta. Estos mensajes de respuesta no pasan por el agente de mensajes y otros clientes o reglas no pueden suscribirse a ellos.

Al suscribirse al trabajo y a los temas de eventos `jobExecution` de su solución de monitoreo de flota, primero habilite los eventos de trabajos y ejecución de trabajos para recibir cualquier evento en la nube. Los mensajes de progreso del trabajo que se procesan a través del agente de mensajes y que las reglas de AWS IoT pueden utilizar se publican como Eventos de trabajos. Como el agente de mensajes publica los mensajes de respuesta, incluso sin una suscripción explícita a ellos, el cliente debe estar configurado para recibir e identificar los mensajes que recibe. El cliente también debe confirmar que el *`thingName`* tema del mensaje entrante se aplica al nombre del cliente antes de que el cliente actúe en función del mensaje.

> **Note**
>
> Los mensajes que se AWS IoT envían en respuesta a los comandos de la API MQTT Jobs se cargan a su cuenta, independientemente de que se haya suscrito a ellos de forma explícita o no.

A continuación se muestran las operaciones de la API MQTT y su sintaxis de solicitud y respuesta. Todas las operaciones de la API MQTT presentan los siguientes parámetros:

clientToken

Un token de cliente opcional utilizado para correlacionar solicitudes y respuestas. Introduzca un valor arbitrario aquí y se reflejará en la respuesta.

`timestamp`

El tiempo, en segundos, desde la fecha de inicio, cuando se envió el mensaje.

GetPendingJobExecutions

Obtiene la lista de todos los trabajos que no están en un estado terminal, respecto de un objeto especificado.

Para invocar esta API, publique un mensaje en `$aws/things/`*`thingName`*`/jobs/get`.

Carga de solicitud:

```
{ "clientToken": "string" }
```

El agente de mensajes publicará `$aws/things/thingName/jobs/get/accepted` e `$aws/things/thingName/jobs/get/rejected` incluso sin una suscripción específica a ellos. Sin embargo, para que su cliente reciba los mensajes, debe estar escuchándolos. Para obtener más información, consulte la nota sobre los mensajes de la API de trabajos.

Carga de respuesta:

```
{
"inProgressJobs" : [ JobExecutionSummary ... ],
"queuedJobs" : [ JobExecutionSummary ... ],
"timestamp" : 1489096425069,
"clientToken" : "client-001"
}
```

Donde `inProgressJobs` y `queuedJobs` devuelven una lista de objetos JobExecutionSummary cuyo estado es `IN_PROGRESS` o `QUEUED`.

StartNextPendingJobExecution

Obtiene y comienza la siguiente ejecución de trabajos pendientes de un objeto (estado `IN_PROGRESS` o `QUEUED`).

- Las ejecuciones de trabajo con el estado `IN_PROGRESS` se devuelven en primer lugar.
- Las ejecuciones de trabajo se devuelven en el orden en el que se pusieron en cola. Cuando añada o elimine algo del grupo de destino de su trabajo, confirme el orden de despliegue de las ejecuciones de trabajos nuevos en comparación con las ejecuciones de trabajos existentes.
- Si la siguiente ejecución de trabajo pendiente es `QUEUED`, su estado cambia a `IN_PROGRESS` y los detalles del estado de la ejecución de trabajo se establecen según se haya especificado.
- Si la siguiente ejecución de trabajo pendiente está ya en `IN_PROGRESS`, los detalles del estado no cambiarán.
- Si no hay ejecuciones de trabajo pendientes, la respuesta no incluirá el campo `execution`.
- Opcionalmente, puede crear un temporizador de pasos estableciendo un valor para la propiedad `stepTimeoutInMinutes`. Si no actualiza el valor de esta propiedad mediante la ejecución de `UpdateJobExecution`, la ejecución del trabajo agotará el tiempo de espera cuando venza el temporizador de pasos.

Para invocar esta API, publique un mensaje en `$aws/things/thingName/jobs/start-next`.

Carga de solicitud:

```
{
"statusDetails": {
    "string": "job-execution-state"
    ...
},
"stepTimeoutInMinutes": long,
"clientToken": "string"
}
```

`statusDetails`

Un conjunto de pares nombre-valor que describen el estado de ejecución del trabajo. Si no se especifica, `statusDetails` no se modifica.

`stepTimeOutInMinutes`

Especifica la cantidad de tiempo que tiene este dispositivo para finalizar la ejecución de este trabajo. Si el estado de ejecución del trabajo no se ha establecido en un estado terminal antes de que este temporizador venza o antes de que se restablezca el temporizador (llamando a `UpdateJobExecution`, estableciendo el estado en `IN_PROGRESS` y especificando un valor de tiempo de espera nuevo en el campo `stepTimeoutInMinutes`) el estado de ejecución se establece automáticamente en `TIMED_OUT`. Configurar este tiempo de espera no tiene ningún efecto en el tiempo de espera de la ejecución de ese trabajo que se pueda haber especificado cuando se creó el trabajo (`CreateJob` con el campo `timeoutConfig`).

Los valores válidos para este parámetro están comprendidos entre 1 y 10 080 (1 minuto a 7 días). El valor -1 también es válido y cancelará el temporizador de pasos actual (creado por el uso anterior de UpdateJobExecutionRequest).

El agente de mensajes publicará `$aws/things/thingName/jobs/start-next/accepted` e `$aws/things/thingName/jobs/start-next/rejected` incluso sin una suscripción específica a ellos. Sin embargo, para que su cliente reciba los mensajes, debe estar escuchándolos. Para obtener más información, consulte la nota sobre los mensajes de la API de trabajos.

Carga de respuesta:

```
{
"execution" : JobExecutionData,
```

```
"timestamp" : timestamp,
"clientToken" : "string"
}
```

Donde `execution` es un objeto JobExecution. Por ejemplo:

```
{
"execution" : {
    "jobId" : "022",
    "thingName" : "MyThing",
    "jobDocument" : "< contents of job document >",
    "status" : "IN_PROGRESS",
    "queuedAt" : 1489096123309,
    "lastUpdatedAt" : 1489096123309,
    "versionNumber" : 1,
    "executionNumber" : 1234567890
},
"clientToken" : "client-1",
"timestamp" : 1489088524284,
}
```

DescribeJobExecution

Obtiene información detallada acerca de una ejecución de trabajo.

Puede establecer `jobId` en `$next` para devolver la siguiente ejecución de trabajo pendiente para un objeto (estado `IN_PROGRESS` o `QUEUED`).

Para invocar esta API, publique un mensaje en `$aws/things/`*`thingName`*`/jobs/`*`jobId`*`/get`.

Carga de solicitud:

```
{
"jobId" : "022",
"thingName" : "MyThing",
"executionNumber": long,
"includeJobDocument": boolean,
"clientToken": "string"
}
```

`thingName`

El nombre del objeto asociado con el dispositivo.

`jobId`

El identificador único asignado a este trabajo cuando se creó.

También puede usar `$next` para devolver la siguiente ejecución de trabajo pendiente para un objeto (estado `IN_PROGRESS` o `QUEUED`). En este caso, las ejecuciones de trabajo con el estado `IN_PROGRESS` se devuelven en primer lugar. Las ejecuciones de trabajo se devuelven en el orden en el que se crearon.

`executionNumber`

(Opcional) Un número que identifica la ejecución de un trabajo en un dispositivo. Si no se especifica, se devuelve la ejecución de trabajo más reciente.

`includeJobDocument`

(Opcional) Cuando no se establece en `false`, la respuesta contiene el documento de trabajo. El valor predeterminado es `true`.

El agente de mensajes publicará `$aws/things/`*`thingName`*`/jobs/`*`jobId`*`/get/accepted` e `$aws/things/`*`thingName`*`/jobs/`*`jobId`*`/get/rejected` incluso sin una suscripción específica a ellos. Sin embargo, para que su cliente reciba los mensajes, debe estar escuchándolos. Para obtener más información, consulte la nota sobre los mensajes de la API de trabajos.

Carga de respuesta:

```
{
"execution" : JobExecutionData,
"timestamp": "timestamp",
"clientToken": "string"
}
```

Donde `execution` es un objeto JobExecution.

UpdateJobExecution

Actualiza el estado de una ejecución de trabajo. Si lo prefiere, puede crear un temporizador de pasos estableciendo un valor para la propiedad `stepTimeoutInMinutes`. Si no actualiza el valor de esta propiedad ejecutando `UpdateJobExecution` otra vez, la ejecución del trabajo agotará el tiempo de espera cuando venza el temporizador de pasos.

Para invocar esta API, publique un mensaje en `$aws/things/`*`thingName`*`/jobs/`*`jobId`*`/update`.

Carga de solicitud:

```
{
"status": "job-execution-state",
"statusDetails": {
    "string": "string"
    ...
},
"expectedVersion": "number",
"executionNumber": long,
"includeJobExecutionState": boolean,
"includeJobDocument": boolean,
"stepTimeoutInMinutes": long,
"clientToken": "string"
}
```

`status`

El nuevo estado de ejecución del trabajo (`IN_PROGRESS`, `FAILED`, `SUCCEEDED` o `REJECTED`). Debe especificarse en cada actualización.

`statusDetails`

Un conjunto de pares nombre-valor que describen el estado de ejecución del trabajo. Si no se especifica, `statusDetails` no se modifica.

`expectedVersion`

La versión actual esperada de la ejecución de trabajos. Cada vez que actualiza la ejecución de trabajos, aumenta su versión. Si la versión de la ejecución del trabajo almacenada en el servicio AWS IoT Jobs no coincide, la actualización se rechaza con un `VersionMismatch` error. También se devuelve una ErrorResponse que contiene los datos del estado actual de ejecución del trabajo. (Esto hace que no sea necesario realizar una solicitud `DescribeJobExecution` aparte para obtener los datos de estado de ejecución del trabajo).

`executionNumber`

(Opcional) Un número que identifica la ejecución de un trabajo en un dispositivo. Si no se especifica, se utiliza la ejecución de trabajo más reciente.

`includeJobExecutionState`

(Opcional) Cuando se incluye y establece en `true`, la respuesta contiene el campo `JobExecutionState`. El valor predeterminado es `false`.

`includeJobDocument`

(Opcional) Cuando se incluye y establece en `true`, la respuesta contiene el `JobDocument`. El valor predeterminado es `false`.

`stepTimeoutInMinutes`

Especifica la cantidad de tiempo que tiene este dispositivo para finalizar la ejecución de este trabajo. Si el estado de ejecución del trabajo no se ha establecido en un estado terminal antes de que este temporizador venza, o antes de que se restablezca, el estado de ejecución del trabajo se establece en `TIMED_OUT`. Configurar o restablecer este tiempo de espera no tiene ningún efecto en el tiempo de espera de la ejecución del trabajo que se pueda haber especificado cuando se creó el trabajo.

El agente de mensajes publicará `$aws/things/`*`thingName`*`/jobs/`*`jobId`*`/update/accepted` e `$aws/things/`*`thingName`*`/jobs/`*`jobId`*`/update/rejected` incluso sin una suscripción específica a ellos. Sin embargo, para que su cliente reciba los mensajes, debe estar escuchándolos. Para obtener más información, consulte la nota sobre los mensajes de la API de trabajos.

Carga de respuesta:

```
{
"executionState": JobExecutionState,
"jobDocument": "string",
"timestamp": timestamp,
"clientToken": "string"
}
```

`executionState`

Un objeto JobExecutionState.

`jobDocument`

Un objeto documento de trabajo.

> **Note**
>
> En las respuestas de MQTT, el campo `jobDocument` es un objeto JSON. En las respuestas de HTTP, es una representación de cadena del objeto JSON.

`timestamp`

El tiempo, en segundos, desde la fecha de inicio, cuando se envió el mensaje.

`clientToken`

Un token de cliente utilizado para correlacionar solicitudes y respuestas.

Al utilizar el protocolo MQTT, también puede realizar las siguientes actualizaciones:

JobExecutionsChanged

Se envía cuando se añade una ejecución de trabajo a la lista de ejecuciones de trabajo pendientes para un objeto, o cuando se quita de dicha lista.

Utilice el tema :

`$aws/things/thingName/jobs/notify`

Carga útil de mensaje:

```
{
"jobs" : {
    "JobExecutionState": [ JobExecutionSummary ... ]
         },
    "timestamp": timestamp
}
```

NextJobExecutionChanged

Se envía cuando se produce un cambio en el que la ejecución de trabajo es la siguiente en la lista de ejecuciones de trabajo pendientes para un objeto, como se define para `DescribeJobExecution` con `jobId` `$next`. Este mensaje no se envía cuando cambian los detalles de ejecución del siguiente trabajo, solo cuando el siguiente trabajo que devolvería `DescribeJobExecution` con `jobId` `$next` ha cambiado. Considere las ejecuciones de trabajo J1 y J2 con el estado `QUEUED`. J1 es el siguiente elemento de la lista de ejecuciones de trabajo pendientes. Si el estado de J2 cambia a `IN_PROGRESS` mientras el estado de J1 permanece invariable, se envía esta notificación y contiene los detalles de J2.

Utilice el tema :

`$aws/things/thingName/jobs/notify-next`

Carga útil de mensaje:

```
{
"execution" : JobExecution,
"timestamp": timestamp,
}
```

API HTTP de dispositivos de Jobs

Los dispositivos pueden comunicarse con AWS IoT Jobs mediante la firma HTTP, versión 4, en el puerto 443. Este es el método utilizado por la CLI AWS SDKs y. Para obtener más información sobre estas herramientas, consulte Referencia de AWS CLI comandos: iot-jobs-data o AWS SDKs and Tools.

Los siguientes comandos están disponibles para los dispositivos que ejecutan los trabajos. Para obtener información sobre el uso de las operaciones de la API con el protocolo MQTT, consulteOperaciones de la API MQTT de dispositivos de Jobs.

GetPendingJobExecutions

Obtiene la lista de todos los trabajos que no están en un estado terminal, respecto de un objeto especificado.

HTTPS request

```
GET /things/thingName/jobs
```

Respuesta:

```
{
"inProgressJobs" : [ JobExecutionSummary ... ],
"queuedJobs" : [ JobExecutionSummary ... ]
}
```

Para obtener más información, consulte GetPendingJobExecutions.

CLI syntax

```
aws iot-jobs-data get-pending-job-executions \
--thing-name <value>  \
[--cli-input-json <value>] \
```

```
[--generate-cli-skeleton]
```

`cli-input-json` formato:

```
{
"thingName": "string"
}
```

Para obtener más información, consulte `get-pending-job-executions`.

StartNextPendingJobExecution

Obtiene y comienza la siguiente ejecución de trabajos pendientes de un objeto (con un estado `IN_PROGRESS` o `QUEUED`).

- Las ejecuciones de trabajo con el estado `IN_PROGRESS` se devuelven en primer lugar.
- Las ejecuciones de trabajo se devuelven en el orden en el que se crearon.
- Si la siguiente ejecución de trabajo pendiente es `QUEUED`, su estado cambia a `IN_PROGRESS` y los detalles del estado de la ejecución de trabajo se establecen según se haya especificado.
- Si la siguiente ejecución de trabajo pendiente está ya en `IN_PROGRESS`, los detalles del estado no cambiarán.
- Si no hay ejecuciones de trabajo pendientes, la respuesta no incluirá el campo `execution`.
- Opcionalmente, puede crear un temporizador de pasos estableciendo un valor para la propiedad `stepTimeoutInMinutes`. Si no actualiza el valor de esta propiedad mediante la ejecución de `UpdateJobExecution`, la ejecución del trabajo agotará el tiempo de espera cuando venza el temporizador de pasos.

HTTPS request

En el siguiente ejemplo se muestra la sintaxis de la solicitud:

```
PUT /things/thingName/jobs/$next
{
"statusDetails": {
    "string": "string"
    ...
},
"stepTimeoutInMinutes": long
```

```
}
```

Para obtener más información, consulte StartNextPendingJobExecution.

CLI syntax

Sinopsis:

```
aws iot-jobs-data  start-next-pending-job-execution \
--thing-name <value> \
{--step-timeout-in-minutes <value>] \
[--status-details <value>]  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

`cli-input-json` formato:

```
{
"thingName": "string",
"statusDetails": {
"string": "string"
},
"stepTimeoutInMinutes": long
}
```

Para obtener más información, consulte start-next-pending-job-execution.

DescribeJobExecution

Obtiene información detallada acerca de una ejecución de trabajo.

Puede establecer el `jobId` en `$next` para devolver la siguiente ejecución de trabajo pendiente para un objeto. El estado de la ejecución del trabajo debe ser `QUEUED` o `IN_PROGRESS`.

HTTPS request

Solicitud:

```
GET /things/thingName/jobs/jobId?
executionNumber=executionNumber&includeJobDocument=includeJobDocument
```

Response: (Respuesta:)

```
{
"execution" : JobExecution,
}
```

Para obtener más información, consulte DescribeJobExecution.

CLI syntax

Sinopsis:

```
aws iot-jobs-data  describe-job-execution \
--job-id <value> \
--thing-name <value> \
[--include-job-document | --no-include-job-document] \
[--execution-number <value>]  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

`cli-input-json` formato:

```
{
"jobId": "string",
"thingName": "string",
"includeJobDocument": boolean,
"executionNumber": long
}
```

Para obtener más información, consulte describe-job-execution.

UpdateJobExecution

Actualiza el estado de una ejecución de trabajo. Opcionalmente, puede crear un temporizador de pasos estableciendo un valor para la propiedad `stepTimeoutInMinutes`. Si no actualiza el valor de esta propiedad ejecutando `UpdateJobExecution` otra vez, la ejecución del trabajo agotará el tiempo de espera cuando venza el temporizador de pasos.

HTTPS request

Solicitud:

```
POST /things/thingName/jobs/jobId
```

```
{
"status": "job-execution-state",
"statusDetails": {
    "string": "string"
    ...
},
"expectedVersion": "number",
"includeJobExecutionState": boolean,
"includeJobDocument": boolean,
"stepTimeoutInMinutes": long,
"executionNumber": long
}
```

Para obtener más información, consulte UpdateJobExecution.

CLI syntax

Sinopsis:

```
aws iot-jobs-data  update-job-execution \
--job-id <value> \
--thing-name <value> \
--status <value> \
[--status-details <value>] \
[--expected-version <value>] \
[--include-job-execution-state | --no-include-job-execution-state] \
[--include-job-document | --no-include-job-document] \
[--execution-number <value>]  \
[--cli-input-json <value>] \
[--step-timeout-in-minutes <value>] \
[--generate-cli-skeleton]
```

`cli-input-json` formato:

```
{
"jobId": "string",
"thingName": "string",
"status": "string",
"statusDetails": {
"string": "string"
},
"stepTimeoutInMinutes": number,
"expectedVersion": long,
"includeJobExecutionState": boolean,
```

```
"includeJobDocument": boolean,
"executionNumber": long
}
```

Para obtener más información, consulte `update-job-execution`.

Proteja los usuarios y los dispositivos con AWS IoT Jobs

Para autorizar a los usuarios a usar AWS IoT Jobs con sus dispositivos, debes concederles permisos mediante las políticas de IAM. Luego, los dispositivos deben autorizarse mediante AWS IoT Core políticas para conectarse de forma segura a los trabajos ejecutados AWS IoT, recibirlos y actualizar el estado de ejecución.

Tipo de política obligatorio para AWS IoT Jobs

La siguiente tabla muestra los diferentes tipos de políticas que debe usar para la autorización. Para obtener más información acerca de la política de uso obligatorio, consulte Autorización.

Tipo de política obligatoria

Caso de uso	Protocol	Autenticación	plane/data Plano de control	Tipo de identidad	Tipo de política obligatoria
Autorizar a un administrador, operador o servicio en la nube para trabajar de forma segura con Jobs	HTTPS	AWS Autenticación de la versión 4 de Signature (puerto 443)	Plano de control y plano de datos	Identidad de Amazon Cognito o de IAM o usuario federado	Política de IAM
Autorizar el dispositivo de IoT para trabajar de	MQTT/HTTPS	Autenticación mutua TCP o TLS (puerto 8883 o 443)	Plano de datos	Certificados X.509	AWS IoT Core política

Caso de uso	Protocol	Autenticación	plane/dat a Plano de control	Tipo de identidad	Tipo de política obligatoria
forma segura con Jobs					

Para autorizar las operaciones de AWS IoT Jobs que se pueden realizar tanto en el plano de control como en el plano de datos, debe utilizar las políticas de IAM. Las identidades deben haberse autenticado con AWS IoT para realizar estas operaciones, que deben ser Identidades de Amazon Cognito o Usuarios, grupos y roles de IAM. Para obtener más información acerca de la autenticación, consulte Autenticación.

Los dispositivos ahora deben estar autorizados en el plano de datos mediante AWS IoT Core políticas para conectarse de forma segura a la puerta de enlace del dispositivo. La pasarela de dispositivos permite a los dispositivos comunicarse de forma segura con los trabajos ejecutados AWS IoT, recibirlos y actualizar el estado de ejecución de los trabajos. La comunicación de los dispositivos se asegura mediante protocolos de comunicación MQTT o Publicación en HTTPS seguros. Estos protocolos utilizan Certificados de cliente X.509 los proporcionados por AWS IoT para autenticar las conexiones del dispositivo.

A continuación, se muestra cómo se autoriza a los usuarios, los servicios de nube y los dispositivos a utilizar AWS IoT Jobs. Para obtener información sobre las operaciones de la API del plano de control y el plano de datos , consulte AWS IoT operaciones de API de trabajos.

Temas

- Autorizar a los usuarios y los servicios en la nube a usar Jobs AWS IoT
- Autorizar tus dispositivos para usar AWS IoT Jobs de forma segura en el plano de los datos

Autorizar a los usuarios y los servicios en la nube a usar Jobs AWS IoT

Para autorizar a los usuarios y servicios en la nube, debe utilizar las políticas de IAM tanto en el plano de control como en el plano de datos. Las políticas deben usarse con el protocolo HTTPS y deben usar la autenticación de AWS Signature Version 4 (puerto 443) para autenticar a los usuarios.

> **Note**
>
> AWS IoT Core las políticas no deben usarse en el plano de control. Para autorizar a los usuarios o los servicios en la nube, solo se utilizan las políticas de IAM. Para obtener más información acerca de usar el tipo de política obligatoria, consulte Tipo de política obligatorio para AWS IoT Jobs.

Las políticas de IAM son documentos JSON que contienen declaraciones de políticas. Las declaraciones de políticas utilizan los elementos Efecto, Acción y Recurso para especificar los recursos, las acciones permitidas o denegadas y las condiciones en las que se permiten o deniegan las acciones. Para obtener más información, consulte Referencia de los elementos de las políticas de JSON de IAM en la guía del usuario de IAM.

> **Warning**
>
> Te recomendamos que no utilices permisos comodín, como `"Action": ["iot:*"]` en tus políticas o AWS IoT Core políticas de IAM. El uso de permisos comodín no es una práctica recomendada de seguridad. Para obtener más información, consulte Políticas de AWS IoT demasiado permisivas.

Políticas de IAM en el plano de control

En el plano de control, las políticas de IAM utilizan el prefijo `iot:` con la acción para autorizar la operación de la API de trabajos correspondiente. Por ejemplo, la acción de política `iot:CreateJob` concede al usuario permiso para usar la API `CreateJob`.

Acciones de políticas

La siguiente tabla muestra una lista de las acciones y permisos de las políticas de IAM para usar las acciones de la API. Para obtener información sobre los tipos de recursos, consulte Tipos de recursos definidos por. AWS IoT Para obtener más información sobre AWS IoT las acciones, consulte Acciones definidas por AWS IoT.

Acciones de la política de IAM en el plano de control

Acción de política	Operación de la API	Tipos de recurso	Description (Descripción)
`iot:AssociateTargetsWithJob`	AssociateTargetsWithJob	• job • thing • thinggroup	Representa el permiso para asociar un grupo a un trabajo continuo. El permiso `iot:AssociateTargetsWithJob` se comprueba cada vez que se presenta una solicitud para asociar destinos.
`iot:CancelJob`	CancelJob	job	Representa el permiso para cancelar un trabajo. El permiso `iot:CancelJob` se comprueba cada vez que se presenta una solicitud para cancelar un trabajo.
`iot:CancelJobExecution`	CancelJobExecution	• job • thing	Representa el permiso para cancelar la ejecución de un trabajo. El permiso `iot:CancelJobExecution` se comprueba cada vez que se presenta una solicitud para cancelar la ejecución de un trabajo.
`iot:CreateJob`	CreateJob	• job • thing • thinggroup • jobtemplate • paquete	Representa el permiso para crear un trabajo. El permiso `iot: CreateJob` se comprueba cada vez que se presenta una solicitud para crear un trabajo.
`iot:CreateJobTemplate`	CreateJobTemplate	• job • jobtemplate • paquete	Representa el permiso para crear una plantilla de trabajo. El permiso `iot: CreateJobTemplate` se comprueba cada vez que se presenta una solicitud para crear una plantilla de trabajo.
`iot:DeleteJob`	DeleteJob	job	Representa el permiso para eliminar un trabajo. El permiso `iot: DeleteJob` se

Acción de política	Operación de la API	Tipos de recurso	Description (Descripción)
			comprueba cada vez que se presenta una solicitud para eliminar un trabajo.
`iot:DeleteJobTemplate`	DeleteJobTemplate	jobtemplate	Representa el permiso para eliminar una plantilla de trabajo. El permiso `iot:CreateJobTemplate` se comprueba cada vez que se presenta una solicitud para eliminar una plantilla de trabajo.
`iot:DeleteJobExecution`	DeleteJobTemplate	• job • thing	Representa el permiso para eliminar una ejecución de trabajo. El permiso `iot:DeleteJobExecution` se comprueba cada vez que se presenta una solicitud para eliminar la ejecución de un trabajo.
`iot:DescribeJob`	DescribeJob	job	Representa el permiso para describir un trabajo. El permiso `iot: DescribeJob` se comprueba cada vez que se presenta una solicitud para describir un trabajo.
`iot:DescribeJobExecution`	DescribeJobExecution	• job • thing	Representa el permiso para describir una ejecución de trabajo. El permiso `iot: DescribeJobExecution` se comprueba cada vez que se presenta una solicitud para describir la ejecución de un trabajo.
`iot:DescribeJobTemplate`	DescribeJobTemplate	jobtemplate	Representa el permiso para describir una plantilla de trabajo. El permiso `iot:DescribeJobTemplate` se comprueba cada vez que se presenta una solicitud para describir una plantilla de trabajo.

Acción de política	Operación de la API	Tipos de recurso	Description (Descripción)
`iot:DescribeManagedJobTemplate`	`DescribeManagedJobTemplate`	jobtemplate	Representa el permiso para describir una plantilla de trabajo administrada. El permiso `iot: DescribeManagedJobTemplate` se comprueba cada vez que se presenta una solicitud para describir una plantilla de trabajo administrada.
`iot:GetJobDocument`	`GetJobDocument`	job	Representa el permiso para obtener el documento de trabajo de un trabajo. El permiso `iot:GetJobDocument` se comprueba cada vez que se presenta una solicitud para obtener un documento de trabajo.
`iot:ListJobExecutionsForJob`	`ListJobExecutionsForJob`	job	Representa el permiso para enumerar las ejecuciones de trabajo de un trabajo. El permiso `iot:ListJobExecutionsForJob` se comprueba cada vez que se presenta una solicitud para enumerar las ejecuciones de trabajo de un trabajo.
`iot:ListJobExecutionsForThing`	`ListJobExecutionsForThing`	thing	Representa el permiso para enumerar las ejecuciones de trabajo de un trabajo. El permiso `iot:ListJobExecutionsForThing` se comprueba cada vez que se presenta una solicitud para enumerar las ejecuciones de trabajo de un objeto.
`iot:ListJobs`	`ListJobs`	none	Representa el permiso para enumerar los trabajos. El permiso `iot:ListJobs` se comprueba cada vez que se presenta una solicitud para enumerar los trabajos.

Acción de política	Operación de la API	Tipos de recurso	Description (Descripción)
iot:ListJobTemplates	ListJobTemplates	none	Representa el permiso para enumerar las plantillas de trabajo. El permiso iot:ListJobTemplates se comprueba cada vez que se presenta una solicitud para enumerar las plantillas de trabajo.
iot:ListManagedJobTemplates	ListManagedJobTemplates	none	Representa el permiso para enumerar las plantillas de trabajo administradas. El permiso iot:ListManagedJobTemplates se comprueba cada vez que se presenta una solicitud para enumerar las plantillas de trabajo administradas.
iot:UpdateJob	UpdateJob	job	Representa el permiso para actualizar un trabajo. El permiso iot:UpdateJob se comprueba cada vez que se presenta una solicitud para actualizar un trabajo.
iot:TagResource	TagResource	• job • jobtemplate • thing	Concede permiso para etiquetar un recurso específico.
iot:UntagResource	UntagResource	• job • jobtemplate • thing	Concede permiso para quitar las etiquetas del recurso específico.

Ejemplo de política de IAM básica

En el siguiente ejemplo se muestra una política de IAM que concede al usuario permiso para realizar las siguientes acciones para su objeto y grupo de objetos de IoT.

En el ejemplo, sustituya:

- *region*con tus Región de AWS, comous-east-1.
- *account-id*con tu Cuenta de AWS número, por ejemplo57EXAMPLE833.
- *thing-group-name*con el nombre del grupo de cosas de IoT al que apunta los trabajos, por ejemploFirmwareUpdateGroup.
- *thing-name*con el nombre de tu actividad de IoT a la que te diriges a los trabajos, por ejemploMyIoTThing.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "iot:CreateJobTemplate",
                "iot:CreateJob"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:iot:us-east-1:123456789012:thinggroup/thing-group-
name"
        },
        {
            "Action": [
                "iot:DescribeJob",
                "iot:CancelJob",
                "iot:DeleteJob"
            ],
            "Effect": "Allow",
     "Resource": "arn:aws:iot:us-east-1:123456789012:job/*"
        },
        {
            "Action": [
                "iot:DescribeJobExecution",
                "iot:CancelJobExecution",
                "iot:DeleteJobExecution"
            ],
            "Effect": "Allow",
            "Resource": [
            "arn:aws:iot:us-east-1:123456789012:thing/thing-123",
                "arn:aws:iot:us-east-1:123456789012:job/*"
```

```
            ]
        }
    ]
}
```

Ejemplo de política de IAM para la autorización basada en IP

Puede impedir que las entidades principales realicen llamadas a la API a su punto de conexión del plano de control desde direcciones IP específicas. Para especificar las direcciones IP que se pueden permitir, en el elemento Condición de la política de IAM, utilice la clave de condición global `aws:SourceIp`.

El uso de esta clave de condición también puede impedir que otras Servicio de AWS personas realicen estas llamadas a la API en tu nombre, por ejemplo AWS CloudFormation. Para permitir el acceso a estos servicios, utilice la clave de condición `aws:ViaAWSService`global junto con la SourceIp clave aws:. Esto asegura que la restricción de acceso a la dirección IP de origen se aplica únicamente a las solicitudes realizadas directamente por una entidad principal. Para obtener más información, consulte AWS: Denega el acceso a en AWS función de la IP de origen.

El siguiente ejemplo muestra cómo permitir que solo una dirección IP específica pueda realizar llamadas a la API al punto de conexión del plano de control. La clave `aws:ViaAWSService` está configurada en`true`, lo que permite a otros servicios realizar llamadas a la API en su nombre.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:CreateJobTemplate",
                "iot:CreateJob"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                 "IpAddress": {
                     "aws:SourceIp": "123.45.167.89"
             },
                 "Bool": {
```

```
                    "aws:ViaAWSService": "false"
        }
        }
    }]
}
```

Políticas de IAM en el plano de datos

Las políticas de IAM en el plano de datos utilizan el prefijo `iotjobsdata:` para autorizar las operaciones de la API de trabajos que los usuarios pueden realizar. En el plano de los datos, se puede conceder a un usuario permiso para usar la API `DescribeJobExecution` mediante la acción de política `iotjobsdata:DescribeJobExecution`.

> **Warning**
>
> No se recomienda utilizar políticas de IAM en el plano de datos cuando se dirija a Jobs de AWS IoT para sus dispositivos. Recomendamos utilizar las políticas de IAM en el plano de control para que los usuarios creen y gestionen los trabajos. En el plano de datos, para autorizar a los dispositivos a recuperar las ejecuciones de trabajos y actualizar el estado de ejecución, utilice AWS IoT Core políticas para el protocolo HTTPS.

Ejemplo de política de IAM básica

Por lo general, las operaciones de la API que se deben autorizar se realizan escribiendo los comandos CLI. A continuación se muestra un ejemplo de un usuario realizando una operación `DescribeJobExecution`.

En el ejemplo, sustituya:

- *`region`*con su Región de AWS, por ejemplo`us-east-1`.
- *`account-id`*con tu Cuenta de AWS número, por ejemplo`57EXAMPLE833`.
- *`thing-name`*con el nombre de tu actividad de IoT a la que te diriges a los trabajos, por ejemplo`myRegisteredThing`.
- *`job-id`* es el identificador único del trabajo al que se dirige la API.

```
aws iot-jobs-data describe-job-execution \
    --endpoint-url "https://account-id.jobs.iot.region.amazonaws.com" \
```

```
    --job-id jobID --thing-name thing-name
```

A continuación se muestra una política de IAM de ejemplo que autoriza esta acción:

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Action": [
            "iotjobsdata:DescribeJobExecution"
        ],
        "Effect": "Allow",
        "Resource": "arn:aws:iot:us-east-1:123456789012:thing/thing-123"
    }
}
```

Ejemplos de política de IAM para la autorización basada en IP

Puede impedir que las entidades principales realicen llamadas a la API a su punto de conexión del plano de datos desde direcciones IP específicas. Para especificar las direcciones IP que se pueden permitir, en el elemento Condición de la política de IAM, utilice la clave de condición global `aws:SourceIp`.

El uso de esta clave de condición también puede impedir que otras Servicio de AWS personas realicen estas llamadas a la API en tu nombre, por ejemplo AWS CloudFormation. Para permitir el acceso a estos servicios, utilice la clave de condición global `aws:ViaAWSService` con la clave de condición `aws:SourceIp`. Esto asegura que la restricción de acceso a la dirección IP se aplica únicamente a las solicitudes realizadas directamente por la entidad principal. Para obtener más información, consulte AWS: Denega el acceso en AWS función de la IP de origen.

El siguiente ejemplo muestra cómo permitir que solo una dirección IP específica pueda realizar llamadas a la API al punto de conexión del plano de datos.

```
{
    "Version":"2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "iotjobsdata:*"
```

```
        ],
        "Resource": [
            "*"
        ],
        "Condition": {
            "IpAddress": {
                "aws:SourceIp": "123.45.167.89"
        },
            "Bool": {
                "aws:ViaAWSService": "false"
        }
        }
    }]
}
```

El siguiente ejemplo muestra cómo restringir direcciones IP o rangos de direcciones específicos para que no puedan realizar llamadas a la API al punto de conexión del plano de datos.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "iotjobsdata:*"
            ],
            "Condition": {
                "IpAddress": {
                    "aws:SourceIp": [
                        "123.45.167.89",
                        "192.0.2.0/24",
                        "203.0.113.0/24"
                    ]
            }
        },
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Ejemplo de política de IAM tanto para el plano de control como para el plano de datos

Si realiza una operación de API tanto en el plano de control como en el plano de datos, la acción de política del plano de control debe usar el prefijo `iot:` y la acción de política del plano de datos debe usar el prefijo `iotjobsdata:`.

Por ejemplo, la API `DescribeJobExecution` se puede usar tanto en el plano de control como en el plano de datos. En el plano de control, la DescribeJobExecutionAPI se utiliza para describir la ejecución de un trabajo. En el plano de los datos, la DescribeJobExecutionAPI se utiliza para obtener detalles de la ejecución de un trabajo.

La siguiente política de IAM autoriza a un usuario a utilizar la API `DescribeJobExecution` tanto en el plano de control como en el plano de datos.

En el ejemplo, sustituya:

- *`region`*con tu Región de AWS, como`us-east-1`.
- *`account-id`*con tu Cuenta de AWS número, por ejemplo`57EXAMPLE833`.
- *`thing-name`*con el nombre de tu actividad de IoT a la que te diriges a los trabajos, por ejemplo`MyIoTThing`.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "iotjobsdata:DescribeJobExecution"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:iot:us-east-1:123456789012:thing/thing-123"
        },
        {
            "Action": [
                "iot:DescribeJobExecution",
                "iot:CancelJobExecution",
                "iot:DeleteJobExecution"
            ],
            "Effect": "Allow",
            "Resource": [
            "arn:aws:iot:us-east-1:123456789012:thing/thing-123",
```

```
    "arn:aws:iot:us-east-1:123456789012:job/*"
            ]
        }
    ]
}
```

Autorización del etiquetado de los recursos de IoT

Para controlar mejor los trabajos y las plantillas de trabajo que puede crear, modificar o usar, puede adjuntarles etiquetas. Estas también le ayudan a determinar la propiedad y a asignar y distribuir los costos, ya que las ubican en grupos de facturación y les asocian etiquetas.

Si un usuario quiere etiquetar sus trabajos o las plantillas de trabajo que ha creado con los Consola de administración de AWS o los AWS CLI, tu política de IAM debe conceder al usuario permisos para etiquetarlos. Para conceder permisos, la política de IAM debe utilizar la acción `iot:TagResource`.

> **Note**
>
> Si la política de IAM no incluye la acción `iot:TagResource`, cualquier `CreateJob` o `CreateJobTemplate` con una etiqueta devolverá un error `AccessDeniedException`.

Si quieres etiquetar tus trabajos o las plantillas de trabajo que has creado con Consola de administración de AWS o la AWS CLI, tu política de IAM debe conceder permiso para etiquetarlos. Para conceder permisos, la política de IAM debe utilizar la acción `iot:TagResource`.

Para obtener información general sobre el etiquetado de recursos, consulte Etiquetar sus recursos AWS IoT.

Ejemplo de política de IAM

Consulte los siguientes ejemplos de políticas de IAM que conceden permisos de etiquetado:

Ejemplo 1

Un usuario que ejecuta el siguiente comando para crear un trabajo y etiquetarlo en un entorno específico.

En este ejemplo, sustituya:

- *region*con las tuyas Región de AWS, comous-east-1.
- *account-id*con tu Cuenta de AWS número, por ejemplo57EXAMPLE833.
- *thing-name*con el nombre de tu actividad de IoT a la que te diriges a los trabajos, por ejemploMyIoTThing.

```
aws iot create-job
    --job-id test_job
    --targets "arn:aws:iot:region:account-id:thing/thingOne"
    --document-source "https://s3.amazonaws.com/amzn-s3-demo-bucket/job-document.json"
    --description "test job description"
    --tags Key=environment,Value=beta
```

Para este ejemplo, debe utilizar la siguiente política de IAM:

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Action": [
            "iot:CreateJob",
            "iot:CreateJobTemplate",
            "iot:TagResource"
        ],
        "Effect": "Allow",
        "Resource": [
           "arn:aws:iot:us-east-1:123456789012:job/*",
           "arn:aws:iot:us-east-1:123456789012:jobtemplate/*"
        ]
    }
}
```

Autorizar tus dispositivos para usar AWS IoT Jobs de forma segura en el plano de los datos

Para autorizar que tus dispositivos interactúen de forma segura con AWS IoT Jobs en el plano de datos, debes usar AWS IoT Core políticas. AWS IoT Core las políticas de empleo son documentos JSON que contienen declaraciones de políticas. Estas políticas también utilizan los elementos Efecto, Acción y Recurso y siguen una convención similar a la de las políticas de IAM. Para obtener

más información sobre los elementos, consulte Referencia de los elementos de las políticas de JSON de IAM en la Guía del usuario de IAM.

Las políticas se pueden usar con los protocolos MQTT y HTTPS y deben usar la autenticación mutua TCP o TLS para autenticar los dispositivos. A continuación, se muestra cómo utilizar estas políticas en los distintos protocolos de comunicación.

> **Warning**
>
> Te recomendamos que no utilices permisos comodín, como `"Action": ["iot:*"]` en tus políticas o AWS IoT Core políticas de IAM. El uso de permisos comodín no es una práctica recomendada de seguridad. Para obtener más información, consulte Políticas de AWS IoT demasiado permisivas.

AWS IoT Core políticas para el protocolo MQTT

AWS IoT Core las políticas del protocolo MQTT le otorgan permisos para usar las acciones de la API MQTT del dispositivo de trabajos. Las operaciones de la API MQTT se utilizan para trabajar con temas MQTT que están reservados para los comandos de trabajos. Para obtener más información sobre estas operaciones de API, consulte Operaciones de la API MQTT de dispositivos de Jobs.

Las políticas MQTT utilizan acciones de política como `iot:Connect`, `iot:Publish`, `iot:Subscribe` y `iot:Receieve` para trabajar con los temas de los trabajos. Estas políticas le permiten conectarse al agente de mensajes, suscribirse a los temas MQTT de trabajos y enviar y recibir mensajes MQTT entre sus dispositivos y la nube. Para obtener más información sobre estas acciones, consulte AWS IoT Core acciones políticas.

Para obtener información sobre los temas de AWS IoT Jobs, consulte. Temas de trabajos

Ejemplo de política MQTT básica

El siguiente ejemplo muestra cómo puede utilizar `iot:Publish` y `iot:Subscribe` para publicar trabajos y ejecuciones de trabajos y suscribirse a ellos.

En el ejemplo, sustituya:

- *`region`*con su Región de AWS, como`us-east-1`.
- *`account-id`*con tu Cuenta de AWS número, por ejemplo`57EXAMPLE833`.

- *thing-name*con el nombre de tu actividad de IoT a la que te diriges a los trabajos, por ejemploMyIoTThing.

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

    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish",
                "iot:Subscribe"
            ],
            "Resource": [
            "arn:aws:iot:us-east-1:123456789012:topic/$aws/events/job/*",
    "arn:aws:iot:us-east-1:123456789012:topic/$aws/events/jobExecution/*",
    "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/thing-123/jobs/*"
            ]
        }
    ]
}
```

AWS IoT Core políticas para el protocolo HTTPS

AWS IoT Core las políticas del plano de datos también pueden utilizar el protocolo HTTPS con el mecanismo de autenticación TLS para autorizar sus dispositivos. En el plano de datos, las políticas utilizan el prefijo `iotjobsdata:` para autorizar las operaciones de la API de trabajos que los dispositivos pueden realizar. Por ejemplo, la acción de política `iotjobsdata:DescribeJobExecution` concede al usuario permiso para usar la API `DescribeJobExecution`.

> **Note**
>
> Las acciones de política del plano de datos deben usar el prefijo `iotjobsdata:`. En el plano de control, las acciones deben usar el prefijo `iot:`. Para ver un ejemplo de política de IAM cuando se utilizan acciones de política tanto del plano de control como del plano de datos,

consulte Ejemplo de política de IAM tanto para el plano de control como para el plano de datos.

Acciones de políticas

En la siguiente tabla se muestra una lista de acciones AWS IoT Core políticas y permisos para autorizar a los dispositivos a utilizar las acciones de la API. Para obtener una lista de las operaciones de API que puede realizar en el plano de datos, consulteAPI HTTP de dispositivos de Jobs.

> **Note**
>
> Estas acciones de política de ejecución de trabajo se aplican únicamente al punto de conexión HTTP TLS. Si utiliza el punto de conexión MQTT, debe utilizar las acciones de política MQTT definidas anteriormente.

AWS IoT Core acciones políticas en el plano de datos

Acción de política	Operación de la API	Tipos de recurso	Description (Descripción)
`iotjobsdata:DescribeJobExecution`	`DescribeJobExecution`	• job • thing	Representa el permiso para recuperar una ejecución de trabajo. El permiso `iotjobsdata:DescribeJobExecution` se comprueba cada vez que se presenta una solicitud para recuperar la ejecución de un trabajo.
`iotjobsdata:GetPendingJobExecutions`	`GetPendingJobExecutions`	thing	Representa el permiso para recuperar la lista de trabajos que no están en un estado final para un objeto. El permiso `iotjobsdata:GetPendingJobExecutions` se comprueba cada vez que se presenta una solicitud para recuperar la lista.

Acción de política	Operación de la API	Tipos de recurso	Description (Descripción)
`iotjobsdata:StartNextPendingJobExecution`	`StartNextPendingJobExecution`	thing	Representa el permiso para obtener e iniciar la próxima ejecución de trabajo pendiente para un objeto, es decir, para actualizar una ejecución de trabajo con un estado `QUEUED` a `IN_PROGRESS` . El permiso `iotjobsdata:StartNextPendingJobExecution` se comprueba cada vez que se presenta una solicitud para iniciar la siguiente ejecución de trabajo pendiente.
`iotjobsdata:UpdateJobExecution`	`UpdateJobExecution`	thing	Representa el permiso para actualizar una ejecución de trabajo. El permiso `iotjobsdata:UpdateJobExecution` se comprueba cada vez que se presenta una solicitud para actualizar el estado de una ejecución de trabajo.

Ejemplo de política básica

A continuación, se muestra un ejemplo de una AWS IoT Core política que otorga permiso para realizar las acciones en las operaciones de la API del plano de datos para cualquier recurso. Puede limitar la política a un recurso específico, como un objeto de IoT. En el ejemplo, sustituya:

- *`region`*con tus Región de AWS , por ejemplo`us-east-1`.
- *`account-id`*con tu Cuenta de AWS número, por ejemplo`57EXAMPLE833`.
- *`thing-name`*con el nombre del IoT, como`MyIoTthing`.

```
{
```

```
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "iotjobsdata:GetPendingJobExecutions",
                "iotjobsdata:StartNextPendingJobExecution",
                "iotjobsdata:DescribeJobExecution",
                "iotjobsdata:UpdateJobExecution"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:iot:us-east-1:123456789012:thing/thing-123"
        }
    ]
}
```

Un ejemplo de cuándo debe usar estas políticas puede ser cuando los dispositivos de IoT usan una política de AWS IoT Core para acceder a una de estas operaciones de API, como el siguiente ejemplo de la API `DescribeJobExecution`:

```
GET /things/thingName/jobs/jobId?
executionNumber=executionNumber&includeJobDocument=includeJobDocument&namespaceId=namespaceId
 HTTP/1.1
```

AWS IoT Límites de puestos

AWS IoT Jobs tiene cuotas de servicio, o límites, que corresponden a la cantidad máxima de recursos u operaciones de servicio para usted Cuenta de AWS.

Temas

- Límites de ejecución de trabajos
- Límites de trabajos activos y simultáneos

Límites de ejecución de trabajos

En esta sección se proporciona información sobre los límites de ejecución de trabajo de AWS IoT Device Management.

> **Note**
>
> Estos límites no forman parte de las Service Quotas que puede encontrar en la documentación de AWS IoT Device Management Service Quotas.

Para obtener información sobre el número de trabajos pendientes, puede utilizar la `GetPendingJobExecutions` API o suscribirse a los temas reservados de MQTT para AWS IoT Jobs and Tipos de notificaciones de trabajo Receiver.

El número de ejecuciones de trabajos pendientes en su cuenta puede variar en función de si tiene habilitada la configuración de programación y utiliza un periodo de mantenimiento periódico.

Número máximo de ejecuciones de trabajos pendientes

Nombre de la API/notificación	Description (Descripción)	Sin configuración de programación	Con configuración de programación
`ListNotification`	Se publica una `ListNotification` cada vez que una ejecución de un trabajo anterior pasa a un estado final o cuando una ejecución de un trabajo nuevo está en cola o cambia a un estado no final. Puede mostrar hasta 15 ejecuciones de trabajos pendientes que tengan el estado `QUEUED` o `IN_PROGRESS` .	10	15 (solo aparecen hasta cinco ejecuciones de trabajos en la `ListNotification` durante el periodo de mantenimiento).
`GetPendingJobExecutions`	Al invocar la API `GetPendingJobExecutions` , se muestra una lista de las ejecuciones de trabajos que aún no se han iniciado y que se pueden iniciar después de la llamada a la API. La API puede devolver hasta un máximo de 10 ejecuciones de trabajos pendientes.	10	15

Nombre de la API/notificación	Description (Descripción)	Sin configuración de programación	Con configuración de programación
	• De las 10 ejecuciones de trabajos pendientes, las que tengan el estado `IN_PROGRESS` se excluirán del resultado. • De las 10 ejecuciones de trabajos pendientes, si los trabajos tienen el estado `SCHEDULED` , se excluirán del resultado.		

Límites de trabajos activos y simultáneos

Esta sección le ayudará a obtener más información sobre los trabajos activos y simultáneos y los límites que se les aplican.

Trabajos activos y su límite

Al crear un trabajo mediante la AWS IoT consola o la `CreateJob` API, el estado del trabajo cambia a`IN_PROGRESS`. Todos los trabajos en curso son trabajos activos y se tienen en cuenta para el límite correspondiente. Esto incluye los trabajos que están desplegando nuevas ejecuciones de trabajos o los que están esperando a que los dispositivos completen la ejecución de sus trabajos. Este límite se aplica tanto a los trabajos continuos como a los instantáneos.

Trabajos simultáneos y límite de simultaneidad

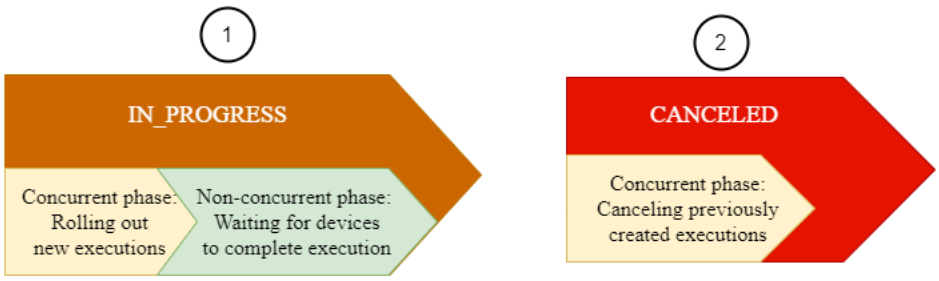
Los trabajos en curso que están implementando nuevas ejecuciones de trabajos o que están cancelando ejecuciones de trabajos creados anteriormente son trabajos simultáneos y se tienen en cuenta a efectos del límite de simultaneidad de trabajos. AWS IoT Los trabajos pueden implementarse y cancelar las ejecuciones de trabajos rápidamente a una velocidad de 1000 dispositivos por minuto. Cada trabajo es `concurrent` y se tiene en cuenta para el límite de simultaneidad de trabajos solo durante un periodo breve. Una vez desplegadas o canceladas las ejecuciones de trabajos, estos dejan de ser simultáneos y no se tienen en cuenta para el límite de simultaneidad de trabajos. Puede utilizar la simultaneidad de trabajos para crear una gran cantidad de trabajos mientras espera a que los dispositivos completen su ejecución.

> **Note**
>
> Si un trabajo con la configuración de programación opcional y el despliegue del documento de trabajo programado para realizarse durante un periodo de mantenimiento alcanza `startTime` seleccionado y está en el límite máximo de simultaneidad de trabajos, ese trabajo programado pasará a un estado `CANCELED`.

Para determinar si un trabajo es simultáneo, puede utilizar la `IsConcurrent` propiedad de un trabajo desde la AWS IoT consola o mediante la API `DescribeJob` o`ListJob`. Este límite se aplica tanto a los trabajos continuos como a los instantáneos.

Para ver los trabajos activos y los límites de simultaneidad de trabajos y otras cuotas de AWS IoT trabajos para usted Cuenta de AWS y solicitar un aumento del límite, consulte los puntos finales y las cuotas de AWS IoT Device Management en. Referencia general de AWS

En el siguiente diagrama se muestra cómo se aplica la simultaneidad de trabajos a los trabajos en curso y a los que se están cancelando.



> **Note**
>
> Los nuevos trabajos que incluyan la opción `SchedulingConfig` mantendrán su estado inicial `SCHEDULED` y se actualizarán a `IN_PROGRESS` al alcanzar el `startTime` seleccionado. Una vez que el nuevo trabajo con la opción `SchedulingConfig` alcance el `startTime` seleccionado y se actualice a `IN_PROGRESS`, se tendrá en cuenta para el límite de trabajos activos y el límite de simultaneidad de trabajos. Los trabajos cuyo estado sea `SCHEDULED` se tendrán en cuenta para el límite de trabajos activos, pero no para el límite de simultaneidad de trabajos.

La siguiente tabla muestra los límites que se aplican a los trabajos activos y simultáneos y las fases simultáneas y no simultáneas de los estados de los trabajos.

Límites de trabajos activos y simultáneos

Estado del trabajo	Fase	Límite de trabajos activos	Límite de simultaneidad de trabajos
`SCHEDULED`	Fase no simultánea: AWS IoT Jobs espera a que la tarea esté programada para iniciar las notificaciones `startTime` de ejecución de la tarea en sus dispositivos. Los trabajos de esta fase solo cuentan para el límite de trabajos activos y tendrán la propiedad `IsConcurrent` establecida en false.	Aplicable	No aplicable
`IN_PROGRESS`	Fase simultánea: AWS IoT Jobs acepta la solicitud de creación de la tarea y comienza a enviar notificaciones de ejecución de la tarea a tus dispositivos. Los trabajos de esta fase son simultáneos, como indica la propiedad `IsConcurrent` establecida en true, y se tienen en cuenta tanto para los trabajos activos como para los límites de simultaneidad de trabajos.	Aplicable	Aplicable
	Fase no simultánea: AWS IoT Jobs espera a que los dispositivos informen de los resultados de la ejecución de sus trabajos. Los trabajos de esta fase solo cuentan para el límite de trabajos activos y tendrán la propiedad `IsConcurrent` establecida en false.	Aplicable	No aplicable
`Canceled`	Fase simultánea: AWS IoT Jobs acepta la solicitud de cancelación de la tarea	No aplicable	Aplicable

Estado del trabajo	Fase	Límite de trabajos activos	Límite de simultaneidad de trabajos
	y comienza a cancelar las ejecuciones de tareas previamente creadas para sus dispositivos. Los trabajos de esta fase son simultáneos y tendrán la propiedad `IsConcurrent` establecida en true. Una vez cancelado el trabajo y sus ejecuciones, el primero deja de ser simultáneo y no se tiene en cuenta para el límite de simultaneidad de trabajos.		

> **Note**
>
> La duración máxima de un periodo de mantenimiento periódico es de 23 horas y 50 minutos.

AWS IoT Device Management comandos

> **Important**
>
> En esta documentación se describe cómo puede utilizar la característica de comandos en AWS IoT Device Management. Para obtener más información sobre cómo usar esta característica para AWS IoT FleetWise, consulte Comandos remotos.
> Usted es el único responsable de implementar los comandos de forma segura y que cumpla con las normativas aplicables. Para obtener más información sobre sus responsabilidades, consulte los términos de servicio de AWS para los servicios de AWS IoT.

Usa AWS IoT Device Management los comandos para enviar una instrucción desde la nube a un dispositivo al que esté conectado AWS IoT. Los comandos se dirigen a un dispositivo a la vez y se pueden usar para aplicaciones de baja latencia y alto rendimiento, como recuperar registros del dispositivo o iniciar un cambio de estado del dispositivo.

El comando es un recurso reutilizable que se administra mediante. AWS IoT Device Management Contiene configuraciones que se aplican antes de publicarse en el dispositivo. Puede predefinir un conjunto de comandos para casos de uso específicos, como encender una bombilla o abrir la puerta de un vehículo.

La función de AWS IoT comandos le permite:

- Cree plantillas de comandos reutilizables con cargas útiles estáticas o dinámicas y ejecútelas en dispositivos específicos.
- Diríjase a dispositivos registrados como AWS IoT cosas o clientes MQTT no registrados.
- Ejecute varios comandos simultáneamente en el mismo dispositivo.
- Habilite las notificaciones de eventos y realice un seguimiento del estado de ejecución a medida que los dispositivos procesan los comandos.

En los temas siguientes se muestra cómo crear comandos, enviarlos al dispositivo y recuperar el estado notificado por el dispositivo.

Temas

- Inicio rápido

- Estado y conceptos de los comandos
- Flujo de trabajo de comandos de
- Creación y administración de comandos
- Inicio y supervisión de las ejecuciones de comandos
- Obsolescencia de un recurso de comandos

Inicio rápido

Complete estos pasos para enviar su primer comando:

1. Requisitos previos: asegúrese de estar incorporado a IoT Core, un dispositivo conectado a IoT Core, tener permisos de IAM para las operaciones de la API de comandos y tener una biblioteca de clientes MQTT instalada en su dispositivo.
2. Crear un comando: defina un comando con una carga útil estática o una plantilla de carga útil que especifique las acciones del dispositivo. Consulte Creación de un recurso de comandos.
3. Identifique su dispositivo: registre su dispositivo como un elemento de IoT o anote su ID de cliente MQTT. Consulte Consideraciones sobre los dispositivos de destino.
4. Configure los permisos: configure las políticas de IAM que permitan a su dispositivo recibir comandos y publicar respuestas.
5. Suscríbase a los temas: configure su dispositivo para suscribirse a los temas de solicitud y respuesta de comandos. Consulte Selección del dispositivo de destino para los comandos y suscripción a los temas de MQTT.
6. Ejecute el comando: inicie la ejecución del comando en el dispositivo de destino. Consulte Inicio de la ejecución de un comando.

Casos de uso comunes:

- Diagnóstico remoto: recupere registros, ejecute diagnósticos y compruebe el estado del dispositivo
- Actualizaciones de configuración: cambie la configuración, actualice los parámetros y cambie las funciones
- Control del dispositivo: Lock/unlock, power on/off cambios de modo

Estado y conceptos de los comandos

Utilice AWS IoT los comandos para enviar instrucciones desde la nube a los dispositivos conectados. Para usar esta característica:

1. Cree un comando con una carga útil que contenga las configuraciones necesarias para ejecutarse en el dispositivo.
2. Especifique el dispositivo de destino que recibirá la carga útil y realizará las acciones.
3. Ejecute el comando en el dispositivo de destino y recupere la información de estado. Para solucionar problemas, consulte los CloudWatch registros.

Para obtener más información acerca de este flujo de trabajo, consulte Flujo de trabajo de comandos de.

Temas

- Conceptos clave de los comandos
- Estados de comando
- Estado de la ejecución del comando

Conceptos clave de los comandos

Los siguientes conceptos clave le ayudan a entender la función de comandos. Los términos se utilizan de forma coherente en toda esta documentación:

- Comando: plantilla reutilizable que define las instrucciones del dispositivo
- Ejecución: instancia de un comando que se ejecuta en un dispositivo
- Nombre de la cosa: identificador de dispositivos registrados en el registro de IoT
- ID de cliente: identificador MQTT para dispositivos no registrados
- Carga útil: los datos de instrucciones que se envían a los dispositivos
- Tema: Canal MQTT para la comunicación de comandos

Comandos

Los comandos son instrucciones que se envían desde la nube a sus dispositivos de IoT como mensajes MQTT. Tras recibir la carga útil, los dispositivos procesan las instrucciones y toman las

medidas correspondientes, como modificar los ajustes de configuración, transmitir las lecturas de los sensores o cargar los registros. Luego, los dispositivos devuelven los resultados a la nube, lo que permite la supervisión y el control remotos.

Namespace

Al crear un comando, especifique su espacio de nombres. Para AWS IoT Device Management los comandos, utilice el espacio de `AWS-IoT` nombres predeterminado y proporcione una carga útil o una plantilla de carga útil. Para los AWS IoT FleetWise comandos, usa el espacio de nombres. `AWS-IoT-FleetWise` Para obtener más información, consulte Comandos remotos en la Guía para AWS IoT FleetWise desarrolladores.

Carga útil

Al crear un comando, proporcione una carga estática que defina las acciones que debe realizar el dispositivo. La carga útil puede utilizar cualquier formato compatible. Para garantizar que los dispositivos interpreten correctamente la carga útil, recomendamos especificar el tipo de formato de la carga útil. Los dispositivos que utilizan el MQTT5 protocolo pueden seguir el estándar MQTT para identificar el formato. Los indicadores de formato para JSON o CBOR están disponibles en el tema de solicitud de comandos.

Plantilla de carga útil

Una plantilla de carga útil define una carga útil de comandos con marcadores de posición que generan diferentes cargas útiles en tiempo de ejecución en función de los valores de los parámetros que proporciones. Por ejemplo, en lugar de crear cargas útiles independientes para diferentes valores de temperatura, crea una plantilla con un marcador de temperatura y especifica el valor durante la ejecución. Esto elimina el mantenimiento de varias cargas útiles similares.

Dispositivo de destino

Para ejecutar un comando, especifique un dispositivo de destino utilizando su nombre (en el caso de los dispositivos registrados AWS IoT) o el ID de cliente MQTT (en el caso de los dispositivos no registrados). El ID de cliente es un identificador único definido en el MQTT protocolo utilizado para conectar los dispositivos. AWS IoT Para obtener más información, consulte Consideraciones sobre los dispositivos de destino.

Temas de comandos

Antes de ejecutar un comando, los dispositivos deben suscribirse al tema de solicitud de comandos. Al ejecutar un comando, la carga útil se envía al dispositivo relacionado con este tema. Tras la ejecución, los dispositivos publican los resultados y el estado en el tema de respuesta al comando. Para obtener más información, consulte Temas de comandos.

Ejecución de comandos

Una ejecución es una instancia de un comando que se ejecuta en un dispositivo de destino. Al iniciar una ejecución, la carga útil se entrega al dispositivo y se genera un identificador de ejecución único. El dispositivo ejecuta el comando e informa a. AWS IoT La lógica del dispositivo determina el comportamiento de ejecución y los informes sobre el estado de los temas reservados.

Condiciones de valor de los parámetros

Al crear comandos con plantillas de carga útil, defina las condiciones de valor para validar los valores de los parámetros antes de la ejecución. Las condiciones de valor garantizan que los parámetros cumplan los requisitos, lo que evita las ejecuciones no válidas.

Operadores compatibles por CommandParameterValuetipo

Tipos numéricos (INTEGER, LONG, DOUBLE, UNSIGNEDLONG)

- `EQUALS`- El valor debe ser igual al número especificado
- `NOT_EQUALS`- El valor no debe ser igual al número especificado
- `GREATER_THAN`- El valor debe ser mayor que el número especificado
- `GREATER_THAN_EQUALS`- El valor debe ser mayor o igual al número especificado
- `LESS_THAN`- El valor debe ser inferior al número especificado
- `LESS_THAN_EQUALS`- El valor debe ser menor o igual al número especificado
- `IN_RANGE`- El valor debe estar dentro del rango especificado (incluido)
- `NOT_IN_RANGE`- El valor debe estar fuera del rango especificado (incluido)
- `IN_SET`- El valor debe coincidir con uno de los números especificados
- `NOT_IN_SET`- El valor no debe coincidir con ninguno de los números especificados

Tipo de cadena (STRING)

- `EQUALS`- El valor debe ser igual a la cadena especificada
- `NOT_EQUALS`- El valor no debe ser igual a la cadena especificada
- `IN_SET`- El valor debe coincidir con una de las cadenas especificadas
- `NOT_IN_SET`- El valor no debe coincidir con ninguna de las cadenas especificadas

Tipo booleano

- No se admiten condiciones de valores

Tipo binario

- No se admiten condiciones de valor

Ejemplo: comando de control de temperatura

```
{
  "commandId": "SetTemperature",
  "namespace": "AWS-IoT",
  "payloadTemplate": "{\"temperature\":
 \"${aws:iot:commandexecution::parameter:temperature}\"}",
  "parameters": [
    {
      "name": "temperature",
      "type": "INTEGER",
      "valueConditions": [
        {
          "comparisonOperator": "IN_RANGE",
          "operand": {
            "numberRange": {
              "min": "60",
              "max": "80"
            }
          }
        }
      ]
    }
  ]
}
```

En este ejemplo, el `temperature` parámetro debe estar entre 60 y 80 (ambos incluidos). Las solicitudes de ejecución con valores fuera de este rango no se validan.

> **Note**
>
> Las condiciones de los valores se evalúan al invocar la StartCommandExecution API. Las validaciones fallidas devuelven un error e impiden la creación de la ejecución.

Prioridad y evaluación del valor del parámetro

Al iniciar la ejecución de comandos con plantillas de carga útil, los valores de los parámetros se resuelven con la siguiente prioridad:

1. Parámetros de la solicitud de ejecución: los valores proporcionados en la `StartCommandExecution` solicitud tienen la máxima prioridad
2. Valores predeterminados del comando: si no se proporciona un parámetro en la solicitud de ejecución, `defaultValue` se utiliza el parámetro
3. Sin valor: si no se proporciona ninguno, se produce un error en la ejecución, ya que es el parámetro necesario para generar la solicitud de ejecución

Las condiciones de valor se evalúan en función del valor final del parámetro obtenido anteriormente en función de la prioridad y antes de la creación de la ejecución. Si la validación falla, la solicitud de ejecución devuelve un error.

Ejemplo: SetTemperature comando con **`defaultValue`**

```
{
  "parameters": [
    {
      "name": "temperature",
      "type": "INTEGER",
      "defaultValue": {"I": 72},
      "valueConditions": [
        {
          "comparisonOperator": "IN_RANGE",
          "operand": {"numberRange": {"min": "60", "max": "80"}}
        }
      ]
    }
  ]
}
```

Al iniciar la ejecución:

- Si proporciona `"temperature": {"I": 75}` la solicitud, se utiliza 75
- Si omite el parámetro de temperatura, se utilizará el valor predeterminado 72
- Ambos valores se validan con respecto a la condición de rango [60,80]

Estados de comando

Los comandos de su Cuenta de AWS ordenador pueden estar en uno de estos tres estados: disponibles, obsoletos o pendientes de eliminación.

Disponible

Una vez creado correctamente, un comando pasa al estado Disponible y se puede ejecutar en los dispositivos.

Obsoleto

Marque los comandos como obsoletos cuando ya no sean necesarios. Los comandos obsoletos no pueden iniciar nuevas ejecuciones, pero las ejecuciones pendientes siguen completándose. Para habilitar nuevas ejecuciones, restaure el comando al estado Disponible.

Eliminación pendiente

Al marcar un comando para su eliminación, se elimina automáticamente si está obsoleto durante más tiempo que el tiempo de espera máximo (predeterminado: 12 horas). Esta acción es permanente. Si no está en desuso o ha pasado menos del tiempo de espera, el comando pasa al estado de eliminación pendiente y se elimina una vez transcurrido el tiempo de espera.

Estado de la ejecución del comando

Al iniciar una ejecución en un dispositivo de destino, entra en `CREATED` estado y puede pasar a otros estados en función de los informes del dispositivo. Puede recuperar la información de estado y realizar un seguimiento de las ejecuciones.

> **Note**
>
> Puede ejecutar varios comandos simultáneamente en un dispositivo. Utilice el control de simultaneidad para limitar las ejecuciones por dispositivo y evitar la sobrecarga. Para ver el número máximo de ejecuciones simultáneas por dispositivo, consulta las cuotas de AWS IoT Device Management comandos.

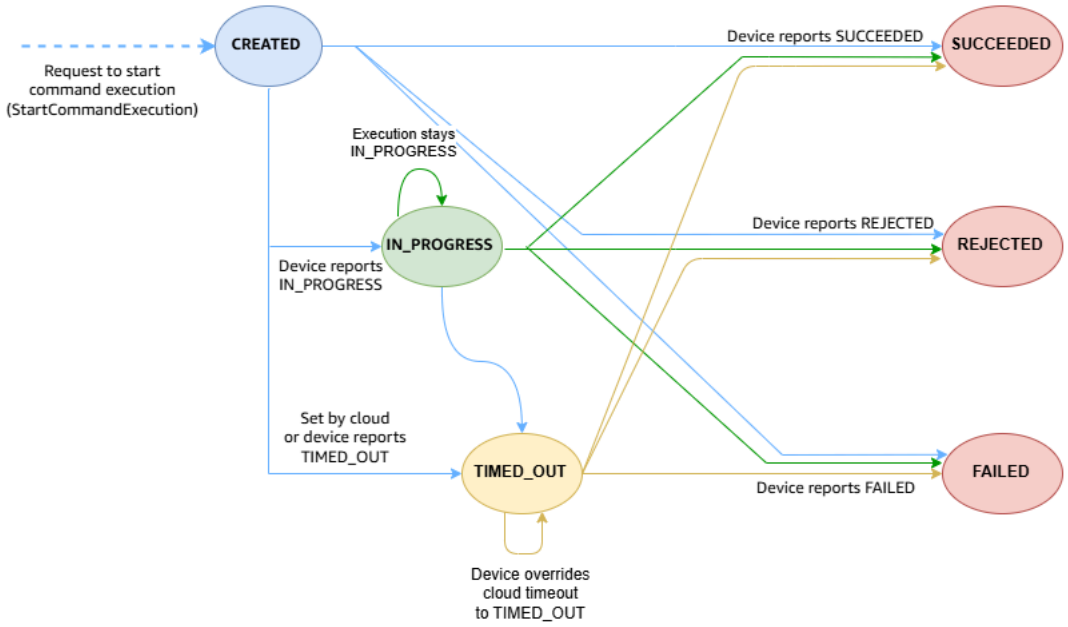
La siguiente tabla muestra los estados de ejecución y sus transiciones en función del progreso de la ejecución.

Estado y origen de la ejecución de los comandos

Estado de la ejecución del comando	¿Iniciado por un dispositivo o la nube?	¿Ejecución final?	Transiciones de estado permitidas
`CREATED`	Cloud	No	• IN_PROGRESS • SUCCEEDED • ERROR • REJECTED • TIMED_OUT
`IN_PROGRESS`	Dispositivo	No	• IN_PROGRESS • SUCCEEDED • ERROR • REJECTED • TIEMPO DE ESPERA
`TIMED_OUT`	Dispositivo y nube	No	• SUCCEEDED • ERROR • REJECTED • TIEMPO DE ESPERA
`SUCCEEDED`	Dispositivo	Sí	No aplicable
`FAILED`	Dispositivo	Sí	No aplicable
`REJECTED`	Dispositivo	Sí	No aplicable

Los dispositivos pueden publicar actualizaciones de estado y resultados en cualquier momento mediante comandos reservados a los temas de MQTT. Para proporcionar un contexto adicional, los dispositivos pueden usar `reasonCode` `reasonDescription` campos del `statusReason` objeto.

El siguiente diagrama muestra las transiciones del estado de ejecución.



> **Note**
>
> Cuando no AWS IoT detecta ninguna respuesta del dispositivo dentro del período de tiempo de espera, se establece `TIMED_OUT` como un estado temporal que permite reintentos y cambios de estado. Si el dispositivo informa de forma explícita`TIMED_OUT`, se convierte en un estado terminal sin más transiciones. Para obtener más información, consulte Ejecuciones de comandos no finales.

En las siguientes secciones se describen las ejecuciones terminales y no terminales y sus estados.

Temas

- Ejecuciones de comandos no finales
- Ejecuciones de comandos finales

Ejecuciones de comandos no finales

Una ejecución no es terminal si puede aceptar actualizaciones de dispositivos. Las ejecuciones que no son terminales se consideran activas. Los siguientes estados no son terminales:

- CREATED

Cuando inicias una ejecución desde la AWS IoT consola o utilizas la `StartCommandExecution` API, las solicitudes correctas cambian el estado a. `CREATED` A partir de este estado, las ejecuciones pueden pasar a cualquier otro estado terminal o no terminal.

- IN_PROGRESS

  Tras recibir la carga útil, los dispositivos pueden empezar a ejecutar instrucciones y realizar acciones específicas. Durante la ejecución, los dispositivos pueden publicar las respuestas al tema de respuesta de los comandos y actualizar el estado a`IN_PROGRESS`. Desde`IN_PROGRESS`, las ejecuciones pueden pasar a cualquier estado terminal o no terminal, excepto`CREATED`.

  > **Note**
  >
  > La `UpdateCommandExecution` API se puede invocar varias veces con el `IN_PROGRESS` estado. Especifique detalles de ejecución adicionales mediante el `statusReason` objeto.

- TIMED_OUT

  Tanto la nube como el dispositivo pueden activar este estado. Las ejecuciones en `CREATED` nuestro `IN_PROGRESS` estado pueden cambiar a `TIMED_OUT` uno por los siguientes motivos:

  - Tras enviar el comando, se inicia un temporizador. Si el dispositivo no responde dentro del tiempo especificado, la nube cambia de estado a`TIMED_OUT`. En este caso, la ejecución no es terminal.
  - El dispositivo puede cambiar el estado a cualquier estado de terminal o informar de un tiempo de espera y establecer el estado en. `TIMED_OUT` En este caso, el estado permanece`TIMED_OUT`, pero los campos del `StatusReason` objeto cambian en función de la información del dispositivo. La ejecución pasa a ser terminal.

  Para obtener más información, consulte Valor de tiempo de espera y estado de ejecución de `TIMED_OUT`.

Ejecuciones de comandos finales

Una ejecución se convierte en terminal cuando ya no acepta actualizaciones de los dispositivos. Los siguientes estados son finales. Las ejecuciones pueden pasar a estados de terminal desde cualquier estado no terminal:`CREATED`,`IN_PROGRESS`, o. `TIMED_OUT`

- SUCCEEDED

Si el dispositivo completa la ejecución correctamente, puede publicar una respuesta en el tema de respuesta del comando y actualizar el estado a. `SUCCEEDED`

- ERROR

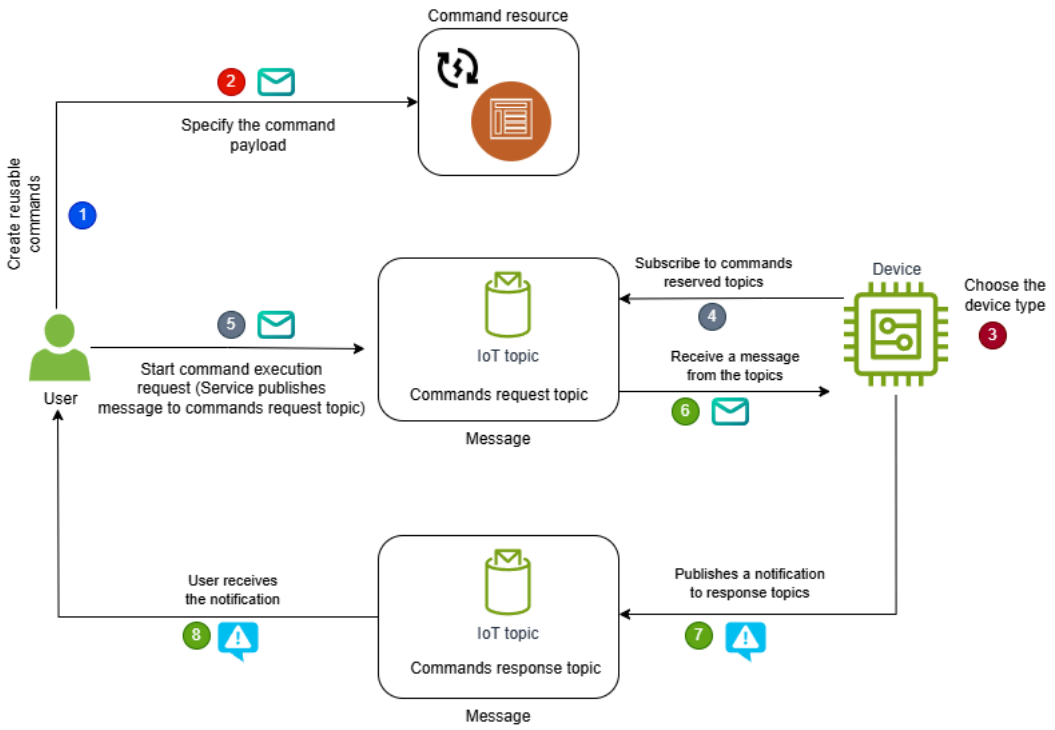
  Cuando un dispositivo no completa la ejecución, puede publicar una respuesta en el tema de respuesta al comando y actualizar el estado a`FAILED`. Utilice los `reasonDescription` campos `reasonCode` y del `statusReason` objeto, o CloudWatch los registros, para solucionar errores.

- REJECTED

  Cuando un dispositivo recibe una solicitud no válida o incompatible, puede invocar la `UpdateCommandExecution` API con el estado. `REJECTED` Usa los `reasonDescription` campos `reasonCode` y del `statusReason` objeto, o CloudWatch los registros, para solucionar problemas.

Flujo de trabajo de comandos de

Este flujo de trabajo muestra cómo los dispositivos interactúan con AWS IoT Device Management los comandos. Todas las solicitudes de la API HTTP utilizan credenciales Sigv4 para firmarlas.



Información general sobre el flujo de trabajo

- Creación y administración de comandos
- Selección del dispositivo de destino para los comandos y suscripción a los temas de MQTT
- Inicio y supervisión de las ejecuciones de comandos en el dispositivo de destino
- (Opcional) Habilitación de las notificaciones para los eventos de comandos

Creación y administración de comandos

Para crear y administrar comandos para los dispositivos, siga los pasos que se indican a continuación.

1. Creación de un recurso de comandos

   Cree un comando desde el centro de comandos o mediante la `CreateCommand`API.

2. Especificación de la carga útil

   Proporcione una carga útil en cualquier formato. Especifique el tipo de contenido para garantizar una interpretación correcta del dispositivo.

   En el caso de los comandos dinámicos con plantillas de carga útil, la carga útil final se genera en tiempo de ejecución con los parámetros proporcionados. Las plantillas solo admiten el formato JSON, pero la carga útil generada se puede enviar como JSON o CBOR.

3. (Opcional) Administración de los comandos creados

   Actualice el nombre para mostrar y la descripción después de la creación. Marque los comandos como obsoletos cuando ya no sean necesarios o elimínelos permanentemente. Para modificar la información de carga útil, crea un comando nuevo.

Selección del dispositivo de destino para los comandos y suscripción a los temas de MQTT

Elija su dispositivo de destino y configure los temas de MQTT para recibir comandos y publicar respuestas.

1. Selección del dispositivo de destino para el comando

   Elija un dispositivo de destino para recibir y ejecutar el comando. Utilice un nombre de cosa para los dispositivos registrados o un ID de cliente para los dispositivos no registrados. Para obtener más información, consulte Consideraciones sobre los dispositivos de destino.

2. Configure la política de AWS IoT dispositivos

   Configure una política de IAM que conceda permisos para recibir ejecuciones y publicar actualizaciones. Consulte Ejemplo de política de IAM para ver ejemplos de políticas.

3. Establecimiento de una conexión con MQTT

   Conecte los dispositivos al agente de mensajes y suscríbase a los temas de solicitud y respuesta. Los dispositivos necesitan `iot:Connect` permiso. Busque el punto final de su plano de datos mediante el comando `DescribeEndpoint` API o `describe-endpoint` CLI:

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

   Al ejecutar este comando, se devolverá el punto de conexión del plano de datos específico de la cuenta, tal como se muestra a continuación.

   ```
   account-specific-prefix.iot.region.amazonaws.com
   ```

4. Suscríbase a los temas de comandos

   Suscríbase al tema de solicitud de comandos. Al iniciar una ejecución, el agente de mensajes publica la carga útil de este tema. El dispositivo recibe y procesa el comando.

   (Opcional) Suscríbase a los temas de respuesta (`acceptedorejected`) para recibir confirmación de si el servicio en la nube ha aceptado o rechazado la respuesta del dispositivo.

   En este ejemplo, sustituya:

   - *`<device>`* con `thing` o `client` en función de si el dispositivo al que se dirige se ha registrado como un objeto de IoT o se ha especificado como un cliente de MQTT.
   - *`<DeviceID>`* con el identificador único del dispositivo de destino. Este ID puede ser el ID único de cliente de MQTT o el nombre de un objeto.

> **Note**
>
> Si el tipo de carga útil no es JSON o CBOR, es posible que el *<PayloadFormat>* campo no esté presente en el tema de solicitud de comandos. Para obtener el formato de carga útil, le recomendamos que lo utilice MQTT5 para obtener la información de formato de los encabezados de los mensajes de MQTT. Para obtener más información, consulte Temas de comandos.

```
$aws/commands/<devices>/<DeviceID>/executions/+/request/<PayloadFormat>
$aws/commands/<devices>/<DeviceID>/executions/+/response/accepted/<PayloadFormat>
$aws/commands/<devices>/<DeviceID>/executions/+/response/rejected/<PayloadFormat>
```

Inicio y supervisión de las ejecuciones de comandos en el dispositivo de destino

Una vez creados los comandos y especificado los destinos del comando, puede iniciar la ejecución en el dispositivo de destino realizando los siguientes pasos.

1. Inicio de la ejecución de comandos en el dispositivo de destino

   Inicie la ejecución desde el centro de comandos o utilice la `StartCommandExecution` API con el punto final específico de su cuenta. Úselo `iot:Data-ATS` para doble pila (IPv4/IPv6) o `iot:Jobs` solo para. IPv4

   La API publica la carga útil en el tema de solicitud de comandos.

   > **Note**
   >
   > Si el dispositivo está desconectado y utiliza sesiones persistentes de MQTT, el comando espera en el intermediario de mensajes. Cuando el dispositivo se vuelva a conectar antes de que se agote el tiempo de espera, puede procesar el comando y publicar los resultados. Si se agota el tiempo de espera, se agota el tiempo de espera de la ejecución y se descarta la carga útil.

2. Actualización del resultado de la ejecución de comandos

   El dispositivo recibe la carga útil, procesa el comando, realiza las acciones especificadas y publica los resultados en el tema de respuesta a los comandos mediante una API basada en `UpdateCommandExecution` MQTT. Si se ha suscrito a los temas aceptados y rechazados, el dispositivo recibe la confirmación de si el servicio en la nube ha aceptado o rechazado la respuesta.

   Según lo que haya especificado en el tema de la solicitud, *<devices>* pueden ser cosas o clientes, y *<DeviceID>* puede ser su nombre o el AWS IoT ID de cliente de MQTT.

   > **Note**
   >
   > Solo *<PayloadFormat>* puede ser JSON o CBOR en el tema de respuesta a los comandos.

   ```
   $aws/commands/<devices>/<DeviceID>/executions/<ExecutionId>/
   response/<PayloadFormat>
   ```

3. (Opcional) Recuperación del resultado de la ejecución de comandos

   Recupere los resultados de la ejecución desde la AWS IoT consola o mediante`GetCommandExecution`. El dispositivo debe publicar los resultados en el tema de respuesta a los comandos para obtener la información más reciente. Vea detalles adicionales, incluida la hora de la última actualización, el resultado y la hora de finalización.

(Opcional) Habilitación de las notificaciones para los eventos de comandos

Suscríbase a los eventos de Commands para recibir notificaciones cuando cambie el estado de ejecución.

1. Creación de una regla del tema

   Suscríbase al tema de eventos de comandos para ver las notificaciones de cambios de estado. Cree una regla temática para enrutar los datos del dispositivo a otros AWS IoT servicios AWS Lambda, como Amazon SQS y AWS Step Functions, mediante la AWS IoT consola o. Crear una AWS IoT regla

En este ejemplo, sustituya *<CommandID>* por el identificador del comando del que desea recibir notificaciones y *<CommandExecutionStatus>* por el estado de la ejecución del comando.

```
$aws/events/commandExecution/<CommandID>/<CommandExecutionStatus>
```

> **Note**
>
> Para recibir notificaciones de todos los comandos y estados de ejecución de los comandos, puede utilizar caracteres comodín y suscribirse al tema siguiente.

```
$aws/events/commandExecution/+/#
```

2. Recepción y procesamiento de los eventos de comandos

   Administre los comandos, envíe notificaciones y cree aplicaciones utilizando los eventos suscritos.

En el siguiente código se muestra un ejemplo de carga útil para las notificaciones de eventos de comandos que recibirá.

```
{
    "executionId": "2bd65c51-4cfd-49e4-9310-d5cbfdbc8554",
    "status":"FAILED",
    "statusReason": {
         "reasonCode": "DEVICE_TOO_BUSY",
         "reasonDescription": ""
    },
    "eventType": "COMMAND_EXECUTION",
    "commandArn":"arn:aws:iot:us-east-1:123456789012:command/0b9d9ddf-
e873-43a9-8e2c-9fe004a90086",
    "targetArn":"arn:aws:iot:us-east-1:123456789012:thing/5006c3fc-
de96-4def-8427-7eee36c6f2bd",
    "timestamp":1717708862107
}
```

Creación y administración de comandos

Usa AWS IoT Device Management los comandos para configurar acciones remotas reutilizables o envía instrucciones inmediatas a los dispositivos. Cree y gestione comandos desde la AWS IoT consola o mediante el AWS CLI.

Creación y administración de operaciones de comandos

- Creación de un recurso de comandos
- Recuperación de información sobre un comando
- Enumere los comandos de su Cuenta de AWS
- Actualización de un recurso de comandos
- Obsolescencia o restauración de un recurso de comandos
- Eliminación de un recurso de comandos

Creación de un recurso de comandos

Proporcione la siguiente información al crear un comando:

- Información general

  Proporcione un identificador de comando único para identificar el comando cuando lo ejecute en los dispositivos de destino. Si lo desea, especifique un nombre para mostrar, una descripción y etiquetas para la administración.

- Carga útil

  En el caso de los comandos estáticos, proporciona una carga útil que defina las acciones del dispositivo. Especifique el tipo de formato de carga útil para una interpretación correcta del dispositivo.

  Para ver los comandos dinámicos, consulte el atributo de plantilla Payload.

- Plantilla de carga útil

  Para los comandos dinámicos, proporciona una plantilla de carga útil con marcadores de posición y parámetros. Si lo desea, proporcione condiciones`defaultValue`. AWS IoT Device Management Los comandos sustituyen a los marcadores de posición en tiempo de ejecución. Los parámetros que faltan utilizan su DefaultValue. Todos los valores deben cumplir las condiciones definidas.

Se admiten los siguientes tipos de marcadores de posición que distinguen mayúsculas de minúsculas:

- `${aws:iot:commandexecution::parameter:`*`parameter1`*`}`— Un marcador de posición para el valor de un parámetro con el nombre. `parameter1`
- `${aws:iot:commandexecution::executionTimeoutSec}`— Un marcador de posición para el parámetro de tiempo de espera de ejecución del comando que se proporciona en tiempo de ejecución.

Temas de comandos y carga útil

Comandos reservados Los temas utilizan un formato basado en el tipo de formato de carga útil.

- Para nuestros `application/json` tipos de `application/cbor` contenido, utilice este tema de solicitud:

```
$aws/commands/<devices>/<DeviceID>/executions/+/request/<PayloadFormat>
```

- Para otros tipos de contenido o formatos no especificados, utilice este tema de solicitud. El formato de carga útil aparece en el encabezado del mensaje MQTT.

```
$aws/commands/<devices>/<DeviceID>/executions/+/request
```

El tema de respuesta a los comandos utiliza `json` o `cbor` formatea independientemente del tipo de carga útil. *`<PayloadFormat>`*debe ser `json` o`cbor`:

```
$aws/commands/<devices>/<DeviceID>/executions/<ExecutionId>/response/<PayloadFormat>
```

Creación de un recurso de comandos (consola)

En las siguientes secciones se describen las consideraciones sobre el formato de carga útil y la creación de comandos desde la consola.

Temas

- Formato de carga útil de comandos estáticos
- Formato de carga útil de comandos dinámicos
- Cómo crear un comando (consola)

Formato de carga útil de comandos estáticos

La carga útil admite cualquier formato de hasta 32 KB. Especifique el tipo de formato de carga útil para una interpretación segura y correcta del dispositivo.

Especifique el tipo de formato de carga útil mediante el `type/subtype` formato (p. ej., `application/json` o`application/cbor`). Predeterminado: `application/octet-stream`. Consulta los tipos MIME más comunes para ver los formatos compatibles.

Formato de carga útil de comandos dinámicos

La plantilla de carga útil debe ser un archivo JSON válido con al menos un marcador de posición (32 KB como máximo).

En el caso `AwsJsonSubstitution` del preprocesador, AWS IoT Device Management Commands envía las notificaciones en formato JSON o CBOR en función de la configuración del preprocesador.

Cómo crear un comando (consola)

Para crear un comando desde la consola, vaya al Centro de comandos y siga estos pasos:

1. Seleccione Crear comando.
2. Especifique un ID de comando único.
3. (Opcional) Especifique el nombre para mostrar, la descripción y las etiquetas.
4. Cargue el archivo de carga útil que contiene las acciones del dispositivo. Especifique el tipo de formato de carga útil para una interpretación correcta del dispositivo.
5. (Opcional) En el caso de las plantillas de carga útil JSON con marcadores de posición, los parámetros se rellenan previamente en la tabla integrada para su edición.
6. (Opcional) Configure el tipo de valor del parámetro (obligatorio), el valor predeterminado y las condiciones.
7. Seleccione Crear comando.

Creación de un recurso de comandos (CLI)

Utilice el comando `CreateCommand`API o `create-command`CLI para crear un comando.

Temas

- Carga útil de comandos
- Ejemplo de política de IAM
- Ejemplo de creación de comandos estáticos
- Ejemplo de creación dinámica de comandos

Carga útil de comandos

Proporcione una carga útil estática o una plantilla de carga útil. Las cargas útiles estáticas están codificadas en base64. En el caso de las plantillas de carga útil, la carga útil final se genera en tiempo de ejecución mediante valores de parámetros. Los dispositivos procesan la carga útil y realizan acciones específicas. Especifique el tipo de contenido de la carga útil para una correcta recepción del dispositivo.

> **Note**
>
> Las cargas útiles no se pueden modificar después de la creación del comando. Crea un comando nuevo para modificar la carga útil.

Ejemplo de política de IAM

Antes de usar esta operación de la API, asegúrese de que la política de IAM le autorice a realizar esta acción en el dispositivo. A continuación, se muestra un ejemplo de una política de IAM que permite a un usuario realizar la acción `CreateCommand`.

En este ejemplo, sustituya:

- *`region`* con la Región de AWS, como, por ejemplo, *`us-east-1`*.
- *`account-id`* con el número de la Cuenta de AWS , como, por ejemplo *`123456789012`*.
- *`command-id`*con un identificador único para el ID de AWS IoT comando, como*`LockDoor`*. Si desea enviar más de un comando, puede especificarlos en la sección Recursos de la política de IAM.

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

```
        "Action": "iot:CreateCommand",
        "Effect": "Allow",
        "Resource": "arn:aws:iot:us-east-1:123456789012:command/command-id"
    }
}
```

Ejemplo de creación de comandos estáticos

El siguiente ejemplo muestra cómo se puede crear un comando estático. En función de la aplicación, sustituya:

- *<command-id>* por un identificador único para el comando. Por ejemplo, para bloquear la puerta de su casa, puede especificarlo*LockDoor*. Le recomendamos que utilice UUID. También puede utilizar caracteres alfanuméricos, “-” y “_”.
- (Opcional) *<display-name>* y *<description>*, que son campos opcionales que puede usar para proporcionar un nombre descriptivo y una descripción significativa para el comando, como, por ejemplo *Lock the doors of my home*.
- `namespace`, que puede utilizar para especificar el espacio de nombres del comando. Debe ser `AWS-IoT`. Para obtener información sobre el uso de esta función AWS IoT FleetWise, consulte Comandos remotos
- `payload` incluye información sobre la carga útil que desea utilizar al ejecutar el comando y su tipo de contenido.

```
aws iot create-command \
    --command-id <command-id> \
    --display-name <display-name> \
    --description <description> \
    --namespace AWS-IoT \
    --payload
 '{"content":"eyAibWVzc2FnZSI6ICJIZWxsbyBJb1QiIH0=","contentType":"application/json"}'
```

La ejecución de este comando genera una respuesta que incluye el ID y el ARN (nombre de recurso de Amazon) del comando. Por ejemplo, si ha especificado el comando *LockDoor* durante la creación, a continuación se muestra un ejemplo del resultado de la ejecución del comando.

```
{
    "commandId": "LockDoor",
```

```
    "commandArn": "arn:aws:iot:us-east-1:123456789012:command/LockDoor"
}
```

Ejemplo de creación dinámica de comandos

El siguiente ejemplo muestra cómo se puede crear un comando dinámico. En función de la aplicación, sustituya:

- *<command-id>* por un identificador único para el comando. Por ejemplo, para establecer el estado de alimentación de la luz, puede especificarlo*Light_Power_Status*. Le recomendamos que utilice UUID. También puede utilizar caracteres alfanuméricos, “-” y “_”.
- (Opcional) *<display-name>* y *<description>*, que son campos opcionales que puede usar para proporcionar un nombre descriptivo y una descripción significativa para el comando, como, por ejemplo *Turn a light ON or OFF*.
- `namespace`, que puede utilizar para especificar el espacio de nombres del comando. Debe ser `AWS-IoT`. Para obtener información sobre el uso de esta función AWS IoT FleetWise, consulte Comandos remotos
- `payloadTemplate`contiene la plantilla de reproducción en formato JSON con marcadores de posición.
- `preprocessor`contiene la configuración del preprocesador que determina cómo se debe procesar la plantilla de carga útil.
- `mandatoryParameter`contiene los parámetros correspondientes a los marcadores de posición de la plantilla de carga útil, su tipo, los valores predeterminados y las condiciones.

```
aws iot create-command \
    --command-id Light_Power_Status \
    --description "Turn a light ON or OFF" \
    --namespace AWS-IoT \
    --payload-
template '{"powerStatus":"${aws:iot:commandexecution::parameter:powerStatus}"}' \
    --preprocessor awsJsonSubstitution={outputFormat=JSON} \
    --mandatory-parameters "name=powerStatus, defaultValue={S=OFF},
 valueConditions=[{comparisonOperator=IN_SET, operand={strings=['ON','OFF']}}]"
```

La ejecución de este comando genera una respuesta que incluye el ID y el ARN (nombre de recurso de Amazon) del comando. Por ejemplo, si ha especificado el comando *Light_Power_Status*

durante la creación, a continuación se muestra un ejemplo del resultado de la ejecución del comando.

```
{
    "commandId": "Light_Power_Status",
    "commandArn": "arn:aws:iot:us-east-1:123456789012:command/Light_Power_Status"
}
```

Recuperación de información sobre un comando

Tras crear un comando, puede recuperar información sobre este desde la consola de AWS IoT y mediante la AWS CLI. Puede obtener la siguiente información.

- El ID del comando, el nombre de recurso de Amazon (ARN) y cualquier nombre y descripción visibles que haya especificado para el comando.
- El estado del comando que indica si un comando está disponible para ejecutarse en el dispositivo de destino o si se va a quedar obsoleto o se va a eliminar.
- La carga útil o la plantilla de carga útil que has proporcionado.
- El preprocesador que ha proporcionado.
- Los parámetros obligatorios que ha proporcionado.
- El momento en que se creó el comando y se actualizó por última vez.

Recuperación de un recurso de comandos (consola)

Para recuperar un comando de la consola, vaya al centro de comandos de la AWS IoT consola y, a continuación, elija el comando que creó para ver sus detalles.

Además de los detalles del comando, puede ver el historial de comandos, que proporciona información sobre las ejecuciones del comando en el dispositivo de destino. Tras ejecutar este comando en el dispositivo, encontrará información sobre las ejecuciones en esta pestaña.

Recuperación de un recurso de comandos (CLI)

Usa la operación de la API del plano de control `GetCommand`HTTP o el `get-command` AWS CLI comando para recuperar información sobre un recurso de comando. Debe haber creado ya el comando mediante la solicitud de la API `CreateCommand` o de la CLI `create-command`.

Ejemplo de política de IAM

Antes de usar esta operación de la API, asegúrese de que la política de IAM le autorice a realizar esta acción en el dispositivo. A continuación, se muestra un ejemplo de una política de IAM que permite a un usuario realizar la acción `GetCommand`.

En este ejemplo, sustituya:

- *`region`* con la Región de AWS, como, por ejemplo, `us-east-1`.
- *`account-id`* con el número de la Cuenta de AWS , como, por ejemplo *`123456789023`*.
- *`command-id`*con su identificador de comando AWS IoT único, como *`LockDoor`* o*`Light_Power_Status`*. Si desea recuperar más de un comando, puede especificarlos en la sección Recursos de la política de IAM.

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Action": "iot:GetCommand",
        "Effect": "Allow",
        "Resource": "arn:aws:iot:us-east-1:123456789012:command/command-id"
    }
}
```

Ejemplo de recuperación de un comando (AWS CLI)

El siguiente ejemplo muestra cómo recuperar información sobre un comando mediante `get-command` AWS CLI. En función de la aplicación, sustituya *`<command-id>`* por el identificador del comando para el que desea recuperar la información. Puede obtener esta información de la respuesta de la CLI `create-command`.

```
aws iot get-command --command-id <command-id>
```

Al ejecutar este comando, se genera una respuesta que incluye información sobre el comando, la carga útil y el momento de creación y de la última actualización. También proporciona información que indica si un comando ha quedado obsoleto o se va a eliminar.

En el siguiente código se muestra una respuesta de ejemplo.

```
{
    "commandId": "LockDoor",
    "commandArn": "arn:aws:iot:<region>:<account>:command/LockDoor",
    "namespace": "AWS-IoT",
    "payload":{
        "content": "eyAibWVzc2FnZSI6ICJIZWxsbyBJb1QiIH0=",
        "contentType": "application/json"
    },
    "createdAt": "2024-03-23T00:50:10.095000-07:00",
    "lastUpdatedAt": "2024-03-23T00:50:10.095000-07:00",
    "deprecated": false,
    "pendingDeletion": false
}
```

Enumere los comandos de su Cuenta de AWS

Una vez que haya creado los comandos, podrá ver los comandos que ha creado en la cuenta. En la lista, puede encontrar información sobre:

- El ID del comando y cualquier nombre para mostrar que haya especificado para los comandos.
- El nombre de recurso de Amazon (ARN) de los comandos.
- El estado del comando que indica si los comandos están disponibles para ejecutarse en el dispositivo de destino o si se van a quedar obsoletos.

> **Note**
>
> En la lista no se muestran los comandos que se van a eliminar de la cuenta. Si los comandos están pendientes de eliminación, aún puede ver los detalles de estos comandos mediante el ID de comando.

- El momento en que se crearon los comandos y su última actualización.

Enumeración de los comandos de la cuenta (consola)

En la AWS IoT consola, puedes encontrar la lista de comandos que has creado y sus detalles en el Centro de Comandos.

Enumeración de los comandos de la cuenta (CLI)

Para enumerar los comandos que ha creado, utilice la operación de la API `ListCommands` o la CLI `list-commands`.

Ejemplo de política de IAM

Antes de usar esta operación de la API, asegúrese de que la política de IAM le autorice a realizar esta acción en el dispositivo. A continuación, se muestra un ejemplo de una política de IAM que permite a un usuario realizar la acción `ListCommands`.

En este ejemplo, sustituya:

- *`region`* con la Región de AWS, como, por ejemplo, `us-east-1`.
- *`account-id`* con el número de la Cuenta de AWS , como, por ejemplo *`123456789012`*.

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

Ejemplo de enumeración de los comandos de la cuenta

En el siguiente comando se muestra cómo enumerar los comandos de la cuenta.

```
aws iot list-commands --namespace "AWS-IoT"
```

Al ejecutar este comando, se genera una respuesta que incluye una lista de los comandos que ha creado, el momento de creación de los comandos y su última actualización. También proporciona información sobre el estado del comando, que indica si un comando ha quedado obsoleto o si está disponible para ejecutarse en el dispositivo de destino. Para obtener más información sobre los distintos estados y el motivo del estado, consulte Estado de la ejecución del comando.

Actualización de un recurso de comandos

Una vez creado un comando, puede actualizar el nombre para mostrar y la descripción del comando.

> **Note**
>
> No se puede actualizar la carga útil del comando. Para actualizar esta información o usar una carga útil modificada, tendrá que crear un comando nuevo.

Actualización de un recurso de comandos (consola)

Para actualizar un comando desde la consola, vaya al centro de comandos de la AWS IoT consola y lleve a cabo los siguientes pasos.

1. Para actualizar un recurso de comandos existente, elija el comando que desee actualizar y, a continuación, en Acciones, seleccione Editar.
2. Especifique un nombre para mostrar, una descripción y cualquier par de nombre y valor como etiquetas para el comando.
3. Seleccione Editar para guardar el comando con la nueva configuración.

Actualización de un recurso de comandos (CLI)

Utilice la operación API del plano de `UpdateCommand`control o la `update-command` AWS CLI para actualizar un recurso de comandos. Con esta API, podrá:

- Editar el nombre para mostrar y la descripción de un comando que haya creado.
- Dar de baja un recurso de comandos o restaurar un comando que ya haya quedado obsoleto.

Ejemplo de política de IAM

Antes de usar esta operación de la API, asegúrese de que la política de IAM le autorice a realizar esta acción en el dispositivo. A continuación, se muestra un ejemplo de una política de IAM que permite a un usuario realizar la acción `UpdateCommand`.

En este ejemplo, sustituya:

- *`region`* con la Región de AWS, como, por ejemplo, `us-east-1`.
- *`account-id`* con el número de la Cuenta de AWS , como, por ejemplo *`123456789012`*.

- *command-id* con el identificador único del comando de AWS IoT , como, por ejemplo, *LockDoor*. Si desea recuperar más de un comando, puede especificarlos en la sección Recursos de la política de IAM.

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Action": "iot:UpdateCommand",
        "Effect": "Allow",
        "Resource": "arn:aws:iot:us-east-1:123456789012:command/command-id"
    }
}
```

Ejemplos de actualización de la información sobre un comando (AWS CLI)

El siguiente ejemplo muestra cómo actualizar la información sobre un comando mediante el `update-command` AWS CLI comando. Para obtener más información sobre cómo usar esta API para dar de baja o restaurar un recurso de comandos, consulte Actualización de un recurso de comandos (CLI).

En el ejemplo se muestra cómo se puede actualizar el nombre para mostrar y la descripción de un comando. En función de la aplicación, sustituya *<command-id>* por el identificador del comando para el que desea recuperar la información.

```
aws iot update-command \
    --command-id <command-id>
    --displayname <display-name> \
    --description <description>
```

Al ejecutar este comando, se genera una respuesta que incluye información sobre el comando y el momento de su última actualización. En el siguiente código se muestra un ejemplo de solicitud y respuesta para actualizar el nombre para mostrar y la descripción de un comando que desactiva la CA.

```
aws iot update-command \
    --command-id <LockDoor> \
    --displayname <Secondary lock door> \
    --description <Locks doors to my home>
```

La ejecución de este comando genera la siguiente respuesta.

```
{
    "commandId": "LockDoor",
    "commandArn": "arn:aws:iot:us-east-1:123456789012:command/LockDoor",
    "displayName": "Secondary lock door",
    "description": "Locks doors to my home",
    "lastUpdatedAt": "2024-05-09T23:15:53.899000-07:00"
}
```

Obsolescencia o restauración de un recurso de comandos

Una vez creado un comando, si ya no desea seguir utilizándolo, puede marcarlo como obsoleto. Al marcar como obsoleto un comando, todas las ejecuciones de comandos pendientes seguirán ejecutándose en el dispositivo de destino hasta que alcancen el estado final. Una vez que un comando haya quedado obsoleto, si quiere usarlo, por ejemplo, para enviar una nueva ejecución de comando al dispositivo de destino, debe restaurarlo.

> **Note**
>
> No puede editar un comando obsoleto ni ejecutarlo. Para ejecutar nuevos comandos en el dispositivo, debe restaurarlo para que el estado del comando cambie a Disponible.

Para obtener información adicional sobre la obsolescencia y la restauración de un comando, así como otros aspectos que hay que tener en cuenta, consulte Obsolescencia de un recurso de comandos.

Eliminación de un recurso de comandos

Si ya no quiere usar un comando, puede eliminarlo permanentemente de la cuenta. Si la acción de eliminación se realiza correctamente:

- Si el comando ha quedado obsoleto durante un periodo superior al tiempo de espera máximo de 12 horas, se eliminará inmediatamente.
- Si el comando no ha quedado obsoleto o ha dejado de utilizarse durante un periodo inferior al tiempo de espera máximo, el comando pasará al estado `pending deletion`. Se eliminará automáticamente de la cuenta una vez transcurrido el tiempo de espera máximo de 12 horas.

> **Note**
>
> Es posible que el comando se elimine incluso si hay alguna ejecución pendiente de ejecución. El comando estará en un estado pendiente de eliminación y se eliminará automáticamente de la cuenta.

Eliminación de un recurso de comandos (consola)

Para eliminar un comando de la consola, vaya al centro de comandos de la AWS IoT consola y lleve a cabo los siguientes pasos.

1. Elija el comando que desea eliminar y, a continuación, en Acciones y elija Eliminar.
2. Confirme que desea eliminar el comando y, a continuación, seleccione Eliminar.

El comando se marcará para su eliminación y se eliminará permanentemente de la cuenta después de 12 horas.

Eliminación de un recurso de comandos (CLI)

Utilice la operación API del plano de control `DeleteCommand` HTTP o el `delete-command` AWS CLI comando para eliminar un recurso de comando. Si la acción de eliminación se realiza correctamente, aparecerá un `statusCode` de HTTP de 204 o 202 y el comando se eliminará automáticamente de la cuenta una vez transcurrido el tiempo máximo de espera de 12 horas. En el caso del estado 204, indica que el comando se ha eliminado.

Ejemplo de política de IAM

Antes de usar esta operación de la API, asegúrese de que la política de IAM le autorice a realizar esta acción en el dispositivo. A continuación, se muestra un ejemplo de una política de IAM que permite a un usuario realizar la acción `DeleteCommand`.

En este ejemplo, sustituya:

- *`region`* con la Región de AWS, como, por ejemplo, `us-east-1`.
- *`account-id`* con el número de la Cuenta de AWS , como, por ejemplo *`123456789012`*.
- *`command-id`* con el identificador único del comando de AWS IoT , como, por ejemplo, *`LockDoor`*. Si desea recuperar más de un comando, puede especificarlos en la sección Recursos de la política de IAM.

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Action": "iot:DeleteCommand",
        "Effect": "Allow",
        "Resource": "arn:aws:iot:us-east-1:123456789012:command/command-id"
    }
}
```

Ejemplo de eliminación de un comando (AWS CLI)

Los siguientes ejemplos muestran cómo eliminar un comando mediante el `delete-command` AWS CLI comando. En función de la aplicación, sustituya *<command-id>* por el identificador del comando que vaya a eliminar.

```
aws iot delete-command --command-id <command-id>
```

Si la solicitud a la API se realiza correctamente, el comando generará un código de estado 202 o 204. Puede usar la API `GetCommand` para comprobar que el comando ya no existe en la cuenta.

Inicio y supervisión de las ejecuciones de comandos

Tras crear un comando, inicie una ejecución en el dispositivo de destino. El dispositivo actualiza los resultados y publica el estado en los temas reservados de MQTT. Recupera y supervisa el estado de ejecución de tu cuenta.

Inicie y supervise los comandos mediante la AWS IoT consola o AWS CLI.

Inicio y supervisión de las operaciones de comandos

- Inicio de la ejecución de un comando
- Actualización del resultado de la ejecución de un comando
- Recuperación de la ejecución de un comando
- Visualización de actualizaciones de comandos mediante el cliente de prueba de MQTT
- Enumera las ejecuciones de comandos en tu Cuenta de AWS
- Eliminación de una ejecución de comandos

Inicio de la ejecución de un comando

> **Important**
>
> Usted es el único responsable de implementar los comandos de forma segura y que cumpla con las normativas aplicables.

Antes de iniciar una ejecución, asegúrese de que:

- Creó un comando en el espacio de AWS IoT nombres con la información de carga útil. Al iniciar la ejecución, el dispositivo procesa las instrucciones de carga útil y realiza acciones específicas. Consulte Creación Creación de un recurso de comandos de comandos.
- Su dispositivo está suscrito a los temas reservados para comandos de MQTT. Al iniciar la ejecución, la información sobre la carga útil se publica en este tema reservado de solicitud de MQTT:

  *<devices>*pueden ser clientes Things o MQTT. *<DeviceID>*es el nombre de la cosa o el ID de cliente. *<PayloadFormat>*Valores admitidos: JSON y CBOR. Para obtener más información, consulte Temas de comandos.

  ```
  $aws/commands/<devices>/<DeviceID>/executions/+/request/<PayloadFormat>
  ```

  Si no es JSON/CBOR*<PayloadFormat>*, utilice este formato de tema de comandos:

  ```
  $aws/commands/<devices>/<DeviceID>/executions/+/request
  ```

Consideraciones sobre los dispositivos de destino

Especifique el dispositivo de destino para recibir y ejecutar el comando. Utilice un nombre de cosa para los dispositivos registrados o un ID de cliente para los dispositivos no registrados. Tras recibir la carga útil, el dispositivo ejecuta el comando y realiza las acciones especificadas.

AWS IoT cosa

Los dispositivos de destino pueden ser cosas registradas en el AWS IoT registro. Las cosas simplifican la búsqueda y la administración de los dispositivos.

Registre los dispositivos como Cosas desde la página Connect device o utilizando `CreateThing`. Encuentra cosas existentes en Thing Hub o usándolas `DescribeThing`. Consulte Administrar cosas con el registro para obtener información sobre el registro.

ID de cliente

En el caso de los dispositivos no registrados, utilice el ID de cliente.

El ID de cliente es un identificador único que se asigna a los dispositivos. Definido en el protocolo MQTT, contiene caracteres alfanuméricos, guiones bajos o guiones. Cada dispositivo al que se conecta AWS IoT necesita un ID de cliente único.

> **Note**
>
> - En el caso de las cosas registradas, el ID de cliente puede coincidir con el nombre de la cosa.
> - Al dirigirse a un ID de cliente específico, los dispositivos deben conectarse AWS IoT utilizando ese ID de cliente para recibir la carga útil.

El ID de cliente es el ID de cliente de MQTT que utilizan los dispositivos al conectarse. AWS IoT Core AWS IoT usa este ID para identificar los dispositivos y administrar las conexiones y suscripciones.

Consideraciones sobre el tiempo de espera de ejecución de comandos

El tiempo de espera especifica el tiempo (en segundos) que tardan los dispositivos en proporcionar los resultados de la ejecución.

Tras crear una ejecución, se inicia un temporizador. Si el dispositivo se desconecta o no informa de los resultados dentro del tiempo de espera, la ejecución se agota con el estado`TIMED_OUT`.

Predeterminado: 10 segundos. Máximo: 12 horas.

Valor de tiempo de espera y estado de ejecución de **`TIMED_OUT`**

Tanto la nube como el dispositivo pueden informar sobre el tiempo de espera.

Tras enviar el comando, se inicia un temporizador. Si el dispositivo no responde dentro del tiempo de espera, la nube establece el estado de ejecución `TIMED_OUT` con el código `$NO_RESPONSE_FROM_DEVICE` de motivo.

Esto ocurre cuando:

- El dispositivo se desconectó durante la ejecución.
- El dispositivo no pudo completar la ejecución dentro del tiempo de espera.
- El dispositivo no pudo informar sobre su estado dentro del tiempo de espera.

En este caso, cuando se informa del estado de ejecución de `TIMED_OUT` desde la nube, la ejecución del comando es no final. El dispositivo puede publicar una respuesta que sustituya el estado por cualquiera de los estados del terminal:`SUCCEEDED`,, `FAILED` o. `REJECTED` La ejecución del comando pasa entonces a ser terminal y no acepta más actualizaciones.

El dispositivo también puede actualizar un `TIMED_OUT` estado iniciado por la nube informando de que se ha agotado el tiempo de espera al ejecutar el comando. En este caso, el estado de ejecución del comando permanece en`TIMED_OUT`, pero el `statusReason` objeto se actualiza en función de la información proporcionada por el dispositivo. La ejecución del comando pasa entonces a ser terminal y no se aceptan más actualizaciones.

Uso de las sesiones persistentes de MQTT

Puede configurar las sesiones persistentes de MQTT para utilizarlas con la función de AWS IoT Device Management comandos. Esta característica resulta especialmente útil en casos en los que el dispositivo se queda sin conexión y quiere asegurarse de que el dispositivo siga recibiendo el comando cuando vuelve a conectarse antes de que se agote el tiempo de espera y siga las instrucciones especificadas.

De forma predeterminada, la caducidad de la sesión persistente de MQTT está establecida en 60 minutos. Si el tiempo de espera de ejecución de comandos está configurado en un valor que supera esta duración, el agente de mensajes puede rechazar las ejecuciones de comandos que se ejecuten durante más de 60 minutos y producir un error. Para ejecutar comandos de más de 60 minutos de duración, puede solicitar un aumento del tiempo de caducidad de la sesión persistente.

> **Note**
>
> Para asegurarse de que está utilizando correctamente la función de sesiones persistentes de MQTT, establezca el indicador de inicio limpio en cero. Para obtener más información, consulte Sesiones persistentes de MQTT.

Inicio de la ejecución de un comando (consola)

Para empezar a ejecutar el comando desde la consola, vaya a la página Command Hub de la AWS IoT consola y lleve a cabo los siguientes pasos.

1. Para ejecutar el comando que ha creado, elija Ejecutar comando.
2. Revise la información sobre el comando que creó, incluidos los temas reservados de MQTT y los parámetros, si corresponde.

   Para los comandos dinámicos, introduzca los valores de los parámetros o déjelos con los valores predeterminados. Para los parámetros que no tienen un valor predeterminado, debe proporcionar un valor para enviarlo como parte de esta ejecución.
3. Especifique el dispositivo de destino para recibir y ejecutar el comando. El dispositivo se puede especificar como una AWS IoT cosa si se ha registrado AWS IoT o mediante el ID de cliente si el dispositivo aún no se ha registrado. Para obtener más información, consulte Consideraciones sobre los dispositivos de destino
4. (Opcional) Configure un valor de tiempo de espera para el comando que determine el tiempo durante el que desea que se ejecute el comando antes de que se agote el tiempo de espera. Si el comando debe ejecutarse durante más de 60 minutos, puede que tenga que aumentar el tiempo de caducidad de las sesiones persistentes de MQTT. Para obtener más información, consulte Consideraciones sobre el tiempo de espera de ejecución de comandos.
5. Elija Run command (Ejecutar comando).

Inicio de la ejecución de un comando (AWS CLI)

Utilice la operación de la API del plano de datos `StartCommandExecution` HTTP para iniciar la ejecución de un comando. La solicitud y la respuesta de la API se correlacionan mediante el ID de ejecución del comando. Una vez que el dispositivo termine de ejecutar el comando, puede informar del estado y el resultado de la ejecución a la nube publicando un mensaje en el tema de respuesta del comando. Para un código de respuesta personalizado, los códigos de aplicación que poseas pueden procesar el mensaje de respuesta y publicar el resultado en él. AWS IoT

Si los dispositivos se han suscrito al tema de solicitud de comandos, la API `StartCommandExecution`l publicará el mensaje de carga útil en ese tema. La carga útil puede usar cualquier formato de su elección. Para obtener más información, consulte Carga útil de comandos.

```
$aws/commands/<devices>/<DeviceID>/executions/+/request/<PayloadFormat>
```

Si el formato de carga útil no es JSON ni CBOR, a continuación se muestra el formato del tema de solicitud de comandos.

```
$aws/commands/<devices>/<DeviceID>/executions/+/request
```

Ejemplo de política de IAM

Antes de usar esta operación de la API, asegúrese de que la política de IAM le autorice a realizar esta acción en el dispositivo. A continuación, se muestra un ejemplo de una política de IAM que permite a un usuario realizar la acción `StartCommandExecution`.

En este ejemplo, sustituya:

- *`region`* con la Región de AWS, como, por ejemplo, `us-east-1`.
- *`account-id`* con el número de la Cuenta de AWS , como, por ejemplo *`123456789012`*.
- *`command-id`*con un identificador único para el AWS IoT comando, como. *`LockDoor`* Si desea enviar más de un comando, puede especificarlos en la política de IAM.
- *`devices`*con `thing` o en `client` función de si sus dispositivos se han registrado como AWS IoT cosas o se han especificado como clientes MQTT.
- *`device-id`*con tu AWS IoT `thing-name` o. `client-id`

```
{
  "Effect": "Allow",
  "Action": [
      "iot:StartCommandExecution"
  ],
  "Resource": [
      "arn:aws:iot:region:account-id:command/command-id",
      "arn:aws:iot:region:account-id:devices/device-id"
  ]
}
```

Para ver una lista de las claves de condición compatibles`StartCommandExecution`, consulte las claves de condición AWS IoT en la Guía del usuario de IAM.

Obtención del punto de conexión del plano de datos específico de la cuenta

Antes de ejecutar el comando de la API, debe obtener la URL del punto de conexión específico de la cuenta para el punto de conexión. Si utiliza puntos finales de doble pila (IPv4y IPv6), utilice el. `iot:Data-ATS` El `iot:Jobs` punto final es solo para. IPv4 Por ejemplo, si ejecuta este comando:

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

Devuelve la URL del punto final específica de la cuenta, como se muestra en el ejemplo de respuesta que aparece a continuación.

```
{
    "endpointAddress":
    "<account-specific-prefix>-ats.iot.<region>.api.com"
}
```

Ejemplo de inicio de la ejecución de un comando (AWS CLI)

En el siguiente ejemplo, se muestra cómo empezar a ejecutar un comando mediante el `start-command-execution` AWS CLI comando.

En este ejemplo, sustituya:

- *<command-arn>* con el ARN del comando que desee ejecutar. Puede obtener esta información de la respuesta del comando `create-command` de la CLI. Por ejemplo, si está ejecutando el comando para cambiar el modo de volante, utilice `arn:aws:iot:`*region*`:`*account-id*`:command/`*SetComfortSteeringMode*.
- *<target-arn>* con el ARN del objeto del dispositivo de destino, que puede ser un objeto de IoT o un cliente de MQTT, para el que desee ejecutar el comando. Por ejemplo, si está ejecutando el comando para el dispositivo de destino `myRegisteredThing`, utilice `arn:aws:iot:`*region*`:`*account-id*`:thing/`*myRegisteredThing*.
- *<endpoint-url>* con el punto de conexión específico de la cuenta que ha obtenido en Obtención del punto de conexión del plano de datos específico de la cuenta, con el prefijo `https://`. Por ejemplo, `https://`*123456789012abcd*`.jobs.iot.`*us-east-1*`.amazonaws.com`.
- (Opcional) También puede especificar un parámetro adicional, `executionTimeoutSeconds`, al realizar la operación de la API`StartCommandExecution`. Este campo opcional especifica el tiempo en segundos dentro del cual el dispositivo debe completar la ejecución del comando. De forma predeterminada, el valor es 10 segundos. Cuando el estado de ejecución del comando sea

`CREATED`, se iniciará un temporizador. Si el resultado de la ejecución del comando no se recibe antes de que caduque el temporizador, el estado cambiará automáticamente a `TIMED_OUT`.

- ```
  aws iot-jobs-data start-command-execution \
      --command-arn <command-arn>  \
      --target-arn <target-arn> \
      --endpoint <endpoint-url> \
      --execution-timeout-seconds 900
  ```

- (Opcional) En el caso de los comandos dinámicos, especifique los parámetros y sus valores que se utilizarán para la sustitución. Debe proporcionar un valor para los parámetros que no tengan un DefaultValue establecido al crear el comando. Si un parámetro tiene un DefaultValue, el valor del parámetro que se proporciona aquí tiene prioridad. En el caso de los parámetros que tienen definidas las condiciones de valor, el valor del parámetro que se proporciona aquí debe cumplir la condición.

  Basado en el ejemplo de `Light_Power_Status` un comando dinámico:

- ```
  aws iot-jobs-data start-command-execution \
      --command-arn arn:aws:iot:us-east-1:123456789012:command/Light_Power_Status  \
      --target-arn arn:aws:iot:us-east-1:123456789012:thing/exampleThing \
      --endpoint <endpoint-url> \
      --execution-timeout-seconds 900 \
      --parameters "powerStatus={S=ON}"
  ```

La ejecución de este comando devuelve un ID de ejecución del comando. Puede usar este ID para consultar el estado de ejecución del comando, los detalles y el historial de ejecución del comando.

> **Note**
>
> Si el comando ha quedado obsoleto, se producirá un error en la solicitud de la API `StartCommandExecution` con una excepción de validación. Para corregir este error, primero restaure el comando mediante la API `UpdateCommand` y, a continuación, ejecute la solicitud `StartCommandExecution`.

```
{
    "executionId": "07e4b780-7eca-4ffd-b772-b76358da5542"
}
```

Actualización del resultado de la ejecución de un comando

Utilice la operación de la API del plano de datos de MQTT `UpdateCommandExecution` para actualizar el estado o el resultado de la ejecución de un comando.

> **Note**
>
> Antes de usar esta API:
>
> - El dispositivo debe haber establecido una conexión de MQTT y estar suscrito a los temas de solicitud y respuesta de comandos. Para obtener más información, consulte Flujo de trabajo de comandos de.
> - Ya debe haber ejecutado este comando mediante la operación de la API `StartCommandExecution`.

Ejemplo de política de IAM

Antes de usar esta operación de la API, asegúrese de que la política de IAM autorice al dispositivo a realizar estas acciones. A continuación, se muestra un ejemplo de política que autoriza al dispositivo a realizar la acción. Para ver ejemplos adicionales de políticas de IAM que permiten al usuario realizar la acción de `UpdateCommandExecution` MQTT, consulte. Ejemplos de políticas de conexión y publicación

En este ejemplo, sustituya:

- *`Region`*con las tuyas Región de AWS, como. `us-east-1`
- *`AccountID`* con el número de la Cuenta de AWS , como, por ejemplo *`123456789012`*.
- *`ThingName`*con el nombre de la AWS IoT cosa a la que apunta la ejecución del comando, como*`myRegisteredThing`*.
- *`commands-request-topic`*y *`commands-response-topic`* con los nombres de los temas de solicitud y respuesta de sus AWS IoT comandos. Para obtener más información, consulte Flujo de trabajo de comandos de.

Ejemplo de política de IAM para el ID de cliente de MQTT

En el siguiente ejemplo de código se muestra una política de dispositivo de ejemplo cuando se utiliza el ID de cliente de MQTT.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:Publish",
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/clients/
${iot:ClientId}/executions/*/response",
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/clients/
${iot:ClientId}/executions/*/response/json"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "iot:Receive",
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/clients/
${iot:ClientId}/executions/*/request",
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/clients/
${iot:ClientId}/executions/*/response/accepted",
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/clients/
${iot:ClientId}/executions/*/response/rejected",
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/clients/
${iot:ClientId}/executions/*/request/json",
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/clients/
${iot:ClientId}/executions/*/response/accepted/json",
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/clients/
${iot:ClientId}/executions/*/response/rejected/json"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "iot:Subscribe",
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/commands/clients/
${iot:ClientId}/executions/+/request",
        "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/commands/clients/
${iot:ClientId}/executions/+/response/accepted",
        "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/commands/clients/
${iot:ClientId}/executions/+/response/rejected",
        "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/commands/clients/
${iot:ClientId}/executions/+/request/json",
```

```
        "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/commands/clients/
${iot:ClientId}/executions/+/response/accepted/json",
        "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/commands/clients/
${iot:ClientId}/executions/+/response/rejected/json"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "iot:Connect",
      "Resource": "arn:aws:iot:us-east-1:123456789012:client/${iot:ClientId}"
    }
  ]
}
```

Ejemplo de política de IAM para un objeto de IoT

En el siguiente ejemplo de código se muestra una política de dispositivo de ejemplo cuando se utiliza un objeto de AWS IoT .

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:Publish",
      "Resource": "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/things/
${iot:Connection.Thing.ThingName}/executions/*/response"
    },
    {
      "Effect": "Allow",
      "Action": "iot:Receive",
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/things/
${iot:Connection.Thing.ThingName}/executions/*/request",
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/things/
${iot:Connection.Thing.ThingName}/executions/*/response/accepted",
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/things/
${iot:Connection.Thing.ThingName}/executions/*/response/rejected",
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/things/
${iot:Connection.Thing.ThingName}/executions/*/request/json",
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/things/
${iot:Connection.Thing.ThingName}/executions/*/response/accepted/json",
```

```
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/things/
${iot:Connection.Thing.ThingName}/executions/*/response/rejected/json"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "iot:Subscribe",
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/commands/things/
${iot:Connection.Thing.ThingName}/executions/+/request",
        "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/commands/things/
${iot:Connection.Thing.ThingName}/executions/+/response/accepted",
        "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/commands/things/
${iot:Connection.Thing.ThingName}/executions/+/response/rejected",
        "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/commands/things/
${iot:Connection.Thing.ThingName}/executions/+/request/json",
        "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/commands/things/
${iot:Connection.Thing.ThingName}/executions/+/response/accepted/json",
        "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/commands/things/
${iot:Connection.Thing.ThingName}/executions/+/response/rejected/json"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "iot:Connect",
      "Resource": "arn:aws:iot:us-east-1:123456789012:client/${iot:ClientId}"
    }
  ]
}
```

Cómo utilizar la API **`UpdateCommandExecution`**

Una vez recibida la ejecución del comando en el tema de solicitud, el dispositivo procesará el comando. A continuación, utiliza la API `UpdateCommandExecution` para actualizar el estado y el resultado de la ejecución del comando en el siguiente tema de respuesta.

```
$aws/commands/<devices>/<DeviceID>/executions/<ExecutionId>/response/<PayloadFormat>
```

En este ejemplo, *<DeviceID>* es el identificador único del dispositivo de destino y *<execution-id>* es el identificador de la ejecución del comando en el dispositivo de destino. *<PayloadFormat>*Pueden ser JSON o CBOR.

> **Note**
>
> Si no has registrado tu dispositivo AWS IoT, puedes usar el ID de cliente como identificador en lugar del nombre de una cosa.

```
$aws/commands/clients/<ClientID>/executions/<ExecutionId>/response/<PayloadFormat>
```

Actualizaciones en el estado de ejecución indicadas por el dispositivo

Los dispositivos pueden usar la API para informar sobre cualquiera de las siguientes actualizaciones de estado relacionadas con la ejecución de un comando. Para obtener más información sobre estos estados, consulte Estado de la ejecución del comando.

- `IN_PROGRESS`: cuando el dispositivo comience a ejecutar el comando, podrá actualizar el estado a `IN_PROGRESS`.
- `SUCCEEDED`: cuando el dispositivo procese correctamente el comando y termine de ejecutarlo, podrá publicar un mensaje en el tema de respuesta como `SUCCEEDED`.
- `FAILED`: si el dispositivo no ha podido ejecutar el comando, puede publicar un mensaje en el tema de respuesta como `FAILED`.
- `REJECTED`: si el dispositivo no ha podido aceptar el comando, puede publicar un mensaje en el tema de respuesta como `REJECTED`.
- `TIMED_OUT`: el estado de ejecución del comando puede cambiar a `TIMED_OUT` debido a cualquiera de los siguientes motivos.
  - No se ha recibido el resultado de la ejecución del comando. Esto puede ocurrir porque la ejecución no se ha completado en el plazo especificado o si el dispositivo no ha podido publicar la información de estado en el tema de respuesta.
  - El dispositivo indica que se ha agotado el tiempo de espera al intentar ejecutar el comando.

Para obtener más información sobre el estado `TIMED_OUT`, consulte Valor de tiempo de espera y estado de ejecución de `TIMED_OUT`.

Consideraciones sobre el uso de la API **`UpdateCommandExecution`**

A continuación, se indican algunas consideraciones importantes a la hora de utilizar la API `UpdateCommandExecution`.

- Sus dispositivos pueden usar un `statusReason` objeto opcional para proporcionar información adicional sobre la ejecución. Si sus dispositivos proporcionan este objeto, el `reasonCode` campo del objeto es obligatorio, pero el `reasonDescription` campo es opcional.
- Cuando los dispositivos utilicen el `statusReason` objeto, `reasonCode` deberán utilizar el patrón `[A-Z0-9_-]+` y no superar los 64 caracteres de longitud. Si lo proporciona`reasonDescription`, asegúrese de que no supere los 1024 caracteres de longitud. Puede utilizar cualquier carácter excepto los caracteres de control, como las líneas nuevas.
- Los dispositivos pueden usar un objeto de `result` opcional para proporcionar información sobre el resultado de la ejecución del comando, como, por ejemplo, el valor devuelto por una llamada a una función remota. Si proporciona el `result`, debe requerir al menos una entrada.
- En el campo `result`, debe especificar las entradas como pares de clave-valor. En cada entrada, debe especificar la información del tipo de datos en forma de cadena, booleano o binario. Un tipo de datos de cadena debe usar la clave `s`, un tipo de datos booleano usa la clave b y un tipo de datos binario debe usar la clave `bin`. Asegúrese de que estas teclas estén en minúsculas.
- Si encuentras un error al ejecutar la `UpdateCommandExecution` API, puedes ver el error en el grupo de `AWSIoTLogsV2` registros de Amazon CloudWatch. Para obtener más información sobre la habilitación de registros y la visualización de los registros, consulte Configure el AWS IoT registro.

Ejemplo de la API **`UpdateCommandExecution`**

En el código siguiente se muestra un ejemplo de cómo un dispositivo puede utilizar la API `UpdateCommandExecution` para informar del estado de la ejecución, el campo `statusReason` para proporcionar información adicional sobre el estado y el campo de resultados para proporcionar información sobre el resultado de la ejecución, como, por ejemplo, el porcentaje de batería del automóvil en este caso.

```
{
  "status": "IN_PROGRESS",
  "statusReason": {
    "reasonCode": "200",
    "reasonDescription": "Execution_in_progress"
  },
  "result": {
        "car_battery": {
            "s": "car battery at 50 percent"
        }
```

```
    }
}
```

Recuperación de la ejecución de un comando

Tras ejecutar un comando, puede recuperar información sobre la ejecución del comando desde la AWS IoT consola y mediante el AWS CLI. Puede obtener la siguiente información.

> **Note**
>
> Para recuperar el último estado de ejecución del comando, el dispositivo debe publicar la información de estado en el tema de respuesta mediante la API de MQTT `UpdateCommandExecution`, tal y como se describe a continuación. Hasta que el dispositivo publique este tema, la API `GetCommandExecution` indicará el estado como `CREATED` o `TIMED_OUT`.

Cada ejecución de comando que cree tendrá:

- Un ID de ejecución, que es un identificador único de la ejecución del comando.
- El Estado de la ejecución del comando. Al ejecutar el comando en el dispositivo de destino, la ejecución del comando entra en un estado `CREATED`. A continuación, puede pasar a otros estados de ejecución de comandos, tal y como se describe a continuación.
- El Resultado de la ejecución del comando.
- El ID de comando único y el dispositivo de destino para el que se han creado las ejecuciones.
- La Fecha de inicio, que muestra el momento en que se creó la ejecución del comando.

Recuperación de la ejecución de un comando (consola)

Puede recuperar una ejecución del comando desde la consola mediante cualquiera de los siguientes métodos.

- Desde la página Centro de comandos

  Vaya a la página Command Hub de la AWS IoT consola y lleve a cabo estos pasos.

  1. Elija el comando para el que ha creado una ejecución en el dispositivo de destino.

2. En la página de detalles del comando, en la pestaña Historial de comandos, aparecerán las ejecuciones que ha creado. Elija la ejecución de la que desee recuperar información.
3. Si los dispositivos han utilizado la API `UpdateCommandExecution` para proporcionar la información de los resultados, puede encontrar esta información en la pestaña Resultados de esta página.

- Desde la página Centro de objetos

  Si has elegido AWS IoT algo como dispositivo de destino al ejecutar el comando, puedes ver los detalles de la ejecución en la página Thing Hub.

  1. Ve a la página Thing Hub de la AWS IoT consola y elige el objeto para el que has creado la ejecución del comando.
  2. En la página de detalles del objeto, en la opción Historial de comandos, aparecerán las ejecuciones que ha creado. Elija la ejecución de la que desee recuperar información.
  3. Si los dispositivos han utilizado la API `UpdateCommandExecution` para proporcionar la información de los resultados, puede encontrar esta información en la pestaña Resultados de esta página.

Recuperación de la ejecución de un comando (CLI)

Utilice la operación de la API HTTP del plano de `GetCommandExecution` AWS IoT Core control para recuperar información sobre la ejecución de un comando. Ya debe haber ejecutado este comando mediante la operación de la API `StartCommandExecution`.

Ejemplo de política de IAM

Antes de usar esta operación de la API, asegúrese de que la política de IAM le autorice a realizar esta acción en el dispositivo. A continuación, se muestra un ejemplo de una política de IAM que permite a un usuario realizar la acción `GetCommandExecution`.

En este ejemplo, sustituya:

- *`region`* con la Región de AWS, como, por ejemplo, `us-east-1`.
- *`account-id`* con el número de la Cuenta de AWS , como, por ejemplo *`123456789012`*.
- *`command-id`*con su identificador de AWS IoT comando único, como*`LockDoor`*.
- *`devices`*con `thing` o en `client` función de si sus dispositivos se han registrado como AWS IoT cosas o se han especificado como clientes MQTT.

- *device-id*con tu AWS IoT thing-name o. client-id

```
{
  "Effect": "Allow",
  "Action": [
      "iot:GetCommandExecution"
  ],
  "Resource": [
      "arn:aws:iot:region:account-id:command/command-id",
      "arn:aws:iot:region:account-id:devices/device-id"
  ]
}
```

Ejemplo de recuperación de la ejecución de un comando

El siguiente ejemplo muestra cómo recuperar información sobre un comando que se ejecutó con el start-command-execution AWS CLI comando. En el siguiente ejemplo se muestra cómo puede recuperar información sobre un comando que se ha ejecutado para desactivar el modo volante.

En este ejemplo, sustituya:

- *<execution-id>* con el identificador de la ejecución del comando del que desea recuperar información.
- *<target-arn>* con el nombre de recurso de Amazon (ARN) del dispositivo al que se dirige la ejecución. Puede obtener esta información de la respuesta del comando start-command-execution de la CLI.
- Si lo desea, si los dispositivos han utilizado la API UpdateCommandExection para proporcionar el resultado de la ejecución, puede especificar si desea incluir el resultado de la ejecución del comando en la respuesta de la API GetCommandExecution mediante la API GetCommandExecution.

```
aws iot get-command-execution
    --execution-id <execution-id> \
    --target-arn <target-arn> \
    --include-result
```

La ejecución de este comando genera una respuesta que incluye información sobre el ARN de la ejecución del comando, el estado de la ejecución y el momento en que se inició y finalizó. También

proporciona un objeto de `statusReason` que incluye información adicional sobre el estado. Para obtener más información sobre los distintos estados y el motivo del estado, consulte Estado de la ejecución del comando.

En el siguiente código se muestra un ejemplo de respuesta de la solicitud de la API.

> **Note**
>
> El campo `completedAt` de la respuesta de ejecución corresponde al momento en que el dispositivo informa a la nube de un estado final. En el caso de un estado `TIMED_OUT`, este campo solo se configurará cuando el dispositivo indique que se ha agotado el tiempo de espera. Cuando la nube establece el estado `TIMED_OUT`, no se actualizará el estado `TIMED_OUT`. Para obtener más información sobre el comportamiento del tiempo de espera, consulte Consideraciones sobre el tiempo de espera de ejecución de comandos.

```
{
    "executionId": "07e4b780-7eca-4ffd-b772-b76358da5542",
    "commandArn": "arn:aws:iot:us-east-1:123456789012:command/LockDoor",
    "targetArn": "arn:aws:iot:us-east-1:123456789012:thing/myRegisteredThing",
    "status": "SUCCEEDED",
    "statusReason": {
        "reasonCode": "DEVICE_SUCCESSFULLY_EXECUTED",
        "reasonDescription": "SUCCESS"
    },
    "result": {
        "sn": { "s": "ABC-001" },
        "digital": { "b": true }
    },
    "createdAt": "2024-03-23T00:50:10.095000-07:00",
    "completedAt": "2024-03-23T00:50:10.095000-07:00"
}
```

Visualización de actualizaciones de comandos mediante el cliente de prueba de MQTT

Puede usar el cliente de prueba de MQTT para ver el intercambio de mensajes a través de MQTT cuando se utiliza la característica de comandos. Una vez que el dispositivo establezca una conexión MQTT con AWS IoT, puede crear un comando, especificar la carga útil y, a continuación, ejecutarlo

en el dispositivo. Al ejecutar el comando, si el dispositivo está suscrito al tema de solicitud reservada de comandos de MQTT, verá el mensaje de carga publicado en este tema.

A continuación, el dispositivo recibe las instrucciones de carga útil y realiza las operaciones especificadas en el dispositivo. AWS IoT A continuación, utiliza la `UpdateCommandExecution` API para publicar el resultado de la ejecución del comando y la información de estado en los temas de respuesta reservados de MQTT para los comandos. AWS IoT Device Management escucha las actualizaciones sobre los temas de respuesta, almacena la información actualizada y publica los registros en Amazon CloudWatch. AWS CloudTrail A continuación, puede recuperar la información más reciente sobre la ejecución de comandos desde la consola o mediante la `GetCommandExecution` API.

En los siguientes pasos se muestra cómo usar el cliente de prueba de MQTT para observar los mensajes.

1. Abra el cliente de pruebas MQTT en la AWS IoT consola.
2. En la pestaña Suscribirse, introduzca el siguiente tema y, a continuación, seleccione Suscribirse, que *<thingId>* es el nombre del dispositivo con AWS IoT el que se ha registrado.

   > **Note**
   >
   > Puedes encontrar el nombre del dispositivo en la página Thing Hub de la AWS IoT consola. Si no has registrado el dispositivo como una cosa, puedes registrarlo al conectarte a AWS IoT desde la página Conectar dispositivo.

   ```
   $aws/commands/things/<thingId>/executions/+/request
   ```

3. (Opcional) En la pestaña Suscribirse, también puede introducir los siguientes temas y seleccionar Suscribirse.

   ```
   $aws/commands/things/+/executions/+/response/accepted/json
   $aws/commands/things/+/executions/+/response/rejected/json
   ```

4. Después de iniciar la ejecución del comando, la carga útil del mensaje se enviará al dispositivo mediante el tema de solicitud al que se ha suscrito el dispositivo, `$aws/commands/things/<thingId>/executions/+/request`. En el cliente de prueba de MQTT, debería aparecer la carga útil del comando que incluye las instrucciones para que el dispositivo procese el comando.

5. Una vez que el dispositivo comience a ejecutar el comando, podrá publicar actualizaciones de estado en el siguiente tema de respuesta reservado de MQTT para comandos.

   ```
   $aws/commands/<devices>/<device-id>/executions/<executionId>/response/json
   ```

   Por ejemplo, considere un comando que ejecutó para encender el aire acondicionado del automóvil y reducir la temperatura al valor deseado. En el siguiente JSON se muestra un mensaje de ejemplo que el vehículo publicó en el tema de respuesta, en el que se indica que no pudo ejecutar el comando.

   ```
   {
     "deviceId": "My_Car",
     "executionId": "07e4b780-7eca-4ffd-b772-b76358da5542",
     "status": "FAILED",
     "statusReason": {
       "reasonCode": "CAR_LOW_ON_BATTERY",
       "reasonDescription": "Car battery is lower than 5 percent"
     }
   }
   ```

   En este caso, puede cargar la batería del automóvil y volver a ejecutar el comando.

Enumera las ejecuciones de comandos en tu Cuenta de AWS

Tras ejecutar un comando, puede recuperar información sobre la ejecución del comando desde la AWS IoT consola y mediante el AWS CLI. Puede obtener la siguiente información.

- Un ID de ejecución, que es un identificador único de la ejecución del comando.
- El Estado de la ejecución del comando. Al ejecutar el comando en el dispositivo de destino, la ejecución del comando entra en un estado `CREATED`. A continuación, puede pasar a otros estados de ejecución de comandos, tal y como se describe a continuación.
- El ID de comando único y el dispositivo de destino para el que se han creado las ejecuciones.
- La Fecha de inicio, que muestra el momento en que se creó la ejecución del comando.

Enumeración de las ejecuciones de comandos de la cuenta (consola)

Puede ver todas las ejecuciones de comandos desde la consola mediante cualquiera de los siguientes métodos.

- Desde la página Centro de comandos

  Vaya a la página Command Hub de la AWS IoT consola y lleve a cabo estos pasos.

  1. Elija el comando para el que ha creado una ejecución en el dispositivo de destino.
  2. En la página de detalles del comando, vaya a la pestaña Historial de comandos y aparecerá una lista de las ejecuciones que ha creado.
- Desde la página Centro de objetos

  Si has elegido un AWS IoT objeto como dispositivo de destino al ejecutar el comando y has creado varias ejecuciones de comandos para un solo dispositivo, puedes ver las ejecuciones del dispositivo en la página Thing Hub.

  1. Ve a la página Thing Hub de la AWS IoT consola y elige el objeto para el que has creado las ejecuciones.
  2. En la página de detalles del objeto, en la opción Historial de comandos, aparecerá una lista de las ejecuciones que ha creado para el dispositivo.

Enumeración de las ejecuciones de comandos de la cuenta (CLI)

Utilice la operación de la API HTTP del plano de `ListCommandExecutions` AWS IoT Core control para enumerar todas las ejecuciones de comandos en su cuenta.

Ejemplo de política de IAM

Antes de usar esta operación de la API, asegúrese de que la política de IAM le autorice a realizar esta acción en el dispositivo. A continuación, se muestra un ejemplo de una política de IAM que permite a un usuario realizar la acción `ListCommandExecutions`.

En este ejemplo, sustituya:

- *`region`* con la Región de AWS, como, por ejemplo, `us-east-1`.
- *`account-id`* con el número de la Cuenta de AWS , como, por ejemplo *`123456789012`*.
- *`command-id`*con su identificador de AWS IoT comando único, como*`LockDoor`*.

```
{
  "Effect": "Allow",
  "Action": "iot:ListCommandExecutions",
```

```
  "Resource": *
}
```

Ejemplo de enumeración de las ejecuciones de comandos

En el siguiente ejemplo se muestra cómo enumerar las ejecuciones de comandos en la Cuenta de AWS.

Al ejecutar el comando, debe especificar si desea filtrar la lista para que muestre únicamente las ejecuciones de comandos que se hayan creado para un determinado dispositivo mediante el `targetArn` o las ejecuciones de un determinado comando especificado mediante el `commandArn`.

En este ejemplo, sustituya:

- *<target-arn>* con el nombre de recurso de Amazon (ARN) del dispositivo al que se dirige la ejecución, como, por ejemplo, `arn:aws:iot:`*us-east-1*`:`*123456789012*`:thing/`*b8e4157c98f332cffb37627f*.
- *<target-arn>* con el nombre de recurso de Amazon (ARN) del dispositivo al que se dirige la ejecución, como, por ejemplo, `arn:aws:iot:`*us-east-1*`:`*123456789012*`:thing/`*b8e4157c98f332cffb37627f*.
- *<after>* con el tiempo transcurrido a partir del cual desea enumerar las ejecuciones que se crearon, por ejemplo, `2024-11-01T03:00`.

```
aws iot list-command-executions \
--target-arn <target-arn> \
--started-time-filter '{after=<after>}' \
--sort-order "ASCENDING"
```

Al ejecutar este comando, se genera una respuesta que incluye una lista de las ejecuciones de comandos que ha creado, el momento en que las ejecuciones comenzaron a ejecutarse y el momento en que finalizaron. También proporciona información sobre el estado, así como el objeto de `statusReason` que incluye información adicional sobre el estado.

```
{
    "commandExecutions": [
        {
            "commandArn": "arn:aws:iot:us-east-1:123456789012:command/TestMe002",
            "executionId": "b2b654ca-1a71-427f-9669-e74ae9d92d24",
```

```
            "targetArn": "arn:aws:iot:us-east-1:123456789012:thing/
b8e4157c98f332cffb37627f",
            "status": "TIMED_OUT",
            "createdAt": "2024-11-24T14:39:25.791000-08:00",
            "startedAt": "2024-11-24T14:39:25.791000-08:00"
        },
        {
            "commandArn": "arn:aws:iot:us-east-1:123456789012:command/TestMe002",
            "executionId": "34bf015f-ef0f-4453-acd0-9cca2d42a48f",
            "targetArn": "arn:aws:iot:us-east-1:123456789012:thing/
b8e4157c98f332cffb37627f",
            "status": "IN_PROGRESS",
            "createdAt": "2024-11-24T14:05:36.021000-08:00",
            "startedAt": "2024-11-24T14:05:36.021000-08:00"
        }
    ]
}
```

Para obtener más información sobre los distintos estados y el motivo del estado, consulte Estado de la ejecución del comando.

Eliminación de una ejecución de comandos

Si ya no quiere usar una ejecución de comandos, puede eliminarla permanentemente de la cuenta.

> **Note**
>
> - La ejecución de un comando solo se puede eliminar si ha pasado a un estado final, como, por ejemplo, `SUCCEEDED`, `FAILED` o `REJECTED`.
> - Esta operación solo se puede realizar mediante la AWS IoT Core API o la AWS CLI. No está disponible en la consola.

Ejemplo de política de IAM

Antes de usar esta operación de la API, asegúrese de que la política de IAM autorice al dispositivo a realizar estas acciones. A continuación, se muestra un ejemplo de política que autoriza al dispositivo a realizar la acción.

En este ejemplo, sustituya:

- *Region*con tu Región de AWS, como`us-east-1`.
- *AccountID* con el número de la Cuenta de AWS , como, por ejemplo *123456789012*.
- *CommandID* con el identificador del comando para el que desea eliminar la ejecución.
- *devices*con `thing` o en `client` función de si sus dispositivos se han registrado como AWS IoT cosas o se han especificado como clientes MQTT.
- *device-id*con tu AWS IoT `thing-name` o. `client-id`

```
{
  "Effect": "Allow",
  "Action": [
      "iot:DeleteCommandExecution"
  ],
  "Resource": [
      "arn:aws:iot:region:account-id:command/command-id",
      "arn:aws:iot:region:account-id:devices/device-id"
  ]
}
```

Ejemplo de eliminación de una ejecución de comandos

El siguiente ejemplo muestra cómo eliminar un comando mediante el `delete-command` AWS CLI comando. Según la aplicación, sustituya *<execution-id>* por el identificador de la ejecución del comando que va a eliminar y el *<target-arn>* por el ARN del dispositivo de destino.

```
aws iot delete-command-execution \
--execution-id <execution-id> \
--target-arn <target-arn>
```

Si la solicitud a la API se realiza correctamente, la ejecución de comandos generará un código de estado 200. Puede usar la API `GetCommandExecution` para comprobar que la ejecución de comandos ya no existe en la cuenta.

Obsolescencia de un recurso de comandos

Desactive los comandos para indicar que están desactualizados y no deben utilizarse. Por ejemplo, desactive los comandos que ya no se mantienen activamente o al crear comandos más nuevos con el mismo identificador pero con cargas útiles diferentes.

Consideraciones clave

Consideraciones importantes a la hora de dejar de usar comandos:

- La desaprobación de un comando no lo elimina. Puede recuperar el comando con su ID y restaurarlo para volver a utilizarlo.
- Al intentar iniciar nuevas ejecuciones de comandos obsoletos, se genera un error que impide el uso de comandos obsoletos.
- Para ejecutar un comando obsoleto, restáurelo primero. Tras la restauración, el comando estará disponible para su uso y ejecución habituales en los dispositivos de destino.
- Si desaprueba un comando mientras las ejecuciones están en curso, seguirán ejecutándose hasta que se complete. Aún puede recuperar el estado de ejecución.

Obsolescencia de un recurso de comandos (consola)

Para anular un comando de la consola, vaya al centro de comandos de la AWS IoT consola y lleve a cabo los siguientes pasos.

1. Elija el comando que desee desaprobar y, a continuación, en Acciones, elija Depredar.
2. Confirme que desea dar de baja el comando y, a continuación, seleccione Dejar de utilizar.

Obsolescencia de un recurso de comandos (CLI)

Marque los comandos como obsoletos mediante la `update-command` CLI. Debe desaprobar un comando antes de eliminarlo. Para usar un comando obsoleto, restaúrelo primero.

```
aws iot update-command \
    --command-id <command-id> \
    --deprecated
```

Por ejemplo, si ha dado de baja el comando *`ACSwitch`* que actualizó en el ejemplo anterior, en el código siguiente se muestra un ejemplo del resultado de la ejecución del comando.

```
{
    "commandId": "turnOffAc",
    "deprecated": true,
    "lastUpdatedAt": "2024-05-09T23:16:51.370000-07:00"
```

```
}
```

Comprobación del tiempo y el estado de obsolescencia

Usa la `GetCommand` API para determinar si un comando está obsoleto y cuándo fue obsoleto por última vez.

```
aws iot get-command --command-id <turnOffAC>
```

Este comando genera una respuesta que contiene información sobre el comando, incluidas las marcas de tiempo de creación y obsolescencia del último campo actualizado. Esto ayuda a determinar la duración del comando y si se debe eliminar o reutilizar. A continuación se muestra un ejemplo de respuesta para el *`turnOffAc`* comando:

```
{
    "commandId": "turnOffAC",
    "commandArn": "arn:aws:iot:us-east-1:123456789012:command/turnOffAC",
    "namespace": "AWS-IoT",
    "payload": {
        "content": "testPayload.json",
        "contentType": "application/json"
    },
    "createdAt": "2024-03-23T00:50:10.095000-07:00",
    "lastUpdatedAt": "2024-05-09T23:16:51.370000-07:00",
    "deprecated": false
}
```

Restauración de un recurso de comando

Para usar o enviar el `ACSwitch` comando a su dispositivo, restaúrelo primero.

Para restaurar un comando desde la consola, ve al centro de comandos de la AWS IoT consola, elige el comando que deseas restaurar y, a continuación, en Acciones, selecciona Restaurar.

Para restaurar un comando mediante la AWS IoT Core API o la AWS CLI, utilice la operación `UpdateCommand` API o la `update-command` CLI. En el siguiente código se muestra una solicitud y una respuesta de ejemplo.

```
aws iot update-command \
    --command-id <command-id>
```

```
    --no-deprecated
```

En el siguiente código se muestra un resultado de ejemplo.

```
{
    "commandId": "ACSwitch",
    "deprecated": false,
    "lastUpdatedAt": "2024-05-09T23:17:21.954000-07:00"
}
```

AWS IoT tunelización segura

Cuando los dispositivos se implementan detrás de firewalls restringidos en sitios remotos, necesita una forma de obtener acceso a esos dispositivos para solucionar problemas, aplicar actualizaciones de configuración y otras tareas operativas. Utilice la tunelización segura para establecer una comunicación bidireccional con dispositivos remotos a través de una conexión segura gestionada por. AWS IoT La tunelización segura no requiere actualizaciones de las reglas de firewall entrantes existentes, por lo que puede mantener el mismo nivel de seguridad proporcionado por las reglas de firewall en un sitio remoto.

Por ejemplo, un sensor instalado en una fábrica que está a varios kilómetros de distancia está teniendo problemas para medir la temperatura de la fábrica. Puede utilizar la tunelización segura para abrir e iniciar rápidamente una sesión en ese sensor. Después de identificar el problema (por ejemplo, un archivo de configuración incorrecto), puede restablecer el archivo y reiniciar el sensor a través de la misma sesión. En comparación con una solución de problemas más tradicional (por ejemplo, enviar a un técnico a la fábrica para que revise el sensor), la tunelización segura reduce la respuesta a incidentes y el tiempo de recuperación, así como los costos de explotación.

¿Qué es la tunelización segura?

Utilice la tunelización segura para acceder a los dispositivos que están desplegados detrás de firewalls con puertos restringidos en sitios remotos. Puede conectarse al dispositivo de destino desde su ordenador portátil o de sobremesa como dispositivo de origen mediante la Nube de AWS El origen y el destino se comunican mediante un proxy local de código abierto que se ejecuta en cada dispositivo. El proxy local se comunica con él Nube de AWS mediante un puerto abierto permitido por el firewall, normalmente el 443. Los datos que se transmiten a través del túnel se cifran mediante seguridad de la capa de transporte (TLS).

Temas

- Conceptos de tunelización segura
- Cómo funciona la tunelización segura
- Ciclo de vida del túnel seguro

Conceptos de tunelización segura

La tunelización segura utiliza los siguientes términos al establecer la comunicación con dispositivos remotos. Para obtener información sobre cómo funciona la tunelización segura, consulte Cómo funciona la tunelización segura.

Token de acceso de cliente (CAT)

: Un par de tokens generados por la tunelización segura cuando se crea un nuevo túnel. El CAT es utilizado por los dispositivos de origen y destino para conectarse al servicio de túnel seguro. El CAT solo puede utilizarse una vez para conectarse al túnel. Para volver a conectarse al túnel, rote los tokens de acceso del cliente mediante la operación RotateTunnelAccessTokenAPI o el comando rotate-tunnel-access-tokenCLI.

Token de cliente

: Un valor único generado por el cliente que AWS IoT Secure Tunneling puede utilizar en todas las conexiones de reintento posteriores al mismo túnel. Este campo es opcional. Si no se proporciona el token del cliente, el token de acceso del cliente (CAT) solo se puede usar una vez para el mismo túnel. Se rechazarán los intentos de conexión posteriores que utilicen el mismo CAT. Para obtener más información sobre el uso de los tokens de cliente, consulte la implementación de referencia del proxy local en. GitHub

Aplicación de destino

: La aplicación que se ejecuta en el dispositivo de destino. Por ejemplo, la aplicación de destino puede ser un daemon SSH para establecer una sesión SSH mediante tunelización segura.

Dispositivo de destino

: El dispositivo remoto al que desea obtener acceso.

Agente de dispositivo

: Una aplicación de IoT que se conecta a la puerta de enlace del AWS IoT dispositivo y escucha las notificaciones de nuevos túneles a través de MQTT. Para obtener más información, consulte Fragmento de agente de IoT.

Proxy local

: Un proxy de software que se ejecuta en los dispositivos de origen y destino y retransmite un flujo de datos entre la tunelización segura y la aplicación del dispositivo. El proxy local se puede ejecutar en modo de origen o modo de destino. Para obtener más información, consulte Proxy local.

Dispositivo de origen

El dispositivo que un operador utiliza para iniciar una sesión en el dispositivo de destino, normalmente un equipo portátil o de sobremesa.

Túnel

Una ruta lógica AWS IoT que permite la comunicación bidireccional entre un dispositivo de origen y un dispositivo de destino.

Cómo funciona la tunelización segura

A continuación, se muestra cómo la tunelización segura establece una conexión entre el dispositivo de origen y el de destino. Para obtener información sobre los distintos términos, como el token de acceso de cliente (CAT), consulte Conceptos de tunelización segura.

1. Abrir un túnel

   Para abrir un túnel para iniciar una sesión con el dispositivo de destino remoto, puede utilizar el Consola de administración de AWS comando AWS CLI open-tunnel o la API. OpenTunnel

2. Descargue el par de tokens de acceso de cliente

   Después de abrir un túnel, puede descargar el token de acceso de cliente (CAT) para su origen y destino y guardarlo en su dispositivo de origen. Debe recuperar el CAT y guardarlo ahora antes de iniciar el proxy local.

3. Inicie el proxy local en modo de destino

   El agente de IoT que se ha instalado y se está ejecutando en el dispositivo de destino se suscribirá al tema MQTT reservado `$aws/things/thing-name/tunnels/notify` y recibirá el CAT. Este *thing-name* es el nombre de AWS IoT lo que has creado para tu destino. Para obtener más información, consulte Temas de tunelización segura.

   Luego, el agente de IoT usa el CAT para iniciar el proxy local en modo de destino y configurar una conexión en el lado de destino del túnel. Para obtener más información, consulte Fragmento de agente de IoT.

4. Inicie el proxy local en modo fuente

   Una vez abierto el túnel, AWS IoT Device Management proporciona el CAT de la fuente que puede descargar en el dispositivo fuente. Puede usar el CAT para iniciar el proxy local en modo

fuente, que luego conecta el lado fuente del túnel. Para obtener más información sobre las sondas locales, consulte Proxy local.

5. Abra una sesión SSH

   Como ambos lados del túnel están conectados, puede iniciar una sesión SSH utilizando el proxy local en el lado de origen.

Para obtener más información sobre cómo utilizar el Consola de administración de AWS para abrir un túnel e iniciar una sesión SSH, consulteAbra un túnel e inicie una sesión SSH en el dispositivo remoto.

En el siguiente vídeo se describe cómo funciona la tunelización segura y se explica el proceso de configuración de una sesión SSH en un dispositivo Raspberry Pi.

Ciclo de vida del túnel seguro

Los túneles pueden tener el estado `OPEN` o `CLOSED`. Las conexiones al túnel pueden tener el estado `CONNECTED` o `DISCONNECTED`. A continuación se muestra cómo funcionan los diferentes estados del túnel y de la conexión.

1. Cuando se abre un túnel, este tiene el estado `OPEN`. El estado de conexión de origen y destino del túnel se establece en `DISCONNECTED`.
2. Cuando un dispositivo (origen o destino) se conecta al túnel, el estado de conexión correspondiente cambia a `CONNECTED`.
3. Cuando un dispositivo se desconecta del túnel mientras el estado del túnel permanece `OPEN`, el estado de la conexión correspondiente vuelve a cambiar a `DISCONNECTED`. Un dispositivo puede conectarse y desconectarse de un túnel varias veces mientras el túnel tenga el estado `OPEN`.

   > **Note**
   >
   > Los tokens de acceso de cliente (CAT) sólo pueden utilizarse una vez para conectarse a un túnel. Para volver a conectarse al túnel, rote los tokens de acceso del cliente mediante la operación RotateTunnelAccessTokenAPI o el comando rotate-tunnel-access-tokenCLI. Para ver ejemplos, consulte Resolver problemas de conectividad de túneles AWS IoT seguros mediante la rotación de los tokens de acceso de los clientes.

4. Cuando se llama a `CloseTunnel` o el túnel permanece `OPEN` durante más tiempo que el valor `MaxLifetimeTimeout`, el estado de un túnel se convierte en `CLOSED`. Puede configurar

`MaxLifetimeTimeout` al llamar a `OpenTunnel`. `MaxLifetimeTimeout` está establecido de forma predeterminada en 12 horas si no se especifica un valor.

> **Note**
>
> Un túnel no se puede volver a abrir cuando tiene el estado `CLOSED`.

5. Mientras el túnel está visible, puede llamar a `DescribeTunnel` y `ListTunnels` para ver los metadatos del túnel. El túnel puede estar visible en la AWS IoT consola durante al menos tres horas antes de que se elimine.

AWS IoT tutoriales de tunelización segura

AWS IoT la tunelización segura ayuda a los clientes a establecer una comunicación bidireccional con los dispositivos remotos que se encuentran detrás de un firewall a través de una conexión segura gestionada por. AWS IoT

Para hacer una demostración de la tunelización AWS IoT segura, utilice nuestra AWS IoT demostración de tunelización segura en. GitHub

Los siguientes tutoriales le ayudarán a aprender cómo empezar a utilizar la tunelización segura. Aprenderá a:

1. Cree un túnel seguro utilizando los métodos de configuración rápida y manual para acceder al dispositivo remoto.
2. Configure el proxy local cuando utilice el método de configuración manual y conéctese al túnel para acceder al dispositivo de destino.
3. Acceda por SSH al dispositivo remoto desde un navegador sin tener que configurar el proxy local.
4. Convierta un túnel creado con el método de configuración manual AWS CLI o utilice el método de configuración rápida.

Tutoriales en esta sección

Los tutoriales de esta sección se centran en la creación de un túnel utilizando la referencia Consola de administración de AWS y la AWS IoT API. En la AWS IoT consola, puedes crear un túnel desde la página central de túneles o desde la página de detalles de un elemento que hayas creado. Para obtener más información, consulte Métodos de creación de túneles en la AWS IoT consola.

Este tutorial contiene las siguientes secciones:

- Abra un túnel y usa SSH basado en un navegador para acceder al dispositivo remoto

  Este tutorial muestra cómo abrir un túnel desde la página Central de túneles mediante el método de configuración rápida. También aprenderás a usar el SSH basado en un navegador para acceder al dispositivo remoto mediante una interfaz de línea de comandos contextual dentro de la consola. AWS IoT

- Abra un túnel mediante la configuración manual y conéctelo a un dispositivo remoto

  Este tutorial muestra cómo abrir un túnel desde la página Central de túneles mediante el método de configuración manual. También aprenderá a configurar e iniciar el proxy local desde una terminal de su dispositivo fuente y a conectarse al túnel.

- Abra un túnel para el dispositivo remoto y utilice SSH basado en navegador

  En este tutorial se muestra cómo abrir un túnel desde la página de detalles de un objeto que haya creado. Aprenderá a crear un túnel nuevo y a utilizar uno existente. El túnel existente corresponde al túnel abierto más reciente que se creó para el dispositivo. También puede utilizar el SSH basado en navegador para acceder al dispositivo remoto.

AWS IoT tutoriales de tunelización segura

- Abra un túnel e inicie una sesión SSH en el dispositivo remoto
- Abra un túnel para el dispositivo remoto y utilice SSH basado en navegador

Abra un túnel e inicie una sesión SSH en el dispositivo remoto

En estos tutoriales, aprenderá a acceder de forma remota a un dispositivo que se encuentra detrás de un firewall. No puede iniciar una sesión SSH directa en el dispositivo porque el firewall bloquea todo el tráfico entrante. En los tutoriales se muestra cómo abrir un túnel y, a continuación, utilizarlo para iniciar una sesión de SSH en un dispositivo remoto.

Requisitos previos para los tutoriales

Los requisitos previos para ejecutar el tutorial pueden variar en función de si se utilizan los métodos de configuración manual o rápida para abrir un túnel y acceder al dispositivo remoto.

> **Note**
>
> Para ambos métodos de configuración, debe permitir el tráfico saliente en el puerto 443.

- Para obtener información sobre los requisitos previos para el tutorial del método de configuración rápida, consulte Requisitos previos para el método de configuración rápida.
- Para obtener información sobre los requisitos previos para el tutorial del método de configuración manual, consulte Requisitos previos para el método de configuración manual. Si utiliza este método de configuración, debe configurar el proxy local en el dispositivo de origen. Para descargar el código fuente del proxy local, consulte Implementación de referencia del proxy local en. GitHub

Métodos de configuración de túnel

En estos tutoriales, aprenderá sobre los métodos de configuración manual y rápida para abrir un túnel y conectarse al dispositivo remoto. La siguiente table muestra las diferencias entre los métodos de configuración. Tras crear el túnel, puede utilizar una interfaz de línea de comandos integrada en el navegador para conectarse mediante SSH al dispositivo remoto. Si pierde los tokens o el túnel se desconecta, puede enviar nuevos tokens de acceso para volver a conectarse al túnel.

Métodos de configuración rápida y manual

Criterios	Configuración rápida	Configuración manual
Creación de túnel	Cree un túnel nuevo con configura ciones editables por defecto. Para acceder a su dispositivo remoto, solo puede usar SSH como servicio de destino.	Cree un túnel especificando manualmente las configuraciones del túnel. Puede usar este método para conectarse al dispositi vo remoto mediante servicios distintos de SSH.
Tokens de acceso	El token de acceso de destino se entregará automáticamente a su dispositivo en el tema reservado de MQTT si se especifica un nombre al crear el túnel. No tiene que descargar ni gestionar el token en su dispositivo de origen.	Tendrá que descargar y gestionar manualmente el token en su dispositivo de origen. El token de acceso de destino se entrega automáticamente al dispositivo remoto en el tema reservado de MQTT si se especifica un nombre al crear el túnel.

Criterios	Configuración rápida	Configuración manual
Proxy local	Se configura automáticamente un proxy local basado en la web para que interactúes con el dispositivo. No hace falta configurar manualmente el proxy local.	Tendrá que configurar e iniciar manualmente el proxy local. Para configurar el proxy local, puede utilizar el cliente del AWS IoT dispositivo o descargar la implementación de referencia del proxy local en GitHub.

Métodos de creación de túneles en la AWS IoT consola

Los tutoriales de esta sección muestran cómo crear un túnel mediante la API Consola de administración de AWS y la OpenTunnelAPI. Si configura el destino al crear un túnel, la tunelización AWS IoT segura envía el token de acceso del cliente de destino al dispositivo remoto a través de MQTT y del tema MQTT reservado,). `$aws/things/RemoteDeviceA/tunnels/notify` Al recibir el mensaje de MQTT, el agente IoT del dispositivo remoto inicia el proxy local en modo de destino. Para obtener más información, consulte Temas reservados.

> **Note**
>
> Puede omitir la configuración de destino si desea entregar el token de acceso de cliente de destino al dispositivo remoto a través de otro método. Para obtener más información, consulte Configuración de un dispositivo remoto y uso de un agente de IoT.

En la AWS IoT consola, puede crear un túnel mediante uno de los métodos siguientes. Para obtener información sobre los tutoriales que le ayudarán a aprender a crear un túnel con estos métodos, consulte Tutoriales en esta sección.

- Central de túneles

  Al crear el túnel, podrá especificar si desea utilizar los métodos de configuración rápida o manual para crear el túnel y proporcionar los detalles de configuración del túnel opcionales. Los detalles de configuración también incluyen el nombre del dispositivo de destino y el servicio que desea utilizar para conectarse al dispositivo. Después de crear un túnel, puedes usar SSH en el navegador o abrir un terminal fuera de la AWS IoT consola para acceder a tu dispositivo remoto.

- Detalles de la cosa

Cuando cree el túnel, también podrá especificar si desea utilizar el túnel abierto más reciente o crear uno nuevo para el dispositivo, además de elegir los métodos de configuración y proporcionar cualquier detalle opcional de configuración del túnel. Los detalles de configuración de un túnel existente no se pueden editar. Puede usar el método de configuración rápida para transferir los tokens de acceso y el SSH al dispositivo remoto desde el navegador. Para abrir un túnel con este método, debe haber creado algo de IoT (por ejemplo,`RemoteDeviceA`) en el AWS IoT registro. Para obtener más información, consulte Registrar un dispositivo en el AWS IoT registro.

Tutoriales en esta sección

- Abra un túnel y usa SSH basado en un navegador para acceder al dispositivo remoto
- Abra un túnel mediante la configuración manual y conéctelo a un dispositivo remoto

Abra un túnel y usa SSH basado en un navegador para acceder al dispositivo remoto

Puede utilizar el método de configuración rápida o manual para crear un túnel. Este tutorial muestra cómo abrir un túnel mediante el método de configuración rápida y cómo utilizar el SSH basado en el navegador para conectarse al dispositivo remoto. Para obtener un ejemplo que muestra cómo abrir un túnel mediante el método de configuración manual, consulte Abra un túnel mediante la configuración manual y conéctelo a un dispositivo remoto.

Con el método de configuración rápida, puede crear un túnel nuevo con las configuraciones predeterminadas que se pueden editar. Se configura un proxy local basado en la web para usted y el token de acceso se entrega automáticamente a su dispositivo de destino remoto mediante MQTT. Tras crear un túnel, puede empezar a interactuar con su dispositivo remoto mediante una interfaz de línea de comandos dentro de la consola.

Con el método de configuración rápida, debe usar SSH como servicio de destino para acceder al dispositivo remoto. Para más información sobre los distintos métodos de configuración, consulte Métodos de configuración de túnel.

Requisitos previos para el método de configuración rápida

- Los firewalls detrás del dispositivo remoto deben permitir el tráfico saliente en el puerto 443. El túnel que cree utilizará este puerto para conectarse al dispositivo remoto.
- Tiene un agente de dispositivo de IoT (consulteFragmento de agente de IoT) ejecutándose en el dispositivo remoto que se conecta a la puerta de enlace del AWS IoT dispositivo y está configurado

con una suscripción a un tema de MQTT. Para obtener más información, consulte conectar un dispositivo a la puerta de enlace del AWS IoT dispositivo.

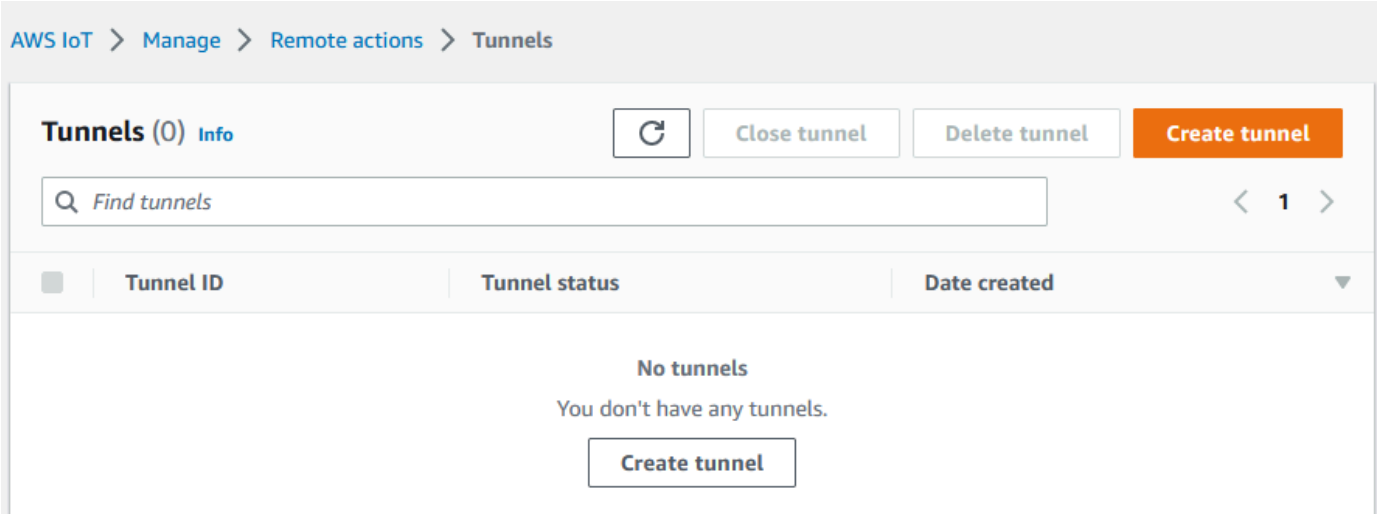
- Debe tener un daemon SSH ejecutándose en el dispositivo remoto.

Abrir un túnel

Puede abrir un túnel seguro mediante la Consola de administración de AWS referencia de la AWS IoT API o la AWS CLI. Si lo desea, puede configurar un nombre de destino, pero no es obligatorio para este tutorial. Si configura el destino, la tunelización segura entregará automáticamente el token de acceso al dispositivo remoto mediante MQTT. Para obtener más información, consulte Métodos de creación de túneles en la AWS IoT consola.

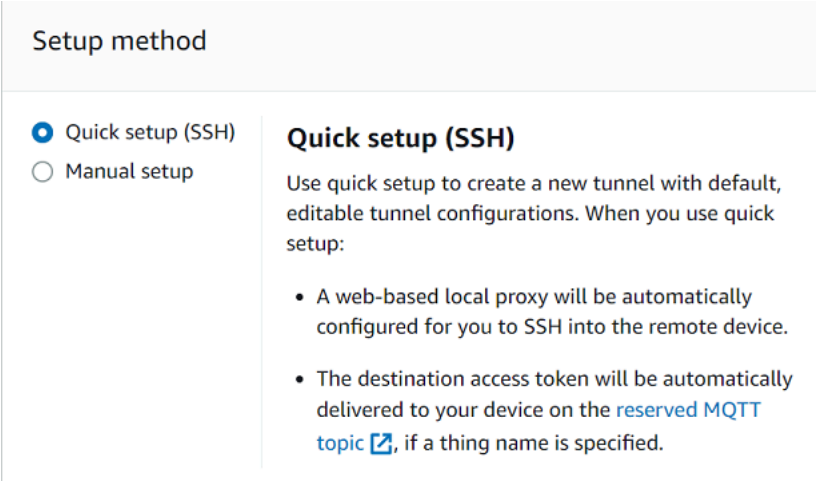
Para abrir un túnel mediante la consola

1. Vaya al Centro de túneles de la consola de AWS IoT y elija Crear túnel.



2. Para este tutorial, seleccione Configuración rápida como método de creación del túnel y, a continuación, seleccione Siguiente.

> **Note**
>
> Si crea un túnel seguro desde la página de detalles de un elemento que ha creado, puede elegir si desea crear un túnel nuevo o utilizar uno existente. Para obtener más información, consulte Abra un túnel para el dispositivo remoto y utilice SSH basado en navegador.

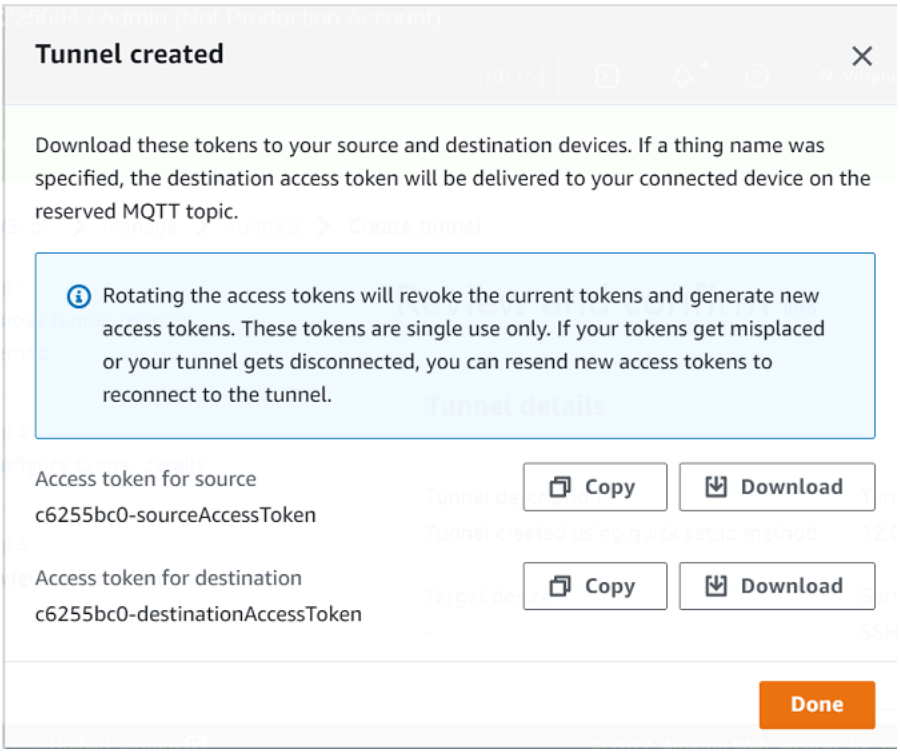


3. Revise y confirme los detalles de configuración del túnel. Para crear un túnel, elija Confirmar y crear. Si desea editar estos detalles, seleccione Anterior para volver a la página anterior y, a continuación, confirme y cree el túnel.

> **Note**
>
> Al utilizar la configuración rápida, no se puede editar el nombre del servicio. Debe usar SSH como servicio.

4. Para crear el túnel, selecciona Listo.

   Para este tutorial, no hace falta descargar los tokens de acceso de origen o destino. Estas fichas solo se pueden utilizar una vez para conectarse al túnel. Si su túnel se desconecta, puede generar y enviar nuevos tokens a su dispositivo remoto para volver a conectarse al túnel. Para obtener más información, consulte Reenviar los tokens de acceso al túnel.



Para abrir un túnel mediante la API

Para abrir un túnel nuevo, puede utilizar la operación OpenTunnelAPI.

> **Note**
>
> Puede crear un túnel mediante el método de configuración rápida solo desde la consola de AWS IoT . Cuando utilices la AWS IoT API de referencia o la AWS CLI, utilizará el método de configuración manual. Puede abrir el túnel existente que creó y, a continuación, cambiar el método de configuración del túnel para utilizar la configuración rápida. Para obtener más información, consulte Abra un túnel existente y use SSH basado en un navegador.

A continuación, mostramos un ejemplo de cómo ejecutar esta operación de API. Si lo desea, si desea especificar el nombre del objeto y el servicio de destino, utilice el parámetro `DestinationConfig`. Para ver un ejemplo que muestra cómo utilizar este parámetro, consulte Abra un túnel nuevo para el dispositivo remoto.

```
aws iotsecuretunneling open-tunnel
```

Al ejecutar este comando, se crea un túnel nuevo y se proporcionan los tokens de acceso de origen y destino.

```
{
    "tunnelId": "01234567-89ab-0123-4c56-789a01234bcd",
    "tunnelArn": "arn:aws:iot:us-
east-1:123456789012:tunnel/01234567-89ab-0123-4c56-789a01234bcd",
    "sourceAccessToken": "<SOURCE_ACCESS_TOKEN>",
    "destinationAccessToken": "<DESTINATION_ACCESS_TOKEN>"
}
```

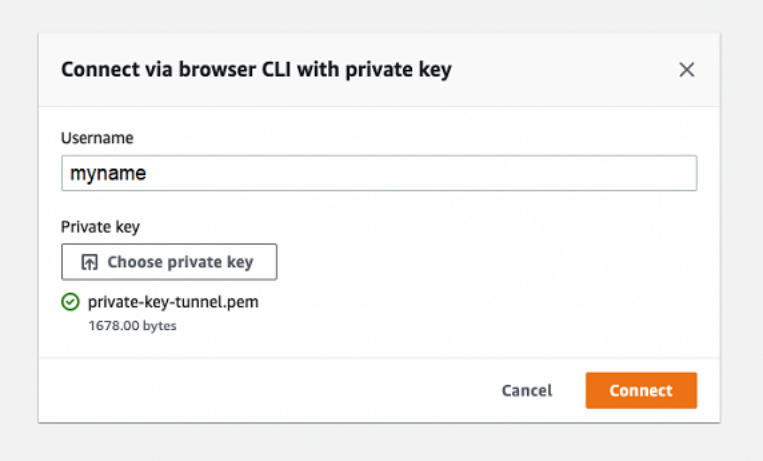
Utilización del SSH basado en navegador

Después de crear un túnel mediante el método de configuración rápida y de que el dispositivo de destino se haya conectado al túnel, podrá acceder al dispositivo remoto mediante un SSH basado en un navegador. Con el SSH basado en un navegador, puede comunicarse directamente con el dispositivo remoto introduciendo comandos en una interfaz de línea de comandos contextual dentro de la consola. Esta característica facilita la interacción con el dispositivo remoto, ya que no es necesario abrir un terminal externo a la consola ni configurar el proxy local.

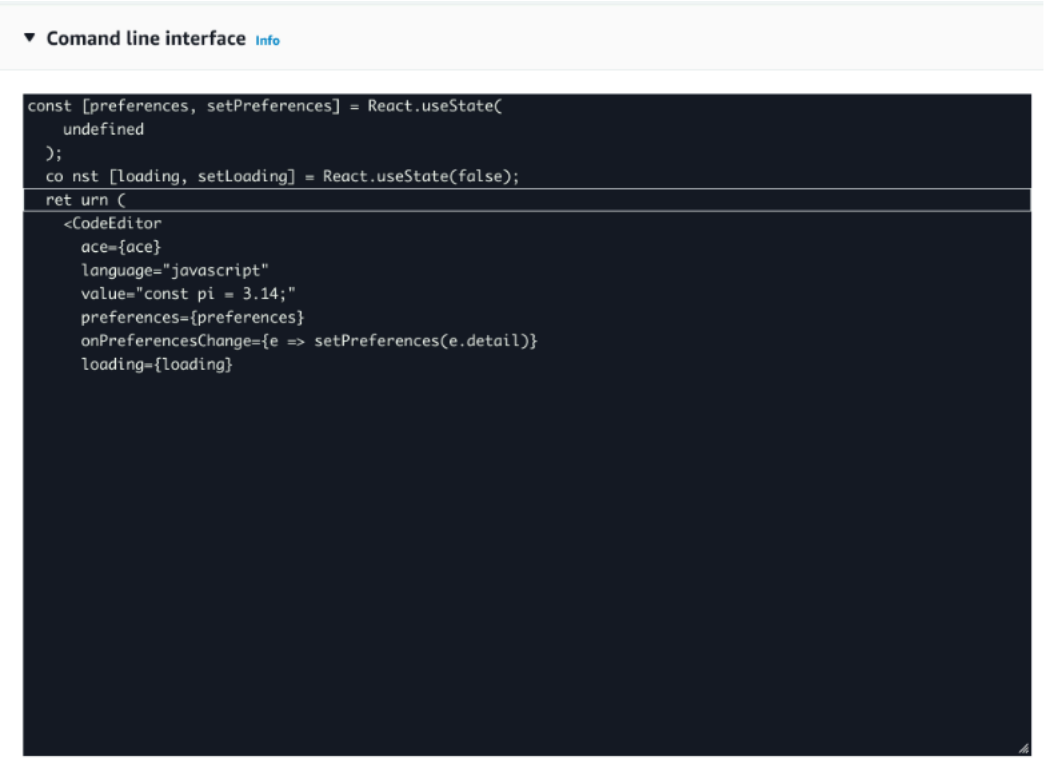
Utilización del SSH basado en navegador

1. Vaya al Centro de túneles de la consola de AWS IoT y elija Crear túnel.
2. Expanda la sección Secure Shell (SSH) y, a continuación, elija Conectar.
3. Elija si desea autenticarse en la conexión SSH proporcionando su nombre de usuario y contraseña o, para una autenticación más segura, puede utilizar la clave privada de su dispositivo. Si te estás autenticando con la clave privada, ten en cuenta que solo las claves RSA con formato PEM (256 y 512) funcionan con la consola SSH. AWS IoT Secure Tunneling

   - Para conectarse con su nombre de usuario y contraseña, elija Usar contraseña. A continuación, puede introducir su nombre de usuario y contraseña y empezar a utilizar la CLI del navegador.
   - Para conectarse con la clave privada del dispositivo de destino, seleccione Usar clave privada. Especifique su nombre de usuario y cargue el archivo de clave privada del dispositivo y, a continuación, elija Conectar para empezar a utilizar la CLI del navegador.



Una vez que se haya autenticado en la conexión SSH, podrá empezar rápidamente a introducir comandos e interactuar con el dispositivo mediante la CLI del navegador, ya que el proxy local ya está configurado para usted.



Si la CLI del navegador permanece abierta después de la duración del túnel, podría agotarse el tiempo de espera y provocar la desconexión de la interfaz de línea de comandos. Puede duplicar el túnel e iniciar otra sesión para interactuar con el dispositivo remoto dentro de la propia consola.

Solucionar problemas al usar SSH basado en navegador

A continuación, se muestra cómo solucionar algunos problemas que pueden surgir al utilizar el SSH basado en un navegador.

- Aparece un error en lugar de la interfaz de línea de comandos

  Es posible que el error se deba a que el dispositivo de destino se ha desconectado. Puede elegir Generar nuevos identificadores de acceso para generar nuevos identificadores de acceso y enviarlos a su dispositivo remoto mediante MQTT. Los nuevos tokens se pueden usar para volver a conectarse al túnel. Al volver a conectarse al túnel, se borra el historial y se actualiza la sesión de línea de comandos.

- Aparece un error de desconexión del túnel al autenticarse con una clave privada

  Es posible que vea el error porque su clave privada no haya sido aceptada por el dispositivo de destino. Para solucionar este error, compruebe el archivo de clave privada que cargó para la autenticación. Si sigue viendo un error, compruebe los registros de su dispositivo. También puede intentar volver a conectarse al túnel enviando nuevos tokens de acceso a su dispositivo remoto.

- El túnel estaba cerrado al usar la sesión

  Si el túnel se cerró porque permaneció abierto durante más tiempo del especificado, es posible que la sesión de línea de comandos se desconecte. Un túnel no se puede volver a abrir una vez cerrado. Para volver a conectarse, debe abrir otro túnel hacia el dispositivo.

  Puede duplicar un túnel para crear uno nuevo con las mismas configuraciones que el túnel cerrado. Puedes duplicar un túnel cerrado desde la AWS IoT consola. Para duplicar el túnel, elija el túnel que estaba cerrado para ver sus detalles y, a continuación, elija Duplicar túnel. Especifique la duración del túnel que quiere usar y, a continuación, cree el túnel nuevo.

Limpieza

- Cerrar el túnel

  Le recomendamos que cierre el túnel una vez que haya terminado de usarlo. Un túnel también puede cerrarse si permanece abierto durante más tiempo del especificado. Un túnel no se puede volver a abrir una vez cerrado. Aún puede duplicar un túnel si selecciona el túnel cerrado y, a continuación, selecciona Duplicar túnel. Especifique la duración del túnel que quiere usar y, a continuación, cree el túnel nuevo.

- Para cerrar un túnel individual o varios túneles desde la AWS IoT consola, vaya al centro de túneles, elija los túneles que desee cerrar y, a continuación, elija Cerrar túnel.
- Para cerrar un túnel individual o varios túneles mediante la AWS IoT API de referencia de la API, utilice la CloseTunnelAPI.

```
aws iotsecuretunneling close-tunnel \
    --tunnel-id "01234567-89ab-0123-4c56-789a01234bcd"
```

- Eliminar túnel

  Puedes eliminar un túnel de forma permanente de tu Cuenta de AWS.

  > **Warning**
  >
  > Las acciones de eliminación son permanentes y no se pueden deshacer.

  - Para eliminar un túnel individual o varios túneles desde la consola de AWS IoT , vaya al centro de túneles, elija los túneles que desee cerrar y, a continuación, elija Eliminar túnel.
  - Para eliminar un túnel individual o varios túneles mediante la AWS IoT API de referencia de la API, usa la CloseTunnelAPI. Cuando utilice la API, ajuste la marca `delete` a `true`.

```
aws iotsecuretunneling close-tunnel \
    --tunnel-id "01234567-89ab-0123-4c56-789a01234bcd"
    --delete true
```

Abra un túnel mediante la configuración manual y conéctelo a un dispositivo remoto

Al abrir un túnel, puede elegir el método de configuración rápida o manual para abrir un túnel en el dispositivo remoto. Este tutorial muestra cómo abrir un túnel mediante el método de configuración manual y cómo configurar e iniciar el proxy local para conectarse al dispositivo remoto.

Si utiliza el método de configuración manual, debe especificar manualmente las configuraciones del túnel al crear el túnel. Tras crear el túnel, puedes usar SSH en el navegador o abrir un terminal fuera de la AWS IoT consola. Este tutorial muestra cómo usar el terminal externo a la consola para acceder al dispositivo remoto. También aprenderá a configurar el proxy local y a conectarse a él para interactuar con el dispositivo remoto. Para conectarse al proxy local, debe descargar el token de acceso de origen al crear el túnel.

Con este método de configuración, puede utilizar servicios distintos de SSH, como FTP para conectarse al dispositivo remoto. Para más información sobre los distintos métodos de configuración, consulte Métodos de configuración de túnel.

Requisitos previos para el método de configuración manual

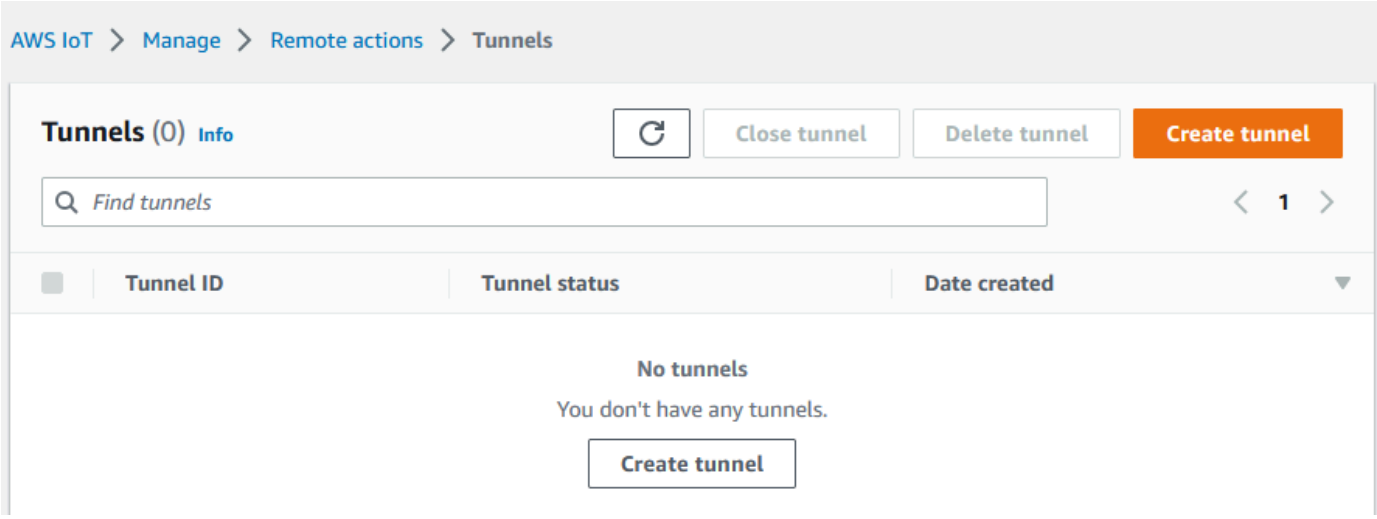
- Los firewalls detrás del dispositivo remoto deben permitir el tráfico saliente en el puerto 443. El túnel que cree utilizará este puerto para conectarse al dispositivo remoto.
- Tiene un agente de dispositivo de IoT (consulteFragmento de agente de IoT) ejecutándose en el dispositivo remoto que se conecta a la puerta de enlace del AWS IoT dispositivo y está configurado con una suscripción a un tema de MQTT. Para obtener más información, consulte conectar un dispositivo a la puerta de enlace del AWS IoT dispositivo.
- Debe tener un daemon SSH ejecutándose en el dispositivo remoto.
- Ha descargado el código fuente del proxy local GitHuby lo ha creado para la plataforma que ha elegido. Nos referiremos al archivo ejecutable del proxy local compilado como `localproxy` en este tutorial.

Abrir un túnel

Puede abrir un túnel seguro utilizando la Consola de administración de AWS referencia de la AWS IoT API o la AWS CLI. Si lo desea, puede configurar un nombre de destino, pero no es obligatorio para este tutorial. Si configura el destino, la tunelización segura entregará automáticamente el token de acceso al dispositivo remoto mediante MQTT. Para obtener más información, consulte Métodos de creación de túneles en la AWS IoT consola.

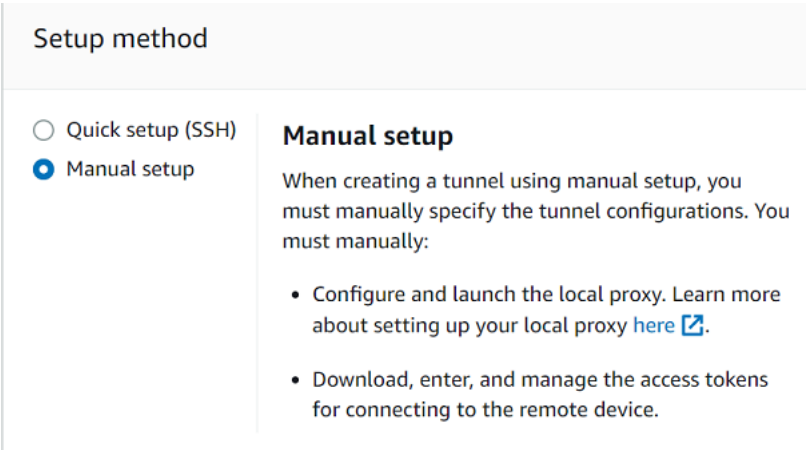
Para abrir un túnel en la consola

1. Vaya al Centro de túneles de la consola de AWS IoT y elija Crear túnel.



2. Para este tutorial, seleccione Configuración manual como método de creación del túnel y, a continuación, seleccione Siguiente. Para obtener información sobre el uso del método de configuración rápida para crear un túnel, consulte Abra un túnel y usa SSH basado en un navegador para acceder al dispositivo remoto.

> **Note**
>
> Si crea un túnel seguro desde la página de detalles de una cosa que ha creado, puede elegir si desea crear un túnel nuevo o utilizar uno existente. Para obtener más información, consulte Abra un túnel para el dispositivo remoto y utilice SSH basado en navegador.



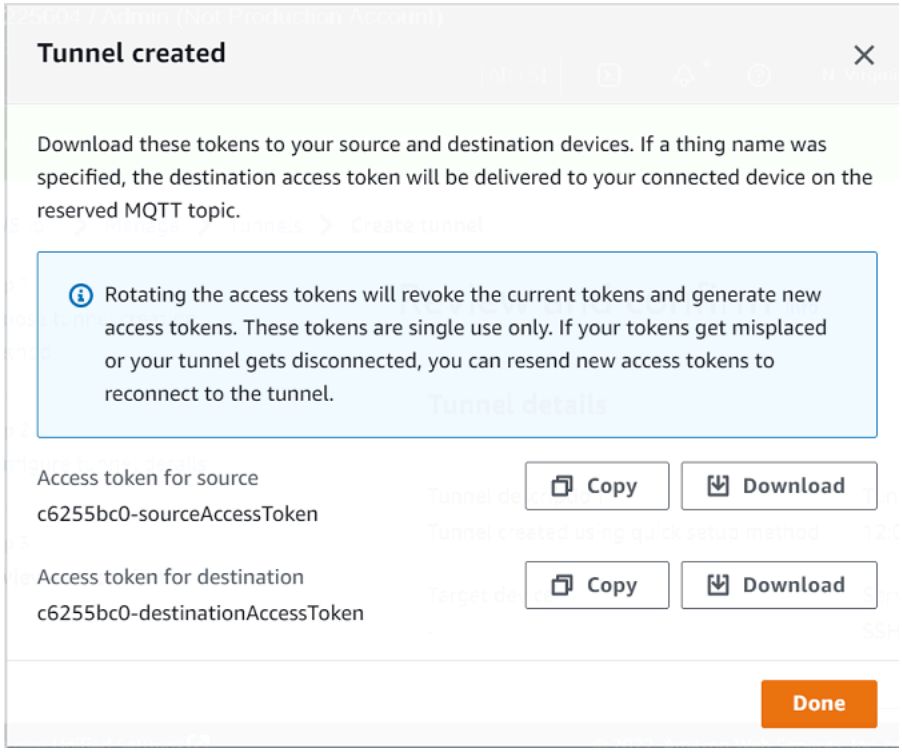
3. (Opcional) Introduzca los ajustes de configuración del túnel. También puede omitir este paso y continuar con el siguiente paso para crear un túnel.

Introduzca la descripción del túnel, la duración del tiempo de espera del túnel y las etiquetas de recursos como pares clave-valor para ayudarle a identificar el recurso. Para este tutorial, puede omitir la configuración de destino.

> **Note**
>
> No se le cobrará en función de la duración durante la que mantenga abierto un túnel. Solo incurrirá en cargos al crear un túnel nuevo. Para obtener información sobre los precios, consulte Tunelización segura en precios de AWS IoT Device Management

4. Descargue los tokens de acceso del cliente y, a continuación, seleccione Listo. Los tokens no estarán disponibles para su descarga después de seleccionar Listo.

   Estas fichas solo se pueden utilizar una vez para conectarse al túnel. Si extravía los tokens o el túnel se desconecta, puede generar y enviar nuevos tokens a su dispositivo remoto para volver a conectarse al túnel.



Para abrir un túnel mediante la API

Para abrir un túnel nuevo, puede utilizar la operación OpenTunnelAPI. También puede especificar configuraciones adicionales mediante la API, como la duración del túnel y la configuración de destino.

```
aws iotsecuretunneling open-tunnel \
    --region us-east-1 \
    --endpoint https://api.us-east-1.tunneling.iot.amazonaws.com
```

Al ejecutar este comando, se crea un túnel nuevo y se proporcionan los tokens de acceso de origen y destino.

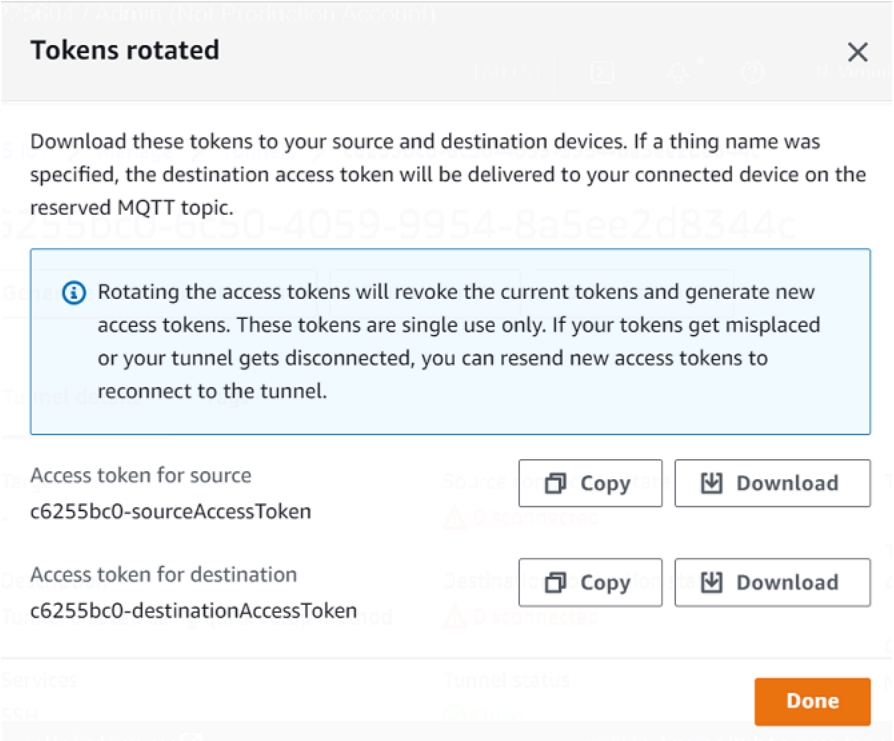
```
{
    "tunnelId": "01234567-89ab-0123-4c56-789a01234bcd",
    "tunnelArn": "arn:aws:iot:us-
east-1:123456789012:tunnel/01234567-89ab-0123-4c56-789a01234bcd",
    "sourceAccessToken": "<SOURCE_ACCESS_TOKEN>",
    "destinationAccessToken": "<DESTINATION_ACCESS_TOKEN>"
}
```

Reenviar los tokens de acceso al túnel

Los tokens que se obtienen al crear un túnel sólo se pueden utilizar una vez para conectarse al mismo. Si pierdes el token de acceso o el túnel se desconecta, puedes volver a enviar nuevos tokens de acceso al dispositivo remoto mediante MQTT sin coste adicional. AWS IoT Un túnel seguro revocará los tokens actuales y devolverá los nuevos al volver a conectarse al túnel.

Para girar los tokens desde la consola

1. Ve al centro de túneles de la AWS IoT consola y elige el túnel que has creado.
2. En la página de detalles del túnel, seleccione Generar nuevos tokens de acceso y, a continuación, seleccione Siguiente.
3. Descargue los nuevos tokens de acceso para tu túnel y seleccione Listo. Estos tokens solo se pueden usar una vez. Si extravía estos tokens o el túnel se desconecta, puede volver a enviar nuevas fichas de acceso.



Para rotar los tokens de acceso mediante la API

Para rotar los tokens de acceso al túnel, puedes usar la operación de la RotateTunnelAccessTokenAPI para revocar los tokens actuales y devolver nuevos tokens de acceso para volver a conectarte al túnel. Por ejemplo, el siguiente comando rota los tokens de acceso del dispositivo de destino, *RemoteThing1*.

```
aws iotsecuretunneling rotate-tunnel-access-token \
    --tunnel-id <tunnel-id> \
    --client-mode DESTINATION \
    --destination-config thingName=<RemoteThing1>,services=SSH \
    --region <region>
```

Al ejecutar este comando, se genera el nuevo token de acceso, como se muestra en el siguiente ejemplo. A continuación, el token se entrega al dispositivo mediante MQTT para conectarse al túnel, si el agente del dispositivo está configurado correctamente.

```
{
    "destinationAccessToken": "destination-access-token",
    "tunnelArn": "arn:aws:iot:region:account-id:tunnel/tunnel-id"
}
```

Para ver ejemplos que muestran cómo y cuándo rotar los tokens de acceso, consulte Resolver problemas de conectividad de túneles AWS IoT seguros mediante la rotación de los tokens de acceso de los clientes.

Configure e inicie el proxy local

Para conectarse al dispositivo remoto, abra un terminal en su portátil y configure e inicie el proxy local. El proxy local transmite los datos enviados por la aplicación que se ejecuta en el dispositivo de origen mediante una tunelización segura a través de una conexión segura. WebSocket Puede descargar la fuente del proxy local desde. GitHub

Después de configurar el proxy local, copie el token de acceso del cliente de origen y utilícelo para iniciar el proxy local en modo origen. A continuación se muestra un ejemplo de comando para iniciar el proxy local. En el siguiente comando, el proxy local está configurado para atender nuevas conexiones en el puerto 5555. En este comando:

- `-r`especifica la Región de AWS, que debe ser la misma región en la que se creó el túnel.
- `-s` especifica el puerto al que debe conectarse el proxy.
- `-t` especifica el texto del token del cliente.

```
./localproxy -r us-east-1 -s 5555 -t source-client-access-token
```

Al ejecutar este comando, se iniciará el proxy local en modo fuente. Si recibe el siguiente error después de ejecutar el comando, configure la ruta CA. Para obtener más información, consulte Proteger el proxy local de tunelización en. GitHub

```
Could not perform SSL handshake with proxy server: certificate verify failed
```

A continuación se muestra un ejemplo de resultado de la ejecución del proxy local en modo `source`.

```
...
...

Starting proxy in source mode
Attempting to establish web socket connection with endpoint wss://
data.tunneling.iot.us-east-1.amazonaws.com:443
Resolved proxy  server IP: 10.10.0.11
Connected successfully with proxy server
```

```
Performing SSL handshake with proxy server
Successfully completed SSL handshake with proxy server
HTTP/1.1 101 Switching Protocols

...

Connection: upgrade
channel-id: 01234567890abc23-00001234-0005678a-b1234c5de677a001-2bc3d456
upgrade: websocket

...

Web socket session ID: 01234567890abc23-00001234-0005678a-b1234c5de677a001-2bc3d456
Web socket subprotocol selected: aws.iot.securetunneling-2.0
Successfully established websocket connection with proxy server: wss://
data.tunneling.iot.us-east-1.amazonaws.com:443
Setting up web socket pings for every 5000 milliseconds
Scheduled next read:

...

Starting web socket read loop continue reading...
Resolved bind IP: 127.0.0.1
Listening for new connection on port 5555
```

Iniciar una sesión SSH

Abra otro terminal y utilice el siguiente comando para iniciar una nueva sesión SSH conectándose al proxy local en el puerto 5555.

```
ssh username@localhost -p 5555
```

Es posible que se pida una contraseña para la sesión SSH. Cuando haya terminado con la sesión SSH, escriba **`exit`** para cerrar la sesión.

Limpieza

- Cerrar el túnel

  Le recomendamos que cierre el túnel una vez que haya terminado de usarlo. Un túnel también puede cerrarse si permanece abierto durante más tiempo del especificado. Un túnel no se puede volver a abrir una vez cerrado. Aún puede duplicar un túnel abriendo el túnel cerrado y, a

continuación, seleccionando Duplicar túnel. Especifique la duración del túnel que quiere usar y, a continuación, cree el túnel nuevo.

- Para cerrar un túnel individual o varios túneles desde la AWS IoT consola, vaya al centro de túneles, elija los túneles que desee cerrar y, a continuación, elija Cerrar túnel.
- Para cerrar un túnel individual o varios túneles mediante la AWS IoT API de referencia de la API, utilice la operación de la CloseTunnelAPI.

```
aws iotsecuretunneling close-tunnel \
    --tunnel-id "01234567-89ab-0123-4c56-789a01234bcd"
```

- Eliminar túnel

  Puedes eliminar un túnel de forma permanente de tu Cuenta de AWS.

  > **Warning**
  >
  > Las acciones de eliminación son permanentes y no se pueden deshacer.

  - Para eliminar un túnel individual o varios túneles desde la consola de AWS IoT , vaya al centro de túneles, elija los túneles que desee cerrar y, a continuación, elija Eliminar túnel.
  - Para eliminar un túnel individual o varios túneles mediante la AWS IoT API de referencia de la API, utilice la operación CloseTunnelAPI. Cuando utilice la API, ajuste la marca `delete` a `true`.

```
aws iotsecuretunneling close-tunnel \
    --tunnel-id "01234567-89ab-0123-4c56-789a01234bcd"
    --delete true
```

Abra un túnel para el dispositivo remoto y utilice SSH basado en navegador

Desde la AWS IoT consola, puedes crear un túnel desde el hub de túneles o desde la página de detalles de un elemento de IoT que hayas creado. Al crear un túnel desde el centro de túneles, puede especificar si desea crear un túnel mediante la configuración rápida o la configuración manual. Para ver un tutorial de ejemplo, consulte Abra un túnel e inicie una sesión SSH en el dispositivo remoto.

Al crear un túnel desde la página de detalles del objeto de la AWS IoT consola, también puede especificar si desea crear un túnel nuevo o abrir uno existente para ese elemento, tal y como se muestra en este tutorial. Si elige un túnel existente, podrá acceder al túnel abierto más reciente que haya creado para este dispositivo. A continuación, puede utilizar la interfaz de línea de comandos dentro del terminal para SSH en el dispositivo.

Requisitos previos

- Los firewalls detrás del dispositivo remoto deben permitir el tráfico saliente en el puerto 443. El túnel que cree utilizará este puerto para conectarse al dispositivo remoto.
- Ha creado una cosa de IoT (por ejemplo,`RemoteDevice1`) en el AWS IoT registro. Esto corresponde a la representación de su dispositivo remoto en la nube. Para obtener más información, consulte Registro de un dispositivo en el registro AWS IoT.
- Tiene un agente de dispositivo de IoT (consulteFragmento de agente de IoT) ejecutándose en el dispositivo remoto que se conecta a la puerta de enlace del AWS IoT dispositivo y está configurado con una suscripción a un tema de MQTT. Para obtener más información, consulte conectar un dispositivo a la puerta de enlace del AWS IoT dispositivo.
- Debe tener un daemon SSH ejecutándose en el dispositivo remoto.

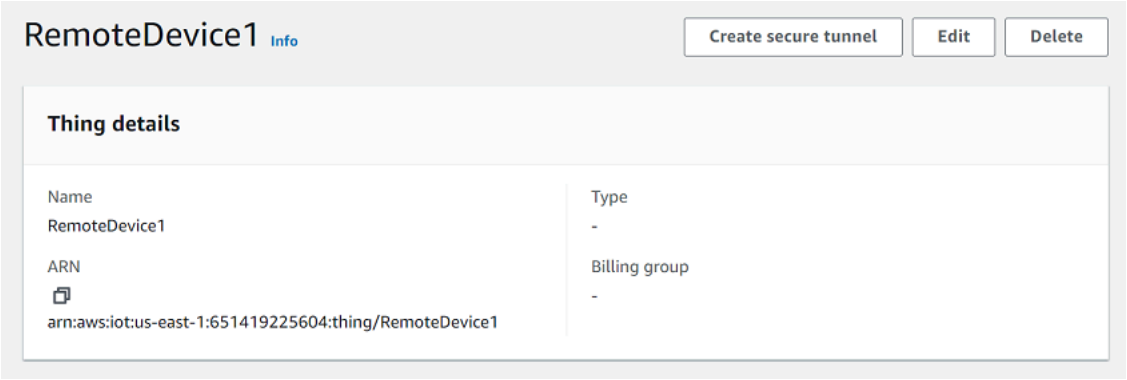
Abra un túnel nuevo para el dispositivo remoto

Supongamos que desea abrir un túnel hacia su dispositivo remoto, `RemoteDevice1`. Primero, cree una cosa de IoT con el nombre `RemoteDevice1` en el registro AWS IoT . A continuación, puede crear un túnel mediante la Consola de administración de AWS AWS IoT API de referencia o la AWS CLI.

Al configurar un destino al crear un túnel, el servicio de tunelización segura entrega el token de acceso del cliente de destino al dispositivo remoto a través de MQTT y el tema MQTT reservado (`$aws/things/RemoteDeviceA/tunnels/notify`). Para obtener más información, consulte Métodos de creación de túneles en la AWS IoT consola.

Para crear un túnel para un dispositivo remoto desde la consola

1. Seleccione el objeto, `RemoteDevice1`, para ver sus detalles y, a continuación, seleccione Crear túnel seguro.



2. Elija si desea crear un túnel nuevo o abrir uno existente. Para crear un túnel nuevo, elija Crear túnel nuevo. A continuación, puede elegir si desea utilizar el método de configuración manual o el de configuración rápida para crear el túnel. Para obtener más información, consulte Abra un túnel mediante la configuración manual y conéctelo a un dispositivo remoto y Abra un túnel y usa SSH basado en un navegador para acceder al dispositivo remoto.

Para crear un túnel para un dispositivo remoto mediante la API

Para abrir un túnel nuevo, puede utilizar la operación OpenTunnelAPI. El siguiente código muestra un ejemplo de ejecución de este comando.

```
aws iotsecuretunneling open-tunnel \
    --region us-east-1 \
    --endpoint https://api.us-east-1.tunneling.iot.amazonaws.com
    --cli-input-json file://input.json
```

A continuación se muestra el contenido del archivo `input.json`. Puede usar el parámetro `destinationConfig` para especificar el nombre del dispositivo de destino (por ejemplo, *`RemoteDevice1`*) y el servicio que desea usar para acceder al dispositivo de destino, por ejemplo, *`SSH`*. Si lo desea, también puede especificar parámetros adicionales, como etiquetas y descripción del túnel.

Contenido de input.json

```
{
   "description": "Tunnel to remote device1",
   "destinationConfig": {
      "services": [ "SSH" ],
      "thingName": "RemoteDevice1"
   }
```

```
}
```

Al ejecutar este comando, se crea un túnel nuevo y se proporcionan los tokens de acceso de origen y destino.

```
{
    "tunnelId": "01234567-89ab-0123-4c56-789a01234bcd",
    "tunnelArn": "arn:aws:iot:us-
east-1:123456789012:tunnel/01234567-89ab-0123-4c56-789a01234bcd",
    "sourceAccessToken": "<SOURCE_ACCESS_TOKEN>",
    "destinationAccessToken": "<DESTINATION_ACCESS_TOKEN>"
}
```

Abra un túnel existente y use SSH basado en un navegador

Supongamos que ha creado el túnel para su dispositivo remoto mediante el método de configuración manual o mediante la AWS IoT API de referencia de la API. `RemoteDevice1` A continuación, puede abrir el túnel existente para el dispositivo y elegir Configuración rápida para utilizar la característica SSH basada en el navegador. Las configuraciones de un túnel existente no se pueden editar, por lo que no puede usar el método de configuración manual.

Para usar la característica SSH basada en el navegador, no tendrás que descargar el token de acceso a la fuente ni configurar el proxy local. Se configurará automáticamente un proxy local basado en la web para que pueda empezar a interactuar con su dispositivo remoto.

Para usar el método de configuración rápida y el SSH basado en el navegador

1. Vaya a la página de detalles de la cosa que ha creado y seleccione Crear un túnel seguro **`RemoteDevice1`**.
2. Seleccione Usar un túnel existente para abrir el túnel abierto más reciente que haya creado para el dispositivo remoto. Las configuraciones del túnel no se pueden editar, por lo que no puede utilizar el método de configuración manual del túnel. Para usar el método de configuración rápida, seleccione Configuración rápida.
3. Proceda a revisar y confirmar los detalles de la configuración del túnel y cree el túnel. Las configuraciones del túnel no se pueden editar.

   Al crear el túnel, la tunelización segura utilizará la operación de la RotateTunnelAccessTokenAPI para revocar los tokens de acceso originales y generar nuevos. Si su dispositivo remoto utiliza MQTT, estos tokens se entregarán automáticamente al dispositivo remoto en el tema MQTT

al que esté suscrito. También puede optar por descargar estos tokens manualmente en su dispositivo de origen.

Una vez creado el túnel, puede usar el SSH basado en el navegador para interactuar con el dispositivo remoto directamente desde la consola mediante la interfaz de línea de comandos contextual. Para usar esta interfaz de línea de comandos, elige el túnel para lo que has creado y, en la página de detalles, expande la sección Interfaz de línea de comandos. Como el proxy local ya está configurado para usted, puede empezar a introducir comandos para empezar rápidamente a acceder a su dispositivo remoto e interactuar con él, `RemoteDevice1`

Para obtener más información sobre el método de configuración rápida y el uso del SSH basado en un navegador, consulte. Abra un túnel y usa SSH basado en un navegador para acceder al dispositivo remoto

Limpieza

- Cerrar el túnel

  Le recomendamos que cierre el túnel una vez que haya terminado de usarlo. Un túnel también puede cerrarse si permanece abierto durante más tiempo del especificado. Un túnel no se puede volver a abrir una vez cerrado. Aún puede duplicar un túnel abriendo el túnel cerrado y, a continuación, seleccionando Duplicar túnel. Especifique la duración del túnel que quiere usar y, a continuación, cree el túnel nuevo.

  - Para cerrar un túnel individual o varios túneles desde la AWS IoT consola, vaya al centro de túneles, elija los túneles que desee cerrar y, a continuación, elija Cerrar túnel.
  - Para cerrar un túnel individual o varios túneles mediante la AWS IoT API de referencia de la API, utilice la operación de la CloseTunnelAPI.

    ```
    aws iotsecuretunneling close-tunnel \
        --tunnel-id "01234567-89ab-0123-4c56-789a01234bcd"
    ```

- Eliminar túnel

  Puedes eliminar un túnel de forma permanente de tu Cuenta de AWS.

  > **Warning**
  >
  > Las acciones de eliminación son permanentes y no se pueden deshacer.

- Para eliminar un túnel individual o varios túneles desde la consola de AWS IoT , vaya al centro de túneles, elija los túneles que desee cerrar y, a continuación, elija Eliminar túnel.
- Para eliminar un túnel individual o varios túneles mediante la AWS IoT API de referencia de la API, utilice la operación CloseTunnelAPI. Cuando utilice la API, ajuste la marca `delete` a `true`.

```
aws iotsecuretunneling close-tunnel \
    --tunnel-id "01234567-89ab-0123-4c56-789a01234bcd"
    --delete true
```

Proxy local

El proxy local transmite los datos enviados por la aplicación que se ejecuta en el dispositivo de origen mediante una tunelización segura a través de una conexión WebSocket segura. Puede descargar la fuente del proxy local desde. GitHub

El proxy local puede ejecutarse en dos modos: `source` o `destination`. En el modo de origen, el proxy local se ejecuta en el mismo dispositivo o red que la aplicación cliente que inicia la conexión TCP. En el modo de destino, el proxy local se ejecuta en el dispositivo remoto, junto con la aplicación de destino. Un único túnel puede admitir hasta tres flujos de datos a la vez mediante la multiplexación de túneles. Para cada flujo de datos, la tunelización segura utiliza varias conexiones TCP, lo que reduce la posibilidad de que se agote el tiempo de espera. Para obtener más información, consulte Multiplex flujos de datos y uso de conexiones TCP simultáneas en un túnel seguro.

Cómo usar el proxy local

Puede ejecutar el proxy local en los dispositivos de origen y destino para transmitir datos a los puntos de conexión de tunelización seguros. Si sus dispositivos están en una red que utiliza un proxy web, este puede interceptar las conexiones antes de reenviarlas a Internet. En este caso, tendrá que configurar su proxy local para utilizar el proxy web. Para obtener más información, consulte Configure el proxy local para los dispositivos que utilizan el proxy web.

Flujo de trabajo de proxy local

Los siguientes pasos muestran cómo se ejecuta el proxy local en los dispositivos de origen y destino.

1. Conecta el proxy local a una tunelización segura

En primer lugar, el proxy local debe establecer una conexión para asegurar el túnel. Al iniciar el proxy local, utilice los siguientes argumentos:

- El `-r` argumento para especificar el lugar Región de AWS en el que se abre el túnel.
- El argumento `-t` para pasar el token de acceso del cliente de origen o de destino devuelto por el `OpenTunnel`.

> **Note**
>
> Dos proxies locales que utilicen el mismo valor de token de acceso de cliente no se pueden conectar al mismo tiempo.

2. Realizar acciones de origen o destino

   Una vez establecida la WebSocket conexión, el proxy local realiza acciones en modo de origen o en modo de destino, según su configuración.

   De forma predeterminada, el proxy local intenta volver a conectarse a un túnel seguro si se produce algún error input/output (de E/S) o si la WebSocket conexión se cierra inesperadamente. Esto hace que la conexión TCP se cierre. Si se produce algún error de socket TCP, el proxy local envía un mensaje a través del túnel para notificar al otro extremo que cierre su conexión TCP. De forma predeterminada, el proxy local siempre usa la comunicación SSL.

3. Detener el proxy local

   Después de utilizar el túnel, puede detener el proceso del proxy local sin problemas. Le recomendamos que cierre explícitamente el túnel llamando a `CloseTunnel`. Es posible que los clientes del túnel activos no se cierren inmediatamente después de llamar a `CloseTunnel`.

Para obtener más información sobre cómo utilizar el Consola de administración de AWS para abrir un túnel e iniciar una sesión SSH, consulte. Abra un túnel e inicie una sesión SSH en el dispositivo remoto

Mejores prácticas del proxy local

Al ejecutar el proxy local, siga estas prácticas recomendadas:

- Evite el uso del argumento `-t` del proxy local para pasar un token de acceso. Se recomienda utilizar la variable de entorno `AWSIOT_TUNNEL_ACCESS_TOKEN` para establecer el token de acceso del proxy local.
- Ejecute el ejecutable del proxy local con privilegios mínimos en el sistema operativo o en el entorno.
  - Evite ejecutar el proxy local como administrador en Windows.
  - Evite ejecutar el proxy local como raíz en Linux y macOS.
- Considere la posibilidad de ejecutar el proxy local en hosts distintos, contenedores, entornos de pruebas, chroot jail o en un entorno virtualizado.
- Cree el proxy local con indicadores de seguridad relevantes, en función de su cadena de herramientas.
- En dispositivos con varias interfaces de red, utilice el argumento `-b` para enlazar el socket TCP a la interfaz de red utilizada para comunicarse con la aplicación de destino.

Comando y salida de ejemplo

A continuación se muestra un ejemplo de comando que se ejecuta y el resultado correspondiente. El ejemplo muestra cómo se puede configurar el proxy local en ambos modos `source` y `destination`. El proxy local actualiza el protocolo HTTPS para establecer una conexión duradera y, WebSockets a continuación, comienza a transmitir datos a través de la conexión a los puntos finales del dispositivo de tunelización segura.

Antes de ejecutar estos comandos:

Debe haber abierto un túnel y haber obtenido los tokens de acceso del cliente para el origen y el destino. También debe haber creado el proxy local tal y como se describió anteriormente. Para crear el proxy local, abra el código fuente del proxy local en el GitHub repositorio y siga las instrucciones para crear e instalar el proxy local.

> **Note**
>
> Los siguientes comandos utilizados en los ejemplos usan la marca `verbosity` para ilustrar una descripción general de los diferentes pasos descritos anteriormente después de ejecutar el proxy local. Le recomendamos que utilice esta marca sólo con fines de prueba.

Ejecutar el proxy local en modo fuente

En los siguientes comandos se muestra cómo ejecutar el proxy local en modo origen.

Linux/macOS

En Linux o macOS, ejecute los siguientes comandos en el terminal para configurar e iniciar el proxy local en su fuente.

```
export AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
./localproxy -s 5555 -v 5 -r us-west-2
```

Donde:

- `-s` es el puerto de escucha de origen, que inicia el proxy local en modo fuente.
- `-v` es la verbosidad de la salida, que puede tener un valor entre cero y seis.
- `-r` es la región del punto de conexión en la que se abre el túnel.

Para obtener más información sobre los parámetros, consulte Opciones configuradas mediante argumentos de línea de comandos.

Windows

En Windows, el proxy local se configura de forma similar a como se hace en Linux o macOS, pero la forma en que se definen las variables de entorno es diferente a la de otras plataformas. Ejecute los siguientes comandos en la ventana `cmd` para configurar e iniciar el proxy local en su fuente.

```
set AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
.\localproxy -s 5555 -v 5 -r us-west-2
```

Donde:

- `-s` es el puerto de escucha de origen, que inicia el proxy local en modo fuente.
- `-v` es la verbosidad de la salida, que puede tener un valor entre cero y seis.
- `-r` es la región del punto de conexión en la que se abre el túnel.

Para obtener más información sobre los parámetros, consulte Opciones configuradas mediante argumentos de línea de comandos.

> **Note**
>
> Si utiliza la última versión del proxy local en modo fuente, debe incluir el AWS CLI parámetro `--destination-client-type V1` en el dispositivo de origen para garantizar la compatibilidad con versiones anteriores. Esto se aplica cuando se conecta a cualquiera de estos modos destino:
>
> - AWS IoT Dispositivo: cliente
> - AWS IoT Componente de tunelización segura o componente de tunelización AWS IoT Greengrass Version 2 segura
> - Cualquier código de demostración de AWS IoT Secure Tunneling escrito antes de 2022
> - Versiones 1.X del proxy local
>
> Este parámetro garantiza una comunicación adecuada entre el proxy de origen actualizado y los clientes de destino anteriores. Para obtener más información sobre las versiones de proxy locales, consulte AWS IoT Secure Tunneling on. GitHub

A continuación, se muestra un ejemplo de resultado de la ejecución del proxy local en el modo `source`.

```
...
...

Starting proxy in source mode
Attempting to establish web socket connection with endpoint wss://
data.tunneling.iot.us-west-2.amazonaws.com:443
Resolved proxy  server IP: 10.10.0.11
Connected successfully with proxy server
Performing SSL handshake with proxy server
Successfully completed SSL handshake with proxy server
HTTP/1.1 101 Switching Protocols

...

Connection: upgrade
channel-id: 01234567890abc23-00001234-0005678a-b1234c5de677a001-2bc3d456
upgrade: websocket
```

```
...

Web socket session ID: 01234567890abc23-00001234-0005678a-b1234c5de677a001-2bc3d456
Web socket subprotocol selected: aws.iot.securetunneling-2.0
Successfully established websocket connection with proxy server: wss://
data.tunneling.iot.us-west-2.amazonaws.com:443
Setting up web socket pings for every 5000 milliseconds
Scheduled next read:

...

Starting web socket read loop continue reading...
Resolved bind IP: 127.0.0.1
Listening for new connection on port 5555
```

Ejecutar el proxy local en el modo de destino

En los siguientes comandos se muestra cómo ejecutar el proxy local en el modo destino.

Linux/macOS

En Linux o macOS, ejecute los siguientes comandos en el terminal para configurar e iniciar el proxy local en su destino.

```
export AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
./localproxy -d 22 -v 5 -r us-west-2
```

Donde:

- `-d` es la aplicación de destino que inicia el proxy local en modo de destino.
- `-v` es la verbosidad de la salida, que puede tener un valor entre cero y seis.
- `-r` es la región del punto de conexión en la que se abre el túnel.

Para obtener más información sobre los parámetros, consulte Opciones configuradas mediante argumentos de línea de comandos.

Windows

En Windows, el proxy local se configura de forma similar a como se hace en Linux o macOS, pero la forma en que se definen las variables de entorno es diferente a la de otras plataformas. Ejecute los siguientes comandos en la ventana `cmd` para configurar e iniciar el proxy local en su destino.

```
set AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
.\localproxy -d 22 -v 5 -r us-west-2
```

Donde:

- `-d` es la aplicación de destino que inicia el proxy local en modo de destino.
- `-v` es la verbosidad de la salida, que puede tener un valor entre cero y seis.
- `-r` es la región del punto de conexión en la que se abre el túnel.

Para obtener más información sobre los parámetros, consulte Opciones configuradas mediante argumentos de línea de comandos.

> **Note**
>
> Si utiliza la última versión del proxy local en el modo de destino, debe incluir el AWS CLI parámetro `--destination-client-type V1` en el dispositivo de destino para garantizar la compatibilidad con versiones anteriores. Esto se aplica cuando se conecta a cualquiera de estos modos origen:
>
> - Tunelización segura basada en el navegador desde la consola. AWS
> - Versiones 1.X del proxy local
>
> Este parámetro garantiza una comunicación adecuada entre el proxy de destino actualizado y los clientes de origen anteriores. Para obtener más información sobre las versiones de proxy locales, consulte AWS IoT Secure Tunneling on. GitHub

A continuación, se muestra un ejemplo de resultado de la ejecución del proxy local en el modo `destination`.

```
...
...

Starting proxy in destination mode
Attempting to establish web socket connection with endpoint wss://
data.tunneling.iot.us-west-2.amazonaws.com:443
Resolved proxy  server IP: 10.10.0.11
```

```
Connected successfully with proxy server
Performing SSL handshake with proxy server
Successfully completed SSL handshake with proxy server
HTTP/1.1 101 Switching Protocols

...

Connection: upgrade
channel-id: 01234567890abc23-00001234-0005678a-b1234c5de677a001-2bc3d456
upgrade: websocket

...

Web socket session ID: 01234567890abc23-00001234-0005678a-b1234c5de677a001-2bc3d456
Web socket subprotocol selected: aws.iot.securetunneling-2.0
Successfully established websocket connection with proxy server: wss://
data.tunneling.iot.us-west-2.amazonaws.com:443
Setting up web socket pings for every 5000 milliseconds
Scheduled next read:

...

Starting web socket read loop continue reading...
```

Configure el proxy local para los dispositivos que utilizan el proxy web

Puede usar el proxy local en AWS IoT los dispositivos para comunicarse mediante túneles AWS IoT seguros. APIs El proxy local transmite los datos enviados por la aplicación del dispositivo mediante una tunelización segura a través de una conexión segura. WebSocket El proxy local puede funcionar en modo `source` o `destination`. En el modo `source`, se ejecuta en el mismo dispositivo o red que inicia la conexión TCP. En el modo `destination`, el proxy local se ejecuta en el dispositivo remoto, junto con la aplicación de destino. Para obtener más información, consulte Proxy local.

El proxy local debe conectarse directamente a Internet para utilizar una tunelización AWS IoT segura. Para una conexión TCP duradera con tunelización segura, el proxy local actualiza la solicitud HTTPS para establecer una WebSockets conexión con uno de los puntos finales de conexión del dispositivo de tunelización segura.

Si sus dispositivos están en una red que utiliza un proxy web, este puede interceptar las conexiones antes de reenviarlas a Internet. Para establecer una conexión duradera con los puntos de conexión

de conexión seguros del dispositivo de tunelización, configura tu proxy local para que utilice el proxy web tal y como se describe en la especificación de websocket.

> **Note**
>
> El Cliente de dispositivo de AWS IoT no es compatible con los dispositivos que utilizan un proxy web. Para trabajar con el proxy web, tendrás que usar un proxy local y configurarlo para que funcione con un proxy web tal y como se describe a continuación.

Los siguientes pasos muestran cómo funciona el proxy local con un proxy web.

1. El proxy local envía una solicitud `CONNECT` HTTP al proxy web que contiene la dirección remota del servicio de tunelización segura, junto con la información de autenticación del proxy web.
2. A continuación, el proxy web creará una conexión duradera con los puntos de conexión remotos de tunelización segura.
3. La conexión TCP está establecida y el proxy local ahora funcionará tanto en el modo de origen como en el de destino para la transmisión de datos.

Para completar los procedimientos de este tutorial, siga los pasos que se indican a continuación.

- Cree el proxy local
- Configure su proxy web
- Configure e inicie el proxy local

Cree el proxy local

Abre el código fuente del proxy local en el GitHub repositorio y sigue las instrucciones para crear e instalar el proxy local.

Configure su proxy web

El proxy local se basa en el mecanismo de tunelización HTTP descrito en la especificación HTTP/1.1. Para cumplir con las especificaciones, el proxy web debe permitir que los dispositivos utilicen el método. `CONNECT`

La forma de configurar el proxy web depende del proxy web que utilice y de la versión del proxy web. Para asegurarse de que configura el proxy web correctamente, consulte la documentación de su proxy web.

Para configurar el proxy web, primero identifique la URL del proxy web y confirme si el proxy web admite la tunelización HTTP. La URL del proxy web se utilizará más adelante cuando configure e inicie el proxy local.

1. Identifique la URL de su proxy web

   La URL del proxy web tendrá el siguiente formato.

   ```
   protocol://web_proxy_host_domain:web_proxy_port
   ```

   AWS IoT La tunelización segura solo admite la autenticación básica para el proxy web. Para utilizar la autenticación básica, debe especificar **`username`** y **`password`** como parte de la URL del proxy web. La URL del proxy web tendrá el siguiente formato.

   ```
   protocol://username:password@web_proxy_host_domain:web_proxy_port
   ```

   - *`protocol`*puede ser `http` o. `https` Le recomendamos que utilice `https`.
   - *`web_proxy_host_domain`*es la dirección IP de su proxy web o un nombre DNS que se convierte en la dirección IP de su proxy web.
   - *`web_proxy_port`*es el puerto en el que escucha el proxy web.
   - El proxy web utiliza este **`username`** y **`password`** para autenticar la solicitud.
2. Pruebe la URL de su proxy web

   Para confirmar si su proxy web soporta el túnel TCP, utilice un comando `curl` y asegúrese de que obtiene una respuesta `2xx` o `3xx`.

   Por ejemplo, si la URL de su proxy web es `https://server.com:1235`, utilice una marca `proxy-insecure` con el comando `curl`, ya que el proxy web podría basarse en un certificado autofirmado.

   ```
   export HTTPS_PROXY=https://server.com:1235
   curl -I https://aws.amazon.com --proxy-insecure
   ```

Si la URL de su proxy web tiene un puerto `http` (por ejemplo, `http://server.com:1234`), no tienes que usar la marca `proxy-insecure`.

```
export HTTPS_PROXY=http://server.com:1234
curl -I https://aws.amazon.com
```

Configure e inicie el proxy local

Para configurar el proxy local para utilizar un proxy web, debe configurar la variable de entorno `HTTPS_PROXY` con los nombres de dominio DNS o las direcciones IP y los números de puerto que utiliza su proxy web.

Después de configurar el proxy local, puede usarlo tal y como se explica en este documento README.

> **Note**
>
> La declaración de variables de entorno distingue entre mayúsculas y minúsculas. Recomendamos que defina cada variable solo con mayúsculas o minúsculas. Los siguientes ejemplos muestran la variable de entorno declarada en mayúsculas. Si la misma variable se especifica con letras mayúsculas y minúsculas, prevalecerá la variable especificada con letras minúsculas.

Los siguientes comandos muestran cómo configurar el proxy local que se ejecuta en el destino para que utilice el proxy web e inicie el proxy local.

- `AWSIOT_TUNNEL_ACCESS_TOKEN`: Esta variable contiene el token de acceso de cliente (CAT) del destino.
- `HTTPS_PROXY`: Esta variable contiene la URL del proxy web o la dirección IP para configurar el proxy local.

Los comandos que se muestran en los siguientes ejemplos dependen del sistema operativo que utilice y de si el proxy web escucha en un puerto HTTP o HTTPS.

El proxy web escucha en un puerto HTTP

Si su proxy web escucha en un puerto HTTP, puede proporcionar la URL o la dirección IP del proxy web para la variable HTTPS_PROXY.

Linux/macOS

En Linux o macOS, ejecute los siguientes comandos en el terminal para configurar e iniciar el proxy local en su destino para utilizar un proxy web a la escucha de un puerto HTTP.

```
export AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
export HTTPS_PROXY=http:proxy.example.com:1234
./localproxy -r us-east-1 -d 22
```

Si tiene que autenticarse con el proxy, debe especificar un **`username`** y **`password`** como parte de la variable HTTPS_PROXY.

```
export AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
export HTTPS_PROXY=http://username:password@proxy.example.com:1234
./localproxy -r us-east-1 -d 22
```

Windows

En Windows, el proxy local se configura de forma similar a como se hace en Linux o macOS, pero la forma en que se definen las variables de entorno es diferente a la de otras plataformas. Ejecute los siguientes comandos en la ventana cmd para configurar e iniciar el proxy local en su destino para utilizar un proxy web a la escucha de un puerto HTTP.

```
set AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
set HTTPS_PROXY=http://proxy.example.com:1234
.\localproxy -r us-east-1 -d 22
```

Si tiene que autenticarse con el proxy, debe especificar un **`username`** y **`password`** como parte de la variable HTTPS_PROXY.

```
set AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
set HTTPS_PROXY=http://username:password@10.15.20.25:1234
.\localproxy -r us-east-1 -d 22
```

El proxy web escucha en un puerto HTTPS

Ejecute los siguientes comandos si su proxy web escucha en un puerto HTTPS.

> **Note**
>
> Si utilizas un certificado autofirmado para el proxy web o si ejecutas el proxy local en un sistema operativo que no admite OpenSSL nativo ni tiene configuraciones predeterminadas, tendrás que configurar tus certificados de proxy web tal y como se describe en la sección Configuración de certificados del repositorio. GitHub

Los siguientes comandos tendrán un aspecto similar al que configuró su proxy web para un proxy HTTP, con la excepción de que también especificará la ruta a los archivos de certificado que instaló, tal como se describió anteriormente.

Linux/macOS

En Linux o macOS, ejecute los siguientes comandos en el terminal para configurar el proxy local que se ejecuta en su destino para utilizar un proxy web que escuche en un puerto HTTPS.

```
export AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
export HTTPS_PROXY=http:proxy.example.com:1234
./localproxy -r us-east-1 -d 22 -c /path/to/certs
```

Si tiene que autenticarse con el proxy, debe especificar un **`username`** y **`password`** como parte de la variable HTTPS_PROXY.

```
export AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
export HTTPS_PROXY=http://username:password@proxy.example.com:1234
./localproxy -r us-east-1 -d 22 -c /path/to/certs
```

Windows

En Windows, ejecute los siguientes comandos en la ventana cmd para configurar e iniciar el proxy local que se ejecuta en su destino para utilizar un proxy web a la escucha de un puerto HTTP.

```
set AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
set HTTPS_PROXY=http://proxy.example.com:1234
.\localproxy -r us-east-1 -d 22 -c \path\to\certs
```

Si tiene que autenticarse con el proxy, debe especificar un **`username`** y **`password`** como parte de la variable `HTTPS_PROXY`.

```
set AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
set HTTPS_PROXY=http://username:password@10.15.20.25:1234
.\localproxy -r us-east-1 -d 22 -c \path\to\certs
```

Comando y salida de ejemplo

A continuación se muestra un ejemplo de un comando que se ejecuta en un sistema operativo Linux y la salida correspondiente. En el ejemplo, se muestra un proxy web que escucha en un puerto HTTP y cómo se puede configurar el proxy local para que utilice el proxy web en ambos modos `source` y `destination`. Antes de poder ejecutar estos comandos, debe haber abierto ya un túnel y obtenido los tokens de acceso de cliente para el origen y el destino. También debe haber construido el proxy local y configurado su proxy web como se ha descrito anteriormente.

A continuación, se muestra información general de los pasos después de iniciar el proxy local. El proxy local:

- Identifica la URL del proxy web para que pueda utilizarla para conectarse al servidor proxy.
- Establece una conexión TCP con el proxy web.
- Envía una solicitud `CONNECT` HTTP al proxy web y espera la respuesta `HTTP/1.1 200`, que indica que se ha establecido la conexión.
- Actualiza el protocolo HTTPS para WebSockets establecer una conexión duradera.
- Comienza a transmitir datos a través de la conexión a los puntos de conexión del dispositivo de tunelización seguro.

> **Note**
>
> Los siguientes comandos utilizados en los ejemplos usan la marca `verbosity` para ilustrar una descripción general de los diferentes pasos descritos anteriormente después de ejecutar el proxy local. Le recomendamos que utilice esta marca sólo con fines de prueba.

Ejecutar el proxy local en modo fuente

En los siguientes comandos se muestra cómo ejecutar el proxy local en modo fuente.

```
export AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
export HTTPS_PROXY=http:username:password@10.15.10.25:1234
./localproxy -s 5555 -v 5 -r us-west-2
```

A continuación se muestra un ejemplo de resultado de la ejecución del proxy local en modo `source`.

```
...

Parsed basic auth credentials for the URL
Found Web proxy information in the environment variables, will use it to connect via
 the proxy.

...

Starting proxy in source mode
Attempting to establish web socket connection with endpoint wss://
data.tunneling.iot.us-west-2.amazonaws.com:443
Resolved Web proxy IP: 10.10.0.11
Connected successfully with Web Proxy
Successfully sent HTTP CONNECT to the Web proxy
Full response from the Web proxy:
HTTP/1.1 200 Connection established
TCP tunnel established successfully
Connected successfully with proxy server
Successfully completed SSL handshake with proxy server
Web socket session ID: 0a109afffee745f5-00001341-000b8138-cc6c878d80e8adb0-f186064b
Web socket subprotocol selected: aws.iot.securetunneling-2.0
Successfully established websocket connection with proxy server: wss://
data.tunneling.iot.us-west-2.amazonaws.com:443
Seting up web socket pings for every 5000 milliseconds
Scheduled next read:

...

Starting web socket read loop continue reading...
Resolved bind IP: 127.0.0.1
Listening for new connection on port 5555
```

Ejecutar el proxy local en el modo de destino

En los siguientes comandos se muestra cómo ejecutar el proxy local en modo destino.

```
export AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
```

```
export HTTPS_PROXY=http:username:password@10.15.10.25:1234
./localproxy -d 22 -v 5 -r us-west-2
```

A continuación se muestra un ejemplo de resultado de la ejecución del proxy local en modo `destination`.

```
...

Parsed basic auth credentials for the URL
Found Web proxy information in the environment variables, will use it to connect via
 the proxy.

...

Starting proxy in destination mode
Attempting to establish web socket connection with endpoint wss://
data.tunneling.iot.us-west-2.amazonaws.com:443
Resolved Web proxy IP: 10.10.0.1
Connected successfully with Web Proxy
Successfully sent HTTP CONNECT to the Web proxy
Full response from the Web proxy:
HTTP/1.1 200 Connection established
TCP tunnel established successfully
Connected successfully with proxy server
Successfully completed SSL handshake with proxy server
Web socket session ID: 06717bfffed3fd05-00001355-000b8315-da3109a85da804dd-24c3d10d
Web socket subprotocol selected: aws.iot.securetunneling-2.0
Successfully established websocket connection with proxy server: wss://
data.tunneling.iot.us-west-2.amazonaws.com:443
Seting up web socket pings for every 5000 milliseconds
Scheduled next read:

...

Starting web socket read loop continue reading...
```

Multiplex flujos de datos y uso de conexiones TCP simultáneas en un túnel seguro

Puede utilizar varios flujos de datos por túnel mediante la característica de multiplexación por túnel segura. Con la multiplexación, puede solucionar problemas de los dispositivos que utilizan

varios flujos de datos. También puede reducir la carga operativa al eliminar la necesidad de crear, implementar e iniciar varios proxies locales o abrir varios túneles en el mismo dispositivo. Por ejemplo, la multiplexación se puede utilizar en el caso de un navegador web que requiera enviar varios flujos de datos HTTP y SSH.

Para cada flujo de datos, la tunelización AWS IoT segura admite conexiones TCP simultáneas. El uso de conexiones simultáneas reduce la posibilidad de que se agote el tiempo de espera en caso de que el cliente presente varias solicitudes. Por ejemplo, puede reducir el tiempo de carga al acceder de forma remota a un servidor web local del dispositivo de destino.

En las siguientes secciones se explica más acerca de la multiplexación y el uso de conexiones TCP simultáneas, así como sus diferentes casos de uso.

Temas

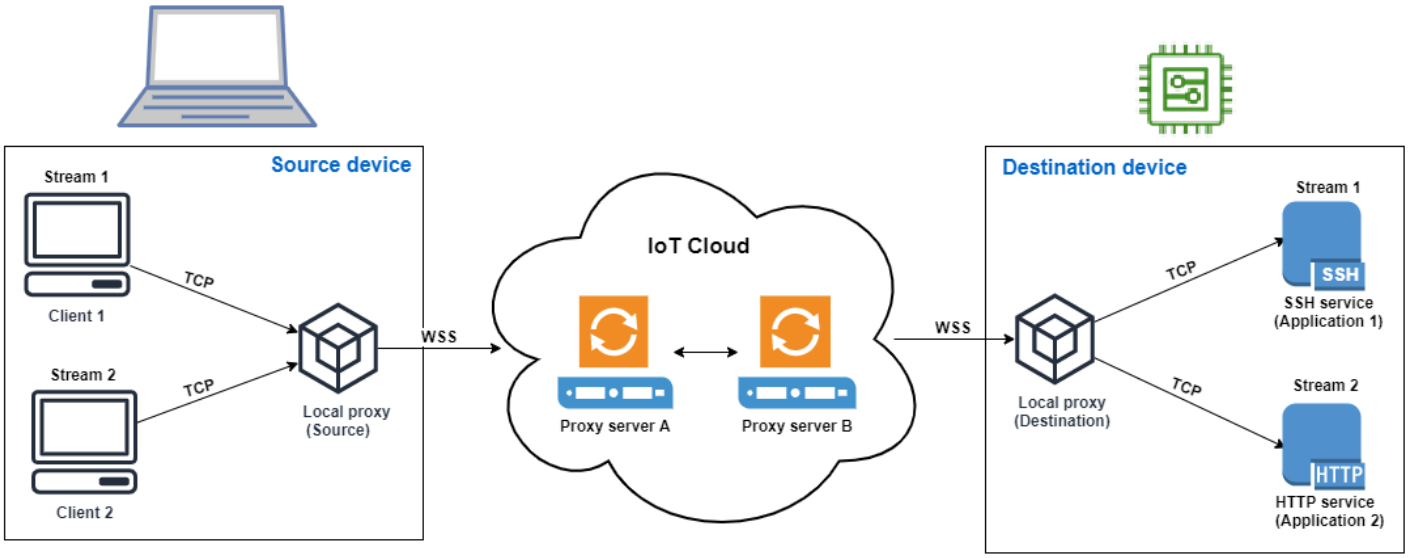
- Multiplexación de varios flujos de datos en un túnel seguro
- Uso de conexiones TCP simultáneas en un túnel seguro

Multiplexación de varios flujos de datos en un túnel seguro

Puede utilizar la característica de multiplexación para dispositivos que utilizan varias conexiones o puertos. La multiplexación también se puede utilizar cuando se necesitan varias conexiones a un dispositivo remoto para solucionar cualquier problema. Por ejemplo, puede utilizarse en el caso de un navegador web que requiera el envío de múltiples flujos de datos HTTP y SSH. Los datos de aplicación de ambos flujos se envían al dispositivo simultáneamente a través del túnel multiplexado.

Ejemplo de caso de uso

Supongamos que necesita conectarse a una aplicación web integrada en el dispositivo para cambiar algunos parámetros de red y, al mismo tiempo, emitir comandos de consola a través del terminal para comprobar que el dispositivo funciona correctamente con los nuevos parámetros de red. En este escenario, es posible que deba conectarse al dispositivo a través de HTTP y SSH y transferir dos flujos de datos paralelos para acceder simultáneamente a la aplicación web y al terminal. Con la característica de multiplexación, estos dos flujos independientes se pueden transferir a través del mismo túnel al mismo tiempo.



Cómo configurar un túnel multiplexado

El siguiente procedimiento explica cómo configurar un túnel multiplexado para solucionar problemas de dispositivos que utilizan aplicaciones que requieren conexiones a varios puertos. Configurará un túnel con dos flujos multiplexados: un flujo HTTP y un flujo SSH.

1. (Opcional) Cree archivos de configuración

   Si lo desea, puede configurar el dispositivo de origen y destino con archivos de configuración. Utilice archivos de configuración si es probable que las asignaciones de puertos cambien con frecuencia. Puede omitir este paso si prefiere especificar la asignación de puertos de forma explícita mediante la CLI o si no necesita iniciar el proxy local en los puertos de escucha designados. Para obtener más información sobre cómo usar los archivos de configuración, consulte las opciones configuradas mediante --config en. GitHub

   1. En su dispositivo fuente, en la carpeta donde se ejecutará su proxy local, cree una carpeta de configuración llamada `Config`. Dentro de esta carpeta, cree un archivo llamado `SSHSource.ini` con el siguiente contenido:

      ```
      HTTP1 = 5555
      SSH1 = 3333
      ```

   2. En su dispositivo de destino, en la carpeta donde se ejecutará su proxy local, cree una carpeta de configuración llamada `Config`. Dentro de esta carpeta, cree un archivo llamado `SSHDestination.ini` con el siguiente contenido:

```
HTTP1 = 80
SSH1 = 22
```

2. Abrir un túnel

   Abra un túnel mediante la operación API `OpenTunnel` o el comando de CLI `open-tunnel`. Configure el destino especificando `SSH1` y `HTTP1` como los servicios y el nombre de AWS IoT lo que corresponde a su dispositivo remoto. Sus aplicaciones SSH y HTTP se ejecutan en este dispositivo remoto. Debes haber creado ya lo de IoT en el AWS IoT registro. Para obtener más información, consulte Administración de objetos con el registro.

   ```
   aws iotsecuretunneling open-tunnel \
    --destination-config thingName=RemoteDevice1,services=HTTP1,SSH1
   ```

   Al ejecutar este comando, se generan los tokens de acceso de origen y destino que utilizarás para ejecutar el proxy local.

   ```
   {
    "tunnelId": "b2de92a3-b8ff-46c0-b0f2-afa28b00cecd",
    "tunnelArn": "arn:aws:iot:us-west-2:431600097591:tunnel/b2de92a3-b8ff-46c0-b0f2-
   afa28b00cecd",
    "sourceAccessToken": source_client_access_token,
    "destinationAccessToken": destination_client_access_token
   }
   ```

3. Configure e inicie el proxy local

   Antes de poder ejecutar el proxy local, configura el cliente del AWS IoT dispositivo o descarga el código fuente del proxy local GitHuby compruébalo para la plataforma que prefieras. A continuación, puede iniciar el proxy local de destino y de origen para conectarse al túnel seguro. Para obtener más información sobre la configuración y el uso del proxy local, consulte Cómo usar el proxy local.

   > **Note**
   >
   > En el dispositivo de origen, si no utiliza ningún archivo de configuración ni especifica la asignación de puertos mediante la CLI, puede seguir utilizando el mismo comando

para ejecutar el proxy local. El proxy local en modo fuente recogerá automáticamente los puertos disponibles para usarlos y las asignaciones por usted.

Start local proxy using configuration files

Ejecute los siguientes comandos para ejecutar el proxy local en los modos de origen y destino mediante archivos de configuración.

```
// ----------------- Start the destination local proxy ----------------------
./localproxy -r us-east-1 -m dst -t destination_client_access_token

// ----------------- Start the source local proxy ---------------------------
// You also run the same command below if you want the local proxy to
// choose the mappings for you instead of using configuration files.
./localproxy -r us-east-1 -m src -t source_client_access_token
```

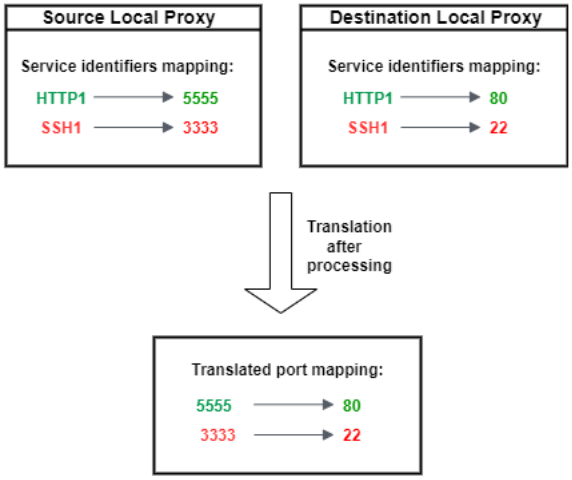
Start local proxy using CLI port mapping

Ejecute los siguientes comandos para ejecutar el proxy local en los modos de origen y destino especificando las asignaciones de puertos de forma explícita mediante la CLI.

```
// ----------------- Start the destination local proxy
 ----------------------------------
./localproxy -r us-east-1 -d HTTP1=80,SSH1=22 -t destination_client_access_token

// ----------------- Start the source local proxy
 ---------------------------------------
./localproxy -r us-east-1 -s HTTP1=5555,SSH1=33 -t source_client_access_token
```

Los datos de la aplicación de la conexión SSH y HTTP ahora se pueden transferir simultáneamente a través del túnel multiplexado. Como se ve en el siguiente mapa, el identificador de servicio actúa como un formato legible para traducir la asignación de puertos entre el dispositivo de origen y el de destino. Con esta configuración, la tunelización segura reenvía todo el tráfico HTTP entrante del puerto *5555* del dispositivo de origen al puerto *80* del dispositivo de destino y cualquier tráfico SSH entrante de un puerto *3333 22* a otro del dispositivo de destino.



Uso de conexiones TCP simultáneas en un túnel seguro

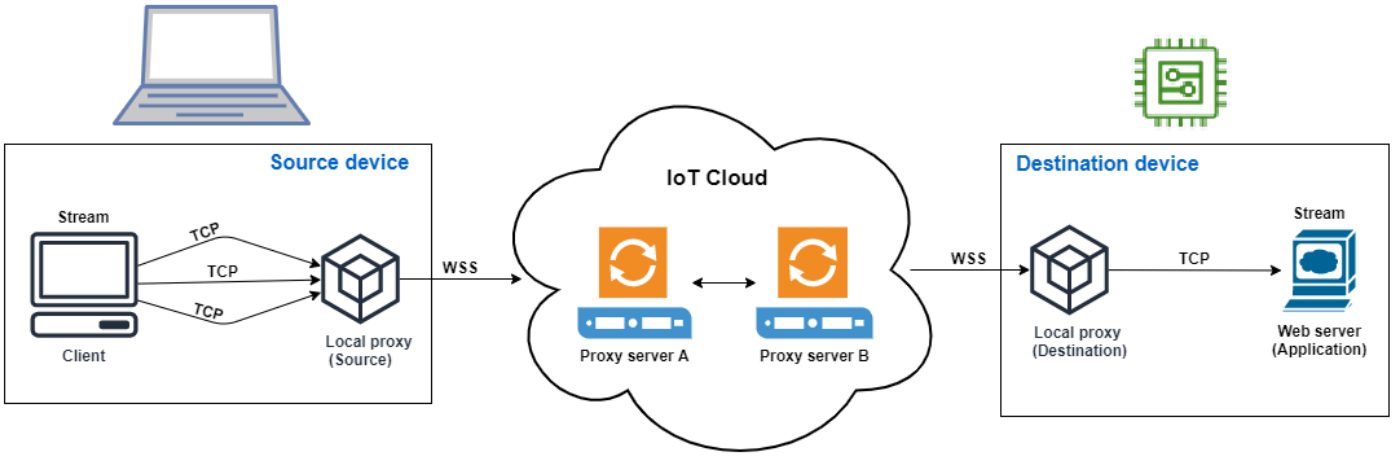
AWS IoT La tunelización segura admite más de una conexión TCP simultáneamente para cada flujo de datos. Puede utilizar esta capacidad cuando necesite conexiones simultáneas a un dispositivo remoto. El uso de conexiones TCP simultáneas reduce la posibilidad de que se agote el tiempo de espera en caso de peticiones múltiples del cliente. Por ejemplo, al acceder a un servidor web que tiene varios componentes ejecutándose, las conexiones TCP simultáneas pueden reducir el tiempo que se tarda en cargar el sitio.

> **Note**
>
> Las conexiones TCP simultáneas tienen un límite de ancho de banda de 800 kilobytes por segundo para cada una. Cuenta de AWS AWS IoT La tunelización segura puede configurar este límite por usted en función del número de solicitudes entrantes.

Ejemplo de caso de uso

Supongamos que necesita acceder de forma remota a un servidor web que sea local en el dispositivo de destino y que tenga varios componentes ejecutándose en él. Con una sola conexión TCP, al intentar acceder al servidor web, la carga secuencial puede aumentar el tiempo que se tarda en cargar los recursos del sitio. Las conexiones TCP simultáneas pueden reducir el tiempo de carga al cumplir con los requisitos de recursos del sitio y, por lo tanto, reducir el tiempo de acceso. El siguiente diagrama muestra cómo se admiten las conexiones TCP simultáneas para el flujo de datos a la aplicación del servidor web que se ejecuta en el dispositivo remoto.

> **Note**
>
> Si desea acceder a varias aplicaciones que se ejecutan en el dispositivo remoto mediante el túnel, puede utilizar la multiplexación por túnel. Para obtener más información, consulte Multiplexación de varios flujos de datos en un túnel seguro.



Cómo utilizar conexiones TCP simultáneas

El siguiente procedimiento explica cómo utilizar conexiones TCP simultáneas para acceder al navegador web del dispositivo remoto. Cuando hay varias solicitudes del cliente, la tunelización AWS IoT segura configura automáticamente conexiones TCP simultáneas para gestionar las solicitudes, lo que reduce el tiempo de carga.

1. Abrir un túnel

   Abra un túnel mediante la operación API `OpenTunnel` o el comando de CLI `open-tunnel`. Configure el destino especificando `HTTP` como servicio y el nombre de la cosa AWS IoT que corresponde a su dispositivo remoto. La aplicación de su servidor web se está ejecutando en este dispositivo remoto. Debes haber creado ya lo de IoT en el AWS IoT registro. Para obtener más información, consulte Administración de objetos con el registro.

   ```
   aws iotsecuretunneling open-tunnel \
    --destination-config thingName=RemoteDevice1,services=HTTP
   ```

Al ejecutar este comando, se generan los tokens de acceso de origen y destino que utilizarás para ejecutar el proxy local.

```
{
 "tunnelId": "b2de92a3-b8ff-46c0-b0f2-afa28b00cecd",
 "tunnelArn": "arn:aws:iot:us-west-2:431600097591:tunnel/b2de92a3-b8ff-46c0-b0f2-
afa28b00cecd",
 "sourceAccessToken": source_client_access_token,
 "destinationAccessToken": destination_client_access_token
}
```

2. Configure e inicie el proxy local

Antes de poder ejecutar el proxy local, descarga el código fuente del proxy local GitHuby compruébalo para la plataforma que prefieras. A continuación, puede iniciar el proxy local de destino y de origen para conectarse al túnel seguro y empezar a utilizar la aplicación del servidor web remoto.

> **Note**
>
> Para que la tunelización AWS IoT segura utilice conexiones TCP simultáneas, debe actualizar el proxy local a la versión más reciente. Esta característica no está disponible si configura el proxy local mediante el cliente del dispositivo AWS IoT .

```
// Start the destination local proxy
./localproxy -r us-east-1 -d HTTP=80 -t destination_client_access_token

// Start the source local proxy
./localproxy -r us-east-1 -s HTTP=5555 -t source_client_access_token
```

Para obtener más información sobre la configuración y el uso del proxy local, consulte Cómo usar el proxy local.

Ahora puede usar el túnel para acceder a la aplicación del servidor web. AWS IoT secure tunneling configurará y gestionará automáticamente las conexiones TCP simultáneas cuando haya varias solicitudes del cliente.

Configuración de un dispositivo remoto y uso de un agente de IoT

El agente IoT se utiliza para recibir el mensaje MQTT que incluye el token de acceso de cliente e iniciar un proxy local en el dispositivo remoto. Debe instalar y ejecutar el agente IoT en el dispositivo remoto si desea un túnel seguro para entregar el token de acceso del cliente mediante MQTT. El agente de IoT debe suscribirse al siguiente tema reservado de MQTT de IoT:

> **Note**
>
> Si desea entregar el token de acceso del cliente de destino al dispositivo remoto mediante métodos distintos de la suscripción al tema MQTT reservado, es posible que necesite un oyente del token de acceso del cliente (CAT) de destino y un proxy local. El oyente CAT debe funcionar con el mecanismo de entrega de los tokens de acceso al cliente que haya elegido y poder iniciar un proxy local en el modo de destino.

Fragmento de agente de IoT

El agente de IoT debe suscribirse al siguiente tema reservado de MQTT sobre IoT para poder recibir el mensaje MQTT e iniciar el proxy local:

`$aws/things/`*`thing-name`*`/tunnels/notify`

¿Dónde `thing-name` está el nombre de la AWS IoT cosa asociada al dispositivo remoto?

A continuación, se muestra un ejemplo de una carga útil de mensaje MQTT:

```
{
    "clientAccessToken": "destination-client-access-token",
    "clientMode": "destination",
    "region": "aws-region",
    "services": ["destination-service"]
}
```

Después de recibir un mensaje MQTT, el agente IoT debe iniciar un proxy local en el dispositivo remoto con los parámetros apropiados.

El siguiente código de Java muestra cómo utilizar el SDK de AWS IoT dispositivos y la biblioteca ProcessBuilderde Java para crear un agente de IoT sencillo que funcione con túneles seguros.

```
// Find the IoT device endpoint for your Cuenta de AWS
final String endpoint = iotClient.describeEndpoint(new
 DescribeEndpointRequest().withEndpointType("iot:Data-ATS")).getEndpointAddress();

// Instantiate the IoT Agent with your AWS credentials
final String thingName = "RemoteDeviceA";
final String tunnelNotificationTopic = String.format("$aws/things/%s/tunnels/notify",
 thingName);
final AWSIotMqttClient mqttClient = new AWSIotMqttClient(endpoint, thingName,
                 "your_aws_access_key", "your_aws_secret_key");

try {
    mqttClient.connect();
    final TunnelNotificationListener listener = new
 TunnelNotificationListener(tunnelNotificationTopic);
    mqttClient.subscribe(listener, true);
}
finally {
    mqttClient.disconnect();
}

private static class TunnelNotificationListener extends AWSIotTopic {
    public TunnelNotificationListener(String topic) {
        super(topic);
    }

    @Override
    public void onMessage(AWSIotMessage message) {
        try {
            // Deserialize the MQTT message
            final JSONObject json = new JSONObject(message.getStringPayload());

            final String accessToken = json.getString("clientAccessToken");
            final String region = json.getString("region");

            final String clientMode = json.getString("clientMode");
            if (!clientMode.equals("destination")) {
                throw new RuntimeException("Client mode " + clientMode + " in the MQTT
 message is not expected");
            }

            final JSONArray servicesArray = json.getJSONArray("services");
            if (servicesArray.length() > 1) {
```

```
                throw new RuntimeException("Services in the MQTT message has more than
 1 service");
            }
            final String service = servicesArray.get(0).toString();
            if (!service.equals("SSH")) {
                throw new RuntimeException("Service " + service + " is not supported");
            }

            // Start the destination local proxy in a separate process to connect to
 the SSH Daemon listening port 22
            final ProcessBuilder pb = new ProcessBuilder("localproxy",
                        "-t", accessToken,
                        "-r", region,
                        "-d", "localhost:22");
            pb.start();
        }
        catch (Exception e) {
            log.error("Failed to start the local proxy", e);
        }
    }
}
```

Control del acceso a los túneles

La tunelización segura proporciona acciones, recursos y claves contextuales de condiciones específicas del servicio para su uso en las políticas de permisos de IAM.

Requisitos previos de acceso al túnel

- Aprenda a proteger los AWS recursos mediante políticas de IAM.
- Aprenda a crear y evaluar condiciones de IAM.
- Aprenda a proteger los AWS recursos mediante etiquetas de recursos.

Políticas de acceso a túnel

Debe utilizar las siguientes políticas para autorizar los permisos de uso de la API de tunelización segura. Para obtener más información sobre AWS IoT la seguridad, consulteGestión de identidad y acceso para AWS IoT.

IoT: OpenTunnel

La acción de política `iot:OpenTunnel` concede un permiso a la entidad principal para llamar a OpenTunnel.

En el elemento `Resource` de la declaración de política de IAM:

- Especifique el ARN del túnel comodín:

  `arn:aws:iot:`*`aws-region`*`:`*`aws-account-id`*`:tunnel/*`
- Especifique un ARN de cosa para administrar el permiso `OpenTunnel` para cosas específicas de IoT:

  `arn:aws:iot:`*`aws-region`*`:`*`aws-account-id`*`:thing/`*`thing-name`*

Por ejemplo, la siguiente instrucción de política le permite abrir un túnel con el objeto de IoT llamado `TestDevice`.

```
{
    "Effect": "Allow",
    "Action": "iot:OpenTunnel",
    "Resource": [
        "arn:aws:iot:aws-region:aws-account-id:tunnel/*",
        "arn:aws:iot:aws-region:aws-account-id:thing/TestDevice"
    ]
}
```

La acción de política `iot:OpenTunnel` admite las siguientes claves de condición:

- `iot:ThingGroupArn`
- `iot:TunnelDestinationService`
- `aws:RequestTag/`*`tag-key`*
- `aws:SecureTransport`
- `aws:TagKeys`

La siguiente instrucción de política le permite abrir un túnel con el objeto si el objeto pertenece a un grupo de objetos con un nombre que comienza por `TestGroup` y el servicio de destino configurado en el túnel es SSH.

```
{
    "Effect": "Allow",
    "Action": "iot:OpenTunnel",
    "Resource": [
        "arn:aws:iot:aws-region:aws-account-id:tunnel/*"
    ],
    "Condition": {
        "ForAnyValue:StringLike": {
            "iot:ThingGroupArn": [
                "arn:aws:iot:aws-region:aws-account-id:thinggroup/TestGroup*"
            ]
        },
        "ForAllValues:StringEquals": {
            "iot:TunnelDestinationService": [
                "SSH"
            ]
        }
    }
}
```

También puede utilizar etiquetas de recursos para controlar los permisos para abrir túneles. Por ejemplo, la siguiente instrucción de política permite abrir un túnel si la clave de etiqueta `Owner` está presente con un valor de `Admin` y no se especifica ninguna otra etiqueta. Para obtener más información acerca del uso de etiquetas, consulte Etiquetar sus recursos AWS IoT.

```
{
    "Effect": "Allow",
    "Action": "iot:OpenTunnel",
    "Resource": [
        "arn:aws:iot:aws-region:aws-account-id:tunnel/*"
    ],
    "Condition": {
        "StringEquals": {
            "aws:RequestTag/Owner": "Admin"
        },
        "ForAllValues:StringEquals": {
            "aws:TagKeys": "Owner"
        }
    }
}
```

IoT: RotateTunnelAccessToken

La acción de política `iot:RotateTunnelAccessToken` concede un permiso a la entidad principal para llamar a RotateTunnelAccessToken.

En el elemento `Resource` de la declaración de política de IAM:

- Especifique un ARN de túnel totalmente cualificado:

  `arn:aws:iot:`*`aws-region`*`: `*`aws-account-id`*`:tunnel/`*`tunnel-id`*

  También puede utilizar el ARN del túnel comodín:

  `arn:aws:iot:`*`aws-region`*`:`*`aws-account-id`*`:tunnel/*`

- Especifique un ARN de cosa para administrar el permiso `RotateTunnelAccessToken` para cosas específicas de IoT:

  `arn:aws:iot:`*`aws-region`*`:`*`aws-account-id`*`:thing/`*`thing-name`*

Por ejemplo, la siguiente declaración de política le permite rotar el token de acceso de origen de un túnel o el token de acceso de destino de un cliente para el elemento de IoT denominado `TestDevice`.

```
{
    "Effect": "Allow",
    "Action": "iot:RotateTunnelAccessToken",
    "Resource": [
        "arn:aws:iot:aws-region:aws-account-id:tunnel/*",
        "arn:aws:iot:aws-region:aws-account-id:thing/TestDevice"
    ]
}
```

La acción de política `iot:RotateTunnelAccessToken` admite las siguientes claves de condición:

- `iot:ThingGroupArn`
- `iot:TunnelDestinationService`
- `iot:ClientMode`
- `aws:SecureTransport`

La siguiente declaración de política le permite rotar el token de acceso de destino a la cosa si la cosa pertenece a un grupo de cosas con un nombre que empiece por `TestGroup`, el servicio de destino configurado en el túnel es SSH, y el cliente está en modo `DESTINATION`.

```
{
    "Effect": "Allow",
    "Action": "iot:RotateTunnelAccessToken",
    "Resource": [
        "arn:aws:iot:aws-region:aws-account-id:tunnel/*"
    ],
    "Condition": {
        "ForAnyValue:StringLike": {
            "iot:ThingGroupArn": [
                "arn:aws:iot:aws-region:aws-account-id:thinggroup/TestGroup*"
            ]
        },
        "ForAllValues:StringEquals": {
            "iot:TunnelDestinationService": [
                "SSH"
            ],
            "iot:ClientMode": "DESTINATION"
        }
    }
}
```

IoT: DescribeTunnel

La acción de política `iot:DescribeTunnel` concede un permiso a la entidad principal para llamar a DescribeTunnel.

En el elemento `Resource` de la declaración de política de IAM, especifique un ARN de túnel totalmente cualificado:

`arn:aws:iot:aws-region: aws-account-id:tunnel/tunnel-id`

También puede utilizar el ARN comodín:

`arn:aws:iot:aws-region:aws-account-id:tunnel/*`

La acción de política `iot:DescribeTunnel` admite las siguientes claves de condición:

- `aws:ResourceTag/tag-key`

- `aws:SecureTransport`

La siguiente instrucción de política le permite llamar a `DescribeTunnel` si el túnel solicitado está etiquetado con la clave `Owner` con un valor de `Admin`.

```
{
    "Effect": "Allow",
    "Action": "iot:DescribeTunnel",
    "Resource": [
        "arn:aws:iot:aws-region:aws-account-id:tunnel/*"
    ],
    "Condition": {
        "StringEquals": {
            "aws:ResourceTag/Owner": "Admin"
        }
    }
}
```

IoT: ListTunnels

La acción de política `iot:ListTunnels` concede un permiso a la entidad principal para llamar a ListTunnels.

En el elemento `Resource` de la declaración de política de IAM:

- Especifique el ARN del túnel comodín:

  `arn:aws:iot:aws-region:aws-account-id:tunnel/*`
- Especifique un ARN de cosa para administrar el permiso `ListTunnelss` en determinadas cosas de IoT:

  `arn:aws:iot:aws-region:aws-account-id:thing/thing-name`

La acción de política `iot:ListTunnels` admite la clave de condición `aws:SecureTransport`:

La siguiente instrucción de política le permite mostrar los túneles del objeto denominado `TestDevice`.

```
{
    "Effect": "Allow",
```

```
    "Action": "iot:ListTunnels",
    "Resource": [
        "arn:aws:iot:aws-region:aws-account-id:tunnel/*",
        "arn:aws:iot:aws-region:aws-account-id:thing/TestDevice"
    ]
}
```

IoT: ListTagsForResource

La acción de política `iot:ListTagsForResource` concede un permiso a la entidad principal para llamar a `ListTagsForResource`.

En el elemento `Resource` de la declaración de política de IAM, especifique un ARN de túnel totalmente cualificado:

`arn:aws:iot:aws-region: aws-account-id:tunnel/tunnel-id`

También puede utilizar el ARN del túnel comodín:

`arn:aws:iot:aws-region:aws-account-id:tunnel/*`

La acción de política `iot:ListTagsForResource` admite la clave de condición `aws:SecureTransport`:

IoT: CloseTunnel

La acción de política `iot:CloseTunnel` concede un permiso a la entidad principal para llamar a CloseTunnel.

En el elemento `Resource` de la declaración de política de IAM, especifique un ARN de túnel totalmente cualificado:

`arn:aws:iot:aws-region: aws-account-id:tunnel/tunnel-id`

También puede utilizar el ARN del túnel comodín:

`arn:aws:iot:aws-region:aws-account-id:tunnel/*`

La acción de política `iot:CloseTunnel` admite las siguientes claves de condición:

- `iot:Delete`
- `aws:ResourceTag/tag-key`
- `aws:SecureTransport`

La siguiente instrucción de política le permite llamar a `CloseTunnel` si el parámetro `Delete` de la solicitud es `false` y el túnel solicitado está etiquetado con la clave `Owner` y el valor `QATeam`.

```
{
    "Effect": "Allow",
    "Action": "iot:CloseTunnel",
    "Resource": [
        "arn:aws:iot:aws-region:aws-account-id:tunnel/*"
    ],
    "Condition": {
        "Bool": {
            "iot:Delete": "false"
        },
        "StringEquals": {
            "aws:ResourceTag/Owner": "QATeam"
        }
    }
}
```

IoT: TagResource

La acción de política `iot:TagResource` concede un permiso a la entidad principal para llamar a `TagResource`.

En el elemento `Resource` de la declaración de política de IAM, especifique un ARN de túnel totalmente cualificado:

`arn:aws:iot:aws-region: aws-account-id:tunnel/tunnel-id`

También puede utilizar el ARN del túnel comodín:

`arn:aws:iot:aws-region:aws-account-id:tunnel/*`

La acción de política `iot:TagResource` admite la clave de condición `aws:SecureTransport`:

IoT: UntagResource

La acción de política `iot:UntagResource` concede un permiso a la entidad principal para llamar a `UntagResource`.

En el elemento `Resource` de la declaración de política de IAM, especifique un ARN de túnel totalmente cualificado:

`arn:aws:iot:aws-region: aws-account-id:tunnel/tunnel-id`

También puede utilizar el ARN del túnel comodín:

`arn:aws:iot:aws-region:aws-account-id:tunnel/*`

La acción de política `iot:UntagResource` admite la clave de condición `aws:SecureTransport`:

Resolver problemas de conectividad de túneles AWS IoT seguros mediante la rotación de los tokens de acceso de los clientes

Al utilizar la tunelización AWS IoT segura, es posible que se produzcan problemas de conectividad incluso si el túnel está abierto. En las siguientes secciones, se muestran algunos posibles problemas y cómo resolverlos rotando los tokens de acceso del cliente. Para rotar el token de acceso del cliente (CAT), utilice la RotateTunnelAccessTokenAPI o el. rotate-tunnel-access-token AWS CLI En función de si se produce un error al utilizar el cliente en el modo de origen o de destino, puede rotar el CAT en el modo de origen o de destino, o en ambos.

> **Note**
>
> - Si no está seguro de si el CAT debe girar en el origen o en el destino, puede girar el CAT tanto en el origen como en el destino ajustando `ClientMode` a TODO cuando utilice la API `RotateTunnelAccessToken`.
> - La rotación del CAT no prolonga la duración del túnel. Por ejemplo, supongamos que la duración del túnel es de 12 horas y que el túnel ya lleva abierto 4 horas. Cuando rote los tokens de acceso, los nuevos tokens que se generen solo podrán utilizarse durante las 8 horas restantes.

Temas

- Error de token de acceso al cliente no válido
- Error de discordancia del token del cliente
- Problemas de conectividad de dispositivos remotos

Error de token de acceso al cliente no válido

Al utilizar la tunelización AWS IoT segura, se puede producir un error de conexión si se utiliza el mismo token de acceso de cliente (CAT) para volver a conectarse al mismo túnel. En este caso, el

proxy local no puede conectarse al proxy host de tunelización segura. Si utiliza un cliente en el modo de origen, puede aparecer el siguiente mensaje de error:

```
Invalid access token: The access token was previously used and cannot be used again
```

El error se produce porque el proxy local solo puede usar el token de acceso de cliente (CAT) una vez y, entonces, deja de ser válido. Para resolver este error, gire el token de acceso del cliente al modo `SOURCE` para generar un nuevo CAT para la fuente. Para obtener un ejemplo que muestra cómo girar el CAT de origen, consulte Rote la fuente (ejemplo de CAT).

Error de discordancia del token del cliente

> **Note**
>
> No se recomienda usar tokens de cliente para reutilizar el CAT. En su lugar, le recomendamos que utilice la API `RotateTunnelAccessToken` para rotar los tokens de acceso del cliente y volver a conectarse al túnel.

Si utiliza tokens de cliente, puede reutilizar el CAT para volver a conectarse al túnel. Para reutilizar el CAT, debe proporcionar el token de cliente junto con el CAT la primera vez que se conecte a un túnel seguro. La tunelización segura almacena el token del cliente, por lo que, para los siguientes intentos de conexión con el mismo token, también se debe proporcionar el token del cliente. Para obtener más información sobre el uso de los tokens de cliente, consulte la implementación de referencia del proxy local en. GitHub

Si está utilizando un cliente en modo fuente, es posible que aparezca el siguiente error:

```
Invalid client token: The provided client token does not match the client token
    that was previously set.
```

El error se produce porque el token de cliente proporcionado no coincide con el token de cliente que se proporcionó con el CAT al acceder al túnel. Para resolver este error, gire el CAT en el modo `SOURCE` para generar un nuevo CAT para la fuente. A continuación se muestra un ejemplo:

Rote la fuente (ejemplo de CAT)

A continuación, se muestra un ejemplo de cómo ejecutar la `RotateTunnelAccessToken` API en el modo `SOURCE` para generar un nuevo CAT para la fuente:

```
aws iotsecuretunneling rotate-tunnel-access-token \
    --region <region> \
    --tunnel-id <tunnel-id> \
    --client-mode SOURCE
```

La ejecución de este comando genera un nuevo token de acceso a la fuente y devuelve el ARN de su túnel.

```
{
    "sourceAccessToken": "<source-access-token>",
    "tunnelArn": "arn:aws:iot:<region>:<account-id>:tunnel/<tunnel-id>"
}
```

Ahora puede utilizar el nuevo token de fuente para conectar el proxy local en modo fuente.

```
export AWSIOT_TUNNEL_ACCESS_TOKEN=<source-access-token>
./localproxy -r <region> -s <port>
```

A continuación se muestra un ejemplo de resultado de la ejecución del proxy local en modo .

```
...

[info]    Starting proxy in source mode
...
[info]    Successfully established websocket connection with proxy server ...
[info]    Listening for new connection on port <port>
...
```

Problemas de conectividad de dispositivos remotos

Al utilizar la tunelización AWS IoT segura, el dispositivo podría desconectarse inesperadamente aunque el túnel esté abierto. Para identificar si un dispositivo sigue conectado al túnel, puedes usar la DescribeTunnelAPI o el describe-tunnel. AWS CLI

Un dispositivo puede desconectarse por varios motivos. Para resolver el problema de conectividad, puede girar el CAT hacia el destino si el dispositivo se desconectó por los siguientes motivos posibles:

- El CAT del destino dejó de ser válido.

- El token no se entregó al dispositivo a través del tema MQTT reservado para la tunelización segura:

  `$aws/things/<thing-name>/tunnels/notify`

El siguiente ejemplo muestra cómo resolver este problema:

Ejemplo de rotación (CAT) de destino

Considere un dispositivo remoto, *<RemoteThing1>*. Para abrir un túnel para esa cosa, puede utilizar el comando siguiente:

```
aws iotsecuretunneling open-tunnel \
    --region <region> \
    --destination-config thingName=<RemoteThing1>,services=SSH
```

Al ejecutar este comando, se generan los detalles del túnel y el CAT para el origen y el destino.

```
{
    "sourceAccessToken": "<source-access-token>",
    "destinationAccessToken": "<destination-access-token>",
    "tunnelId": "<tunnel-id>",
    "tunnelArn": "arn:aws:iot:<region>:<account-id>:tunnel/tunnel-id"
}
```

Sin embargo, cuando utilizas la DescribeTunnelAPI, el resultado indica que el dispositivo se ha desconectado, como se muestra a continuación:

```
aws iotsecuretunneling describe-tunnel \
    --tunnel-id <tunnel-id> \
    --region <region>
```

Al ejecutar este comando, se muestra que el dispositivo aún no está conectado.

```
{
    "tunnel": {
        ...
        "destinationConnectionState": {
            "status": "DISCONNECTED"
```

```
        },
        ...
    }
}
```

Para resolver este error, ejecute la API `RotateTunnelAccessToken` con el cliente en modo `DESTINATION` y con las configuraciones del destino. Al ejecutar este comando, se revoca el token de acceso anterior, se genera un token nuevo y se reenvía este token al tema MQTT:

`$aws/things/<thing-name>/tunnels/notify`

```
aws iotsecuretunneling rotate-tunnel-access-token \
    --tunnel-id <tunnel-id> \
    --client-mode DESTINATION \
    --destination-config thingName=<RemoteThing1>,services=SSH \
    --region <region>
```

La ejecución de este comando genera el nuevo token de acceso como se muestra a continuación. A continuación, el token se entrega al dispositivo mediante MQTT para conectarse al túnel, si el agente del dispositivo está configurado correctamente.

```
{
    "destinationAccessToken": "destination-access-token",
    "tunnelArn": "arn:aws:iot:region:account-id:tunnel/tunnel-id"
}
```

Aprovisionamiento de dispositivos

AWS proporciona varias formas diferentes de aprovisionar un dispositivo e instalar certificados de cliente únicos en él. En esta sección se describe cada forma y cómo seleccionar la mejor para su solución de IoT. Estas opciones se describen en detalle en el documento técnico titulado Fabricación y aprovisionamiento de dispositivos con certificados X.509 en AWS IoT Core.

Seleccionar la opción que mejor se adapte a su situación

- Puede instalar certificados en dispositivos de IoT antes de que se entreguen

  Si puede instalar de forma segura certificados de cliente únicos en sus dispositivos de IoT antes de que se entreguen para su uso por el usuario final, querrá usar el just-in-timeaprovisionamiento (JITP) o el just-in-timeregistro (JITR).

  Con el JITP y el JITR, la autoridad de certificación (CA) utilizada para firmar el certificado del dispositivo se registra AWS IoT y se reconoce AWS IoT cuando el dispositivo se conecta por primera vez. El dispositivo se aprovisiona AWS IoT en su primera conexión mediante los detalles de su plantilla de aprovisionamiento.

  Para obtener más información sobre el aprovisionamiento de un solo objeto, el aprovisionamiento JITP, el aprovisionamiento JITR y el aprovisionamiento masivo de dispositivos con certificados únicos, consulte the section called “Aprovisionamiento de dispositivos que tienen certificados de dispositivo”.
- Los usuarios finales o los instaladores pueden usar una aplicación para instalar certificados en sus dispositivos de IoT

  Si no puede instalar de forma segura certificados únicos de cliente en su dispositivo de IoT antes de entregarlos al usuario final, pero el usuario final o un instalador pueden usar una aplicación para registrar los dispositivos e instalar los certificados de dispositivo únicos, es recomendable utilizar el proceso de aprovisionamiento por usuario de confianza.

  El uso de un usuario de confianza, como un usuario final o un instalador con una cuenta conocida, puede simplificar el proceso de fabricación del dispositivo. En lugar de un certificado de cliente único, los dispositivos tienen un certificado temporal que permite al dispositivo conectarse solo AWS IoT durante 5 minutos. Durante ese periodo de 5 minutos, el usuario de confianza obtiene un certificado de cliente único con una vida útil más larga y lo instala en el dispositivo.

La vida útil limitada del certificado de reclamación minimiza el riesgo de que el certificado se vea comprometido.

Para obtener más información, consulte the section called “Aprovisionamiento por usuario de confianza”.

- Los usuarios finales NO PUEDEN usar una aplicación para instalar certificados en sus dispositivos de IoT

  Si ninguna de las opciones anteriores funciona en su solución de IoT, el proceso de aprovisionamiento por reclamación es una opción. Con este proceso, sus dispositivos de IoT disponen de un certificado de reclamación que comparten otros dispositivos de la flota. La primera vez que un dispositivo se conecta con un certificado de reclamación, lo AWS IoT registra mediante su plantilla de aprovisionamiento y emite al dispositivo su certificado de cliente exclusivo para poder acceder a AWS IoTél posteriormente.

  Esta opción permite el aprovisionamiento automático de un dispositivo cuando se conecta a él AWS IoT, pero podría suponer un riesgo mayor en caso de que el certificado de reclamación se viera comprometido. Si un certificado de reclamación se ve comprometido, puede desactivarlo. La desactivación del certificado de reclamación impide que todos los dispositivos con ese certificado de reclamación se registren en el futuro. Sin embargo, la desactivación del certificado de reclamación no bloquea los dispositivos que ya se han aprovisionado.

  Para obtener más información, consulte the section called “Aprovisionamiento por reclamación”.

Aprovisionamiento de dispositivos en AWS IoT

Al aprovisionar un dispositivo AWS IoT, debe crear recursos para que sus dispositivos y AWS IoT puedan comunicarse de forma segura. Se pueden crear otros recursos que le ayuden a administrar su flota de dispositivos. Durante el proceso de aprovisionamiento se pueden crear los siguientes recursos:

- Un objeto de IoT.

  Las cosas de IoT son entradas en el registro de AWS IoT dispositivos. Cada objeto tiene un nombre único y un conjunto de atributos, y está asociado con un dispositivo físico. Los objetos se pueden definir usando un tipo de objeto o agruparse en grupos de objetos. Para obtener más información, consulte Administrar dispositivos con AWS IoT.

Aunque no es necesario, la creación de objetos permite administrar la flota de dispositivos de manera más eficaz mediante la búsqueda de dispositivos por tipo de objeto, grupo de objetos y atributos de objetos. Para obtener más información, consulte Indexación de flotas.

> **Note**
>
> Para indexar los datos de estado de conectividad de su objeto, aprovisione su objeto y configúrelo de manera que el nombre del objeto coincida con el ID de cliente utilizado en la solicitud Connect.

- Un certificado X.509

  Los dispositivos utilizan certificados X.509 para realizar la autenticación mutua. AWS IoT Puede registrar un certificado existente o hacer que AWS IoT genere y registre uno nuevo para usted. Un certificado se asocia a un dispositivo asociándolo al objeto que representa el dispositivo. También debe copiar el certificado y la clave privada asociada en el dispositivo. Los dispositivos presentan el certificado al conectarse a AWS IoT. Para obtener más información, consulte Autenticación.

- Una política de IoT

  Las políticas de IoT definen qué operaciones puede realizar un dispositivo en AWS IoT. Las políticas de IoT se asocian a certificados de dispositivo. Cuando un dispositivo presenta el certificado AWS IoT, se le conceden los permisos especificados en la política. Para obtener más información, consulte Autorización. Cada dispositivo necesita un certificado para comunicarse con AWS IoT.

AWS IoT admite el aprovisionamiento automatizado de flotas mediante plantillas de aprovisionamiento. Las plantillas de aprovisionamiento describen los recursos necesarios para AWS IoT aprovisionar el dispositivo. Las plantillas contienen variables que permiten utilizar una plantilla para aprovisionar varios dispositivos. Cuando aprovisione un dispositivo, especifique valores para las variables específicas de este utilizando un diccionario o mapa. Para aprovisionar otro dispositivo, especifique nuevos valores en el diccionario.

Puede utilizar el aprovisionamiento automatizado tanto si sus dispositivos tienen certificados únicos (y su clave privada asociada) como si no.

Aprovisionamiento de flota APIs

Existen varias categorías que se APIs utilizan en el aprovisionamiento de flotas:

- Estas funciones de plano de control crean y administran las plantillas de aprovisionamiento de flotas y configuran políticas de usuario de confianza.
  - CreateProvisioningTemplate
  - CreateProvisioningTemplateVersion
  - DeleteProvisioningTemplate
  - DeleteProvisioningTemplateVersion
  - DescribeProvisioningTemplate
  - DescribeProvisioningTemplateVersion
  - ListProvisioningTemplates
  - ListProvisioningTemplateVersions
  - UpdateProvisioningTemplate
- Los usuarios de confianza pueden utilizar esta función de plano de control para generar una notificación de incorporación temporal. Esta reclamación temporal se pasa al dispositivo durante la configuración de wifi o un método similar.
  - CreateProvisioningClaim
- La API MQTT utilizada durante el proceso de aprovisionamiento por los dispositivos con un certificado de notificación de aprovisionamiento incrustado en un dispositivo o pasado a él por un usuario de confianza.
  - the section called “CreateCertificateFromCsr”
  - the section called “CreateKeysAndCertificate”
  - the section called “RegisterThing”

Aprovisionamiento de dispositivos que no tienen certificados de dispositivo mediante el aprovisionamiento de flotas

Al utilizar el aprovisionamiento de AWS IoT flotas, AWS IoT puede generar y entregar de forma segura certificados de dispositivos y claves privadas a sus dispositivos cuando se conectan a ellos AWS IoT por primera vez. AWS IoT proporciona certificados de cliente firmados por la autoridad de certificación (CA) raíz de Amazon.

Existen dos formas de utilizar el aprovisionamiento de flotas:

- Aprovisionamiento por reclamación
- Aprovisionamiento por usuario de confianza

Aprovisionamiento por reclamación

Los dispositivos se pueden fabricar con un certificado de notificación de aprovisionamiento y una clave privada (que son credenciales de uso especial) incrustados. Si estos certificados están registrados AWS IoT, el servicio puede cambiarlos por certificados de dispositivo únicos que el dispositivo puede usar para sus operaciones habituales. El proceso consta de los pasos siguientes:

Antes de entregar el dispositivo

1. Llame a `CreateProvisioningTemplate` para crear una plantilla de aprovisionamiento. Esta API devuelve un ARN de plantilla. Para obtener más información, consulte API de MQTT de aprovisionamiento de dispositivos.

   También puede crear una plantilla de aprovisionamiento de flota en la AWS IoT consola.

   a. En el panel de navegación, seleccione el menú desplegable Conectar muchos dispositivos. A continuación, elija Conectar muchos dispositivos.
   b. Seleccione Crear plantilla de aprovisionamiento.
   c. Elija la opción Situación de aprovisionamiento que mejor se adapte a los procesos de instalación. A continuación, elija Siguiente.
   d. Complete el flujo de trabajo de la plantilla.
2. Cree los certificados y las claves privadas asociadas que se utilizarán como certificados de reclamación de aprovisionamiento.
3. Registre estos certificados AWS IoT y asocie una política de IoT que restrinja el uso de los certificados. El siguiente ejemplo de política de IoT restringe el uso del certificado asociado a esta política para aprovisionar dispositivos.

```
{
    "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/$aws/certificates/create/
*",
    "arn:aws:iot:us-east-1:123456789012:topic/$aws/provisioning-
templates/templateName/provision/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "iot:Subscribe",
            "Resource": [
            "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/certificates/
create/*",
    "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/provisioning-
templates/templateName/provision/*"
            ]
        }
    ]
}
```

4. Al aprovisionar dispositivos, otorga al AWS IoT servicio permiso para crear o actualizar recursos de IoT, como cosas y certificados, en tu cuenta. Para ello, asocie la política `AWSIoTThingsRegistration` gestionada a una función de IAM (denominada función de aprovisionamiento) que confíe en el principal del servicio. AWS IoT
5. Fabricar el dispositivo con el certificado de reclamación de aprovisionamiento incrustado de forma segura en él.

El dispositivo ya está listo para ser entregado a donde se instalará para su uso.

> **Important**
>
> Las claves privadas de la notificación de aprovisionamiento deben estar protegidas en todo momento, también en el dispositivo. Le recomendamos que utilice AWS IoT CloudWatch métricas y registros para supervisar los indicios de un uso indebido. Si detecta un uso incorrecto, desactive el certificado de reclamación de aprovisionamiento para que no se pueda utilizar para el aprovisionamiento de dispositivos.

Para inicializar el dispositivo para su uso

1. El dispositivo lo utiliza SDK de dispositivos, SDK para móviles y cliente de dispositivo de AWS IoT para conectarse y autenticarse AWS IoT mediante el certificado de notificación de aprovisionamiento que está instalado en el dispositivo.

   > **Note**
   >
   > Por motivos de seguridad, el `certificateOwnershipToken` devuelto por `CreateCertificateFromCsr` y `CreateKeysAndCertificate` caduca al cabo de una hora. Debe llamarse a `RegisterThing` antes de que `certificateOwnershipToken` venza. Si el certificado creado por `CreateCertificateFromCsr` o `CreateKeysAndCertificate` no se ha activado y no se ha asociado a una política o a un objeto en el momento de caducar el token, el certificado se eliminará. Si el token caduca, el dispositivo puede volver a llamar a `CreateCertificateFromCsr` o a `CreateKeysAndCertificate` para generar un nuevo certificado.

2. El dispositivo obtiene un certificado permanente y una clave privada mediante una de estas opciones. El dispositivo utilizará el certificado y la clave para todas las autenticaciones futuras AWS IoT.

   a. Llame `CreateKeysAndCertificate`para crear un certificado y una clave privada nuevos mediante la autoridad de AWS certificación.

      O

   b. Llame a `CreateCertificateFromCsr` para generar un certificado desde una solicitud de firma de certificado que mantiene su clave privada segura.

3. Desde el dispositivo, llame a `RegisterThing` para registrar el dispositivo con AWS IoT y crear recursos en la nube.

   El servicio de aprovisionamiento de flotas utiliza una plantilla de aprovisionamiento para definir y crear recursos en la nube como objetos de IoT. La plantilla puede especificar los atributos y grupos a los que pertenece el objeto. Los grupos de objetos deben existir para poder agregarles el nuevo objeto.
4. Después de guardar el certificado permanente en el dispositivo, este debe desconectarse de la sesión que inició con el certificado de reclamación de aprovisionamiento y volver a conectarse con el certificado permanente.

El dispositivo ya está listo para comunicarse con normalidad con él AWS IoT.

Aprovisionamiento por usuario de confianza

En muchos casos, un dispositivo se conecta AWS IoT por primera vez cuando un usuario de confianza, como un usuario final o un técnico de instalación, utiliza una aplicación móvil para configurar el dispositivo en su ubicación de despliegue.

> **Important**
>
> Debe administrar el acceso y el permiso del usuario de confianza para realizar este procedimiento. Una forma de hacerlo es proporcionar y mantener una cuenta para el usuario de confianza que lo autentica y le otorga acceso a las características y las operaciones API de AWS IoT necesarias para realizar este procedimiento.

Antes de entregar el dispositivo

1. Llame `CreateProvisioningTemplate`para crear una plantilla de aprovisionamiento y devuelva su *`templateArn`* y*`templateName`*.
2. Cree un rol de IAM que utilice un usuario de confianza para iniciar el proceso de aprovisionamiento. La plantilla de aprovisionamiento solo permite a ese usuario aprovisionar un dispositivo. Por ejemplo:

```
{
    "Effect": "Allow",
    "Action": [
```

```
            "iot:CreateProvisioningClaim"
        ],
        "Resource": [
            "arn:aws:iot:aws-region:aws-account-id:provisioningtemplate/templateName"
        ]
}
```

3. Al aprovisionar dispositivos, otorga al AWS IoT servicio permiso para crear o actualizar recursos de IoT, como cosas y certificados, en tu cuenta. Para ello, asocie la política `AWSIoTThingsRegistration` gestionada a una función de IAM (denominada función de aprovisionamiento) que confíe en el principal del servicio. AWS IoT
4. Proporcione los medios para identificar a sus usuarios de confianza, por ejemplo, proporcionándoles una cuenta que pueda autenticarlos y autorizar sus interacciones con las operaciones de la AWS API necesarias para registrar sus dispositivos.

Para inicializar el dispositivo para su uso

1. El usuario de confianza inicia sesión en su aplicación móvil o servicio web de aprovisionamiento.
2. La aplicación móvil o la aplicación web utiliza el rol de IAM y llama a `CreateProvisioningClaim` para obtener un certificado de reclamación de aprovisionamiento temporal de AWS IoT.

> **Note**
>
> Por motivos de seguridad, el certificado de reclamación de aprovisionamiento temporal devuelto por `CreateProvisioningClaim` solo es válido durante cinco minutos. Los pasos siguientes deben devolver correctamente un certificado válido antes de que caduque el certificado de reclamación de aprovisionamiento temporal. Los certificados de reclamación de aprovisionamiento temporal no aparecen en la lista de certificados de su cuenta.

3. La aplicación móvil o la aplicación web proporciona el certificado de reclamación de aprovisionamiento temporal al dispositivo junto con cualquier información de configuración necesaria, como credenciales Wi-Fi.
4. El dispositivo utiliza el certificado de reclamación de aprovisionamiento temporal para conectarse a AWS IoT mediante el SDK de dispositivos, SDK para móviles y cliente de dispositivo de AWS IoT.

5. El dispositivo obtiene un certificado permanente y una clave privada mediante una de estas opciones a los cinco minutos de conectarse AWS IoT con el certificado de notificación de aprovisionamiento temporal. El dispositivo utilizará el certificado y la clave que estas opciones devuelven para todas las autenticaciones futuras AWS IoT.

   a. Llame `CreateKeysAndCertificate`para crear un certificado y una clave privada nuevos mediante la autoridad de AWS certificación.

      O

   b. Llame a `CreateCertificateFromCsr` para generar un certificado desde una solicitud de firma de certificado que mantiene su clave privada segura.

   > **Note**
   >
   > Recuerde `CreateKeysAndCertificate`o `CreateCertificateFromCsr`debe devolver un certificado válido en un plazo de cinco minutos a partir de la fecha de conexión AWS IoT con el certificado de solicitud de aprovisionamiento temporal.

6. El dispositivo llama `RegisterThing`para registrarlo AWS IoT y crear recursos en la nube.

   El servicio de aprovisionamiento de flotas utiliza una plantilla de aprovisionamiento para definir y crear recursos en la nube como objetos de IoT. La plantilla puede especificar los atributos y grupos a los que pertenece el objeto. Los grupos de objetos deben existir para poder agregarles el nuevo objeto.

7. Después de guardar el certificado permanente en el dispositivo, el dispositivo debe desconectarse de la sesión que inició con el certificado de reclamación de aprovisionamiento temporal y volver a conectarse con el certificado permanente.

El dispositivo ya está listo para comunicarse con normalidad AWS IoT.

Uso de ganchos de aprovisionamiento previo con la CLI AWS

El siguiente procedimiento crea una plantilla de aprovisionamiento con enlaces de preaprovisionamiento. La función de Lambda utilizada aquí es un ejemplo que se puede modificar.

Para crear y aplicar un enlace de preaprovisionamiento a una plantilla de aprovisionamiento

1. Cree una función de Lambda que tenga una entrada y una salida definidas. Las funciones de Lambda son altamente personalizables. `allowProvisioning` y `parameterOverrides` son necesarias para crear enlaces previos al aprovisionamiento. Para obtener más información sobre la creación de funciones Lambda, consulte Utilización AWS Lambda con la interfaz de línea de AWS comandos.

   El siguiente es un ejemplo de una salida de función de Lambda:

   ```
   {
     "allowProvisioning": True,
     "parameterOverrides": {
       "incomingKey0": "incomingValue0",
       "incomingKey1": "incomingValue1"
     }
   }
   ```

2. AWS IoT utiliza políticas basadas en recursos para llamar a Lambda, por lo que debe conceder AWS IoT permiso para llamar a su función Lambda.

   > **Important**
   >
   > Asegúrese de incluir las claves contextuales de condición global `source-arn` o `source-account` en las de las políticas adjuntas a su acción de Lambda para evitar la manipulación de permisos. Para obtener más información acerca de este tema, consulte Prevención de la sustitución confusa entre servicios.

   El siguiente ejemplo utiliza add-permission para dar permiso de IoT a su Lambda.

   ```
   aws lambda add-permission \
       --function-name myLambdaFunction \
       --statement-id iot-permission \
       --action lambda:InvokeFunction \
       --principal iot.amazonaws.com
   ```

3. Añada un enlace de preaprovisionamiento a una plantilla mediante el comando o. create-provisioning-templateupdate-provisioning-template

El siguiente ejemplo de CLI utiliza el create-provisioning-templatepara crear una plantilla de aprovisionamiento que tiene ganchos de preaprovisionamiento:

```
aws iot create-provisioning-template \
    --template-name myTemplate \
    --provisioning-role-arn arn:aws:iam:us-east-1:1234564789012:role/myRole \
    --template-body file://template.json \
    --pre-provisioning-hook file://hooks.json
```

El resultado de este comando tendrá un aspecto similar al siguiente.

```
{
    "templateArn": "arn:aws:iot:us-east-1:1234564789012:provisioningtemplate/
myTemplate",
    "defaultVersionId": 1,
    "templateName": myTemplate
}
```

También puede cargar un parámetro desde un archivo en lugar de escribirlo todo como un valor de parámetro de línea de comandos para ahorrar tiempo. Para obtener más información, consulte Carga de parámetros de AWS CLI desde un archivo. A continuación se muestra el parámetro `template` en formato JSON expandido:

```
{
    "Parameters" : {
        "DeviceLocation": {
            "Type": "String"
        }
    },
    "Mappings": {
        "LocationTable": {
            "Seattle": {
                "LocationUrl": "https://example.aws"
            }
        }
    },
    "Resources" : {
        "thing" : {
            "Type" : "AWS::IoT::Thing",
            "Properties" : {
```

```
                "AttributePayload" : {
                    "version" : "v1",
                    "serialNumber" : "serialNumber"
                },
                "ThingName" : {"Fn::Join":["",["ThingPrefix_",
{"Ref":"SerialNumber"}]]},
                "ThingTypeName" : {"Fn::Join":["",["ThingTypePrefix_",
{"Ref":"SerialNumber"}]]},
                "ThingGroups" : ["widgets", "WA"],
                "BillingGroup": "BillingGroup"
            },
            "OverrideSettings" : {
                "AttributePayload" : "MERGE",
                "ThingTypeName" : "REPLACE",
                "ThingGroups" : "DO_NOTHING"
            }
        },
        "certificate" : {
            "Type" : "AWS::IoT::Certificate",
            "Properties" : {
                "CertificateId": {"Ref": "AWS::IoT::Certificate::Id"},
                "Status" : "Active",
                "ThingPrincipalType" : "EXCLUSIVE_THING"
            }
        },
        "policy" : {
            "Type" : "AWS::IoT::Policy",
            "Properties" : {
                "PolicyDocument" : {
                    "Version": "2012-10-17",
                    "Statement": [{
                        "Effect": "Allow",
                        "Action":["iot:Publish"],
                        "Resource": ["arn:aws:iot:us-east-1:504350838278:topic/foo/
bar"]
                    }]
                }
            }
        }
    },
    "DeviceConfiguration": {
        "FallbackUrl": "https://www.example.com/test-site",
        "LocationUrl": {
```

```
            "Fn::FindInMap": ["LocationTable",{"Ref": "DeviceLocation"},
 "LocationUrl"]}
    }
}
```

A continuación se muestra el parámetro `pre-provisioning-hook` en formato JSON expandido:

```
{
     "targetArn" : "arn:aws:lambda:us-
east-1:765219403047:function:pre_provisioning_test",
     "payloadVersion" : "2020-04-01"
}
```

Aprovisionamiento de dispositivos que tienen certificados de dispositivo

AWS IoT proporciona tres formas de aprovisionar dispositivos cuando ya tienen un certificado de dispositivo (y una clave privada asociada):

- Aprovisionamiento de un solo objeto con una plantilla de aprovisionamiento. Esta es una buena opción si solo necesita aprovisionar los dispositivos de uno en uno.
- Just-in-time aprovisionamiento (JITP) con una plantilla que aprovisiona un dispositivo cuando se conecta por primera vez a. AWS IoT Esta es una buena opción si necesita registrar un gran número de dispositivos, pero no dispone de información sobre ellos que se pueda incluir en una lista de aprovisionamiento por lotes.
- Registro masivo Esta opción le permite especificar una lista de valores de aprovisionamiento de un solo objeto que se almacenan en un archivo en un bucket de S3. Este enfoque funciona bien si tiene un gran número de dispositivos conocidos cuyas características puede incluir en una lista.

Temas

- Aprovisionamiento de un solo objeto
- Just-in-time aprovisionamiento
- Registro masivo

Aprovisionamiento de un solo objeto

Para aprovisionar algo, utilice la RegisterThingAPI o el comando `register-thing` CLI. El comando de CLI `register-thing` adopta los argumentos siguientes:

--template-body

La plantilla de aprovisionamiento.

--parameters

Una lista de pares de nombre-valor para los parámetros utilizados en la plantilla de aprovisionamiento, en formato JSON (por ejemplo, `{"ThingName" : "MyProvisionedThing", "CSR" : "csr-text"}`).

Consulte Aprovisionamiento de plantillas.

RegisterThingo `register-thing` devuelve ARNs los recursos y el texto del certificado que creó:

```
{
    "certificatePem": "certificate-text",
    "resourceArns": {
    "PolicyLogicalName": "arn:aws:iot:us-
west-2:123456789012:policy/2A6577675B7CD1823E271C7AAD8184F44630FFD7",
    "certificate": "arn:aws:iot:us-west-2:123456789012:cert/
cd82bb924d4c6ccbb14986dcb4f40f30d892cc6b3ce7ad5008ed6542eea2b049",
    "thing": "arn:aws:iot:us-west-2:123456789012:thing/MyProvisionedThing"
    }
}
```

Si se omite el parámetro en el diccionario, se utilizará el valor predeterminado. Si no se especifica el valor predeterminado, el parámetro no se reemplazará por un valor.

Just-in-time aprovisionamiento

Puede utilizar el just-in-time aprovisionamiento (JITP) para aprovisionar sus dispositivos la primera vez que intenten conectarse a ellos. AWS IoT Para aprovisionar el dispositivo, debe habilitar el registro automático y asociar una plantilla de aprovisionamiento al certificado de CA utilizado para firmar el certificado de dispositivo. Los éxitos y errores de aprovisionamiento se registran como Métricas de aprovisionamiento de dispositivos en Amazon CloudWatch.

Temas

- Información general de JITP
- Registrar CA mediante una plantilla de aprovisionamiento
- Registrar una CA mediante el nombre de la plantilla de aprovisionamiento

Información general de JITP

Cuando un dispositivo intenta conectarse mediante un certificado firmado AWS IoT por un certificado de CA registrado, AWS IoT carga la plantilla del certificado de CA y la utiliza para realizar la llamada RegisterThing. El flujo de trabajo de JITP registra primero un certificado con un valor de estado de `PENDING_ACTIVATION`. Cuando se completa el flujo de aprovisionamiento del dispositivo, el estado del certificado cambia a `ACTIVE`.

AWS IoT define los siguientes parámetros que puede declarar y hacer referencia en las plantillas de aprovisionamiento:

- `AWS::IoT::Certificate::Country`
- `AWS::IoT::Certificate::Organization`
- `AWS::IoT::Certificate::OrganizationalUnit`
- `AWS::IoT::Certificate::DistinguishedNameQualifier`
- `AWS::IoT::Certificate::StateName`
- `AWS::IoT::Certificate::CommonName`
- `AWS::IoT::Certificate::SerialNumber`
- `AWS::IoT::Certificate::Id`

Los valores de estos parámetros de plantilla de aprovisionamiento se limitan a lo que JITP puede extraer del campo de asunto del certificado del dispositivo que se va a aprovisionar. El certificado debe contener valores para todos los parámetros en el cuerpo de la plantilla. El parámetro `AWS::IoT::Certificate::Id` se refiere a un ID generado internamente, no un ID que se encuentra en el certificado. Puede obtener el valor de este ID mediante la `principal()` función incluida en una AWS IoT regla.

> **Note**
>
> Puede aprovisionar dispositivos mediante la función de AWS IoT Core just-in-time aprovisionamiento (JITP) sin tener que enviar toda la cadena de confianza en la primera conexión de un dispositivo a una. AWS IoT Core La presentación del certificado de CA es opcional, pero es necesario que el dispositivo envíe la extensión Indicación del nombre del servidor (SNI) cuando se conecte a AWS IoT Core.

Ejemplo de cuerpo de plantilla

El siguiente archivo JSON es un ejemplo de cuerpo de plantilla de JITP completa.

```
{
   "Parameters":{
      "AWS::IoT::Certificate::CommonName":{
         "Type":"String"
      },
      "AWS::IoT::Certificate::SerialNumber":{
         "Type":"String"
      },
      "AWS::IoT::Certificate::Country":{
         "Type":"String"
      },
      "AWS::IoT::Certificate::Id":{
         "Type":"String"
      }
   },
   "Resources":{
      "thing":{
         "Type":"AWS::IoT::Thing",
         "Properties":{
            "ThingName":{
               "Ref":"AWS::IoT::Certificate::CommonName"
            },
            "AttributePayload":{
               "version":"v1",
               "serialNumber":{
                  "Ref":"AWS::IoT::Certificate::SerialNumber"
               }
            },
            "ThingTypeName":"lightBulb-versionA",
```

```
            "ThingGroups":[
               "v1-lightbulbs",
               {
                  "Ref":"AWS::IoT::Certificate::Country"
               }
            ]
         },
         "OverrideSettings":{
            "AttributePayload":"MERGE",
            "ThingTypeName":"REPLACE",
            "ThingGroups":"DO_NOTHING"
         }
      },
      "certificate":{
         "Type":"AWS::IoT::Certificate",
         "Properties":{
            "CertificateId":{
               "Ref":"AWS::IoT::Certificate::Id"
            },
            "Status":"ACTIVE"
         }
      },
      "policy":{
         "Type":"AWS::IoT::Policy",
         "Properties":{
            "PolicyDocument":"{ \"Version\": \"2012-10-17\",        \"Statement
\": [{ \"Effect\": \"Allow\", \"Action\":[\"iot:Publish\"], \"Resource\":
 [\"arn:aws:iot:us-east-1:123456789012:topic/foo/bar\"] }] }"
         }
      }
   }
}
```

Esta plantilla de ejemplo declara valores para los parámetros de aprovisionamiento `AWS::IoT::Certificate::CommonName`, `AWS::IoT::Certificate::SerialNumber`, `AWS::IoT::Certificate::Country` y `AWS::IoT::Certificate::Id` que se extraen del certificado y se usan en la sección `Resources`. El flujo de trabajo de JITP utiliza después esta plantilla para llevar a cabo las siguientes acciones:

- Registrar un certificado y establecer su estado en PENDING_ACTIVE
- Crear un recurso de objeto
- Crear un recurso de política

- Asociar la política al certificado
- Asocie el certificado al objeto .
- Actualizar el estado del certificado a ACTIVE

El aprovisionamiento del dispositivo falla si el certificado no tiene todas las propiedades mencionadas en la sección `Parameters` del `templateBody`. Por ejemplo, si `AWS::IoT::Certificate::Country` está incluido en la plantilla, pero el certificado no tiene ninguna propiedad `Country`, se produce un error en el aprovisionamiento del dispositivo.

También puedes utilizarla CloudTrail para solucionar problemas con tu plantilla JITP. Para obtener información sobre las métricas que se registran en Amazon CloudWatch, consulteMétricas de aprovisionamiento de dispositivos. Para obtener más información acerca del aprovisionamiento de dispositivos, consulte Plantillas de aprovisionamiento.

> **Note**
>
> Durante el proceso de aprovisionamiento, el just-in-time aprovisionamiento (JITP) llama a otras operaciones de la API del plano AWS IoT de control. Estas llamadas pueden superar las cuotas de limitación de AWS IoT establecidas en su cuenta y provocar una limitación controlada de las llamadas. Póngase en contacto con el servicio de atención al cliente de AWS para aumentar las cuotas de limitación controlada, si es necesario.

Registrar CA mediante una plantilla de aprovisionamiento

Para registrar una CA mediante una plantilla de aprovisionamiento completa, siga estos pasos:

1. Guarde la plantilla de aprovisionamiento y la información del ARN del rol como en el siguiente ejemplo como un archivo JSON:

```
{
     "templateBody" : "{\r\n    \"Parameters\" : {\r\n
 \"AWS::IoT::Certificate::CommonName\": {\r\n            \"Type\": \"String\"\r
\n        },\r\n        \"AWS::IoT::Certificate::SerialNumber\": {\r\n
  \"Type\": \"String\"\r\n        },\r\n        \"AWS::IoT::Certificate::Country
\": {\r\n            \"Type\": \"String\"\r\n        },\r\n
 \"AWS::IoT::Certificate::Id\": {\r\n            \"Type\": \"String\"\r
\n        }\r\n    },\r\n    \"Resources\": {\r\n        \"thing\": {\r
```

```
\n            \"Type\": \"AWS::IoT::Thing\",\r\n            \"Properties
\": {\r\n                \"ThingName\": {\r\n                    \"Ref\":
 \"AWS::IoT::Certificate::CommonName\"\r\n                },\r\n
     \"AttributePayload\": {\r\n                    \"version\": \"v1\",\r\n
                   \"serialNumber\": {\r\n                        \"Ref\":
 \"AWS::IoT::Certificate::SerialNumber\"\r\n                    }\r\n
     },\r\n                \"ThingTypeName\": \"lightBulb-versionA\",\r\n
        \"ThingGroups\": [\r\n                    \"v1-lightbulbs\",\r\n
           {\r\n                        \"Ref\": \"AWS::IoT::Certificate::Country
\"\r\n                    }\r\n                ]\r\n            },\r\n
  \"OverrideSettings\": {\r\n                \"AttributePayload\": \"MERGE\",\r\n
               \"ThingTypeName\": \"REPLACE\",\r\n                \"ThingGroups
\": \"DO_NOTHING\"\r\n            }\r\n        },\r\n        \"certificate\": {\r
\n            \"Type\": \"AWS::IoT::Certificate\",\r\n            \"Properties
\": {\r\n                \"CertificateId\": {\r\n                    \"Ref\":
 \"AWS::IoT::Certificate::Id\"\r\n                },\r\n                \"Status\":
 \"ACTIVE\"\r\n            },\r\n            \"OverrideSettings\": {\r\n
     \"Status\": \"DO_NOTHING\"\r\n            }\r\n        },\r\n        \"policy
\": {\r\n            \"Type\": \"AWS::IoT::Policy\",\r\n            \"Properties
\": {\r\n                \"PolicyDocument\": \"{ \\\"Version\\\": \\\"2012-10-17
     \\\", \\\"Statement\\\": [{ \\\"Effect\\\": \\\"Allow\\\", \\\"Action\\\":[\\
\"iot:Publish\\\"], \\\"Resource\\\": [\\\"arn:aws:iot:us-east-1:123456789012:topic
\/foo\/bar\\\"] }] }\"\r\n            }\r\n        }\r\n    }\r\n}",
     "roleArn" : "arn:aws:iam::123456789012:role/JITPRole"
}
```

En este ejemplo, el valor del campo `templateBody` debe ser un objeto JSON especificado como una cadena de escape y solo puede utilizar los valores de la lista anterior. Puede utilizar distintas herramientas para crear la salida JSON necesaria, como `json.dumps` (Python) o `JSON.stringify` (Node). El valor del campo `roleARN` debe ser el ARN de un rol que tenga `AWSIoTThingsRegistration` asociado. Además, la plantilla puede utilizar un `PolicyName` existente en lugar del `PolicyDocument` insertado en el ejemplo.

2. Registre un certificado de CA con la operación Register CACertificate API o el comando `register-ca-certificate`CLI. Especificará el directorio de la plantilla de aprovisionamiento y la información del ARN del rol que guardó en el paso anterior:

   A continuación, se muestra un ejemplo de cómo registrar un certificado de CA en modo `DEFAULT` mediante la AWS CLI:

   ```
   aws iot register-ca-certificate --ca-certificate file://your-ca-cert --
   verification-cert file://your-verification-cert
   ```

```
                --set-as-active --allow-auto-registration --registration-config
 file://your-template
```

A continuación, se muestra un ejemplo de cómo registrar un certificado de CA en modo SNI_ONLY mediante la AWS CLI:

```
aws iot register-ca-certificate --ca-certificate file://your-ca-cert --certificate-
mode SNI_ONLY
                 --set-as-active --allow-auto-registration --registration-config
 file://your-template
```

Para obtener más información, consulte Registrar sus certificados de CA.

3. (Opcional) Actualice la configuración de un certificado de CA mediante la operación Actualizar CACertificate API o el comando update-ca-certificateCLI.

   El siguiente ejemplo muestra cómo actualizar un certificado de CA con la AWS CLI:

```
aws iot update-ca-certificate --certificate-id caCertificateId
                --new-auto-registration-status ENABLE --registration-config
 file://your-template
```

Registrar una CA mediante el nombre de la plantilla de aprovisionamiento

Para registrar una CA mediante el nombre de una plantilla de aprovisionamiento, siga estos pasos:

1. Guarde el cuerpo de la plantilla de aprovisionamiento como un archivo JSON. Puede encontrar un cuerpo de plantilla de ejemplo en el cuerpo de la plantilla de ejemplo.
2. Para crear una plantilla de aprovisionamiento, utilice la CreateProvisioningTemplateAPI o el comando create-provisioning-templateCLI:

```
aws iot create-provisioning-template --template-name your-template-name \
        --template-body file://your-template-body.json --type JITP \
        --provisioning-role-arn arn:aws:iam::123456789012:role/test
```

> **Note**
>
> Para el just-in-time aprovisionamiento (JITP), debe especificar el tipo de plantilla que se utilizará JITP al crear la plantilla de aprovisionamiento. Para obtener más información

sobre el tipo de plantilla, consulta la referencia de CreateProvisioningTemplatela AWS API.

3. Para registrar la CA con el nombre de la plantilla, utilice la CACertificate API de registro o el comando `register-ca-certificate`CLI:

```
aws iot register-ca-certificate --ca-certificate file://your-ca-cert --
verification-cert file://your-verification-cert \
        --set-as-active --allow-auto-registration --registration-config
 templateName=your-template-name
```

Registro masivo

Puede usar el comando `start-thing-registration-task` para registrar varios objetos a la vez. Este comando adopta una plantilla de aprovisionamiento, un nombre de bucket de S3; un nombre de clave y un ARN de rol que permite el acceso al archivo en el bucket de S3. El archivo en el bucket de S3 contiene los valores utilizados para reemplazar los parámetros de la plantilla. El archivo debe ser un archivo JSON delimitado por nueva línea. Cada línea contiene todos los valores de los parámetros para el registro de un único dispositivo. Por ejemplo:

```
{"ThingName": "foo", "SerialNumber": "123", "CSR": "csr1"}
{"ThingName": "bar", "SerialNumber": "456", "CSR": "csr2"}
```

Las siguientes operaciones de API relacionadas con el registro masivo pueden ser útiles:

- ListThingRegistrationTasks: Enumera las tareas actuales de aprovisionamiento masivo de productos.
- DescribeThingRegistrationTask: proporciona información sobre una tarea específica de registro masivo de productos.
- StopThingRegistrationTask: Detiene una tarea de registro masivo.
- ListThingRegistrationTaskReports: Se utiliza para comprobar los resultados y los errores de una tarea de registro masivo.

> **Note**
>
> - Solo puede ejecutarse una tarea de operación de registro masivo a la vez (por cuenta).

- Las operaciones de registro masivo llaman a otras operaciones de API del plano de AWS IoT control. Estas llamadas pueden superar las cuotas de limitación controlada de AWS IoT establecidas en la cuenta y provocar errores de limitación. Póngase en contacto con AWS Customer Support para aumentar AWS IoT sus cuotas de regulación, si es necesario.

Aprovisionamiento de plantillas

Una plantilla de aprovisionamiento es un documento JSON que utiliza parámetros para describir los recursos con los que el dispositivo debe interactuar. AWS IoT Una plantilla de aprovisionamiento contiene dos secciones: `Parameters` y `Resources`. Existen dos tipos de plantillas de aprovisionamiento. AWS IoT Una se usa para el just-in-time aprovisionamiento (JITP) y el registro masivo, y la segunda se usa para el aprovisionamiento de flotas.

Temas

- Sección de parámetros
- Sección de recursos
- Ejemplo de plantilla para el registro masivo
- Ejemplo de plantilla para just-in-time aprovisionamiento (JITP)
- Aprovisionamiento de flotas

Sección de parámetros

La sección `Parameters` declara los parámetros utilizados en la sección `Resources`. Cada parámetro declara un nombre, un tipo y un valor predeterminado opcional. El valor predeterminado se usa cuando el diccionario trasladado con la plantilla no contiene un valor para el parámetro. La sección `Parameters` de un documento de plantilla tiene el siguiente aspecto:

```
{
    "Parameters" : {
        "ThingName" : {
            "Type" : "String"
        },
        "SerialNumber" : {
            "Type" : "String"
```

```
        },
        "Location" : {
            "Type" : "String",
            "Default" : "WA"
        },
        "CSR" : {
            "Type" : "String"
        }
    }
}
```

Este snippet de cuerpo de plantilla declara cuatro parámetros: `ThingName`, `SerialNumber`, `Location` y `CSR`. Todos esos parámetros son de tipo `String`. El parámetro `Location` declara un valor predeterminado de `"WA"`.

Sección de recursos

La `Resources` sección del cuerpo de la plantilla declara los recursos necesarios para que el dispositivo se comunique con AWS IoT: una cosa, un certificado y una o más políticas de IoT. Cada recurso especifica un nombre lógico, un tipo y un conjunto de propiedades.

Un nombre lógico le permite hacer referencia a un recurso en cualquier lugar de la plantilla.

El tipo especifica la clase de recurso que está declarando. Los tipos válidos son:

- `AWS::IoT::Thing`
- `AWS::IoT::Certificate`
- `AWS::IoT::Policy`

Las propiedades que especifique dependen del tipo de recurso que está declarando.

Recursos de objetos

Los recursos de objetos se declaran mediante las siguientes propiedades:

- `ThingName`: cadena.
- `AttributePayload`: opcional. Una lista de pares de nombre-valor.
- `ThingTypeName`: opcional. Cadena para un tipo de objeto asociado para el objeto.
- `ThingGroups`: opcional. Una lista de grupos a los que pertenece el objeto.

- `BillingGroup`: opcional. Cadena para el nombre de un grupo de facturación asociado.
- `PackageVersions`: opcional. Cadena para un paquete asociado y los nombres de las versiones.

Recursos de certificados

Puede especificar certificados de una de las siguientes maneras:

- Una solicitud de firma de certificado (CSR).
- Un ID de certificado de un certificado de dispositivo existente. (Solo el certificado se IDs puede usar con una plantilla de aprovisionamiento de flota).
- Un certificado de dispositivo creado con un certificado de CA registrado con AWS IoT. Si tiene más de un certificado de CA registrado con el mismo campo de asunto, también debe trasladar el certificado de CA utilizado para firmar el certificado de dispositivo.

> **Note**
>
> Cuando declare un certificado en una plantilla, use solo uno de estos métodos. Por ejemplo, si utiliza un CSR, no puede especificar también un ID de certificado o un certificado de dispositivo. Para obtener más información, consulte Certificados de cliente X.509.

Para obtener más información, consulte Información general del certificado X.509.

Los recursos de certificados se declaran mediante las siguientes propiedades:

- `CertificateSigningRequest`: cadena.
- `CertificateId`: cadena.
- `CertificatePem`: cadena.
- `CACertificatePem`: cadena.
- `Status`: opcional. Cadena, que puede ser `ACTIVE` o `INACTIVE`. El valor predeterminado es ACTIVE.
- `ThingPrincipalType`: opcional. Cadena que especifica el tipo de relación entre el objeto y la entidad principal (el certificado).
  - `EXCLUSIVE_THING`: establece una relación exclusiva. La entidad principal solo puede asociarse a este objeto específico y no a otros.

- `NON_EXCLUSIVE_THING`: asocia la entidad principal especificada a los objetos. Puede asociar varios objetos a la entidad principal. Este es el valor predeterminado si no se especifica otro valor.

> **Note**
>
> También puede aprovisionar dispositivos sin certificados de dispositivo. Para obtener más información, consulte Aprovisionamiento de dispositivos que no tienen certificados de dispositivo mediante el aprovisionamiento de flotas.

Ejemplos:

- Certificado especificado con un CSR:

```
{
    "certificate" : {
        "Type" : "AWS::IoT::Certificate",
        "Properties" : {
            "CertificateSigningRequest": {"Ref" : "CSR"},
            "Status" : "ACTIVE"
        }
    }
}
```

- Certificado especificado con un ID de certificado existente:

```
{
    "certificate" : {
        "Type" : "AWS::IoT::Certificate",
        "Properties" : {
            "CertificateId": {"Ref" : "CertificateId"}
        }
    }
}
```

- Certificado especificado con un archivo .pem de certificado existente y un archivo .pem de certificado de CA:

```
{
    "certificate" : {
```

```
        "Type" : "AWS::IoT::Certificate",
        "Properties" : {
            "CACertificatePem": {"Ref" : "CACertificatePem"},
            "CertificatePem": {"Ref" : "CertificatePem"}
        }
    }
}
```

- Asocie exclusivamente un objeto a una entidad principal:

```
{
    "certificate" : {
        "Type" : "AWS::IoT::Certificate",
        "Properties" : {
            "ThingPrincipalType" : "EXCLUSIVE_THING"
        }
    }
}
```

Recursos de políticas

Los recursos de políticas se declaran mediante una de las siguientes propiedades:

- `PolicyName`: opcional. Cadena. Hash es el valor predeterminado del documento de políticas. Solo `PolicyName` puede hacer referencia a las políticas de AWS IoT , pero no a las políticas de IAM. Si utiliza una política de AWS IoT existente, escriba el nombre de la política para la propiedad `PolicyName`. No incluya la propiedad `PolicyDocument`.
- `PolicyDocument`: opcional. Un objeto JSON especificado como una cadena de escape. Si `PolicyDocument` no se proporciona, la política debe haberse creado ya.

> **Note**
>
> Si una sección `Policy` está presente, `PolicyName` o `PolicyDocument`, pero no ambas, debe especificarse.

Anular la configuración

Si una plantilla especifica un recurso que ya existe, la sección `OverrideSettings` le permite especificar la acción que se debe realizar:

`DO_NOTHING`

Dejar el recurso como está.

`REPLACE`

Sustituir el recurso por el recurso especificado en la plantilla.

`FAIL`

Provocar un error en la solicitud con `ResourceConflictsException`.

`MERGE`

Solo es válida para las propiedades `ThingGroups` y `AttributePayload` de un objeto (`thing`). Combinar los atributos o las pertenencias a grupos existentes de la cosa con los especificados en la plantilla.

Cuando declara un recurso de objeto, puede especificar `OverrideSettings` para las siguientes propiedades:

- `ATTRIBUTE_PAYLOAD`
- `THING_TYPE_NAME`
- `THING_GROUPS`

Cuando declara un recurso de certificado, puede especificar `OverrideSettings` para la propiedad `Status`.

`OverrideSettings` no están disponibles para recursos de políticas.

Ejemplo de recursos

El siguiente fragmento de plantilla declara un objeto, un certificado y una política:

```
{
```

```
    "Resources" : {
        "thing" : {
            "Type" : "AWS::IoT::Thing",
            "Properties" : {
                "ThingName" : {"Ref" : "ThingName"},
                "AttributePayload" : { "version" : "v1", "serialNumber" :  {"Ref" :
 "SerialNumber"}},
                "ThingTypeName" :  "lightBulb-versionA",
                "ThingGroups" : ["v1-lightbulbs", {"Ref" : "Location"}]
            },
            "OverrideSettings" : {
                "AttributePayload" : "MERGE",
                "ThingTypeName" : "REPLACE",
                "ThingGroups" : "DO_NOTHING"
            }
        },
        "certificate" : {
            "Type" : "AWS::IoT::Certificate",
            "Properties" : {
                "CertificateSigningRequest": {"Ref" : "CSR"},
                "Status" : "ACTIVE"
            }
        },
        "policy" : {
            "Type" : "AWS::IoT::Policy",
            "Properties" : {
                "PolicyDocument" : "{ \"Version\": \"2012-10-17\",
 \"Statement\": [{ \"Effect\": \"Allow\", \"Action\":[\"iot:Publish\"], \"Resource\":
 [\"arn:aws:iot:us-east-1:123456789012:topic/foo/bar\"] }] }"
            }
        }
    }
}
```

El objeto se declara con:

- El nombre lógico `"thing"`.
- El tipo `AWS::IoT::Thing`.
- Un conjunto de propiedades de objeto.

  Las propiedades de objeto incluyen el nombre de objeto, un conjunto de atributos, un nombre de tipo de objeto opcional y una lista opcional de grupos de objetos a los que pertenece el objeto.

Se hace referencia a los parámetros mediante `{"Ref":"parameter-name"}`. Cuando se evalúa la plantilla, los parámetros se reemplazan por el valor de los parámetros desde el diccionario trasladado con la plantilla.

El certificado se declara con:

- El nombre lógico `"certificate"`.
- El tipo `AWS::IoT::Certificate`.
- Un conjunto de propiedades.

  Las propiedades incluyen el CSR para el certificado y el establecimiento del estado en `ACTIVE`. El texto CSR se transfiere como parámetro en el diccionario transferido con la plantilla.

La política se declara con:

- El nombre lógico `"policy"`.
- El tipo `AWS::IoT::Policy`.
- O el nombre de una política existente o un documento de políticas.

Ejemplo de plantilla para el registro masivo

El siguiente archivo JSON es un ejemplo de una plantilla de aprovisionamiento completa que especifica el certificado con un CSR:

(El valor del campo `PolicyDocument` debe ser un objeto JSON especificado como una cadena de escape).

```
{
    "Parameters" : {
        "ThingName" : {
            "Type" : "String"
        },
        "SerialNumber" : {
            "Type" : "String"
        },
        "Location" : {
            "Type" : "String",
            "Default" : "WA"
        },
```

```
        "CSR" : {
            "Type" : "String"
        }
    },
    "Resources" : {
        "thing" : {
            "Type" : "AWS::IoT::Thing",
            "Properties" : {
                "ThingName" : {"Ref" : "ThingName"},
                "AttributePayload" : { "version" : "v1", "serialNumber" :  {"Ref" :
 "SerialNumber"}},
                "ThingTypeName" :  "lightBulb-versionA",
                "ThingGroups" : ["v1-lightbulbs", {"Ref" : "Location"}]
            }
        },
        "certificate" : {
            "Type" : "AWS::IoT::Certificate",
            "Properties" : {
                "CertificateSigningRequest": {"Ref" : "CSR"},
                "Status" : "ACTIVE",
                "ThingPrincipalType" : "EXCLUSIVE_THING"
            }
        },
        "policy" : {
            "Type" : "AWS::IoT::Policy",
            "Properties" : {
                "PolicyDocument" : "{ \"Version\": \"2012-10-17\",
 \"Statement\": [{ \"Effect\": \"Allow\", \"Action\":[\"iot:Publish\"], \"Resource\":
 [\"arn:aws:iot:us-east-1:123456789012:topic/foo/bar\"] }] }"
            }
        }
    }
}
```

Ejemplo de plantilla para just-in-time aprovisionamiento (JITP)

El siguiente archivo JSON es un ejemplo de una plantilla de aprovisionamiento completa que especifica un certificado existente con un ID de certificado:

```
{
   "Parameters":{
      "AWS::IoT::Certificate::CommonName":{
         "Type":"String"
```

```
      },
      "AWS::IoT::Certificate::SerialNumber":{
         "Type":"String"
      },
      "AWS::IoT::Certificate::Country":{
         "Type":"String"
      },
      "AWS::IoT::Certificate::Id":{
         "Type":"String"
      }
   },
   "Resources":{
      "thing":{
         "Type":"AWS::IoT::Thing",
         "Properties":{
            "ThingName":{
               "Ref":"AWS::IoT::Certificate::CommonName"
            },
            "AttributePayload":{
               "version":"v1",
               "serialNumber":{
                  "Ref":"AWS::IoT::Certificate::SerialNumber"
               }
            },
            "ThingTypeName":"lightBulb-versionA",
            "ThingGroups":[
               "v1-lightbulbs",
               {
                  "Ref":"AWS::IoT::Certificate::Country"
               }
            ]
         },
         "OverrideSettings":{
            "AttributePayload":"MERGE",
            "ThingTypeName":"REPLACE",
            "ThingGroups":"DO_NOTHING"
         }
      },
      "certificate":{
         "Type":"AWS::IoT::Certificate",
         "Properties":{
            "CertificateId":{
               "Ref":"AWS::IoT::Certificate::Id"
            },
```

```
            "Status":"ACTIVE",
            "ThingPrincipalType" : "EXCLUSIVE_THING"
         }
      },
      "policy":{
         "Type":"AWS::IoT::Policy",
         "Properties":{
            "PolicyDocument":"{ \"Version\": \"2012-10-17\",        \"Statement
\": [{ \"Effect\": \"Allow\", \"Action\":[\"iot:Publish\"], \"Resource\":
 [\"arn:aws:iot:us-east-1:123456789012:topic/foo/bar\"] }] }"
         }
      }
   }
}
```

> **Important**
>
> Debe utilizar `CertificateId` en una plantilla que se utilice para el aprovisionamiento JIT.

Para obtener más información sobre el tipo de plantilla de aprovisionamiento, consulta la referencia de `CreateProvisioningTemplate`la AWS API.

Para obtener más información sobre cómo utilizar esta plantilla para el just-in-time aprovisionamiento, consulta: J ust-in-time Provisioning.

Aprovisionamiento de flotas

Las plantillas de aprovisionamiento de flotas se utilizan AWS IoT para configurar la nube y los dispositivos. Estas plantillas utilizan los mismos parámetros y recursos que las plantillas de registro masivo y JITP. Para obtener más información, consulte Aprovisionamiento de plantillas. Las plantillas de aprovisionamiento de flotas pueden contener una sección `Mapping` y una sección `DeviceConfiguration`. Puede utilizar funciones intrínsecas dentro de una plantilla de aprovisionamiento de flotas para generar una configuración específica del dispositivo. Las plantillas de aprovisionamiento de flotas se denominan recursos y se identifican mediante ARNs (por ejemplo,`arn:aws:iot:us-west-2:1234568788:provisioningtemplate/`*`templateName`*).

Mapeos

La sección `Mappings` opcional hace coincidir una clave con el conjunto correspondiente de valores identificados. Por ejemplo, si desea establecer valores en función de una AWS región, puede crear

un mapeo que utilice el Región de AWS nombre como clave y contenga los valores que desee especificar para cada región específica. Puede utilizar la función intrínseca `Fn::FindInMap` para recuperar valores en una asignación.

No puede incluir parámetros, pseudoparámetros ni funciones de llamadas intrínsecas en la sección `Mappings`.

Configuración del dispositivo

La sección de configuración del dispositivo contiene los datos arbitrarios que desea enviar a sus dispositivos al realizar el aprovisionamiento. Por ejemplo:

```
{
    "DeviceConfiguration": {
        "Foo":"Bar"
    }
}
```

Si envías mensajes a tus dispositivos mediante el formato de carga útil de notación de JavaScript objetos (JSON), AWS IoT Core formatea estos datos como JSON. Si utiliza el formato de carga de representación concisa de objetos binarios (CBOR), AWS IoT Core formatea estos datos como CBOR. La sección `DeviceConfiguration` no admite objetos JSON anidados.

Funciones intrínsecas

Las funciones intrínsecas se utilizan en cualquier sección de la plantilla de aprovisionamiento, excepto en la sección `Mappings`.

`Fn::Join`

Agrega un conjunto de valores a un único valor separado por el delimitador especificado. Si un delimitador es la cadena vacía, los valores se concatena sin delimitador.

> **Important**
>
> `Fn::Join` no admite the section called “Recursos de políticas”.

`Fn::Select`

Devuelve un único objeto de una lista de objetos por índice.

> **Important**
>
> `Fn::Select` no comprueba si hay valores `null` o si el índice queda fuera de los límites de la matriz. Ambas condiciones dan lugar a un error de aprovisionamiento, por lo que debe asegurarse de elegir un valor de índice válido y de que la lista contenga valores distintos de null.

`Fn::FindInMap`

Devuelve el valor correspondiente a claves en una asignación de dos niveles declarada en la sección `Mappings`.

`Fn::Split`

Divide una cadena en una lista de valores de cadena para que pueda seleccionar un elemento de la lista de cadenas. Especifique un delimitador que determine dónde se divide la cadena (por ejemplo, una coma). Después de dividir una cadena, utilice `Fn::Select` para seleccionar un elemento.

Por ejemplo, si se importa una cadena de subred IDs delimitada por comas a tu plantilla de pila, puedes dividir la cadena en cada coma. En la lista de subredes IDs, se utiliza `Fn::Select` para especificar un ID de subred para un recurso.

`Fn::Sub`

Sustituye variables en una cadena de entrada por los valores que especifique. Puede utilizar esta función para crear comandos o salidas que incluyan valores que no están disponibles hasta que crea o actualiza una pila.

Ejemplo de plantilla de aprovisionamiento de flotas

```
{
    "Parameters" : {
        "ThingName" : {
            "Type" : "String"
        },
        "SerialNumber": {
            "Type": "String"
        },
        "DeviceLocation": {
```

```
            "Type": "String"
        }
    },
    "Mappings": {
        "LocationTable": {
            "Seattle": {
                "LocationUrl": "https://example.aws"
            }
        }
    },
    "Resources" : {
        "thing" : {
            "Type" : "AWS::IoT::Thing",
            "Properties" : {
                "AttributePayload" : {
                    "version" : "v1",
                    "serialNumber" : "serialNumber"
                },
                "ThingName" : {"Ref" : "ThingName"},
                "ThingTypeName" : {"Fn::Join":["",["ThingPrefix_",
{"Ref":"SerialNumber"}]]},
                "ThingGroups" : ["v1-lightbulbs", "WA"],
                "BillingGroup": "LightBulbBillingGroup"
            },
            "OverrideSettings" : {
                "AttributePayload" : "MERGE",
                "ThingTypeName" : "REPLACE",
                "ThingGroups" : "DO_NOTHING"
            }
        },
        "certificate" : {
            "Type" : "AWS::IoT::Certificate",
            "Properties" : {
                "CertificateId": {"Ref": "AWS::IoT::Certificate::Id"},
                "Status" : "Active",
                "ThingPrincipalType" : "EXCLUSIVE_THING"
            }
        },
        "policy" : {
            "Type" : "AWS::IoT::Policy",
            "Properties" : {
                "PolicyDocument" : {
                    "Version": "2012-10-17",
                    "Statement": [{
```

```
                        "Effect": "Allow",
                        "Action":["iot:Publish"],
                        "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/foo/
bar"]
                    }]
                }
            }
        }
    },
    "DeviceConfiguration": {
        "FallbackUrl": "https://www.example.com/test-site",
        "LocationUrl": {
            "Fn::FindInMap": ["LocationTable",{"Ref": "DeviceLocation"},
 "LocationUrl"]}
        }
}
```

> **Note**
>
> Se puede actualizar una plantilla de aprovisionamiento existente para agregar un enlace de preaprovisionamiento.

Enlaces de preaprovisionamiento

AWS recomienda utilizar funciones de enlace previas al aprovisionamiento al crear plantillas de aprovisionamiento para tener un mayor control sobre qué dispositivos incorpora tu cuenta y cuántos. Los enlaces de preaprovisionamiento son funciones de Lambda que validan los parámetros transferidos desde el dispositivo antes de permitir que se aprovisione el dispositivo. Esta función de Lambda debe existir en su cuenta antes de aprovisionar un dispositivo porque se llama cada vez que un dispositivo envía una solicitud a través de the section called “RegisterThing”.

> **Important**
>
> Asegúrese de incluir las claves contextuales de condición global `source-arn` o `source-account` en las de las políticas adjuntas a su acción de Lambda para evitar la manipulación de permisos. Para obtener más información acerca de este tema, consulte Prevención de la sustitución confusa entre servicios.

Para que los dispositivos se aprovisionen, la función de Lambda debe aceptar el objeto de entrada y devolver el objeto de salida descrito en esta sección. El aprovisionamiento continúa solo si la función de Lambda devuelve un objeto con `"allowProvisioning": True`.

Preaprovisionamiento de entrada de enlace

AWS IoT envía este objeto a la función Lambda cuando un dispositivo se registra en. AWS IoT

```
{
    "claimCertificateId" : "string",
    "certificateId" : "string",
    "certificatePem" : "string",
    "templateArn" : "arn:aws:iot:us-east-1:1234567890:provisioningtemplate/MyTemplate",
    "clientId" : "221a6d10-9c7f-42f1-9153-e52e6fc869c1",
    "parameters" : {
        "string" : "string",
        ...
    }
}
```

El objeto `parameters` pasado a la función de Lambda contiene las propiedades en el argumento `parameters` pasado en la carga de solicitud the section called “RegisterThing”.

Valor de retorno del enlace previo a la provisión

La función de Lambda debe devolver una respuesta que indique si ha autorizado la solicitud de aprovisionamiento y los valores de las propiedades para anular.

A continuación se muestra un ejemplo de una respuesta exitosa de la función de preaprovisionamiento.

```
{
    "allowProvisioning": true,
    "parameterOverrides" : {
        "Key": "newCustomValue",
        ...
    }
}
```

Se agregarán valores `"parameterOverrides"` al parámetro `"parameters"` en la carga de solicitud de the section called “RegisterThing”.

> **Note**
>
> - Si se produce un error en la función Lambda, se produce un error en la solicitud de aprovisionamiento `ACCESS_DENIED` y se registra un error en Logs. CloudWatch
> - Si la función de Lambda no devuelve `"allowProvisioning": "true"` en la respuesta, se produce un error `ACCESS_DENIED` en la solicitud de aprovisionamiento.
> - La función de Lambda debe terminar de ejecutarse y volver en 5 segundos; de lo contrario, la solicitud de aprovisionamiento falla.

Ejemplo de enlace Lambda de preaprovisionamiento

Python

Un ejemplo de enlace Lambda de preaprovisionamiento en Python.

```
import json

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

    return {
        'allowProvisioning': True,
        'parameterOverrides': {
            'DeviceLocation': 'Seattle'
        }
    }
```

Java

Un ejemplo de enlace Lambda de preaprovisionamiento en Java.

Clase Handler:

```
package example;

import java.util.Map;
import java.util.HashMap;
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
```

```
public class PreProvisioningHook implements
 RequestHandler<PreProvisioningHookRequest, PreProvisioningHookResponse> {

    public PreProvisioningHookResponse handleRequest(PreProvisioningHookRequest
 object, Context context) {
        Map<String, String> parameterOverrides = new HashMap<String, String>();
        parameterOverrides.put("DeviceLocation", "Seattle");

        PreProvisioningHookResponse response = PreProvisioningHookResponse.builder()
                .allowProvisioning(true)
                .parameterOverrides(parameterOverrides)
                .build();

        return response;
    }

}
```

Clase Request:

```
package example;

import java.util.Map;
import lombok.Builder;
import lombok.Data;
import lombok.AllArgsConstructor;
import lombok.NoArgsConstructor;

@Data
@Builder
@AllArgsConstructor
@NoArgsConstructor
public class PreProvisioningHookRequest {
    private String claimCertificateId;
    private String certificateId;
    private String certificatePem;
    private String templateArn;
    private String clientId;
    private Map<String, String> parameters;
}
```

Clase Response:

```
package example;

import java.util.Map;
import lombok.Builder;
import lombok.Data;
import lombok.AllArgsConstructor;
import lombok.NoArgsConstructor;


@Data
@Builder
@AllArgsConstructor
@NoArgsConstructor
public class PreProvisioningHookResponse {
    private boolean allowProvisioning;
    private Map<String, String> parameterOverrides;
}
```

JavaScript

Un ejemplo de enlace previo al aprovisionamiento de Lambda. JavaScript

```
exports.handler = function(event, context, callback) {
    console.log(JSON.stringify(event, null, 2));
    var reply = {
        allowProvisioning: true,
        parameterOverrides: {
            DeviceLocation: 'Seattle'
        }
     };
     callback(null, reply);
}
```

Firma de certificados autogestionada mediante un proveedor de certificados AWS IoT Core

Puede crear un proveedor de AWS IoT Core certificados para firmar las solicitudes de firma de certificados (CSRs) en el aprovisionamiento de AWS IoT flotas. Un proveedor de certificados hace referencia a una función de Lambda y a la API de MQTT `CreateCertificateFromCsr` para el

aprovisionamiento de flotas. La función de Lambda acepta una CSR y devuelve un certificado de cliente firmado.

Si no cuenta con un proveedor de certificados Cuenta de AWS, se recurre a la API CreateCertificateFromCsr MQTT en el aprovisionamiento de flotas para generar el certificado a partir de una CSR. Tras crear un proveedor de certificados, el comportamiento de la API de CreateCertificateFromCsr MQTT cambiará y todas las llamadas a esta API de MQTT invocarán al proveedor de certificados para que emita el certificado.

Con el proveedor de AWS IoT Core certificados, puede implementar soluciones que utilicen autoridades de certificación privadas (CAs) AWS Private CA, como otras de confianza pública CAs, o su propia infraestructura de clave pública (PKI) para firmar la CSR. Además, puede utilizar el proveedor de certificados para personalizar los campos del certificado de su cliente, como los períodos de validez, los algoritmos de firma, los emisores y las extensiones.

> **Important**
>
> Solo puede crear un proveedor de certificados por Cuenta de AWS. El cambio en el comportamiento de firma se aplica a toda la flota que llama a la API de CreateCertificateFromCsr MQTT hasta que elimines el proveedor de certificados de tu empresa. Cuenta de AWS

En este tema:

- Funcionamiento de la firma de certificados autoadministrada en el aprovisionamiento de flotas
- Entrada de la función de Lambda del proveedor de certificados
- Valor devuelto por la función de Lambda del proveedor de certificados
- Ejemplo de función de Lambda
- Firma de certificados autoadministrada para el aprovisionamiento de flotas
- AWS CLI comandos para el proveedor de certificados

Funcionamiento de la firma de certificados autoadministrada en el aprovisionamiento de flotas

Conceptos clave

Los siguientes conceptos proporcionan detalles que pueden ayudarle a entender cómo funciona la firma de certificados autogestionada en el aprovisionamiento de AWS IoT flotas. Para obtener más información, consulte Aprovisionamiento de dispositivos que no tienen certificados de dispositivo mediante el aprovisionamiento de flotas.

AWS IoT aprovisionamiento de flotas

Con el aprovisionamiento de AWS IoT flotas (abreviatura de aprovisionamiento de flotas), AWS IoT Core genera y entrega de forma segura certificados de dispositivo a sus dispositivos cuando se conectan a ellos AWS IoT Core por primera vez. Puede utilizar el aprovisionamiento de flotas para conectar dispositivos que no tienen certificados de dispositivo con AWS IoT Core.

Solicitud de firma de certificado (CSR)

En el proceso de aprovisionamiento de flotas, un dispositivo realiza una solicitud a AWS IoT Core través del MQTT de aprovisionamiento de flota. APIs Esta solicitud incluye una solicitud de firma de certificado (CSR), la cual se firmará para crear un certificado de cliente.

AWS firma gestionada de certificados en el aprovisionamiento de flotas

AWS gestionado es la configuración predeterminada para la firma de certificados en el aprovisionamiento de flotas. Con la firma de certificados AWS gestionada, AWS IoT Core firmará CSRs con la suya propia. CAs

Firma de certificados autoadministrada en el aprovisionamiento de flotas

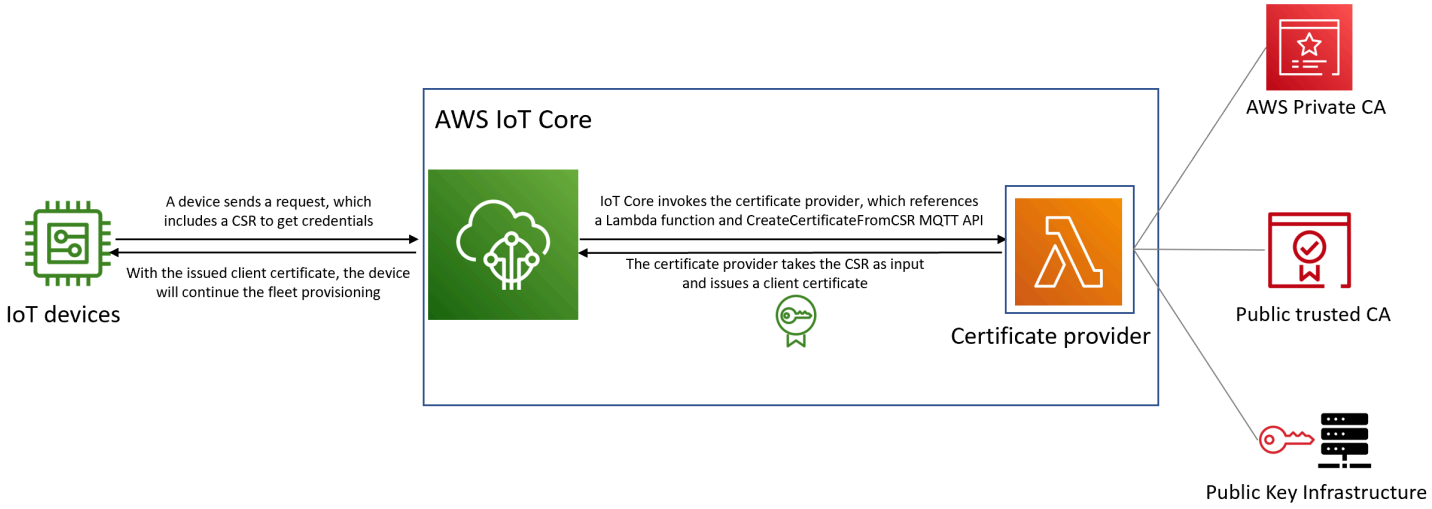
La autoadministración es otra opción para la firma de certificados en el aprovisionamiento de flotas. Con la firma de certificados autogestionada, se crea un proveedor de AWS IoT Core certificados para CSRs firmarlos. Puede usar la firma de certificados autogestionada para firmar CSRs con una CA generada por una CA AWS privada, otra CA de confianza pública o su propia infraestructura de clave pública (PKI).

AWS IoT Core proveedor de certificados

AWS IoT Core el proveedor de certificados (abreviatura de proveedor de certificados) es un recurso administrado por el cliente que se utiliza para la firma autogestionada de certificados en el aprovisionamiento de flotas.

Diagrama

El siguiente diagrama es una ilustración simplificada de cómo funciona la firma automática de certificados en el aprovisionamiento de flotas. AWS IoT



- Cuando se fabrica o se introduce un nuevo dispositivo de IoT en la flota, necesita certificados de cliente para autenticarse AWS IoT Core.
- Como parte del proceso de aprovisionamiento de la flota, el dispositivo solicita los certificados de los clientes a través del AWS IoT Core MQTT de aprovisionamiento de la flota. APIs Esta solicitud incluye una solicitud de firma de certificado (CSR).
- AWS IoT Core invoca al proveedor de certificados y pasa la CSR como entrada al proveedor.
- El proveedor de certificados toma la CSR como entrada y emite un certificado de cliente.

  Para la firma de certificados AWS gestionada, AWS IoT Core firma la CSR con su propia CA y emite un certificado de cliente.
- Con el certificado de cliente emitido, el dispositivo continuará con el aprovisionamiento de flotas y establecerá una conexión segura con AWS IoT Core.

Entrada de la función de Lambda del proveedor de certificados

AWS IoT Core envía el siguiente objeto a la función Lambda cuando un dispositivo se registra en ella. El valor de `certificateSigningRequest` es la CSR en formato de correo con privacidad mejorada (PEM) que se proporciona en la solicitud `CreateCertificateFromCsr`. `principalId`Es el ID del principal al que se conecta AWS IoT Core al realizar la

`CreateCertificateFromCsr` solicitud. `clientId`es el ID de cliente establecido para la conexión MQTT.

```
{
 "certificateSigningRequest": "string",
 "principalId": "string",
 "clientId": "string"
}
```

Valor devuelto por la función de Lambda del proveedor de certificados

La función de Lambda debe devolver una respuesta que contenga el valor `certificatePem`. A continuación se muestra un ejemplo de una respuesta satisfactoria. AWS IoT Core utilizará el valor devuelto (`certificatePem`) para crear el certificado.

```
{
 "certificatePem": "string"
}
```

Si el registro se realiza correctamente, `CreateCertificateFromCsr` devolverá el mismo `certificatePem` en la respuesta `CreateCertificateFromCsr`. Para obtener más información, consulte el ejemplo de carga útil de respuesta de CreateCertificateFromCsr.

Ejemplo de función de Lambda

Antes de crear un proveedor de certificados, debe crear una función de Lambda para firmar una CSR. El siguiente es un ejemplo de función de Lambda en Python. Esta función llama a AWS Private CA para firmar la CSR de entrada con una CA privada y el algoritmo de firma `SHA256WITHRSA`. El certificado de cliente devuelto tendrá una validez de un año. Para obtener más información sobre AWS Private CA cómo crear una CA privada, consulte ¿Qué es una CA AWS privada? y Crear una CA privada.

```
import os
import time
import uuid
import boto3

def lambda_handler(event, context):
    ca_arn = os.environ['CA_ARN']
```

```
    csr = (event['certificateSigningRequest']).encode('utf-8')

    acmpca = boto3.client('acm-pca')
    cert_arn = acmpca.issue_certificate(
        CertificateAuthorityArn=ca_arn,
        Csr=csr,
        Validity={"Type": "DAYS", "Value": 365},
        SigningAlgorithm='SHA256WITHRSA',
        IdempotencyToken=str(uuid.uuid4())
    )['CertificateArn']

    # Wait for certificate to be issued
    time.sleep(1)
    cert_pem = acmpca.get_certificate(
        CertificateAuthorityArn=ca_arn,
        CertificateArn=cert_arn
    )['Certificate']

    return {
        'certificatePem': cert_pem
    }
```

Important

- Los certificados devueltos por la función de Lambda deben tener el mismo nombre de sujeto y clave pública que la solicitud de firma de certificado (CSR).
- La función de Lambda debe terminar de ejecutarse en cinco segundos.
- La función Lambda debe estar en la misma región Cuenta de AWS y región que el recurso del proveedor de certificados.
- Se debe conceder AWS IoT al director del servicio el permiso de invocación para la función Lambda. Para evitar problemas de suplente confuso, le recomendamos que defina `sourceArn` y `sourceAccount` para los permisos de invocación. Para obtener más información, consulte Prevención de la sustitución confisa entre servicios.

El siguiente ejemplo de política basada en recursos para Lambda concede a AWS IoT el permiso para invocar la función de Lambda:

```
{
```

```
 "Version":"2012-10-17",
 "Id": "InvokePermission",
 "Statement": [
  {
   "Sid": "LambdaAllowIotProvider",
   "Effect": "Allow",
   "Principal": {
    "Service": "iot.amazonaws.com"
   },
   "Action": "lambda:InvokeFunction",
   "Resource": "arn:aws:lambda:us-east-1:123456789012:function:my-function",
   "Condition": {
    "StringEquals": {
     "AWS:SourceAccount": "123456789012"
    },
    "ArnLike": {
    "AWS:SourceArn": "arn:aws:iot:us-east-1:123456789012:certificateprovider/my-
certificate-provider"
    }
   }
  }
 ]
}
```

Firma de certificados autoadministrada para el aprovisionamiento de flotas

Puede elegir la firma de certificados autoadministrada para el aprovisionamiento de flotas mediante la AWS CLI o la Consola de administración de AWS.

AWS CLI

Para elegir la firma de certificados autogestionada, debe crear un proveedor de AWS IoT Core certificados para iniciar sesión CSRs en Fleet Provisioning. AWS IoT Core invoca al proveedor de certificados, que toma una CSR como entrada y devuelve un certificado de cliente. Para crear un perfil de certificado, utilice la operación de la API `CreateCertificateProvider` o el comando de la CLI `create-certificate-provider`.

> **Note**
>
> Tras crear un proveedor de certificados, el comportamiento de la API `CreateCertificateFromCsr` para el aprovisionamiento de flotas cambiará, de modo que

todas las llamadas a `CreateCertificateFromCsr` invocarán al proveedor de certificados para crear los certificados. Puede que este comportamiento tarde unos minutos en cambiar una vez creado un proveedor de certificados.

```
aws iot create-certificate-provider \
               --certificateProviderName my-certificate-provider \
               --lambdaFunctionArn arn:aws:lambda:us-east-1:123456789012:function:my-function-1 \
               --accountDefaultForOperations CreateCertificateFromCsr
```

A continuación se muestra un ejemplo del resultado asociado a la ejecución de este comando:

```
{
 "certificateProviderName": "my-certificate-provider",
 "certificateProviderArn": "arn:aws:iot:us-east-1:123456789012:certificateprovider:my-certificate-provider"
}
```

Para obtener más información, consulte `CreateCertificateProvider` en la Referencia de la API de AWS IoT.

Consola de administración de AWS

Para elegir la firma de certificados autogestionada mediante Consola de administración de AWS, siga estos pasos:

1. Vaya a la consola de AWS IoT.
2. En el panel de navegación de la izquierda, bajo Seguridad, elija Firma de certificado.
3. En la página Firma de certificado, en Detalles de firma de certificado, elija Editar el método de firma de certificados.
4. En la página Editar el método de firma de certificados, en Método de firma de certificados, elija Autoadministrado.
5. En la sección Configuración autoadministrada, introduzca un nombre para el proveedor de certificados y, a continuación, cree o elija una función de Lambda.
6. Seleccione Actualizar firma de certificado.

AWS CLI comandos para el proveedor de certificados

Creación de un proveedor de certificados

Para crear un perfil de certificado, utilice la operación de la API `CreateCertificateProvider` o el comando de la CLI `create-certificate-provider`.

> **Note**
>
> Tras crear un proveedor de certificados, el comportamiento de la API `CreateCertificateFromCsr` para el aprovisionamiento de flotas cambiará, de modo que todas las llamadas a `CreateCertificateFromCsr` invocarán al proveedor de certificados para crear los certificados. Puede que este comportamiento tarde unos minutos en cambiar una vez creado un proveedor de certificados.

```
aws iot create-certificate-provider \
                --certificateProviderName my-certificate-provider \
                --lambdaFunctionArn arn:aws:lambda:us-east-1:123456789012:function:my-function-1 \
                --accountDefaultForOperations CreateCertificateFromCsr
```

A continuación se muestra un ejemplo del resultado asociado a la ejecución de este comando:

```
{
 "certificateProviderName": "my-certificate-provider",
 "certificateProviderArn": "arn:aws:iot:us-east-1:123456789012:certificateprovider:my-certificate-provider"
}
```

Para obtener más información, consulte `CreateCertificateProvider` en la Referencia de la API de AWS IoT.

Actualización del proveedor de certificados

Para actualizar un proveedor de certificados, utilice la operación de la API `UpdateCertificateProvider` o el comando de la CLI `update-certificate-provider`.

```
aws iot update-certificate-provider \
                --certificateProviderName my-certificate-provider \
```

```
                --lambdaFunctionArn arn:aws:lambda:us-east-1:123456789012:function:my-
function-2 \
                --accountDefaultForOperations CreateCertificateFromCsr
```

A continuación se muestra un ejemplo del resultado asociado a la ejecución de este comando:

```
{
 "certificateProviderName": "my-certificate-provider",
 "certificateProviderArn": "arn:aws:iot:us-east-1:123456789012:certificateprovider:my-
certificate-provider"
}
```

Para obtener más información, consulte `UpdateCertificateProvider` y en la Referencia de la API de AWS IoT.

Descripción del proveedor de certificados

Para describir un proveedor de certificados, utilice la operación de la API `DescribeCertificateProvider` o el comando de la CLI `describe-certificate-provider`.

```
aws iot describe-certificate-provider --certificateProviderName my-certificate-provider
```

A continuación se muestra un ejemplo del resultado asociado a la ejecución de este comando:

```
{
 "certificateProviderName": "my-certificate-provider",
 "lambdaFunctionArn": "arn:aws:lambda:us-east-1:123456789012:function:my-function",
 "accountDefaultForOperations": [
  "CreateCertificateFromCsr"
 ],
 "creationDate": "2022-11-03T00:15",
 "lastModifiedDate": "2022-11-18T00:15"
}
```

Para obtener más información, consulte `DescribeCertificateProvider` en la Referencia de la API de AWS IoT.

Eliminación de un proveedor de certificados

Para eliminar un proveedor de certificados, utilice la operación de la API `DeleteCertificateProvider` o el comando de la CLI `delete-certificate-provider`. Si

elimina el recurso del proveedor de certificados, `CreateCertificateFromCsr` se reanudará el comportamiento de y AWS IoT se crearán certificados AWS IoT firmados por una CSR.

```
aws iot delete-certificate-provider --certificateProviderName my-certificate-provider
```

Este comando no proporciona ninguna salida.

Para obtener más información, consulte `DeleteCertificateProvider` en la Referencia de la API de AWS IoT.

Visualización de la lista de proveedores de certificados

Para enumerar los proveedores de certificados de su Cuenta de AWS empresa, utilice la operación `ListCertificateProviders` API o el comando `list-certificate-providers` CLI.

```
aws iot list-certificate-providers
```

A continuación se muestra un ejemplo del resultado asociado a la ejecución de este comando:

```
{
 "certificateProviders": [
  {
   "certificateProviderName": "my-certificate-provider",
   "certificateProviderArn": "arn:aws:iot:us-
east-1:123456789012:certificateprovider:my-certificate-provider"
  }
 ]
}
```

Para obtener más información, consulte `ListCertificateProvider` en la Referencia de la API de AWS IoT.

Creación de políticas y roles de IAM para un usuario que instala un dispositivo

> **Note**
>
> Estos procedimientos se utilizan únicamente cuando lo indique la AWS IoT consola.
> Para ir a esta página desde la consola, abra Crear una nueva plantilla de aprovisionamiento.

¿Por qué no se puede hacer esto en la AWS IoT consola?

Para una experiencia más segura, las acciones de IAM se realizan en la consola de IAM. Los procedimientos de esta sección explican los pasos necesarios para crear las funciones y políticas de IAM necesarias para utilizar la plantilla de aprovisionamiento.

Creación de una política de IAM para el usuario que va a instalar un dispositivo

Este procedimiento describe cómo crear una política de IAM que autorice a un usuario a instalar un dispositivo mediante una plantilla de aprovisionamiento.

Mientras realiza este procedimiento, cambiará entre la consola de IAM y la AWS IoT consola. Se recomienda tener ambas consolas abiertas al mismo tiempo mientras completa este procedimiento.

Para crear una política de IAM para el usuario que va a instalar un dispositivo

1. Abra la Central de políticas en la consola de IAM.
2. Elija Crear política.
3. En la página Crear política, elija la pestaña JSON.
4. Diríjase a la página de la AWS IoT consola en la que eligió Configurar la política y el rol del usuario.
5. En el Ejemplo de política de aprovisionamiento, seleccione Copiar.
6. Vuelva a la consola de IAM.
7. En el editor JSON, pegue la política que copió de la AWS IoT consola. Esta política es específica de la plantilla que estás creando en la AWS IoT consola.
8. Para continuar, elija Siguiente: Etiquetas.
9. En la página Agregar etiquetas (opcional), seleccione Agregar etiqueta para cada etiqueta que quiera agregar a esta política. Puede omitir este paso si no tiene etiquetas para agregar.
10. Elija Siguiente: Revisión para continuar.
11. En la página Revisar política haga lo siguiente:

    a. En Nombre*, introduzca un nombre para la política que le ayude a recordar el propósito de la política.

       Anote el nombre que le dé a esta política porque lo usará en el siguiente procedimiento.

b. Puede elegir introducir una descripción opcional para la política que está creando.

c. Revise el resto de esta política y sus etiquetas.

12. Elija Crear política para finalizar la creación de su política.

Después de crear la nueva política, vaya a the section called “Creación de un rol de IAM para el usuario que instalará un dispositivo” para crear la entrada de rol de usuario a la que adjuntará esta política.

Creación de un rol de IAM para el usuario que instalará un dispositivo

En estos pasos se describe cómo crear un rol de IAM que autentique al usuario que instalará un dispositivo mediante una plantilla de aprovisionamiento.

Para crear una política de IAM para el usuario que va a instalar un dispositivo

1. Abra la Central de roles en la consola de IAM.
2. Elija Crear rol.
3. En Seleccione una entidad de confianza, elija el tipo de entidad de confianza a la que quiere dar acceso a la plantilla que va a crear.
4. Elija o introduzca la identificación de la entidad de confianza a la que desea conceder acceso y, a continuación, seleccione Siguiente.
5. En la página Agregar permisos, en las Políticas de permisos, en el cuadro de búsqueda, escriba el nombre de la política que creó en el procedimiento anterior.
6. En la lista de políticas, elija la política que creó en el procedimiento anterior y, a continuación, elija Siguiente.
7. En la sección Nombrar, revisar y crear, haga lo siguiente:

   a. En Nombre del rol, introduzca un nombre de rol que le sea útil para identificar su propósito.

   b. En Descripción, puede optar por introducir una descripción opcional del rol. Esto no es necesario para continuar.

   c. Revise los valores de los pasos 1 y 2.

   d. En Agregar etiquetas (opcional), puede elegir agregar etiquetas a este rol. Esto no es necesario para continuar.

   e. Compruebe que la información de esta página esté completa y sea correcta y, a continuación, seleccione Crear función.

Tras crear el nuevo rol, vuelve a la AWS IoT consola para seguir creando la plantilla.

Actualización de una política existente para autorizar una plantilla nueva

En los siguientes pasos se describe cómo agregar una nueva plantilla a una política de IAM que autorice a un usuario a instalar un dispositivo mediante una plantilla de aprovisionamiento.

Para agregar una plantilla nueva a una política de IAM existente

1. Abra la Central de políticas en la consola de IAM.
2. En el cuadro de búsqueda, escriba el nombre de la política que desea actualizar.
3. En la lista que aparece debajo del cuadro de búsqueda, busque la política que desea actualizar y elija el nombre de la política.
4. En Resumen de la política, seleccione la pestaña JSON si ese panel aún no está visible.
5. Elija Editar política para editar el documento de política JSON.
6. En el editor, selecciona la pestaña JSON si ese panel aún no está visible.
7. En el documento de política, busque la declaración de política que contiene la acción `iot:CreateProvisioningClaim`.

   Si el documento de política no contiene una declaración de política con la acción `iot:CreateProvisioningClaim`, copie el siguiente fragmento de declaración y péguelo como una entrada adicional en la matriz `Statement` del documento de política.

   > **Note**
   >
   > Este fragmento debe colocarse antes del carácter `]` de cierre de la matriz `Statement`. Puede que tenga que agregar una coma antes o después de este fragmento para corregir cualquier error de sintaxis.

```
{
    "Effect": "Allow",
    "Action": [
        "iot:CreateProvisioningClaim"
    ],
    "Resource": [
        "--PUT YOUR NEW TEMPLATE ARN HERE--"
    ]
```

```
}
```

8. Diríjase a la página de la AWS IoT consola en la que eligió Modificar los permisos de los roles de usuario.
9. Busque el ARN del recurso de la plantilla y elija Copiar.
10. Vuelva a la consola de IAM.
11. Pegue el nombre de recurso de Amazon (ARN) copiado en la parte superior de la lista de plantillas ARNs de la `Statement` matriz para que sea la primera entrada.

    Si este es el único ARN de la matriz, quite la coma al final del valor que acabas de pegar.
12. Revise la declaración de política actualizada y corrija los errores indicados por el editor.
13. Para guardar el documento de política actualizado, seleccione Revisar política.
14. Revise la política y, a continuación, seleccione Guardar los cambios.
15. Regrese a la AWS IoT consola.

API de MQTT de aprovisionamiento de dispositivos

El servicio de aprovisionamiento de flotas admite las siguientes operaciones de API de MQTT:

- the section called “CreateCertificateFromCsr”
- the section called “CreateKeysAndCertificate”
- the section called “RegisterThing”

Esta API admite búferes de respuesta en formato de representación concisa de objetos binarios (CBOR) y notación de JavaScript objetos (JSON), según el *`payload-format`* tema. Para una mayor claridad, los ejemplos de respuesta y solicitud de esta sección se muestran en formato JSON.

`payload-format`	Tipo de datos de formato de respuesta
cbor	Concise Binary Object Representation (Representación concisa de objetos binarios, CBOR)
json	JavaScript Notación de objetos (JSON)

> **Important**
>
> Antes de publicar un tema de mensaje de solicitud, suscríbase a los temas de respuesta para recibir la respuesta. Los mensajes utilizados por esta API utilizan el protocolo de publicación/suscripción de MQTT para proporcionar una interacción de solicitud y respuesta.
> Si no se suscribe a los temas de respuesta antes de publicar una solicitud, es posible que no reciba los resultados de dicha solicitud.
> Aprovisionamiento de flotas de IoT Core devuelve los resultados de la API de MQTT de aprovisionamiento del dispositivo a través de la misma conexión con MQTT utilizada para publicar la solicitud de API.

CreateCertificateFromCsr

Crea un certificado a partir de una solicitud de firma de certificado (CSR). AWS IoT proporciona certificados de cliente firmados por la autoridad de certificación (CA) raíz de Amazon. El nuevo certificado tiene un estado `PENDING_ACTIVATION`. Cuando llama a `RegisterThing` para aprovisionar un objeto con este certificado, el estado del certificado cambia a la plantilla `ACTIVE` o `INACTIVE` tal como se describe en ella.

Para obtener más información sobre cómo crear un certificado de cliente con el certificado de la autoridad de certificación y una solicitud de firma de certificado, consulte Crear un certificado de cliente mediante el certificado de entidad de certificación.

> **Note**
>
> Por motivos de seguridad, el `certificateOwnershipToken` devuelto por `CreateCertificateFromCsr` caduca al cabo de una hora. Se debe llamar a `RegisterThing` antes de que venza `certificateOwnershipToken`. Si el certificado creado por `CreateCertificateFromCsr` no se ha activado y no se ha asociado a una política o a un objeto cuando caduque el token, se elimina el certificado. Si el token caduca, el dispositivo puede realizar una llamada `CreateCertificateFromCsr` para generar un certificado nuevo.

CreateCertificateFromCsr solicitud

Publicar un mensaje con el tema `$aws/certificates/create-from-csr/`*`payload-format`*.

`payload-format`

El formato de carga del mensaje como `cbor` o `json`.

CreateCertificateFromCsr solicitar carga útil

```
{
    "certificateSigningRequest": "string"
}
```

`certificateSigningRequest`

La CSR, en formato PEM.

CreateCertificateFromCsr respuesta

Suscripción a `$aws/certificates/create-from-csr/`*`payload-format`*`/accepted`

`payload-format`

El formato de carga del mensaje como `cbor` o `json`.

CreateCertificateFromCsr carga útil de respuesta

```
{
    "certificateOwnershipToken": "string",
    "certificateId": "string",
    "certificatePem": "string"
}
```

`certificateOwnershipToken`

El token para comprobar la propiedad del certificado durante el aprovisionamiento.

`certificateId`

El ID del certificado. Las operaciones de administración de certificados solo adoptan un certificateId.

`certificatePem`

Los datos del certificado, en formato PEM.

CreateCertificateFromCsr error

Para recibir respuestas de error, suscríbase a `$aws/certificates/create-from-csr/payload-format/rejected`.

`payload-format`

El formato de carga del mensaje como `cbor` o `json`.

CreateCertificateFromCsr carga útil de error

```
{
    "statusCode": int,
    "errorCode": "string",
    "errorMessage": "string"
}
```

`statusCode`

El código del estado.

`errorCode`

Código de error.

`errorMessage`

Mensaje de error.

CreateKeysAndCertificate

Crea nuevas claves y un certificado. AWS IoT proporciona certificados de cliente firmados por la autoridad de certificación (CA) raíz de Amazon. El nuevo certificado tiene un estado `PENDING_ACTIVATION`. Cuando llama a `RegisterThing` para aprovisionar un objeto con este certificado, el estado del certificado cambia a la plantilla `ACTIVE` o `INACTIVE` tal como se describe en ella.

> **Note**
>
> Por motivos de seguridad, el `certificateOwnershipToken` devuelto por `CreateKeysAndCertificate` caduca al cabo de una hora. Se debe llamar a

RegisterThing antes de que venza certificateOwnershipToken. Si el certificado creado por CreateKeysAndCertificate no se ha activado y no se ha asociado a una política o a un objeto cuando caduque el token, se elimina el certificado. Si el token caduca, el dispositivo puede realizar una llamada CreateKeysAndCertificate para generar un certificado nuevo.

CreateKeysAndCertificate solicitud

Publicar un mensaje en $aws/certificates/create/*payload-format* con una carga de mensajes vacía.

payload-format

El formato de carga del mensaje como cbor o json.

CreateKeysAndCertificate respuesta

Suscripción a $aws/certificates/create/*payload-format*/accepted

payload-format

El formato de carga del mensaje como cbor o json.

CreateKeysAndCertificate respuesta

```
{
    "certificateId": "string",
    "certificatePem": "string",
    "privateKey": "string",
    "certificateOwnershipToken": "string"
}
```

certificateId

El ID del certificado.

certificatePem

Los datos del certificado, en formato PEM.

`privateKey`

La clave privada.

`certificateOwnershipToken`

El token para comprobar la propiedad del certificado durante el aprovisionamiento.

CreateKeysAndCertificate error

Para recibir respuestas de error, suscríbase a `$aws/certificates/create/`*`payload-format`*`/rejected`.

`payload-format`

El formato de carga del mensaje como `cbor` o `json`.

CreateKeysAndCertificate carga útil de error

```
{
    "statusCode": int,
    "errorCode": "string",
    "errorMessage": "string"
}
```

`statusCode`

El código del estado.

`errorCode`

Código de error.

`errorMessage`

Mensaje de error.

RegisterThing

Aprovisiona un objeto mediante una plantilla predefinida.

RegisterThing solicitud

Publica un mensaje en `$aws/provisioning-templates/`*`templateName`*`/provision/`*`payload-format`*.

`payload-format`

El formato de carga del mensaje como `cbor` o `json`.

`templateName`

El nombre de la plantilla de aprovisionamiento.

RegisterThing solicitar carga útil

```
{
    "certificateOwnershipToken": "string",
    "parameters": {
        "string": "string",
        ...
    }
}
```

`certificateOwnershipToken`

Es el token que demuestra quién es propietario del certificado. AWS IoT genera el token al crear un certificado a través de MQTT.

`parameters`

Opcional. Pares clave-valor del dispositivo que utilizan los enlaces de preaprovisionamiento para evaluar la solicitud de registro.

RegisterThing respuesta

Suscripción a `$aws/provisioning-templates/`*`templateName`*`/provision/`*`payload-format`*`/accepted`

`payload-format`

El formato de carga del mensaje como `cbor` o `json`.

`templateName`

El nombre de la plantilla de aprovisionamiento.

RegisterThing carga útil de respuesta

```
{
    "deviceConfiguration": {
        "string": "string",
        ...
    },
    "thingName": "string"
}
```

`deviceConfiguration`

La configuración del dispositivo definida en la plantilla.

`thingName`

El nombre de la cosa de IoT creado durante el aprovisionamiento.

RegisterThing respuesta de error

Para recibir respuestas de error, suscríbase a `$aws/provisioning-templates/`*`templateName`*`/provision/`*`payload-format`*`/rejected`.

`payload-format`

El formato de carga del mensaje como `cbor` o `json`.

`templateName`

El nombre de la plantilla de aprovisionamiento.

RegisterThing carga útil de respuesta a errores

```
{
    "statusCode": int,
    "errorCode": "string",
    "errorMessage": "string"
```

```
}
```

`statusCode`

El código del estado.

`errorCode`

Código de error.

`errorMessage`

Mensaje de error.

Indexación de flotas

Puede utilizar la indexación de flotas para indexar, buscar y agregar los datos de sus dispositivos de los siguientes orígenes: registro de AWS IoT, AWS IoT Device Shadow, conectividad de AWS IoT, Catálogo de paquetes de software de AWS IoT Device Management e infracciones de AWS IoT Device Defender. Puede consultar un grupo de dispositivos y agregar estadísticas en los registros de dispositivos que se basen en diferentes combinaciones de atributos del dispositivo, como el estado, la conectividad y las infracciones de los dispositivos. Con la indexación de flotas, puede organizar, investigar y solucionar los problemas de su flota de dispositivos.

La indexación de flotas ofrece las siguientes funciones.

Administración de actualizaciones de índices

Puede configurar un índice de flota para indexar las actualizaciones de sus grupos de objetos, registros de objetos, dispositivos ocultos, conectividad de dispositivos e infracciones de dispositivos. Cuando activa la indexación de flotas, AWS IoT crea un índice para sus objetos o grupos de objetos. `AWS_Things` es el índice creado para todos los objetos. `AWS_ThingGroups` es el índice que contiene todos los grupos de objetos. Una vez activa la indexación de flotas, puede ejecutar consultas en su índice. Por ejemplo, puede encontrar todos los dispositivos portátiles que tengan una duración de batería superior al 70 por ciento. AWS IoT actualiza el índice continuamente con sus datos más recientes. Para obtener más información, consulte Administración de la indexación de flotas.

Consulta de estado de conectividad de un dispositivo específico

Esta API proporciona acceso de baja latencia y alto rendimiento a la información de conectividad específica del dispositivo más reciente. Para obtener más información, consulte Estado de conexión del dispositivo.

Buscar en todos los orígenes de datos

Puede crear una cadena de consulta basada en un lenguaje de consulta y utilizarla para buscar en todos los orígenes de datos. También debe configurar los orígenes de datos en la configuración de indexación de flotas para que la configuración de indexación contenga los orígenes de datos en los que desea buscar. En la cadena de consulta se describen los elementos que desea buscar. Puede

crear consultas mediante campos AWS gestionados, campos personalizados y cualquier atributo de sus fuentes de datos indexadas. Para obtener más información sobre los orígenes de datos que admiten la indexación de flotas, consulte Administración de la indexación de objetos.

Consulta de datos agregados

Puede buscar en sus dispositivos datos agregados y obtener estadísticas, percentiles, cardinales o una lista de elementos con consultas de búsqueda sobre campos específicos. Puede ejecutar agregaciones en los campos AWS gestionados o en cualquier atributo que configure como campos personalizados en la configuración de indexación de la flota. Para obtener más información sobre las consultas de agregación, vaya a Consulta de datos agregados.

Monitorización de datos agregados y creación alarmas mediante métricas de flota

Puedes usar las métricas de la flota para enviar datos agregados CloudWatch automáticamente, analizar tendencias y crear alarmas para monitorear el estado agregado de tu flota en función de umbrales predefinidos. Para obtener información sobre las métricas de la flota, consulte Métricas de flota.

Administración de la indexación de flotas

La indexación de flotas administra dos tipos de índices por usted: la indexación de objetos y la indexación de grupos de objetos.

Indexación de objetos

El índice creado para todos sus objetos se llama `AWS_Things`. La indexación de objetos admite los siguientes orígenes de datos: datos de registro de AWS IoT, datos de AWS IoT Device Shadow, datos de conectividad de AWS IoT y datos de infracciones de AWS IoT Device Defender. Al agregar estos orígenes de datos a la configuración de indexación de su flota, puede buscar objetos, consultar datos agregados y crear grupos de objetos dinámicos y métricas de flota basados en sus consultas de búsqueda.

Registro:AWS IoT proporciona un registro que le ayuda a gestionar las cosas. Puede agregar los datos de registro a la configuración de indexación de su flota para buscar dispositivos en función de

los nombres, las descripciones y otros atributos del registro. Para obtener más información acerca del registro, consulte Cómo administrar objetos con el registro.

Sombra: el servicio AWS IoT Device Shadow proporciona sombras que le ayudan a almacenar los datos de estado del dispositivo. La indexación de objetos admite tanto las sombras clásicas sin nombre como las sombras con nombre. Para indexar sombras con nombre, active la configuración de sombras guardadas y especifique los nombres de las sombras en la configuración de indexación de objetos. De forma predeterminada, puede añadir hasta 10 nombres ocultos por cada uno Cuenta de AWS. Para ver cómo aumentar el límite del número de nombres ocultos, consulte Cuotas de AWS IoT Device Management en la Referencia general de AWS .

Para agregar sombras con nombre para la indexación:

- Si utiliza la consola de AWS IoT, active Indexación de objetos, seleccione Agregar sombras con nombre y agregue los nombres de las sombras en Selección de sombra con nombre.
- Si usa AWS Command Line Interface (AWS CLI), `namedShadowIndexingMode` configúrelo en be `ON` y especifique los nombres de las sombras en `IndexingFilter`. Para ver ejemplos de comandos de la CLI, consulte Administración de la indexación de objetos.

> **Important**
>
> El 20 de julio de 2022 estará disponible para el público general (GA) de la integración de indexación de flotas de AWS IoT Device Management, que incluye sombras AWS IoT Core nombradas y AWS IoT Device Defender detección de infracciones. Con esta versión GA, puede indexar sombras con nombres específicos especificando nombres de sombras. Si ha agregado sombras con nombre para indexarlas durante el período de vista previa pública de esta característica, del 30 de noviembre de 2021 al 19 de julio de 2022, le recomendamos que vuelva a configurar los ajustes de indexación de su flota y elija nombres de sombras específicos para reducir los costes de indexación y optimizar el rendimiento.

Para obtener información sobre las sombras, consulte el servicio AWS IoT Device Shadow.

Conectividad: los datos de conectividad de los dispositivos le ayudan a identificar el estado de conexión de sus dispositivos. Estos datos de conectividad dependen de los eventos del ciclo de vida. Cuando un cliente se conecta o se desconecta, AWS IoT publica los eventos del ciclo de vida con mensajes sobre temas de MQTT. Un mensaje de conexión o desconexión puede ser una lista de

elementos JSON que proporcionan detalles del estado de la conexión. Para obtener información sobre la conectividad de los dispositivos, consulte Eventos del ciclo de vida.

Infracciones de Device Defender: los datos sobre AWS IoT Device Defender infracciones ayudan a identificar los comportamientos anómalos de los dispositivos en comparación con los comportamientos normales que se definen en un perfil de seguridad. Un perfil de seguridad contiene un conjunto de comportamientos esperados del dispositivo. Cada comportamiento utiliza una métrica que especifica el comportamiento normal de los dispositivos. Para obtener más información sobre las infracciones de Device Defender, consulte AWS IoT Device Defender detect.

Para obtener más información, consulte Administración de la indexación de flotas.

Indexación de grupos de objetos

`AWS_ThingGroups` es el índice que contiene todos sus grupos de objetos. Puede usar este índice para buscar grupos en función de su nombre, descripción, atributos y todos los nombres de grupos principales.

Para obtener más información, consulte Administración de la indexación de grupos de objetos.

Campos administrados

Los campos gestionados contienen datos asociados a cosas, grupos de cosas, dispositivos ocultos, conectividad de los dispositivos e infracciones de Device Defender. AWS IoT define el tipo de datos en los campos gestionados. Los valores de cada campo gestionado se especifican al crear cualquier AWS IoT cosa. Por ejemplo, los nombres de objetos, los grupos de objetos y las descripciones de objetos son todos campos administrados. La indexación de flotas indexa los campos administrados en función del modo de indexación que usted especifique. Los campos administrados no se pueden cambiar ni pueden aparecer en `customFields`. Para obtener más información, consulte Campos personalizados.

A continuación se enumeran los campos administrados para la indexación de objetos:

- Campos administrados del registro

```
"managedFields" : [
  {name:thingId, type:String},
  {name:thingName, type:String},
  {name:registry.version, type:Number},
  {name:registry.thingTypeName, type:String},
  {name:registry.thingGroupNames, type:String},
```

```
]
```

- Campos administrados para sombras clásicas sin nombre

```
"managedFields" : [
  {name:shadow.version, type:Number},
  {name:shadow.hasDelta, type:Boolean}
]
```

- Campos administrados para sombras con nombre

```
"managedFields" : [
  {name:shadow.name.shadowName.version, type:Number},
  {name:shadow.name.shadowName.hasDelta, type:Boolean}
]
```

- Campos administrados de conectividad de objetos

```
"managedFields" : [
  {name:connectivity.timestamp, type:Number},
  {name:connectivity.version, type:Number},
  {name:connectivity.connected, type:Boolean},
  {name:connectivity.disconnectReason, type:String}
]
```

- Campos administrados para Device Defender

```
"managedFields" : [
  {name:deviceDefender.violationCount, type:Number},
  {name:deviceDefender.securityprofile.behaviorname.metricName, type:String},
  {name:deviceDefender.securityprofile.behaviorname.lastViolationTime, type:Number},
  {name:deviceDefender.securityprofile.behaviorname.lastViolationValue, type:String},
  {name:deviceDefender.securityprofile.behaviorname.inViolation, type:Boolean}
]
```

- Campos administrados de grupos de objetos

```
"managedFields" : [
  {name:description, type:String},
  {name:parentGroupNames, type:String},
  {name:thingGroupId, type:String},
  {name:thingGroupName, type:String},
  {name:version, type:Number},
```

```
]
```

En la siguiente tabla se muestran los campos administrados que no se pueden buscar.

Origen de datos	Campo administrado que no se puede buscar
Registro	`registry.version`
Sombras sin nombre	`shadow.version`
Sombras con nombre	`shadow.name.*.version`
Device Defender	`deviceDefender.version`
Grupos de objetos	`version`

Campos personalizados

Puede agregar atributos de objetos, datos de Device Shadow y datos de infracciones de Device Defender creando campos personalizados para indexarlos. El atributo `customFields` es una lista de pares de campos y tipos de datos con nombre. Puede realizar consultas de agregación en función del tipo de datos. El modo de indexación que elija afecta a los campos se puede especificar en `customFields`. Por ejemplo, si especifica el modo de indexación `REGISTRY`, no puede especificar un campo personalizado de una sombra de objeto. Puede usar el comando update-indexing-configurationCLI para crear o actualizar los campos personalizados (consulte un comando de ejemplo en Actualización de ejemplos de configuración de indexación).

- Nombres de campos personalizados

Los nombres de los campos personalizados para los atributos objetos y grupos de objetos comienzan por `attributes.` seguidos del nombre del atributo. Si la indexación de sombras sin nombre está activada, los objetos pueden tener nombres de campo personalizados que comiencen por `shadow.desired` o `shadow.reported`, seguidos del nombre del valor de los datos de sombras sin nombre. Si la indexación de sombras sin nombre está activada, los objetos pueden tener nombres de campo personalizados que comiencen por `shadow.name.*.desired.` o `shadow.name.*.reported.`, seguidos del valor de los datos de sombras sin nombre. Si la indexación de infracciones de Device Defender está activada, los objetos pueden tener nombres de

campo personalizados que comiencen por `deviceDefender.`, seguidos del valor de los datos de infracciones de Device Defender.

El nombre de atributo o valor de datos que sigue al prefijo puede tener solo caracteres alfanuméricos, - (guion) y _ (guion bajo). No puede tener ningún espacio.

Si existe una incoherencia de tipo entre un campo personalizado de su configuración y el valor que se indexa, la indexación de flotas ignora el valor incoherente para las consultas de agregación. CloudWatch Los registros son útiles para solucionar problemas de consultas de agregación. Para obtener más información, consulte Solución de problemas de consultas de agregación en el servicio de indexación de flotas.

- Tipos de campos personalizados

Los tipos de campos personalizados tienen los siguientes valores admitidos: `Number`, `String` y `Boolean`.

Administración de la indexación de objetos

El índice creado para todos sus objetos es `AWS_Things`. Puede controlar qué indexar de los siguientes orígenes de datos: datos de registro de AWS IoT, datos de AWS IoT Device Shadow, datos de conectividad de AWS IoT y datos de infracciones de AWS IoT Device Defender.

En este tema:

- Habilitación de la indexación de objetos
- Descripción de un índice de objeto
- Consulta de un índice de objeto
- Restricciones y limitaciones
- Autorización

Habilitación de la indexación de objetos

Utilice el comando update-indexing-configurationCLI o la operación UpdateIndexingConfigurationAPI para crear el `AWS_Things` índice y controlar su configuración. Utilizando el parámetro `--thing-indexing-configuration` (`thingIndexingConfiguration`), controla qué tipo de datos se indexan (por ejemplo, registro, sombra, datos de conectividad del dispositivo y datos de infracciones de Device Defender).

El parámetro `--thing-indexing-configuration` toma una cadena con la siguiente estructura:

```
{
  "thingIndexingMode": "OFF"|"REGISTRY"|"REGISTRY_AND_SHADOW",
  "thingConnectivityIndexingMode": "OFF"|"STATUS",
  "deviceDefenderIndexingMode": "OFF"|"VIOLATIONS",
  "namedShadowIndexingMode": "OFF"|"ON",
  "managedFields": [
    {
      "name": "string",
      "type": "Number"|"String"|"Boolean"
    },
    ...
  ],
  "customFields": [
    {
      "name": "string",
      "type": "Number"|"String"|"Boolean"
    },
    ...
  ],
  "filter": {
     "namedShadowNames": [ "string" ],
     "geoLocations": [
        {
            "name": "String",
            "order": "LonLat|LatLon"
        }
    ]
  }
}
```

Modos de indexación de objetos

Puede especificar distintos modos de indexación en su configuración de indexación, en función de los orígenes de datos que desee indexar y de los dispositivos desde los que desee buscar:

- `thingIndexingMode`: controla si el registro o la sombra están indexados. Cuando `thingIndexingMode` se establece en `OFF`, la indexación de objetos está desactivada.

- `thingConnectivityIndexingMode`: especifica si los datos de conectividad de objetos están indexados.

- `deviceDefenderIndexingMode`: especifica si los datos de infracciones de Device Defender están indexados.
- `namedShadowIndexingMode`: especifica si los datos ocultos con nombre están indexados. Para seleccionar sombras con nombre y agregarlas a la configuración de indexación de su flota, establezca `namedShadowIndexingMode` en `ON` y especifique los nombres de las sombras con nombre en `filter`.

La siguiente tabla muestra los valores válidos para cada modo de indexación y el origen de datos que está indexado para cada valor.

Atributo	Valores válidos	Registro	Sombra	Conectividad	Infracciones de DD	Sombra con nombre
`thingIndexingMode`	OFF					
	REGISTRY	✓				
	REGISTRY_AND_SHADOW	✓	✓			
`thingConnectivityIndexingMode`	No especificado.					
	OFF					
	STATUS			✓		
`deviceDefenderIndexingMode`	No especificado.					
	OFF					
	VIOLATIONS				✓	
`namedShadowIndexingMode`	No especificado.					

Atributo	Valores válidos	Registro	Sombra	Conectividad	Infracciones de DD	Sombra con nombre
	OFF					
	ON					✓

Campos administrados y campos personalizados

Campos administrados

Los campos gestionados contienen datos asociados a cosas, grupos de cosas, dispositivos ocultos, conectividad de los dispositivos e infracciones de Device Defender. AWS IoT define el tipo de datos en los campos gestionados. El usuario especifica los valores de cada campo administrado cuando se crea un objeto de AWS IoT . Por ejemplo, los nombres de objetos, los grupos de objetos y las descripciones de objetos son todos campos administrados. La indexación de flotas indexa los campos administrados en función del modo de indexación que usted especifique. Los campos administrados no se pueden cambiar ni pueden aparecer en `customFields`.

Campos personalizados

Puede agregar atributos, datos de Device Shadow y datos de infracciones de Device Defender creando campos personalizados para indexarlos. El atributo `customFields` es una lista de pares de campos y tipos de datos con nombre. Puede realizar consultas de agregación en función del tipo de datos. El modo de indexación que elija afecta a los campos se puede especificar en `customFields`. Por ejemplo, si especifica el modo de indexación `REGISTRY`, no puede especificar un campo personalizado de una sombra de objeto. Puede usar el comando update-indexing-configurationCLI para crear o actualizar los campos personalizados (consulte un comando de ejemplo en Actualización de ejemplos de configuración de indexación). Para obtener más información, consulte Campos personalizados.

Filtro de indexación

El filtro de indexación proporciona selecciones adicionales para las sombras con nombre y los datos de geolocalización.

`namedShadowNames`

Para añadir sombras con nombre a su configuración de indexación de flotas, establezca `namedShadowIndexingMode` en `ON` y especifique los nombres de las sombras con nombre con el filtro `namedShadowNames`.

Ejemplo

```
"filter": {
     "namedShadowNames": [ "namedShadow1", "namedShadow2" ]
}
```

`geoLocations`

Para añadir datos de geolocalización a su configuración de indexación de flotas:

- Si sus datos de geolocalización se almacenan en una sombra clásica (sin nombre), establezca `thingIndexingMode` en REGISTRY_AND_SHADOW y especifique los datos de geolocalización en el filtro `geoLocations`.

  El siguiente filtro de ejemplo especifica un objeto geoLocation en una sombra clásica (sin nombre):

  ```
  "filter": {
       "geoLocations": [
          {
              "name": "shadow.reported.location",
              "order": "LonLat"
          }
       ]
    }
  ```

- Si los datos de geolocalización están almacenados en una sombra con nombre, establezca `namedShadowIndexingMode` en ON, añada el nombre de la sombra al filtro `namedShadowNames` y especifique los datos de geolocalización en el filtro `geoLocations`.

  El siguiente filtro de ejemplo especifica un objeto geoLocation en una sombra con nombre (`nameShadow1`):

  ```
  "filter": {
       "namedShadowNames": [ "namedShadow1" ],
       "geoLocations": [
          {
  ```

```
            "name": "shadow.name.namedShadow1.reported.location",
            "order": "LonLat"
        }
    ]
}
```

Para obtener más información, consulte la referencia IndexingFilterde la AWS IoTAPI.

Actualización de ejemplos de configuración de indexación

Para actualizar la configuración de indexación, utilice el comando AWS IoT update-indexing-configuration CLI. En los siguientes ejemplos se muestra cómo utilizar update-indexing-configuration.

Sintaxis corta:

```
aws iot update-indexing-configuration --thing-indexing-configuration \
'thingIndexingMode=REGISTRY_AND_SHADOW, deviceDefenderIndexingMode=VIOLATIONS,
namedShadowIndexingMode=ON,filter={namedShadowNames=[thing1shadow]},
 thingConnectivityIndexingMode=STATUS,
customFields=[{name=attributes.version,type=Number},
{name=shadow.name.thing1shadow.desired.DefaultDesired, type=String},
 {name=shadow.desired.power, type=Boolean},
{name=deviceDefender.securityProfile1.NUMBER_VALUE_BEHAVIOR.lastViolationValue.number,
 type=Number}]'
```

Sintaxis de JSON:

```
aws iot update-indexing-configuration --cli-input-json \ '{
          "thingIndexingConfiguration": { "thingIndexingMode": "REGISTRY_AND_SHADOW",
          "thingConnectivityIndexingMode": "STATUS",
          "deviceDefenderIndexingMode": "VIOLATIONS",
          "namedShadowIndexingMode": "ON",
          "filter": { "namedShadowNames": ["thing1shadow"]},
          "customFields": [ { "name": "shadow.desired.power", "type": "Boolean" },
          {"name": "attributes.version", "type": "Number"},
          {"name": "shadow.name.thing1shadow.desired.DefaultDesired", "type":
 "String"},
          {"name":
 "deviceDefender.securityProfile1.NUMBER_VALUE_BEHAVIOR.lastViolationValue.number",
 "type": Number} ] } }'
```

Este comando no proporciona ninguna salida.

Para comprobar el estado del índice del objeto, ejecute el comando de la CLI `describe-index`:

```
aws iot describe-index --index-name "AWS_Things"
```

El resultado del comando `describe-index` tendrá un aspecto similar al siguiente:

```
{
    "indexName": "AWS_Things",
    "indexStatus": "ACTIVE",
    "schema": "MULTI_INDEXING_MODE"
}
```

> **Note**
>
> La indexación de la flota puede tardar un momento en actualizar el índice de la flota. Recomendamos esperar a que `indexStatus` aparezca activo antes de usarla. Puede tener valores diferentes en el campo de esquema en función de los orígenes de datos que haya configurado. Para obtener más información, consulte Descripción de un índice de objeto.

Para obtener los detalles de la configuración de indexación, ejecute el comando de la CLI `get-indexing-configuration`:

```
aws iot get-indexing-configuration
```

El resultado del comando `get-indexing-configuration` tendrá un aspecto similar al siguiente:

```
{
    "thingIndexingConfiguration": {
        "thingIndexingMode": "REGISTRY_AND_SHADOW",
        "thingConnectivityIndexingMode": "STATUS",
        "deviceDefenderIndexingMode": "VIOLATIONS",
        "namedShadowIndexingMode": "ON",
        "managedFields": [
            {
                "name": "connectivity.disconnectReason",
                "type": "String"
            },
            {
                "name": "registry.version",
```

```
            "type": "Number"
        },
        {
            "name": "thingName",
            "type": "String"
        },
        {
            "name": "deviceDefender.violationCount",
            "type": "Number"
        },
        {
            "name": "shadow.hasDelta",
            "type": "Boolean"
        },
        {
            "name": "shadow.name.*.version",
            "type": "Number"
        },
        {
            "name": "shadow.version",
            "type": "Number"
        },
        {
            "name": "connectivity.version",
            "type": "Number"
        },
        {
            "name": "connectivity.timestamp",
            "type": "Number"
        },
        {
            "name": "shadow.name.*.hasDelta",
            "type": "Boolean"
        },
        {
            "name": "registry.thingTypeName",
            "type": "String"
        },
        {
            "name": "thingId",
            "type": "String"
        },
        {
            "name": "connectivity.connected",
```

```
                "type": "Boolean"
            },
            {
                "name": "registry.thingGroupNames",
                "type": "String"
            }
        ],
        "customFields": [
            {
                "name": "shadow.name.thing1shadow.desired.DefaultDesired",
                "type": "String"
            },

            {
                "name":
 "deviceDefender.securityProfile1.NUMBER_VALUE_BEHAVIOR.lastViolationValue.number",
                "type": "Number"
            },
            {
                "name": "shadow.desired.power",
                "type": "Boolean"
            },
            {
                "name": "attributes.version",
                "type": "Number"
            }
        ],
        "filter": {
              "namedShadowNames": [
                  "thing1shadow"
              ]
          }
      },
    "thingGroupIndexingConfiguration": {
        "thingGroupIndexingMode": "OFF"
    }
}
```

Para actualizar los campos personalizados, puede ejecutar el comando `update-indexing-configuration`. A continuación tiene un ejemplo:

```
aws iot update-indexing-configuration --thing-indexing-configuration
```

```
 'thingIndexingMode=REGISTRY_AND_SHADOW,customFields=[{name=attributes.version,type=Number},
{name=attributes.color,type=String},{name=shadow.desired.power,type=Boolean},
{name=shadow.desired.intensity,type=Number}]'
```

Este comando se agregó `shadow.desired.intensity` a la configuración de indexación.

> **Note**
>
> La actualización de los campos personalizados en la configuración de indexación sobrescribe todos los campos personalizados existentes. Asegúrese de especificar todos los campos personalizados cuando llame a update-indexing-configuration.

Después de reconstruir el índice, puede utilizar una consulta de agregación en los campos recién agregados, los datos de registro de búsqueda, los datos de sombras y los datos de estado de conectividad de objetos.

Al cambiar el modo de indexación, asegúrese de que todos los campos personalizados son válidos en el nuevo modo de indexación. Por ejemplo, si utiliza el modo `REGISTRY_AND_SHADOW` con un campo personalizado llamado `shadow.desired.temperature`, debe eliminar el campo personalizado `shadow.desired.temperature` antes de cambiar el modo de indexación a `REGISTRY`. Si la configuración de indexación contiene campos personalizados que no están indexados por el modo de indexación, se producirá un error en la actualización.

Descripción de un índice de objeto

El comando siguiente muestra cómo utilizar el comando describe-index de la CLI para recuperar el estado actual del índice del objeto:

```
aws iot describe-index --index-name "AWS_Things"
```

La respuesta del comando tendrá un aspecto similar al siguiente:

```
{
    "indexName": "AWS_Things",
    "indexStatus": "BUILDING",
    "schema": "REGISTRY_AND_SHADOW_AND_CONNECTIVITY_STATUS"
}
```

La primera vez que indexa la flota, AWS IoT crea su índice. Cuando `indexStatus` está en el estado `BUILDING`, no se puede consultar el índice. El valor `schema` del índice de objetos indica qué tipo de datos (`REGISTRY_AND_SHADOW_AND_CONNECTIVITY_STATUS`) se indexan.

Si se cambia la configuración del índice, el índice se vuelve a compilar. Durante este proceso, `indexStatus` es `REBUILDING`. Puede ejecutar consultas en los datos del índice de objetos mientras se está generando. Por ejemplo, si cambia la configuración del índice de `REGISTRY` a `REGISTRY_AND_SHADOW`, cuando el índice se vuelve a generar, puede consultar los datos del registro, incluidas las últimas actualizaciones. Sin embargo, no puede consultar los datos de sombra hasta que se complete la recompilación. El tiempo que tarda en compilar o recompilar el índice depende de la cantidad de datos.

Puede ver diferentes valores en el campo de esquema en función de los orígenes de datos que haya configurado. En la siguiente tabla, se muestran los diferentes valores de esquema y las descripciones correspondientes:

Esquema	Description (Descripción)
OFF	No hay ningún origen de datos configurado ni indexado.
REGISTRY	Solo se indexan los datos de registro.
REGISTRY_AND_SHADOW	Se indexan los datos del registro y los datos de sombras sin nombre (clásicos).
REGISTRY_AND_CONNECTIVITY	Se indexan los datos de registro y conectividad.
REGISTRY_AND_SHADOW_AND_CONNECTIVITY_STATUS	Se indexan los datos de registro, los datos de sombras sin nombre (clásicos) y los datos de conectividad.
MULTI_INDEXING_MODE	Se indexan los datos de sombras con nombre o las infracciones de Device Defender, además de los datos de registro, de sombras sin nombre (clásicos) o de conectividad.

Consulta de un índice de objeto

Puede utilizar el comando search-index de la CLI para consultar los datos del índice.

```
aws iot search-index --index-name "AWS_Things" --query-string
          "thingName:mything*"
```


```
{
    "things":[{
         "thingName":"mything1",
         "thingGroupNames":[
            "mygroup1"
         ],
         "thingId":"a4b9f759-b0f2-4857-8a4b-967745ed9f4e",
         "attributes":{
            "attribute1":"abc"
         },
         "connectivity": {
            "connected":false,
            "timestamp":1556649874716,
            "disconnectReason": "CONNECTION_LOST"
         }
    },
    {
        "thingName":"mything2",
        "thingTypeName":"MyThingType",
        "thingGroupNames":[
            "mygroup1",
            "mygroup2"
        ],
        "thingId":"01014ef9-e97e-44c6-985a-d0b06924f2af",
        "attributes":{
            "model":"1.2",
            "country":"usa"
        },
        "shadow":{
            "desired":{
                "location":"new york",
                "myvalues":[3, 4, 5]
            },
            "reported":{
                "location":"new york",
                "myvalues":[1, 2, 3],
```

```
                "stats":{
                    "battery":78
                }
            },
            "metadata":{
                 "desired":{
                       "location":{
                            "timestamp":123456789
                        },
                       "myvalues":{
                            "timestamp":123456789
                       }
                 },
                 "reported":{
                       "location":{
                            "timestamp":34535454
                        },
                       "myvalues":{
                            "timestamp":34535454
                       },
                       "stats":{
                            "battery":{
                                  "timestamp":34535454
                            }
                       }
                }
            },
            "version":10,
            "timestamp":34535454
        },
        "connectivity": {
            "connected":true,
            "timestamp":1556649855046
        }
    }],
    "nextToken":"AQFCuvk7zZ3D9pOYMbFCeHbdZ+h=G"
}
```

En la respuesta JSON,"`connectivity`" (según lo permita configuración de `thingConnectivityIndexingMode=STATUS`) proporciona un valor booleano, una marca temporal y un parámetro disconnectReason que indica si el dispositivo está conectado a AWS IoT Core. El dispositivo "`mything1`" desconectado (`false`) en tiempo POSIX 1556649874716

debido a CONNECTION_LOST. Para obtener información sobre los motivos de desconexión, consulte Eventos del ciclo de vida.

```
"connectivity": {
    "connected":false,
    "timestamp":1556649874716,
    "disconnectReason": "CONNECTION_LOST"
}
```

El dispositivo "mything2" conectado (true) en tiempo POSIX 1556649855046:

```
"connectivity": {
    "connected":true,
    "timestamp":1556649855046
}
```

Las marcas temporales se indican en milisegundos desde la fecha de inicio, por lo que 1556649855046 representa 6:44:15.046 p.m. el martes 30 de abril de 2019 (UTC).

> **Important**
>
> Si un dispositivo se ha desconectado durante aproximadamente una hora, el valor "timestamp" y el valor "disconnectReason" del estado de conectividad podrían no aparecer.

Restricciones y limitaciones

Estas son las restricciones y limitaciones de AWS_Things.

Campos de sombra con tipos complejos

Un campo de sombra se indexa solo si el valor del campo es un tipo sencillo, como un objeto JSON que no incluya una matriz o una matriz que conste completamente de tipos sencillos. Un tipo sencillo es una cadena, un número o uno de los literales true o false. Por ejemplo, dado el siguiente estado de sombra, el valor del campo "palette" no se indexa porque es una matriz que incluye elementos de tipos complejos. El valor del campo "colors" sí se indexará porque cada valor de la matriz es una cadena.

```
{
```

```
    "state": {
        "reported": {
            "switched": "ON",
            "colors": [ "RED", "GREEN", "BLUE" ],
            "palette": [
                {
                    "name": "RED",
                    "intensity": 124
                },
                {
                    "name": "GREEN",
                    "intensity": 68
                },
                {
                    "name": "BLUE",
                    "intensity": 201
                }
            ]
        }
    }
}
```

Nombres de campos de sombra anidados

Los nombres de los campos de sombra anidados se almacenan como una cadena delimitada por punto (.). Por ejemplo, dado un documento de sombra:

```
{
  "state": {
    "desired": {
      "one": {
        "two": {
          "three": "v2"
        }
      }
    }
  }
}
```

El nombre del campo `three` se almacena como `desired.one.two.three`. Si también tiene un documento de sombra, se guarda del modo siguiente:

```
{
```

```
  "state": {
    "desired": {
      "one.two.three": "v2"
    }
  }
}
```

Ambos coinciden con una consulta para `shadow.desired.one.two.three:v2`. La práctica recomendada es no utilizar puntos en los nombres de campos de sombra.

Metadatos de sombra

Un campo en una sección de metadatos de sombra se indexa, pero solo si se indexa el campo correspondiente en la sección `"state"` de la sombra. (En el ejemplo anterior, el campo `"palette"` de la sección de metadatos de la sombra tampoco se indexa).

Dispositivos no registrados

La indexación de flotas indexa el estado de conectividad de un dispositivo cuya conexión `clientId` es la misma que la `thingName` de un dispositivo registrado en el Registro.

Sombras no registradas

Si antes UpdateThingShadowcreabas una sombra con el nombre de un objeto que no estaba registrado en tu AWS IoT cuenta, los campos de esa sombra no se indexarán. Esto se aplica tanto a la sombra sin nombre clásica como a la sombra con nombre.

Valores numéricos

Si el servicio reconoce como valor numérico algún dato de sombra o de registro, se indexa como tal. Puede formar consultas que impliquen rangos y operadores de comparación en valores numéricos (por ejemplo `"attribute.foo<5"` o `"shadow.reported.foo:[75 TO 80]"`). Para que se reconozca como numérico, el valor de los datos debe ser un número JSON válido y de tipo literal. El valor puede ser un entero en el rango -2^53...2^53-1, un punto flotante de doble precisión con notación exponencial opcional, o parte de una matriz que contenga solo estos valores.

Valores nulos

Los valores nulos no se indexan.

Valores máximos

El número máximo de campos personalizados para consultas de agregación es 5.

El número máximo de percentiles solicitados para consultas de agregación es 100.

Autorización

Puede especificar el índice de cosas como un nombre de recurso de Amazon (ARN) en una acción AWS IoT política, de la siguiente manera.

Action	Recurso
`iot:SearchIndex`	El ARN de un índice (por ejemplo, `arn:aws:iot:` *`your-aws-region your-aws-account`* `:index/AWS_Things`).
`iot:DescribeIndex`	El ARN de un índice (por ejemplo, `arn:aws:iot:` *`your-aws-region`* `:index/AWS_Things`).

> **Note**
>
> Si tiene los permisos para consultar el índice de la flota, podrá obtener acceso a los datos de los objetos en toda la flota.

Administración de la indexación de grupos de objetos

`AWS_ThingGroups` es el índice que contiene todos sus grupos de objetos. Puede usar este índice para buscar grupos en función de su nombre, descripción, atributos y todos los nombres de grupos principales.

Habilitación de la indexación de grupos de objetos

Puede utilizar la `thing-group-indexing-configuration` configuración de la UpdateIndexingConfigurationAPI para crear el `AWS_ThingGroups` índice y controlar su configuración. Puede usar la GetIndexingConfigurationAPI para recuperar la configuración de indexación actual.

Para actualizar las configuraciones de indexación del grupo de objetos, ejecute el comando de la CLI update-indexing-configuration:

```
aws iot update-indexing-configuration --thing-group-indexing-configuration
 thingGroupIndexingMode=ON
```

También puede actualizar las configuraciones de la indexación de objetos y grupos de objetos en un único comando, como en el siguiente ejemplo.

```
aws iot update-indexing-configuration --thing-indexing-configuration
 thingIndexingMode=REGISTRY --thing-group-indexing-configuration
 thingGroupIndexingMode=ON
```

Los siguientes valores son válidos para `thingGroupIndexingMode`.

OFF

Sin indexing/delete índice.

ON

Cree o configure el índice `AWS_ThingGroups`.

Para recuperar las configuraciones actuales de indexación de objetos y grupos de objetos, ejecute el comando de la CLI get-indexing-configuration:

```
aws iot get-indexing-configuration
```

La respuesta del comando tendrá un aspecto similar al siguiente:

```
{
   "thingGroupIndexingConfiguration": {
        "thingGroupIndexingMode": "ON"
    }
}
```

Descripción de índices de grupos

Para recuperar el estado actual del índice `AWS_ThingGroups`, utilice el comando de la CLI describe-index:

```
aws iot describe-index --index-name "AWS_ThingGroups"
```

La respuesta del comando tendrá un aspecto similar al siguiente:

```
{
   "indexStatus": "ACTIVE",
   "indexName": "AWS_ThingGroups",
   "schema": "THING_GROUPS"
}
```

AWS IoT crea el índice la primera vez que indexa. No se puede consultar el índice si `indexStatus` es `BUILDING`.

Consulta de un índice de grupo de objetos

Para consultar los datos del índice, utilice el comando de la CLI search-index:

```
aws iot search-index --index-name "AWS_ThingGroups" --query-string
 "thingGroupName:mythinggroup*"
```

Autorización

Puede especificar el índice de grupos de cosas como un ARN de recurso en una acción de AWS IoT política, de la siguiente manera.

Action	Recurso
`iot:SearchIndex`	El ARN de un índice (por ejemplo, `arn:aws:iot:` *`your-aws-region`* `:index/AWS_ThingGroups`).
`iot:DescribeIndex`	El ARN de un índice (por ejemplo, `arn:aws:iot:` *`your-aws-region`* `:index/AWS_ThingGroups`).

Consultas de estado de conectividad de los dispositivos

AWS IoT La indexación de flotas admite consultas de conectividad de dispositivos individuales, lo que le permite recuperar de manera eficiente el estado de la conectividad y los metadatos relacionados para dispositivos específicos. Esta característica complementa las capacidades de indexación y consulta existentes en toda la flota.

Funcionamiento

La compatibilidad con las consultas de conectividad de los dispositivos se puede utilizar para recuperar de forma optimizada el estado de la conectividad de un solo dispositivo. Esta API proporciona acceso de baja latencia y alto rendimiento a la información de conectividad específica del dispositivo más reciente. Una vez que habilite la indexación de conectividad, tendrá acceso a esta API de consultas, que se cobrará como consultas estándar. Para obtener más información, consulte Precios de AWS IoT Device Management.

Características

Gracias a la compatibilidad con las consultas de conectividad de los dispositivos, puede:

1. Consultar el estado de conectividad actual (conectado o desconectado) de un dispositivo determinado mediante su `thingName`.
2. Recuperar metadatos de conectividad adicionales, como, por ejemplo:
   a. Motivo de desconexión
   b. Marcas de tiempo de los eventos de conexión o desconexión más recientes.

> **Note**
>
> La indexación de flotas indexa el estado de conectividad de un dispositivo cuya conexión `clientId` es la misma que la `thingName` de un dispositivo registrado en el Registro.

Ventajas

1. Baja latencia: refleja el estado de conectividad del dispositivo más reciente y ofrece una latencia baja para reflejar los cambios en el estado de la conexión con respecto a IoT Core. IoT Core determina que un dispositivo está desconectado en cuanto recibe una solicitud de desconexión del dispositivo o en caso de que un dispositivo se desconecte sin enviar una solicitud de desconexión. IoT Core esperará 1,5 veces el tiempo de keep-alive configurado antes de que se determine que el cliente está desconectado. La API de estado de conectividad reflejará estos cambios normalmente en menos de un segundo después de que IoT Core determine el cambio de estado de conexión de un dispositivo.
2. Alto rendimiento: admite 350 transacciones por segundo (TPS) de forma predeterminada y se puede ajustar en un nivel superior si se solicita.

3. Retención de datos: almacena los datos de los eventos de forma indefinida cuando el ConnectivityIndexing modo de indexación de flotas (FI) está activado y el elemento no se elimina. Si deshabilita la opción Indexación de conectividad, los registros no se conservarán.

> **Note**
>
> Si la indexación del estado de la conectividad estaba habilitada antes del lanzamiento de esta API, la opción Indexación de flotas comienza a realizar un seguimiento de los cambios en el estado de la conectividad después del lanzamiento de la API y refleja el estado actualizado en función de esos cambios.

Requisitos previos

Cómo utilizar el soporte de consultas sobre conectividad de dispositivos:

1. Configura una cuenta AWS
2. Incorpora y registra dispositivos AWS IoT Core en la región que prefieras
3. Habilitar Indexación de flotas con Indexación de conectividad

> **Note**
>
> No se requiere ninguna configuración adicional si ya tiene habilitada la indexación de conectividad.

Para obtener instrucciones detalladas de configuración, consulte la Guía para desarrolladores de AWS IoT.

Ejemplos

```
aws iot get-thing-connectivity-data --thing-name myThingName
```

```
{
   "connected": true,
   "disconnectReason": "NONE",
```

```
   "thingName": "myThingName",
   "timestamp": "2024-12-19T10:00:00.000000-08:00"
}
```

- `thingName`: el nombre del dispositivo, tal y como se indica en la solicitud. También coincide con el clientId utilizado para conectarse a AWS IoT Core.
- `disconnectReason`: motivo de la desconexión. Será NONE para un dispositivo conectado.
- `connected`: el valor booleano true indica que este dispositivo está conectado actualmente.
- `timestamp`: la marca de tiempo que representa la última desconexión del dispositivo en milisegundos.

```
aws iot get-thing-connectivity-data --thing-name myThingName
```

```
{
   "connected": false,
   "disconnectReason": "CLIENT_INITIATED_DISCONNECT",
   "thingName": "myThingName",
   "timestamp": "2024-12-19T10:30:00.000000-08:00"
}
```

- `thingName`: el nombre del dispositivo, tal y como se indica en la solicitud. También coincide con el clientId utilizado para conectarse a AWS IoT Core.
- `disconnectReason`: El motivo de la desconexión es CLIENT_INITIATED_DISCONNECT, que indica el cliente que quiere desconectarse AWS IoT Core .
- `connected`: el valor booleano false indica que este dispositivo está desconectado actualmente.
- `timestamp`: la marca de tiempo que representa la última desconexión del dispositivo en milisegundos.

```
aws iot get-thing-connectivity-data --thing-name neverConnectedThing
```

```
{
   "connected": false,
   "disconnectReason": "UNKNOWN",
```

```
    "thingName": "neverConnectedThing"
}
```

- `thingName`: el nombre del dispositivo, tal y como se indica en la solicitud. También coincide con el clientId utilizado para conectarse a AWS IoT Core.
- `disconnectReason`: motivo de la desconexión. Será “UNKNOWN” para un dispositivo que nunca se haya conectado o para el que la opción Indexación de flotas no tenga almacenado el motivo de la última desconexión.
- `connected`: el valor booleano false indica que este dispositivo está desconectado actualmente.
- `timestamp`: la marca de tiempo no se devuelve para un dispositivo que nunca se haya conectado o para el que la opción Indexación de flotas no tenga almacenada la última marca de tiempo.

Consulta de datos agregados

AWS IoT proporciona cuatro APIs (`GetStatistics`, `GetCardinalityGetPercentiles`, y`GetBucketsAggregation`) que le permiten buscar datos agregados en su flota de dispositivos.

> **Note**
>
> Si hay problemas relacionados con valores faltantes o inesperados en la agregación APIs, consulta la guía de solución de problemas de indexación de flotas.

GetStatistics

La GetStatisticsAPI y el comando get-statistics CLI devuelven el recuento, el promedio, la suma, el mínimo, el máximo, la suma de los cuadrados, la varianza y la desviación estándar del campo agregado especificado.

El comando get-statistics de la CLI usa los siguientes parámetros:

`index-name`

El nombre del índice que se buscará. El valor predeterminado es `AWS_Things`.

`query-string`

La consulta utilizada para buscar el índice. Puede especificar si desea `"*"` obtener el recuento de todos los elementos indexados de su. Cuenta de AWS

`aggregationField`

(Opcional) El campo que se va a agregar. Este campo debe ser un campo administrado o personalizado definido al llamar a update-indexing-configuration. Si no especifica un campo de agregación, se utiliza `registry.version` como el campo de agregación.

`query-version`

La versión de la consulta que se va a utilizar. El valor predeterminado es `2017-09-30`.

El tipo de campo de agregación puede afectar a las estadísticas devueltas.

GetStatistics con valores de cadena

Si realiza la agregación en un campo de cadena, la llamada a `GetStatistics` devuelve el número de dispositivos que tienen atributos que coinciden con la consulta. Por ejemplo:

```
aws iot get-statistics --aggregation-field 'attributes.stringAttribute'
            --query-string '*'
```

Este comando devuelve el número de dispositivos que contienen un atributo llamado `stringAttribute`:

```
{
  "statistics": {
    "count": 3
  }
}
```

GetStatistics con valores booleanos

Cuando llama a `GetStatistics` con un campo de agregación booleano:

- AVERAGE es el porcentaje de dispositivos que coinciden con la consulta.
- MINIMUM es 0 o 1 de acuerdo con las siguientes reglas:
  - Si todos los valores del campo de agregación son `false`, MINIMUM es 0.
  - Si todos los valores del campo de agregación son `true`, MINIMUM es 1.
  - Si los valores del campo de agregación son una mezcla de `false` y `true`, MINIMUM es 0.
- MAXIMUM es 0 o 1 de acuerdo con las siguientes reglas:

  - Si todos los valores del campo de agregación son `false`, MAXIMUM es 0.
  - Si todos los valores del campo de agregación son `true`, MAXIMUM es 1.
  - Si los valores del campo de agregación son una mezcla de `false` y `true`, MAXIMUM es 1.
- SUM es la suma del entero equivalente de los valores booleanos.
- COUNT es el recuento de objetos que coinciden con los criterios de la cadena de consulta y contienen un valor de campo de agregación válido.

GetStatistics con valores numéricos

Cuando se llama a `GetStatistics` y se especifica un campo de agregación de tipo `Number`, `GetStatistics` devuelve los siguientes valores:

count

El número de objetos que coinciden con los criterios de la cadena de consulta y contienen un valor de campo de agregación válido.

average

El promedio de los valores numéricos que coinciden con la consulta.

sum

La suma de los valores numéricos que coinciden con la consulta.

minimum

El menor de los valores numéricos que coinciden con la consulta.

maximum

El mayor de los valores numéricos que coinciden con la consulta.

sumOfSquares

La suma de los cuadrados de los valores numéricos que coinciden con la consulta.

variance

La varianza de los valores numéricos que coinciden con la consulta. La varianza de un conjunto de valores es la media de los cuadrados de las diferencias de cada valor con respecto al valor medio del conjunto.

stdDeviation

La desviación estándar de los valores numéricos que coinciden con la consulta. La desviación estándar de un conjunto de valores es una medida de la distribución de los valores.

El siguiente ejemplo muestra cómo llamar a get-statistics con un campo numérico personalizado.

```
aws iot get-statistics --aggregation-field 'attributes.numericAttribute2'
            --query-string '*'
```

```
{
  "statistics": {
    "count": 3,
    "average": 33.333333333333336,
    "sum": 100.0,
    "minimum": -125.0,
    "maximum": 150.0,
    "sumOfSquares": 43750.0,
    "variance": 13472.22222222222,
    "stdDeviation": 116.06990230986766
  }
}
```

Para los campos de agregación numérica, si los valores del campo superan el valor «double» máximo, los valores de las estadísticas están vacíos.

GetCardinality

La GetCardinalityAPI y el comando get-cardinality CLI devuelven el recuento aproximado de valores únicos que coinciden con la consulta. Por ejemplo, es posible que desee encontrar el número de dispositivos con niveles de batería inferiores al 50 por ciento:

```
aws iot get-cardinality --index-name AWS_Things --query-string "batterylevel
          > 50" --aggregation-field "shadow.reported.batterylevel"
```

Este comando devuelve el número de objetos con niveles de batería de más del 50 por ciento:

```
{
    "cardinality": 100
}
```

get-cardinality siempre devuelve `cardinality`, aunque no haya campos coincidentes. Por ejemplo:

```
aws iot get-cardinality --query-string "thingName:Non-existent*"
          --aggregation-field "attributes.customField_STR"
```

```
{
    "cardinality": 0
}
```

El comando get-cardinality de la CLI usa los siguientes parámetros:

`index-name`

El nombre del índice que se buscará. El valor predeterminado es `AWS_Things`.

`query-string`

La consulta utilizada para buscar el índice. Puede especificar si desea `"*"` obtener el recuento de todos los elementos indexados de su Cuenta de AWS.

`aggregationField`

El campo que se va a agregar.

`query-version`

La versión de la consulta que se va a utilizar. El valor predeterminado es `2017-09-30`.

GetPercentiles

La GetPercentilesAPI y el comando get-percentiles CLI agrupan los valores agregados que coinciden con la consulta en grupos de percentiles. Los grupos de percentiles predeterminados son: 1,5,25,50,75,95,99, aunque puede especificar los suyos propios cuando llame a `GetPercentiles`. Esta función devuelve un valor para cada grupo de percentiles especificado (o para los grupos de percentiles predeterminados). El grupo de percentiles "1" contiene el valor agregado del campo que se obtiene aproximadamente en el uno por ciento de los valores que coinciden con la consulta. El grupo de percentiles "5" contiene el valor agregado del campo que se obtiene en aproximadamente el cinco por ciento de los valores que coinciden con la consulta, y así sucesivamente. El resultado es una aproximación: cuantos más valores coincidan con la consulta, más precisos serán los valores de percentil.

El siguiente ejemplo muestra cómo llamar al comando get-percentiles de la CLI.

```
aws iot get-percentiles --query-string "thingName:*" --aggregation-field
          "attributes.customField_NUM" --percents 10 20 30 40 50 60 70 80 90 99
```


```
{
    "percentiles": [
        {
            "value": 3.0,
            "percent": 80.0
        },
        {
            "value": 2.5999999999999996,
            "percent": 70.0
        },
        {
            "value": 3.0,
            "percent": 90.0
        },
        {
            "value": 2.0,
            "percent": 50.0
        },
        {
            "value": 2.0,
            "percent": 60.0
        },
        {
            "value": 1.0,
            "percent": 10.0
        },
        {
            "value": 2.0,
            "percent": 40.0
        },
        {
            "value": 1.0,
            "percent": 20.0
        },
        {
            "value": 1.4,
            "percent": 30.0
        },
        {
            "value": 3.0,
```

```
            "percent": 99.0
        }
    ]
}
```

El siguiente comando muestra la salida devuelta get-percentiles cuando no hay documentos coincidentes.

```
aws iot get-percentiles --query-string "thingName:Non-existent*"
          --aggregation-field "attributes.customField_NUM"
```

```
{
    "percentiles": []
}
```

El comando get-percentile de la CLI usa los siguientes parámetros:

`index-name`

El nombre del índice que se buscará. El valor predeterminado es `AWS_Things`.

`query-string`

La consulta utilizada para buscar el índice. Puede especificar si desea `"*"` obtener el recuento de todos los elementos indexados de su. Cuenta de AWS

`aggregationField`

El campo que se va a agregar, que debe ser del tipo `Number`.

`query-version`

La versión de la consulta que se va a utilizar. El valor predeterminado es `2017-09-30`.

`percents`

(Opcional) Puede utilizar este parámetro para especificar grupos de percentiles personalizados.

GetBucketsAggregation

La GetBucketsAggregationAPI y el comando get-buckets-aggregation CLI devuelven una lista de depósitos y el número total de elementos que se ajustan a los criterios de la cadena de consulta.

El siguiente ejemplo muestra cómo llamar al comando get-buckets-aggregation de la CLI.

```
aws iot get-buckets-aggregation --query-string '*' --index-name AWS_Things --
aggregation-field 'shadow.reported.batterylevelpercent' --buckets-aggregation-type
 'termsAggregation={maxBuckets=5}'
```

Este comando devuelve la siguiente salida:

```
{
    "totalCount": 20,
    "buckets": [
        {
            "keyValue": "100",
            "count": 12
        },
        {
            "keyValue": "90",
            "count": 5
        },
        {
            "keyValue": "75",
            "count": 3
        }
    ]
}
```

El comando get-buckets-aggregation de la CLI usa los siguientes parámetros:

`index-name`

El nombre del índice que se buscará. El valor predeterminado es `AWS_Things`.

`query-string`

La consulta utilizada para buscar el índice. Puede especificar si desea `"*"` obtener el recuento de todos los elementos indexados de su. Cuenta de AWS

`aggregation-field`

El campo que se va a agregar.

`buckets-aggregation-type`

El control básico de la forma de la respuesta y el tipo de agregación de buckets que se va a realizar.

Autorización

Puede especificar el índice de grupos de cosas como un ARN de recurso en una acción de AWS IoT política, de la siguiente manera.

Action	Recurso
`iot:GetStatistics`	El ARN de un índice (por ejemplo, `arn:aws:iot:` *`your-aws-region`* `:index/AWS_Things` o `arn:aws:iot:` *`your-aws-region`* `:index/AWS_ThingGroups`).

Sintaxis de la consulta

En la indexación de flotas, se utiliza una sintaxis de consulta para especificar las consultas.

Características admitidas

La sintaxis de consulta es compatible con las siguientes características.

- Términos y frases
- Búsqueda de campos
- Búsqueda de prefijos
- Búsqueda de intervalos
- Operadores booleanos `AND`, `OR`, `NOT` y `-`. El guion se utiliza para excluir algo de los resultados de la búsqueda (por ejemplo, `thingName:(tv* AND -plasma)`).
- Agrupación
- Agrupación de campos
- Utilizar secuencias de escape con caracteres especiales (por ejemplo, con \)
- El uso de caracteres comodín al principio está limitado a un término de consulta por consulta. Por ejemplo, no puede buscar `thingName:*my` ni `thingGroupNames:*echo` en la misma consulta. Las consultas que incluyen un carácter comodín al principio tienen una longitud máxima de 100 caracteres.

> **Note**
>
> Las cuentas con un límite de consultas en la API de indexación de flotas superior a 30 solicitudes por segundo están sujetas a restricciones cuando se utilizan los primeros términos de consulta con caracteres comodín. Este requests-per-second límite de 30 se aplica a nivel de cuenta en todas las consultas de indexación de Fleet. APIs

Características no admitidas

La sintaxis de consulta no es compatible con las siguientes características:

- Expresiones regulares
- Impulso
- Clasificación
- Búsquedas confusas
- Búsqueda de proximidad
- Ordenar
- Agregación
- Caracteres especiales: `` ` ``, `@`, `#`, `%`, `\`, `/`, `'`, `;` y `,`. Tenga en cuenta que `,` solo se admite en las geoconsultas.

Notas

Algunas cosas que tener en cuenta acerca del idioma de consulta:

- El operador predeterminado es AND. La consulta `"thingName:abc thingType:xyz"` es igual que `"thingName:abc AND thingType:xyz"`.
- Si no se especifica ningún campo, AWS IoT busca el término en todos los campos del registro, Device Shadow y Device Defender.
- Todos los nombres de campo distinguen entre mayúsculas y minúsculas.
- La búsqueda distingue entre mayúsculas y minúsculas. Las palabras están separadas por caracteres de espacio en blanco, tal como define la especificación `Character.isWhitespace(int)` de Java.

- La indexación de los datos de Device Shadow (sombras sin nombre y sombras con nombre) incluye las secciones reported, desired, delta, y de metadatos.
- Las versiones de Device Shadow y del registro no permiten búsquedas, pero están presentes en la respuesta.
- El número máximo de términos en una consulta es 12.
- El carácter especial `,` solo se admite en las geoconsultas.

Ejemplo de consultas de objetos

Especifique las consultas en una cadena de consulta mediante una sintaxis de consulta. Las consultas se pasan a la API `SearchIndex`. En la siguiente tabla se enumeran algunas cadenas de consulta de ejemplo.

Cadena de consulta	Resultado
`abc`	Consulta “abc” en cualquier campo de registro, sombra (la clásica sombra sin nombre y sombra con nombre) o Device Defender.
`thingName:myThingName`	Consulta un elemento con el nombre "myThingName».
`thingName:my*`	Consulta los objetos cuyos nombres que comienzan por "my".
`thingName:*my`	Consultas de objetos cuyos nombres finalizan por “my”. El uso de caracteres comodín al principio está limitado a un término de consulta por consulta. Las consultas que incluyen un carácter comodín al principio tienen una longitud máxima de 100 caracteres.
`thingName:*my*`	Consultas de objetos que incluyen la subcadena “my”. El uso de caracteres comodín al principio está limitado a un término de consulta por consulta. Las consultas que incluyen un carácter comodín al principio tienen una longitud máxima de 100 caracteres.

Cadena de consulta	Resultado
`thingName:ab?`	Consulta los objetos cuyos nombres tienen “ab“ además de un carácter adicional (por ejemplo: “aba”, “abb”, “abc”, etc.)
`thingTypeName:aa`	Consulta los objetos que están asociados con el tipo “aa”.
`thingGroupNames:a`	Consulta los objetos con nombre de grupo de objetos principal o grupo de facturación “a”.
`thingGroupNames:a*`	Consulta los objetos con nombre de grupo de objetos principal o grupo de facturación que coincide con el patrón “a*”.
`attributes.myAttribute:75`	Consulta los objetos con un atributo denominado "myAttribute" que tiene el valor 75.
`attributes.myAttribute:[75 TO 80]`	Consulta los objetos con un atributo denominado “myAttribute”, cuyo valor se encuentra dentro de un rango numérico (entre 75-80, ambos inclusive).
`attributes.myAttribute:{75 TO 80]`	Consulta los objetos con un atributo denominado “myAttribute”, cuyo valor se encuentra dentro del rango numérico (>75 y <=80).
`attributes.serialNumber:["abcd" TO "abcf"]`	Consulta los objetos con un atributo llamado “serialNumber”, cuyo valor se encuentra dentro del rango de cadenas alfanuméricas. Esta consulta devuelve objetos con un atributo "serialNumber" con valores "abcd", "abce" o "abcf".
`attributes.myAttribute:i*t`	Consulta los objetos con un atributo llamado “MyAttribute” cuyo valor es 'i', seguido de un número de caracteres, seguido por 't'.

Cadena de consulta	Resultado
`attributes.attr1:abc AND attributes.attr2<5 NOT attributes.attr3>10`	Consultas de objetos que combinan términos mediante expresiones booleanas. Esta consulta devuelve objetos que tengan un atributo llamado “attr1” con un valor “abc”, un atributo denominado “attr2” inferior a 5 y un atributo llamado “attr3” que no sea superior a 10.
`shadow.hasDelta:true`	Consulta los objetos con una sombra sin nombre que tenga un elemento delta.
`NOT attributes.model:legacy`	Consultas de objetos donde el atributo llamado "model" no es "legacy".
`shadow.reported.stats.battery:{70 TO 100} (v2 OR v3) NOT attributes.model:legacy`	Consulta los objetos que cumplen lo siguiente: • El atributo `stats.battery` de sombra del objeto tiene un valor entre 70 y 100. • El texto "v2" o "v3" aparece en los valores del atributo, el nombre del tipo o el nombre del objeto. • El atributo `model` del objeto no está establecido en "legacy".
`shadow.reported.myvalues:2`	Consulta los objetos cuya matriz `myvalues` de la sección reported de la sombra contiene el valor 2.
`shadow.reported.location:* NOT shadow.desired.stats.battery:*`	Consulta los objetos que cumplen lo siguiente: • El atributo `location` existe en la sección `reported` de la sombra. • El atributo `stats.battery` no existe en la sección `desired` de la sombra.
`shadow.name.<shadowName>.hasDelta:true`	Consulta los objetos que tienen una sombra con el nombre dado y también un elemento delta.

Cadena de consulta	Resultado
`shadow.name.<shadowName>.desired.filament:*`	Consulta los objetos que tienen una sombra con el nombre dado y también una propiedad de “filament” deseada.
`shadow.name.<shadowName>.reported.location:*`	Consulta los elementos que tienen una sombra con el nombre dado y donde existe el atributo `location` en la sección de informes de la sombra con nombre.
`connectivity.connected:true`	Consulta sobre todos los dispositivos conectados.
`connectivity.connected:false`	Consulta de todos los dispositivos desconectados.
`connectivity.connected:true AND connectivity.timestamp : [1557651600000 TO 1557867600000]`	Consulta de todos los dispositivos conectados con una marca temporal de conexión >= 1557651600000 y <= 1557867600000. Las marcas temporales se indican en milisegundos desde la fecha de inicio.
`connectivity.connected:false AND connectivity.timestamp : [1557651600000 TO 1557867600000]`	Consulta de todos los dispositivos desconectados con una marca temporal de desconexión >= 1557651600000 y <= 1557867600000. Las marcas temporales se indican en milisegundos desde la fecha de inicio.
`connectivity.connected:true AND connectivity.timestamp > 1557651600000`	Consulta de todos los dispositivos conectados con una marca temporal de conexión > 1557651600000. Las marcas temporales se indican en milisegundos desde la fecha de inicio.
`connectivity.connected:*`	Consulta todos los dispositivos para los que hay información de conectividad.
`connectivity.disconnectReason:*`	Consulta todos los dispositivos con conectividad disconnectReason presentes.

Cadena de consulta	Resultado
`connectivity.disconnectReason:CLIENT_INITIATED_DISCONNECT`	Consulta todos los dispositivos desconectados debido a CLIENT_INITIATED_DISCONNECT.
`deviceDefender.violationCount:[0 TO 100]`	Consultas los objetos con un número de infracciones de Device Defender dentro del rango numérico (0-100, ambos inclusive).
`deviceDefender.<device-SecurityProfile>.disconnectBehavior.inViolation:true`	Consulta los objetos que infringen el comportamiento `disconnectBehavior` definido en el perfil de seguridad `device-SecurityProfile`. Tenga en cuenta que inViolation:false no es una consulta válida.
`deviceDefender.<device-SecurityProfile>.disconnectBehavior.lastViolationValue.number>2`	Busca elementos que infrinjan el comportamiento `disconnectBehavior` definido en el dispositivo del perfil de seguridad, SecurityProfile con un valor de última infracción superior a 2.
`deviceDefender.<device-SecurityProfile>.disconnectBehavior.lastViolationTime>1634227200000`	Realiza consultas sobre elementos que infringen el comportamiento `disconnectBehavior` definido en el dispositivo del perfil de seguridad, SecurityProfile con un evento de última infracción transcurrido un período de tiempo especificado.
`shadow.name.gps-tracker.reported.coordinates:geo_distance,47.6204,-122.3491,15.5km`	Consulta objetos que se encuentran dentro de una distancia radial de 15,5 km desde las coordenadas 47.6204, -122.3491. Esta cadena de consulta se aplica cuando los datos de ubicación se almacenan en una sombra con nombre.
`shadow.reported.coordinates:geo_distance,47.6204,-122.3491,15.5km`	Consulta objetos que se encuentran dentro de una distancia radial de 15,5 km desde las coordenadas 47.6204, -122.3491. Esta cadena de consulta se aplica cuando los datos de ubicación se almacenan en una sombra clásica.

Ejemplo de consultas de grupo de objetos

Las consultas se especifican en una cadena de consulta mediante una sintaxis de consulta y se trasladan a la API de `SearchIndex`. En la siguiente tabla se enumeran algunas cadenas de consulta de ejemplo.

Cadena de consulta	Resultado
`abc`	Consulta "abc" en cualquier campo.
`thingGroupName:myGroupThingName`	Consultas para un grupo de cosas con el nombre "myGroupThingNombre».
`thingGroupName:my*`	Consulta los grupos de objetos cuyos nombres que comienzan por "my".
`thingGroupName:ab?`	Consulta los grupos de objetos cuyos nombres tienen “ab“ además de un carácter adicional (por ejemplo: “aba”, “abb”, “abc”, etc.)
`attributes.myAttribute:75`	Consulta los grupos de con un atributo llamado "MyAttribute" que tiene el valor de 75.
`attributes.myAttribute:[75 TO 80]`	Consulta los grupos de con un atributo llamado “MyAttribute”, cuyo valor se encuentra dentro de un rango numérico (75-80, ambos inclusive).
`attributes.myAttribute:[75 TO 80]`	Consulta los grupos de objetos con un atributo llamado "MyAttribute", cuyo valor se encuentra dentro del rango numérico (>75 y <=80).
`attributes.myAttribute:["abcd" TO "abcf"]`	Consulta los grupos de con un atributo llamado "myAttribute", cuyo valor se encuentra dentro del rango de cadenas alfanuméricas. Esta consulta devuelve grupos de objetos con un atributo "serialNumber" con valores "abcd", "abce" o "abcf".

Cadena de consulta	Resultado
`attributes.myAttribute:i*t`	Consulta los grupos de objetos con un atributo llamado "MyAttribute" cuyo valor es 'i', seguido de un número de caracteres, seguido por 't'.
`attributes.attr1:abc AND attributes.attr2<5 NOT attributes.attr3>10`	Consultas de grupos de objetos que combinan términos mediante expresiones booleanas. Esta consulta devuelve grupos de objetos que tengan un atributo denominado “attr1” con un valor “abc”, un atributo denominado “attr2” inferior a 5 y un atributo denominado “attr3” que no sea superior a 10.
`NOT attributes.myAttribute:cde`	Consulta los grupos de objetos donde el atributo llamado "MyAttribute" no es "cde".
`parentGroupNames:(myParentThingGroupName)`	Consultas para grupos de cosas cuyo nombre de grupo principal coincide con "myParentThingGroupName».
`parentGroupNames:(myParentThingGroupName OR myRootThingGroupName)`	Consultas para grupos de cosas cuyo nombre de grupo principal coincide con myParentThing GroupName "" o "myRootThingGroupName».
`parentGroupNames:(myParentThingGroupNa*)`	Consultas para grupos de cosas cuyo nombre de grupo principal comienza por "myParentThingGroupNa».

Indexación de datos de ubicación

Puede usar la indexación de flotas de AWS IoT para indexar los últimos datos de ubicación enviados por sus dispositivos y buscar dispositivos realizando consultas geográficas. Esta característica resuelve los casos de uso de la supervisión y la administración de dispositivos, como el seguimiento de la ubicación y la búsqueda de proximidad. La indexación de ubicaciones funciona de manera similar a otras características de indexación de flotas y con configuraciones adicionales que puede especificar en la indexación de objetos.

Los casos de uso más comunes son: buscar y añadir dispositivos ubicados dentro de los límites geográficos deseados, obtener información específica de la ubicación mediante términos de consulta relacionados con los metadatos y el estado del dispositivo a partir de orígenes de datos indexados,

proporcionar una vista granular, como filtrar los resultados según un área geográfica específica para reducir los retrasos de representación en los mapas de supervisión de la flota y rastrear la ubicación del último dispositivo notificado, e identificar los dispositivos que están fuera de los límites deseados y generar alarmas utilizando las métricas de flota. Para empezar a utilizar la indexación de ubicaciones y las geoconsultas, consulte ???.

Formatos de datos admitidos

AWS IoT La indexación de flotas admite los siguientes formatos de datos de ubicación:

1. Una representación de texto muy conocida de sistemas de coordenadas de referencia

   Es una cadena que sigue el formato Geographic information - Well-known text representation of coordinate reference systems. Por ejemplo, `"POINT(long lat)"`.
2. Una cadena que representa las coordenadas

   Es una cadena con el formato `"latitude, longitude"` o `"longitude, latitude"`. Si usa `"longitude, latitude"`, también debe especificar `order` en `geoLocations`. Por ejemplo, `"41.12,-71.34"`.
3. Un objeto con claves de lat (latitud) y lon (longitud)

   Este formato se aplica tanto a la sombra clásica como a la sombra con nombre. Claves admitidas: `lat`, `latitude`, `lon`, `long` y `longitude`. Por ejemplo, `{"lat": 41.12, "lon": -71.34}`.
4. Una matriz que representa las coordenadas

   Una matriz con el formato `[lat,lon]` o `[lon,lat]`. Si utiliza el formato `[lon,lat]`, que es el mismo que el de las coordenadas de GeoJSON (aplicable a la sombra clásica y a la sombra con nombre), también debe especificar `order` y `geoLocations`.

   Por ejemplo:

```
{
  "location": {
    "coordinates": [
      **Longitude**,
      **Latitude**
    ],
    "type": "Point",
    "properties": {
```

```
      "country": "United States",
      "city": "New York",
      "postalCode": "*****",
      "horizontalAccuracy": 20,
      "horizontalConfidenceLevel": 0.67,
      "state": "New York",
      "timestamp": "2023-01-04T20:59:13.024Z"
    }
  }
}
```

Cómo indexar los datos de ubicación

Los siguientes pasos muestran cómo actualizar la configuración de indexación de los datos de ubicación y cómo usar consultas geográficas para buscar dispositivos.

1. Saber dónde se almacenan los datos de ubicación

   Actualmente, la indexación de flotas permite indexar los datos de ubicación almacenados en sombras clásicas o sombras con nombre.

2. Utilizar los formatos de datos de ubicación compatibles

   Asegúrese de que el formato de los datos de ubicación siga uno de los formatos de datos compatibles.

3. Actualizar la configuración de indexación

   Como mínimo, active la configuración de indexación de objetos (registro). También debe activar la indexación en la sombra clásica o en la sombra con nombre que contenga sus datos de ubicación. Al actualizar la indexación de objetos, debe incluir los datos de ubicación en la configuración de indexación.

4. Crear y ejecutar geoconsultas

   Según los casos de uso, cree geoconsultas y ejecútelas para buscar dispositivos. La geoconsulta que componga debe seguir la sintaxis de consulta. Encontrará algunos ejemplos en ???.

Actualización de la configuración de indexación de objetos

Para indexar los datos de ubicación, debe actualizar la configuración de indexación e incluir los datos de ubicación. Según dónde estén almacenados los datos de ubicación, siga los pasos para actualizar la configuración de indexación:

Los datos de ubicación se almacenan en sombras clásicas

Si los datos de ubicación se almacenan en una sombra clásica, debe establecer el `thingIndexingMode` en `REGISTRY_AND_SHADOW` y especificar los datos de ubicación en los campos `geoLocations` (`name` y `order`) de `filter`.

En el siguiente ejemplo de configuración de indexación de objetos, se especifica la ruta de los datos de ubicación `shadow.reported.coordinates` como `name` y `LonLat` como `order`.

```
{
 "thingIndexingMode": "REGISTRY_AND_SHADOW",
 "filter": {
  "geoLocations": [
   {
    "name": "shadow.reported.coordinates",
    "order": "LonLat"
   }
  ]
 }
}
```

- `thingIndexingMode`

  El modo de indexación controla si el registro o la sombra están indexados. Cuando `thingIndexingMode` se establece en `OFF`, la indexación de objetos está desactivada.

  Para indexar los datos de ubicación almacenados en una sombra clásica, debe configurar el `thingIndexingMode` en `REGISTRY_AND_SHADOW`. Para obtener más información, consulte ???.

- `filter`

  El filtro de indexación proporciona selecciones adicionales para las sombras con nombre y los datos de geolocalización. Para obtener más información, consulte ???.

- `geoLocations`

Es la lista de objetivos de geolocalización que selecciona para indexar. El número máximo predeterminado de objetivos de geolocalización para la indexación es 1. Para solicitar un aumento de límite, consulte AWS IoT Device Management Quotas.

- `name`

  Es el nombre del campo objetivo de geolocalización. Un ejemplo de valor de `name` puede ser la ruta de los datos de ubicación de la sombra: `shadow.reported.coordinates`.

- `order`

  Es el orden del campo objetivo de geolocalización. Los valores válidos son `LatLon` y `LonLat`. `LatLon` significa latitud y longitud. `LonLat` significa longitud y latitud. Este campo es opcional. El valor predeterminado es `LatLon`.

Los datos de ubicación se almacenan en sombras con nombre

Si los datos de ubicación se almacenan en una sombra con nombre, defina el `namedShadowIndexingMode` en `ON`, añada los nombres de la sombra con nombre al campo `namedShadowNames` en `filter` y especifique la ruta de los datos de ubicación en el campo `geoLocations` en `filter`.

En el siguiente ejemplo de configuración de indexación de objetos, se especifica la ruta de los datos de ubicación `shadow.name.namedShadow1.reported.coordinates` como `name` y `LonLat` como `order`.

```
{
 "thingIndexingMode": "REGISTRY",
 "namedShadowIndexingMode": "ON",
 "filter": {
  "namedShadowNames": [
   "namedShadow1"
  ],
  "geoLocations": [
   {
    "name": "shadow.name.namedShadow1.reported.coordinates",
    "order": "LonLat"
   }
  ]
 }
}
```

- `thingIndexingMode`

  El modo de indexación controla si el registro o la sombra están indexados. Cuando `thingIndexingMode` se establece en `OFF`, la indexación de objetos está desactivada.

  Para indexar los datos de ubicación almacenados en una sombra con nombre, debe configurar el `thingIndexingMode` en `REGISTRY` (o `REGISTRY_AND_SHADOW`). Para obtener más información, consulte ???.

- `filter`

  El filtro de indexación proporciona selecciones adicionales para las sombras con nombre y los datos de geolocalización. Para obtener más información, consulte ???.

- `geoLocations`

  Es la lista de objetivos de geolocalización que selecciona para indexar. El número máximo predeterminado de objetivos de geolocalización para la indexación es 1. Para solicitar un aumento de límite, consulte AWS IoT Device Management Quotas.

- `name`

  Es el nombre del campo objetivo de geolocalización. Un ejemplo de valor de `name` puede ser la ruta de los datos de ubicación de la sombra: `shadow.name.namedShadow1.reported.coordinates`.

- `order`

  Es el orden del campo objetivo de geolocalización. Los valores válidos son `LatLon` y `LonLat`. `LatLon` significa latitud y longitud. `LonLat` significa longitud y latitud. Este campo es opcional. El valor predeterminado es `LatLon`.

Ejemplo de geoconsultas

Tras completar la configuración de indexación de los datos de ubicación, ejecute geoconsultas para buscar dispositivos. También puede combinar las geoconsultas con otras cadenas de consulta. Para obtener más información, consulte ??? y ???.

Ejemplo de consulta 1

En este ejemplo, se parte de la base de que los datos de ubicación se almacenan en una sombra con nombre `gps-tracker`. El resultado de este comando es la lista de dispositivos que se

encuentran a una distancia radial de 15,5 km del punto central de las coordenadas (47.6204, -122.3491).

```
aws iot search-index --query-string \
"shadow.name.gps-tracker.reported.coordinates:geo_distance,47.6204,-122.3491,15.5km"
```

Ejemplo de consulta 2

En este ejemplo, se parte de la base de que los datos de ubicación se almacenan en una sombra clásica. El resultado de este comando es la lista de dispositivos que se encuentran a una distancia radial de 15,5 km del punto central de las coordenadas (47.6204, -122.3491).

```
aws iot search-index --query-string \
"shadow.reported.coordinates:geo_distance,47.6204,-122.3491,15.5km"
```

Ejemplo de consulta 3

En este ejemplo, se parte de la base de que los datos de ubicación se almacenan en una sombra clásica. El resultado de este comando es la lista de dispositivos que no están conectados y se encuentran fuera de la distancia radial de 15,5 km del punto central de las coordenadas (47.6204, -122.3491).

```
aws iot search-index --query-string \
"connectivity.connected:false AND (NOT
 shadow.reported.coordinates:geo_distance,47.6204,-122.3491,15.5km)"
```

Tutorial introductorio

En este tutorial se explica cómo utilizar la indexación de flotas para indexar los datos de ubicación. Para simplificar, puede crear un objeto que represente su dispositivo y almacenar los datos de ubicación en una sombra con nombre, actualizar la configuración de indexación de objetos para la indexación de ubicaciones y ejecutar geoconsultas de ejemplo para buscar dispositivos dentro de un límite radial.

Para completar este tutorial se necesitan aproximadamente 15 minutos.

En este tema:

- Requisitos previos

- Creación de objetos y sombras
- Actualización de la configuración de indexación de objetos
- Ejecución de una geoconsulta

Requisitos previos

- Instale la versión más reciente de AWS CLI.
- Familiarícese con la indexación de ubicaciones y las geoconsultas, la administración de la indexación de objetos y la sintaxis de las consultas.

Creación de objetos y sombras

Cree un objeto para representar su dispositivo y una sombra con nombre para almacenar sus datos de ubicación (coordenadas 47.61564, -122.33584).

1. Ejecute el siguiente comando para crear un objeto que represente la bicicleta llamada Bike-1. Para obtener más información sobre cómo crear algo mediante el uso AWS CLI, consulte create-thing from Reference. AWS CLI

   ```
   aws iot create-thing --thing-name "Bike-1" \
   --attribute-payload '{"attributes": {"model":"OEM-2302-12", "battery":"35",
    "acqDate":"06/09/23"}}'
   ```

   El resultado de este comando puede tener un aspecto similar al siguiente.

   ```
   {
       "thingName": "Bike-1",
       "thingArn": "arn:aws:iot:us-east-1:123456789012:thing/Bike-1",
       "thingId": "df9cf01d-b0c8-48fe-a2e2-e16cff6b23df"
   }
   ```

2. Ejecute el siguiente comando para crear una sombra con nombre para almacenar los datos de ubicación de Bike-1 (coordenadas 47.61564, -122.33584). Para obtener más información sobre cómo crear una sombra con nombre utilizando AWS CLI, consulte update-thing-shadowFrom AWS CLIReference.

   ```
   aws iot-data update-thing-shadow \
   --thing-name Bike-1 \
   ```

```
--shadow-name Bike1-shadow \
--cli-binary-format raw-in-base64-out \
--payload '{"state":{"reported":{"coordinates":{"lat": 47.6153, "lon": -122.3333}}}}'
 \
"output.txt" \
```

Este comando no proporciona ninguna salida. Para ver la sombra con nombre que ha creado, puede ejecutar el comando CLI list-named-shadows-for-thing.

```
aws iot-data list-named-shadows-for-thing --thing-name Bike-1
```

El resultado de este comando puede tener un aspecto similar al siguiente.

```
{
    "results": [
        "Bike1-shadow"
    ],
    "timestamp": 1699574309
}
```

Actualización de la configuración de indexación de objetos

Para indexar los datos de ubicación, debe actualizar la configuración de indexación de objetos para incluir los datos de ubicación. Como en este tutorial los datos de ubicación se almacenan en una sombra con nombre, defina el `thingIndexingMode` en `REGISTRY` (con un requisito mínimo), defina el `namedShadowIndexingMode` en `ON` y añada los datos de ubicación a la configuración. En este ejemplo, debe añadir el nombre de la sombra con nombre y la ruta de los datos de ubicación de la sombra al `filter`.

1. Ejecute el comando para actualizar la configuración de indexación para indexar las ubicaciones.

```
aws iot update-indexing-configuration --cli-input-json '{
"thingIndexingConfiguration": { "thingIndexingMode": "REGISTRY",
"thingConnectivityIndexingMode": "OFF",
"deviceDefenderIndexingMode": "OFF",
"namedShadowIndexingMode": "ON",
"filter": {
    "namedShadowNames": ["Bike1-shadow"],
    "geoLocations":[{
```

```
        "name":"shadow.name.Bike1-shadow.reported.coordinates"
    }]
},
"customFields": [
{ "name":"attributes.battery",
"type":"Number"}] } }'
```

El comando no genera ningún resultado. Puede que tenga que esperar un momento hasta que se complete la actualización. Para comprobar el estado, ejecute el comando de la CLI describe-index. Si ve que `indexStatus` muestra `ACTIVE`, significa que se ha completado la actualización de la indexación de objetos.

2. Ejecute el comando para comprobar la configuración de la indexación. Este paso es opcional.

```
aws iot get-indexing-configuration
```

El resultado puede ser similar al siguiente:

```
{
    "thingIndexingConfiguration": {
        "thingIndexingMode": "REGISTRY",
        "thingConnectivityIndexingMode": "OFF",
        "deviceDefenderIndexingMode": "OFF",
        "namedShadowIndexingMode": "ON",
        "managedFields": [
            {
                "name": "shadow.name.*.hasDelta",
                "type": "Boolean"
            },
            {
                "name": "registry.version",
                "type": "Number"
            },
            {
                "name": "registry.thingTypeName",
                "type": "String"
            },
            {
                "name": "registry.thingGroupNames",
                "type": "String"
            },
            {
```

```
                "name": "shadow.name.*.version",
                "type": "Number"
            },
            {
                "name": "thingName",
                "type": "String"
            },
            {
                "name": "thingId",
                "type": "String"
            }
        ],
        "customFields": [
            {
                "name": "attributes.battery",
                "type": "Number"
            }
        ],
        "filter": {
            "namedShadowNames": [
                "Bike1-shadow"
            ],
            "geoLocations": [
                {
                    "name": "shadow.name.Bike1-shadow.reported.coordinates",
                    "order": "LatLon"
                }
            ]
        }
    },
    "thingGroupIndexingConfiguration": {
        "thingGroupIndexingMode": "OFF"
    }
}
```

Ejecución de una geoconsulta

Ahora ha actualizado su configuración de indexación de objetos para incluir los datos de ubicación. Intente crear algunas geoconsultas y ejecútelas para ver si puede obtener los resultados de búsqueda deseados. Una geoconsulta debe seguir la sintaxis de consulta. Puede consultar algunos ejemplos útiles de geoconsultas en ???.

En el siguiente comando de ejemplo, se utiliza la geoconsulta `shadow.name.Bike1-shadow.reported.coordinates:geo_distance,47.6204,-122.3491,15.5km` para buscar dispositivos que se encuentren dentro de una distancia radial de 15,5 km desde el punto central de las coordenadas (47.6204, -122.3491).

```
aws iot search-index --query-string "shadow.name.Bike1-
shadow.reported.coordinates:geo_distance,47.6204,-122.3491,15.5km"
```

Como tiene un dispositivo ubicado en las coordenadas lat: 47.6153, lon: -122.3333, que se encuentra a una distancia de 15,5 km del punto central, debería ver este dispositivo (Bike-1) en el resultado. El resultado puede ser similar al siguiente:

```
{
    "things": [
        {
            "thingName": "Bike-1",
            "thingId": "df9cf01d-b0c8-48fe-a2e2-e16cff6b23df",
            "attributes": {
                "acqDate": "06/09/23",
                "battery": "35",
                "model": "OEM-2302-12"
            },
            "shadow": "{\"reported\":{\"coordinates\":{\"lat\":47.6153,\"lon
\":-122.3333}},\"metadata\":{\"reported\":{\"coordinates\":{\"lat\":{\"timestamp
\":1699572906},\"lon\":{\"timestamp\":1699572906}}}},\"hasDelta\":false,\"version\":1}"
        }
    ]
}
```

Para obtener más información, consulte ???.

Métricas de flota

Las métricas de flota son una función de la indexación de flotas, un servicio gestionado que te permite indexar, buscar y agregar los datos de tus dispositivos. AWS IoT Puedes usar las métricas de la flota para monitorear el estado agregado de los dispositivos de tu flota a CloudWatchlo largo del tiempo, lo que incluye revisar la tasa de desconexión de los dispositivos de tu flota o los cambios promedio en el nivel de batería durante un período específico.

Con las métricas de la flota, puedes crear consultas de agregación cuyos resultados se emitan continuamente CloudWatchcomo métricas para analizar tendencias y crear alarmas. Para sus tareas de monitorización, puede especificar las consultas de agregación de diferentes tipos de agregación (estadísticas, cardinalidad y percentil). Puede guardar todas sus consultas de agregación para crear métricas de flota para reutilizarlas en el futuro.

Tutorial introductorio

En este tutorial, creará una métrica de flota para monitorizar las temperaturas de sus sensores y detectar posibles anomalías. Al crear la métrica de flota, defina una consulta de agregación que detecte el número de sensores con temperaturas superiores a 80 grados Fahrenheit. Usted especifica que la consulta se ejecute cada 60 segundos y los resultados de la consulta se CloudWatch emitirán allí, donde podrá ver la cantidad de sensores que presentan posibles riesgos de alta temperatura y configurar las alarmas. Para completar este tutorial utilizará la AWS CLI.

En este tutorial, aprenderá a:

- Configurar.
- Crear métricas de flota
- Vea las métricas en CloudWatch
- Limpiar recursos

Para completar este tutorial se necesitan aproximadamente 15 minutos.

Requisitos previos

- Instale la versión más reciente de AWS CLI
- Familiarícese con las consultas de datos agregados
- Familiarízate con el uso de CloudWatch las métricas de Amazon

Configuración

Para utilizar las métricas de flota, active la indexación de flotas. Para habilitar la indexación de flotas de sus objetos o grupos de objetos con orígenes de datos específicos y configuraciones asociadas, siga las instrucciones de Administrar la indexación de objetos y Administrar la indexación de grupos de objetos.

Para configurar

1. Ejecute el siguiente comando para habilitar la indexación de flotas y especifique los orígenes de datos en los que realizar la búsqueda.

```
aws iot update-indexing-configuration \
--thing-indexing-configuration
 "thingIndexingMode=REGISTRY_AND_SHADOW,customFields=[{name=attributes.temperature,type=Num
{name=attributes.rackId,type=String},
{name=attributes.stateNormal,type=Boolean}],thingConnectivityIndexingMode=STATUS" \
```

   El comando de la CLI del ejemplo anterior permite que la indexación de flotas admita la búsqueda de datos de registro, datos de sombras y el estado de conectividad de las objetos mediante el índice `AWS_Things`.

   El cambio de la configuración puede tardar algunos minutos en completarse. Compruebe que la indexación de flotas esté habilitada antes de crear las métricas de flota.

   Para comprobar si la indexación de flotas está habilitada, ejecute el siguiente comando de la CLI:

```
aws --region us-east-1 iot describe-index --index-name "AWS_Things"
```

   Para obtener más información, consulte Habilitar la indexación de objetos.

2. Ejecute el siguiente script bash para crear diez objetos y describirlos.

```
# Bash script. Type `bash` before running in other shells.

Temperatures=(70 71 72 73 74 75 47 97 98 99)
Racks=(Rack1 Rack1 Rack2 Rack2 Rack3 Rack4 Rack5 Rack6 Rack6 Rack6)
IsNormal=(true true true true true true false false false false)

for ((i=0; i < 10; i++))
do
  thing=$(aws iot create-thing --thing-name "TempSensor$i" --attribute-
payload attributes="{temperature=${Temperatures[@]:$i:1},rackId=${Racks[@]:
$i:1},stateNormal=${IsNormal[@]:$i:1}}")
  aws iot describe-thing --thing-name "TempSensor$i"
done
```

Este script crea diez objetos para representar diez sensores. Cada objeto tiene los atributos de `temperature`, `rackId`, y `stateNormal`, tal como se describe en la siguiente tabla:

Atributo	Tipo de datos:	Description (Descripción)
`temperature`	Número	Valor de temperatura en grados Fahrenheit
`rackId`	Cadena	ID del rack de servidores que contiene los sensores
`stateNormal`	Booleano	Si el valor de temperatura del sensor es normal o no

La salida de este script contiene diez archivos JSON. Uno de los archivos JSON tiene el siguiente aspecto:

```
{
    "version": 1,
    "thingName": "TempSensor0",
    "defaultClientId": "TempSensor0",
    "attributes": {
        "rackId": "Rack1",
        "stateNormal": "true",
        "temperature": "70"
    },
    "thingArn": "arn:aws:iot:region:account:thing/TempSensor0",
    "thingId": "example-thing-id"
}
```

Para obtener más información, consulte la sección Crear un objeto.

Crear métricas de flota

Para crear una métrica de flota

1. Ejecute el siguiente comando para crear una métrica de flota llamada*high_temp_FM*. Debe crear la métrica de flota para monitorear la cantidad de sensores con temperaturas superiores a 80 grados Fahrenheit in. CloudWatch

```
aws iot create-fleet-metric --metric-name "high_temp_FM" --query-string
 "thingName:TempSensor* AND attributes.temperature >80" --period 60 --aggregation-
field "attributes.temperature" --aggregation-type name=Statistics,values=count
```

--metric-name

Tipo de datos: cadena. El parámetro `--metric-name` especifica el nombre de una métrica de flota. En este ejemplo, va a crear una métrica de flota llamada high_temp_FM.

--query-string

Tipo de datos: cadena. El parámetro `--query-string` especifica la cadena de consulta. En este ejemplo, la cadena de consulta significa consultar todos los elementos cuyos nombres comiencen por TempSensory con temperaturas superiores a 80 grados Fahrenheit. Para obtener más información, consulte Sintaxis de consultas.

--period

Tipo de datos: entero. El parámetro `--period` especifica el tiempo necesario para recuperar los datos agregados en segundos. En este ejemplo, se especifica que la métrica de flota que está creando recupera los datos agregados cada 60 segundos.

--aggregation-field

Tipo de datos: cadena. El parámetro `--aggregation-field` especifica el atributo que se va a evaluar. En este ejemplo, se va a evaluar el atributo de temperatura.

--aggregation-type

El parámetro `--aggregation-type` especifica el resumen estadístico que se mostrará en la métrica de la flota. Para sus tareas de monitorización, puede personalizar las propiedades de consulta de agregación para los diferentes tipos de agregación (estadísticas, cardinalidad y

percentil). En este ejemplo, se especifica el recuento para el tipo de agregación y Statistics para que devuelva el recuento de dispositivos que tienen atributos que coinciden con la consulta, es decir, para devolver el recuento de los dispositivos cuyos nombres comiencen por TempSensory con temperaturas superiores a 80 grados Fahrenheit. Para obtener mas información, consulte Consulta de datos agregados.

El resultado de este comando tendrá un aspecto similar al siguiente.

```
{
    "metricArn": "arn:aws:iot:region:111122223333:fleetmetric/high_temp_FM",
    "metricName": "high_temp_FM"
}
```

> **Note**
>
> Los puntos de datos pueden tardar un momento en mostrarse. CloudWatch

Para obtener más información sobre cómo crear una métrica de flota, consulte Administración de métricas de flota.

Si no puede crear una métrica de flota, vaya a Solución de problemas de métricas de flota.

2. (Opcional) Ejecute el siguiente comando para describir su métrica de flota llamada high_temp_FM:

```
aws iot describe-fleet-metric --metric-name "high_temp_FM"
```

El resultado de este comando tendrá un aspecto similar al siguiente.

```
{
    "queryVersion": "2017-09-30",
    "lastModifiedDate": 1625249775.834,
    "queryString": "*",
    "period": 60,
    "metricArn": "arn:aws:iot:region:111122223333:fleetmetric/high_temp_FM",
    "aggregationField": "registry.version",
    "version": 1,
    "aggregationType": {
        "values": [
```

```
            "count"
        ],
        "name": "Statistics"
    },
    "indexName": "AWS_Things",
    "creationDate": 1625249775.834,
    "metricName": "high_temp_FM"
}
```

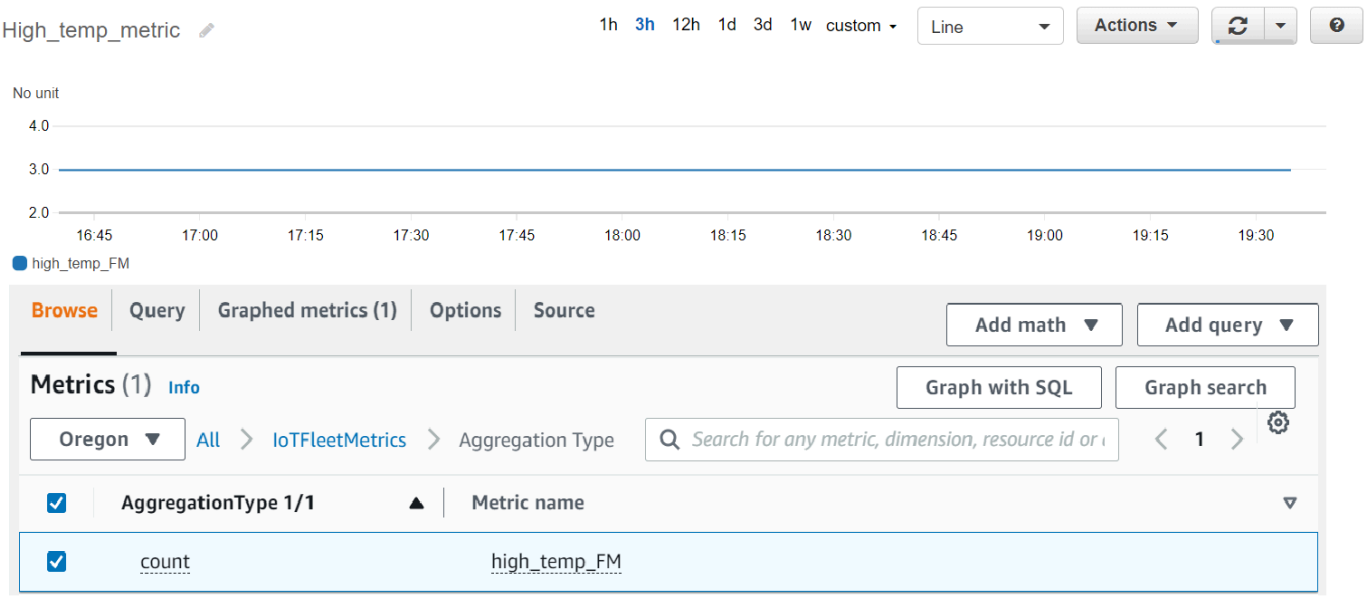
Vea las métricas de la flota en CloudWatch

Tras crear la métrica de flota, puede ver los datos de la métrica en CloudWatch. En este tutorial, verá la métrica que muestra el número de sensores cuyos nombres comienzan por TempSensory con temperaturas superiores a 80 grados Fahrenheit.

Para ver los puntos de datos en CloudWatch

1. Abra la CloudWatch consola en https://console.aws.amazon.com/cloudwatch/.
2. En el CloudWatch menú del panel izquierdo, selecciona Métricas para expandir el submenú y, a continuación, selecciona Todas las métricas. Esto abre la página con la mitad superior para mostrar el gráfico y la mitad inferior contiene cuatro secciones con pestañas.
3. La primera sección con pestañas Todas las métricas muestra todas las métricas que puedes ver en grupos. Selecciona lo TFleet Metrics. Contiene todas las métricas de su flota.
4. En la sección Tipo de agregación de la pestaña Todas las métricas, seleccione Tipo de agregación para ver todas las métricas de flota que ha creado.
5. Elija la métrica de flota para mostrar el gráfico a la izquierda de la sección Tipos de agregación. Verás el valor *`count`* a la izquierda del nombre de tu métrica, y este es el valor del tipo de agregación que especificaste en la sección Crear métricas de flota de este tutorial.
6. Seleccione la segunda pestaña, denominada Métricas graficadas, a la derecha de la pestaña Todas las métricas para ver la métrica de flota que eligió en el paso anterior.

   Debería poder ver un gráfico que muestre el número de sensores con temperaturas superiores a 80 grados Fahrenheit, como el siguiente:



> **Note**
>
> El atributo Periodo CloudWatch por defecto es de 5 minutos. Es el intervalo de tiempo entre los puntos de datos que aparecen CloudWatch. Puede cambiar la configuración de Period en función de sus necesidades.

7. (Opcional) Puede configurar una alarma métrica.

   1. En el CloudWatch menú del panel izquierdo, selecciona Alarmas para expandir el submenú y, a continuación, selecciona Todas las alarmas.
   2. En la página Alarmas, seleccione Crear alarma en la esquina superior derecha. Sigue las instrucciones Crear alarma en la consola para crear una alarma según sea necesario. Para obtener más información, consulta Cómo usar CloudWatch las alarmas de Amazon.

Para obtener más información, consulta Cómo usar CloudWatch las métricas de Amazon.

Si no puedes ver los puntos de datos CloudWatch, lee Solución de problemas con las métricas de flota.

Limpieza

Para eliminar métricas de flota

El comando de la CLI delete-fleet-metric se utiliza para eliminar las métricas de flota.

Para eliminar la métrica de flota denominada high_temp_FM, ejecute el comando siguiente.

```
aws iot delete-fleet-metric --metric-name "high_temp_FM"
```

Para limpiar objetos

Puede utilizar el comando de la CLI delete-thing para eliminar objetos.

Para eliminar los diez objetos que ha creado, ejecute el siguiente script:

```
# Bash script. Type `bash` before running in other shells.

for ((i=0; i < 10; i++))
do
  thing=$(aws iot delete-thing --thing-name "TempSensor$i")
done
```

Para limpiar las métricas en CloudWatch

CloudWatch no admite la eliminación de métricas. Las métricas caducan en función de sus programas de retención. Para obtener más información, consulta Cómo usar CloudWatch las métricas de Amazon.

Administración de métricas de flota

En este tema se muestra cómo utilizar la AWS IoT consola y cómo AWS CLI gestionar las métricas de tu flota.

Temas

- Administración de métricas de flota (consola)
- Administración de métricas de flota (CLI)
- Autorización del etiquetado de los recursos de IoT

Administración de métricas de flota (consola)

En las siguientes secciones, se muestra cómo utilizar la AWS IoT consola para gestionar las métricas de la flota. Asegúrese de activar la indexación de flotas con los orígenes de datos y las configuraciones asociadas antes de crear las métricas de flota.

Habilitar la indexación de flotas

Si ya ha habilitado la indexación de flotas, puede omitir esta sección.

Si no ha habilitado la indexación de flotas, siga estas instrucciones.

1. Abre la AWS IoT consola en https://console.aws.amazon.com/iot/.
2. En el AWS IoT menú, selecciona Configuración.
3. Para ver la configuración detallada, en la página Configuración desplácese hacia abajo hasta la sección Indexación de flotas.
4. Para actualizar la configuración de indexación de flotas, a la derecha de la sección Indexación de flotas, seleccione Administrar indexación.
5. En la página Administrar indexación de flotas, actualice la configuración de indexación de flotas en función de sus necesidades.
   - Configuración

     Para activar la indexación de objetos, active Indexación de objetos y, a continuación, seleccione los orígenes de datos desde los que desee indexar.

     Para activar la indexación de grupos de objetos, active Indexación de grupo de objetos.
   - Campos personalizados para la agregación: opcional

     Los campos personalizados son una lista de pares de nombres y tipos de campo.

     Para agregar un par de campos personalizados, seleccione Agregar un campo nuevo. Introduzca un nombre de campo personalizado, por ejemplo `attributes.temperature`, y, a continuación, seleccione un tipo de campo en el menú Tipo de campo. Tenga en cuenta que el nombre de un campo personalizado comienza por `attributes.` y se guardará como un atributo para ejecutar consultas de agregación de objetos.

     Para actualizar y guardar la configuración, seleccione Actualizar.

Crear una métrica de flota

1. Abre la AWS IoT consola en https://console.aws.amazon.com/iot/.
2. En el AWS IoT menú, selecciona Administrar y, a continuación, selecciona Estadísticas de flota.
3. En la página Métricas de flota, seleccione Crear métrica de flota y complete los pasos de creación.

4. En el paso 1, Configurar las métricas de flota
   - En la sección Consulta, introduzca una cadena de consulta para especificar los objetos o grupos de objetos que quiere que realicen la búsqueda agregada. La cadena de consulta consta de un atributo y un valor. En Propiedades, elija el atributo que desee o, si no aparece en la lista, introdúzcalo en el campo. Especifique el valor después de `:`. Un ejemplo de cadena de consulta puede ser `thingName:TempSensor*`. Para cada cadena de consulta que introduzca, pulse Intro en el teclado. Si introduce varias cadenas de consulta, especifique su relación seleccionando los operadores and, or, and not u or not.
   - En Propiedades del informe, elija las opciones Nombre del índice, Tipo de agregación y Campo de agregación en sus respectivas listas. A continuación, seleccione los datos que desee agregar en Seleccionar datos, donde puede seleccionar varios valores de datos.
   - Elija Siguiente.
5. En el paso 2, Especificar las propiedades de las métricas de flota
   - En el campo Nombre de la métrica de flota, introduzca un nombre para la métrica de flota que está creando.
   - En el campo Descripción: opcional, introduzca una descripción para la métrica de flota que está creando. Este campo es opcional.
   - En los campos Horas y Minutos, introduce la hora (con qué frecuencia) quieres que la métrica de flota emita datos CloudWatch.
   - Elija Siguiente.
6. En el paso 3, Revisar y crear
   - Revise la configuración de los pasos 1 y 2. Para editar esta configuración, elija Editar.
   - Seleccione Crear métrica de flota.

Una vez creada correctamente, la métrica de flota aparece en la página Métrica de flota.

Actualizar una métrica de flota

1. En la página Métricas de flota, seleccione la métrica de flota que quiera actualizar.
2. En la página Detalles de la métrica de flota, seleccione Editar. Se abren los pasos de creación, en los que puede actualizar la métrica de su flota en cualquiera de los tres pasos.
3. Cuando termine de actualizar la métrica de flota, seleccione Actualizar métrica de flota.

Eliminar una métrica de flota

1. En la página Métricas de flota, seleccione la métrica de flota que quiera eliminar.
2. En la página siguiente que muestra los detalles de la métrica de su flota, seleccione Eliminar.
3. En el cuadro de diálogo, escriba el nombre de la métrica de flota para confirmar la eliminación.
4. Elija Eliminar. Este paso elimina la métrica de flota de forma permanente.

Administración de métricas de flota (CLI)

En las siguientes secciones, se muestra cómo utilizarlos AWS CLI para gestionar las métricas de la flota. Asegúrese de activar la indexación de flotas con los orígenes de datos y las configuraciones asociadas antes de crear las métricas de flota. Para habilitar la indexación de flotas de sus objetos o grupos de objetos, siga las instrucciones de Administrar la indexación de objetos o Administrar la indexación de grupos de objetos.

Crear una métrica de flota

Puede usar el comando create-fleet-metric CLI para crear una métrica de flota.

```
aws iot create-fleet-metric --metric-name "YourFleetMetricName" --query-string
 "*" --period 60 --aggregation-field "registry.version" --aggregation-type
 name=Statistics,values=sum
```

La salida de este comando contiene el nombre y el nombre de recurso de Amazon (ARN) de su métrica de flota. El resultado es similar al siguiente:

```
{
    "metricArn": "arn:aws:iot:us-east-1:111122223333:fleetmetric/YourFleetMetricName",
    "metricName": "YourFleetMetricName"
}
```

Enumerar métricas de flota

Puede usar el comando list-fleet-metric CLI para enumerar todas las métricas de la flota de su cuenta.

```
aws iot list-fleet-metrics
```

La salida de este comando contiene todas las métricas de flota. El resultado es similar al siguiente:

```
{
    "fleetMetrics": [
        {
            "metricArn": "arn:aws:iot:us-east-1:111122223333:fleetmetric/
YourFleetMetric1",
            "metricName": "YourFleetMetric1"
        },
        {
            "metricArn": "arn:aws:iot:us-east-1:111122223333:fleetmetric/
YourFleetMetric2",
            "metricName": "YourFleetMetric2"
        }
    ]
}
```

Describir una métrica de flota

Puede usar el comando describe-fleet-metric CLI para mostrar información más detallada sobre una métrica de flota.

```
aws iot describe-fleet-metric --metric-name "YourFleetMetricName"
```

La salida del comando contiene información acerca de la métrica de flota especificada. El resultado es similar al siguiente:

```
{
    "queryVersion": "2017-09-30",
    "lastModifiedDate": 1625790642.355,
    "queryString": "*",
    "period": 60,
    "metricArn": "arn:aws:iot:us-east-1:111122223333:fleetmetric/YourFleetMetricName",
    "aggregationField": "registry.version",
    "version": 1,
    "aggregationType": {
        "values": [
            "sum"
        ],
        "name": "Statistics"
    },
    "indexName": "AWS_Things",
    "creationDate": 1625790642.355,
    "metricName": "YourFleetMetricName"
```

```
}
```

Actualizar una métrica de flota

Puede usar el comando update-fleet-metric CLI para actualizar una métrica de flota.

```
aws iot update-fleet-metric --metric-name "YourFleetMetricName" --query-string
 "*" --period 120 --aggregation-field "registry.version" --aggregation-type
 name=Statistics,values=sum,count --index-name AWS_Things
```

El comando update-fleet-metric no genera ninguna salida. Puede usar el comando describe-fleet-metric CLI para ver el resultado.

```
{
    "queryVersion": "2017-09-30",
    "lastModifiedDate": 1625792300.881,
    "queryString": "*",
    "period": 120,
    "metricArn": "arn:aws:iot:us-east-1:111122223333:fleetmetric/YourFleetMetricName",
    "aggregationField": "registry.version",
    "version": 2,
    "aggregationType": {
        "values": [
            "sum",
            "count"
        ],
        "name": "Statistics"
    },
    "indexName": "AWS_Things",
    "creationDate": 1625792300.881,
    "metricName": "YourFleetMetricName"
}
```

Eliminar una métrica de flota

Utilice el comando delete-fleet-metric CLI para eliminar una métrica de flota.

```
aws iot delete-fleet-metric --metric-name "YourFleetMetricName"
```

Este comando no produce ningún resultado si la eliminación se realiza correctamente o si se especifica una métrica de flota que no existe.

Para obtener más información, consulte la Solución de problemas de métricas de flota.

Autorización del etiquetado de los recursos de IoT

Para controlar mejor las métricas de flota que puede crear, modificar o usar, puede asociarles etiquetas.

Para etiquetar las métricas de flota que cree utilizando Consola de administración de AWS o AWS CLI, debe incluir la `iot:TagResource` acción en su política de IAM para conceder permisos al usuario. Si la política de IAM no incluye la acción `iot:TagResource`, cualquier acción para crear una métrica de flota con una etiqueta devolverá un error `AccessDeniedException`.

Para obtener más información general sobre los recursos, consulte Tagging your AWS IoT resources.

Ejemplo de política de IAM

Consulte el siguiente ejemplo de política de IAM que concede permisos de etiquetado al crear una métrica de flota:

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Action": [
    "iot:TagResource"
   ],
   "Effect": "Allow",
   "Resource": [
    "arn:aws:iot:*:*:fleetmetric/*"
   ]
  },
  {
   "Action": [
    "iot:CreateFleetMetric"
   ],
   "Effect": "Allow",
   "Resource": [
    "arn:aws:iot:*:*:index/*",
    "arn:aws:iot:*:*:fleetmetric/*"
   ]
  }
```

```
  ]
}
```

Para obtener información, consulte Acciones, recursos y claves de condición para AWS IoT.

Entrega de archivos basada en MQTT

Una opción que puede utilizar para gestionar los archivos y transferirlos a los AWS IoT dispositivos de su flota es la entrega de archivos mediante MQTT. Con esta función de la AWS nube, puede crear una transmisión que contenga varios archivos, actualizar los datos de la transmisión (la lista de archivos y las descripciones), obtener los datos de la transmisión y mucho más. AWS IoT La entrega de archivos basada en MQTT puede transferir datos en pequeños bloques a sus dispositivos de IoT, utilizando el protocolo MQTT con soporte para mensajes de solicitud y respuesta en JSON o CBOR.

Para obtener más información sobre las formas de transferir datos hacia y desde dispositivos de IoT mediante AWS IoT, consulteConnect dispositivos a AWS IoT.

Temas

- ¿Qué es una transmisión?
- Administrar una transmisión en la nube AWS
- Uso de la entrega de archivos AWS IoT basada en MQTT en los dispositivos
- Un ejemplo de caso de uso en Freertos OTA

¿Qué es una transmisión?

En AWS IoT, una transmisión es un recurso de acceso público que es una abstracción de una lista de archivos que se pueden transferir a un dispositivo de IoT. Una transmisión típica contiene la siguiente información:

- Un nombre de recurso de Amazon (ARN) que identifica de forma exclusiva una transmisión en un momento dado. Este ARN tiene el patrón `arn:`*`partition`*`:iot:`*`region`*`:`*`account-ID`*`:stream/`*`stream ID`*.
- Un ID de transmisión que identifica tu transmisión y que se usa (y suele ser obligatorio) en los comandos AWS Command Line Interface (AWS CLI) o del SDK.
- Una descripción de la transmisión que proporciona una descripción del recurso de la transmisión.
- Una versión de la transmisión que identifica una versión concreta de la transmisión. Como los datos de transmisión se pueden modificar inmediatamente antes de que los dispositivos inicien la transferencia de datos, los dispositivos pueden utilizar la versión de transmisión para aplicar una comprobación de coherencia.

- Una lista de archivos que se pueden transferir a los dispositivos. Para cada archivo de la lista, la transmisión registra un ID de archivo, el tamaño del archivo y la información de dirección del archivo, que consta, por ejemplo, del nombre del bucket de Amazon S3, la clave de la cosa y la versión de la cosa.
- Una función AWS Identity and Access Management (IAM) que otorga a la entrega de archivos AWS IoT basada en MQTT el permiso para leer los archivos de transmisión almacenados en el almacenamiento de datos.

AWS IoT La entrega de archivos basada en MQTT proporciona las siguientes funciones para que los dispositivos puedan transferir datos desde la nube: AWS

- Transferencia de datos mediante el protocolo MQTT.
- Soporte para los formatos JSON o CBOR.
- La capacidad de describir una transmisión (API `DescribeStream`) para obtener una lista de archivos de transmisión, una versión de la transmisión e información relacionada.
- La capacidad de enviar datos en bloques pequeños (API `GetStream`) para que los dispositivos con limitaciones de hardware puedan recibir los bloques.
- Soporte para un tamaño de bloque dinámico por solicitud, para admitir dispositivos que tienen diferentes capacidades de memoria.
- Optimización para solicitudes de transmisión simultáneas cuando varios dispositivos solicitan bloques de datos del mismo archivo de transmisión.
- Amazon S3 como almacenamiento de datos para archivos de transmisión.
- Support para la publicación de registros de transferencia de datos desde la entrega de archivos AWS IoT basados en MQTT a. CloudWatch

Para conocer las cuotas de entrega de archivos basada en MQTT, consulte Service Quotas AWS IoT Core en Referencia general de AWS.

Administrar una transmisión en la nube AWS

AWS IoT proporciona el AWS SDK y AWS CLI los comandos que puede usar para administrar una transmisión en la AWS nube. Puede utilizar estas métricas para hacer lo siguiente:

- Crear una transmisión. CLI / SDK
- Describir una transmisión para obtener su información. CLI / SDK

- Enumere las transmisiones en su Cuenta de AWS. CLI / SDK
- Actualizar la lista de archivos o la descripción de la transmisión en una transmisión. CLI / SDK
- Eliminar una transmisión. CLI / SDK

> **Note**
>
> En este momento, las transmisiones no están visibles en la Consola de administración de AWS. Debe usar el AWS SDK AWS CLI o el SDK para administrar una transmisión AWS IoT. Además, el SDK para Embedded C es el único SDK que admite transferencias de archivos basadas en MQTT.

Antes de utilizar la entrega de archivos AWS IoT basada en MQTT desde sus dispositivos, debe asegurarse de que se cumplen las siguientes condiciones en sus dispositivos, como se muestra en las siguientes secciones:

- Una política que refleje los permisos correctos necesarios para transmitir datos a través de MQTT.
- El dispositivo se puede conectar a la puerta de enlace de AWS IoT dispositivos.
- Una declaración de política que indique que puede etiquetar los recursos. Si se llama a `CreateStream` con etiquetas, entonces `iot:TagResource` es obligatorio.

Antes de utilizar la entrega de archivos AWS IoT basada en MQTT desde sus dispositivos, debe seguir los pasos de las siguientes secciones para asegurarse de que los dispositivos están debidamente autorizados y se pueden conectar a AWS IoT Device Gateway.

Conceder permisos a sus dispositivos

Puede seguir los pasos que se indican en Crear una política AWS IoT para crear una política de dispositivos o usar una política de dispositivos existente. Adjunte la política a los certificados asociados a sus dispositivos y añada los siguientes permisos a la política del dispositivo.

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

```
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
            "arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive",
                "iot:Publish"
            ],
            "Resource": [
            "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/
${iot:Connection.Thing.ThingName}/streams/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "iot:Subscribe",
            "Resource": [
            "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/things/
${iot:Connection.Thing.ThingName}/streams/*"
            ]
        }
    ]
}
```

Conecta tus dispositivos a AWS IoT

Es necesario conectar los dispositivos que utilizan la entrega de archivos AWS IoT basada en MQTT. AWS IoT AWS IoT La entrega de archivos basada AWS IoT en MQTT se integra con la AWS nube, por lo que sus dispositivos deberían conectarse directamente al punto final del plano de AWS IoT datos.

> **Note**
>
> El punto final del plano de AWS IoT datos es específico de la región Cuenta de AWS y. Debe usar el punto final para la región en la que están registrados sus dispositivos Cuenta de AWS y para la región en la que están registrados AWS IoT.

Para obtener más información, consulte Connect to AWS IoT Core.

TagResource Uso

La acción de la API `CreateStream` crea una transmisión para entregar uno o varios archivos grandes en trozos a través de MQTT.

Una llamada a la API `CreateStream` correcta requiere los siguientes permisos:

- `iot:CreateStream`
- `iot:TagResource`(si `CreateStream` es con etiquetas)

La política que respalda esos dos permisos se muestra a continuación:

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Action": [
            "iot:CreateStream",
            "iot:TagResource"
        ],
        "Effect": "Allow",
        "Resource": "arn:aws:iot:us-east-1:123456789012:stream/streamId"
    }
}
```

La acción de declaración de política `iot:TagResource` es necesaria para garantizar que un usuario no pueda crear o actualizar una etiqueta en un recurso sin los permisos adecuados. Sin la acción de declaración de política específica de `iot:TagResource`, la llamada a la API `CreateStream` devolverá un `AccessDeniedException` si la solicitud viene con etiquetas.

Para obtener más información, consulte los siguientes enlaces:

- CreateStream
- TagResource
- Etiqueta

Uso de la entrega de archivos AWS IoT basada en MQTT en los dispositivos

Para iniciar el proceso de transferencia de datos, el dispositivo debe recibir un conjunto de datos inicial, que incluya como mínimo un ID de transmisión. Puede utilizar un AWS IoT Empleos para programar tareas de transferencia de datos para sus dispositivos incluyendo el conjunto de datos inicial en el documento de trabajo. Cuando un dispositivo recibe el conjunto de datos inicial, debería iniciar la interacción con la entrega de archivos basada en AWS IoT MQTT. Para intercambiar datos con la entrega de archivos AWS IoT basada en MQTT, el dispositivo debe:

- Utilizar el protocolo MQTT para suscribirse al Temas relacionados con la entrega de archivos basada en MQTT.
- Enviar las solicitudes y esperar a recibir las respuestas mediante mensajes MQTT.

Si lo desea, puede incluir un ID de archivo de transmisión y una versión de la transmisión en el conjunto de datos inicial. El envío de un ID de archivo de transmisión a un dispositivo puede simplificar la programación del firmware/software del dispositivo, ya que elimina la necesidad de realizar una solicitud `DescribeStream` desde el dispositivo para obtener este ID. El dispositivo puede especificar la versión de la transmisión en una solicitud `GetStream` para aplicar una comprobación de coherencia en caso de que la transmisión se actualice de forma inesperada.

Se usa DescribeStream para obtener datos de transmisión

AWS IoT La entrega de archivos basada en MQTT proporciona la `DescribeStream` API para enviar datos de transmisión a un dispositivo. Los datos de transmisión devueltos por esta API incluyen el ID de la transmisión, la versión de la transmisión, la descripción de la transmisión y una lista de archivos de transmisión, cada uno de los cuales tiene un ID de archivo y el tamaño del archivo en bytes. Con esta información, un dispositivo puede seleccionar archivos arbitrarios para iniciar el proceso de transferencia de datos.

> **Note**
>
> No necesitas usar la `DescribeStream` API si tu dispositivo recibe todos los archivos de transmisión necesarios IDs en el conjunto de datos inicial.

Siga estos pasos para realizar una solicitud `DescribeStream`.

1. Suscríbase al filtro de temas “aceptados” `$aws/things/`*`ThingName`*`/streams/`*`StreamId`*`/description/json`.
2. Suscríbase al filtro de temas “rechazados” `$aws/things/`*`ThingName`*`/streams/`*`StreamId`*`/rejected/json`.
3. Publique una solicitud `DescribeStream` enviando un mensaje a `$aws/things/`*`ThingName`*`/streams/`*`StreamId`*`/describe/json`.
4. Si la solicitud se ha aceptado, el dispositivo recibirá una respuesta `DescribeStream` en el filtro de temas “aceptados”.
5. Si la solicitud se rechazó, el dispositivo recibirá la respuesta de error en el filtro de temas “rechazados”.

> **Note**
>
> Si sustituye `json` con `cbor` en los temas y filtros de temas mostrados, su dispositivo recibirá los mensajes en el formato CBOR, que es más compacto que JSON.

DescribeStream solicitud

Una solicitud `DescribeStream` típica en JSON se parece al siguiente ejemplo.

```
{
    "c": "ec944cfb-1e3c-49ac-97de-9dc4aaad0039"
}
```

- (Opcional) “c” es el campo del token del cliente.

  El token de cliente no puede ser superior a 64 bytes. Un token de cliente que supere los 64 bytes provoca una respuesta de error y un mensaje de error `InvalidRequest`.

DescribeStream respuesta

Una respuesta `DescribeStream` en JSON se parece al siguiente ejemplo.

```
{
    "c": "ec944cfb-1e3c-49ac-97de-9dc4aaad0039",
    "s": 1,
    "d": "This is the description of stream ABC.",
    "r": [
        {
            "f": 0,
            "z": 131072
        },
        {
            "f": 1,
            "z": 51200
        }
    ]
}
```

- “`c`” es el campo de token del cliente. Esto se devuelve si se especificó en la solicitud `DescribeStream`. Use el token del cliente para asociar la respuesta a su solicitud.
- “`s`” es la versión de transmisión en forma de número entero. Puede utilizar esta versión para comprobar la coherencia de sus solicitudes `GetStream`.
- “`r`” contiene una lista de los archivos de la transmisión.
  - “`f`” es el ID del archivo de transmisión en forma de número entero.
  - “`z`” es el tamaño del archivo de transmisión en número de bytes.
- “`d`” contiene la descripción de la transmisión.

Obtener bloques de datos de un archivo de transmisión

Puede usar la API `GetStream` para que un dispositivo pueda recibir archivos de transmisión en bloques de datos pequeños, de modo que puedan utilizarla aquellos dispositivos que tienen limitaciones a la hora de procesar bloques de gran tamaño. Para recibir un archivo de datos completo, es posible que un dispositivo deba enviar o recibir varias solicitudes y respuestas hasta que se reciban y procesen todos los bloques de datos.

GetStream solicitud

Siga estos pasos para realizar una solicitud `GetStream`.

1. Suscríbase al filtro de temas “aceptados” `$aws/things/`*`ThingName`*`/streams/`*`StreamId`*`/data/json`.
2. Suscríbase al filtro de temas “rechazados” `$aws/things/`*`ThingName`*`/streams/`*`StreamId`*`/rejected/json`.
3. Publique una `GetStream` solicitud sobre el tema`$aws/things/`*`ThingName`*`/streams/`*`StreamId`*`/get/json`.
4. Si la solicitud ha sido aceptada, su dispositivo recibirá una o varias respuestas `GetStream` en el filtro de temas “aceptados”. Cada mensaje de respuesta contiene información básica y una carga de datos para un solo bloque.
5. Repita los pasos 3 y 4 para recibir todos los bloques de datos. Debe repetir estos pasos si la cantidad de datos solicitada es superior a 128 KB. Debe programar el dispositivo para que utilice varias solicitudes `GetStream` a fin de recibir todos los datos solicitados.
6. Si la solicitud ha sido rechazada, su dispositivo recibirá la respuesta de error en el filtro de temas “rechazados”.

> **Note**
>
> - Si sustituye “json” por “cbor” en los temas y filtros de temas mostrados, su dispositivo recibirá mensajes en el formato CBOR, que es más compacto que JSON.
> - AWS IoT La entrega de archivos basada en MQTT limita el tamaño de un bloque a 128 KB. Si realiza una solicitud de un bloque que supere los 128 KB, la solicitud fallará.
> - Puede realizar una solicitud para varios bloques cuyo tamaño total sea superior a 128 KB (por ejemplo, si solicita 5 bloques de 32 KB cada uno para un total de 160 KB de datos). En este caso, la solicitud no falla, pero el dispositivo debe realizar varias solicitudes para recibir todos los datos solicitados. El servicio enviará bloques adicionales a medida que tu dispositivo realice solicitudes adicionales. Le recomendamos que continúe con una nueva solicitud sólo después de haber recibido y procesado correctamente la respuesta anterior.
> - Independientemente del tamaño total de los datos solicitados, debe programar su dispositivo para que inicie reintentos cuando no se reciban bloques, o no se reciban correctamente.

Una solicitud `GetStream` típica en JSON se parece al siguiente ejemplo.

```
{
    "c": "1bb8aaa1-5c18-4d21-80c2-0b44fee10380",
    "s": 1,
    "f": 0,
    "l": 4096,
    "o": 2,
    "n": 100,
    "b": "..."
}
```

- [opcional] “`c`” es el campo del token del cliente.

  El token de cliente no puede ser superior a 64 bytes. Un token de cliente que supere los 64 bytes provoca una respuesta de error y un mensaje de error `InvalidRequest`.

- [opcional] “`s`” es el campo de versión de la transmisión (un entero).

  La entrega de archivos basada en MQTT aplica una comprobación de coherencia basada en esta versión solicitada y en la última versión de la transmisión en la nube. Si la versión de transmisión enviada desde un dispositivo en una solicitud `GetStream` no coincide con la última versión de transmisión en la nube, el servicio envía una respuesta de error y un mensaje de error `VersionMismatch`. Normalmente, un dispositivo recibe la versión esperada (más reciente) de la transmisión en el conjunto de datos inicial o en la respuesta a `DescribeStream`.

- “`f`” es el ID del archivo de transmisión (un número entero comprendido entre 0 y 255).

  El ID del archivo de transmisión es obligatorio al crear o actualizar una transmisión mediante el SDK AWS CLI o el SDK. Si un dispositivo solicita un archivo de transmisión con un ID que no existe, el servicio envía una respuesta de error y un mensaje de error `ResourceNotFound`.

- “`l`” es el tamaño del bloque de datos en bytes (un entero comprendido entre 256 y 131 072).

  Consulte Crea un mapa de bits para una solicitud GetStream para obtener instrucciones sobre cómo usar los campos de mapa de bits para especificar qué parte del archivo de transmisión se devolverá en la respuesta `GetStream`. Si un dispositivo especifica un tamaño de bloque que está fuera del rango, el servicio envía una respuesta de error y un mensaje de error `BlockSizeOutOfBounds`.

- [opcional] “`o`” es el desplazamiento del bloque en el archivo de transmisión (un número entero en el rango de 0 a 98.304).

Consulte Crea un mapa de bits para una solicitud GetStream para obtener instrucciones sobre cómo usar los campos de mapa de bits para especificar qué parte del archivo de transmisión se devolverá en la respuesta `GetStream`. El valor máximo de 98.304 se basa en un límite de tamaño de archivo de transmisión de 24 MB y 256 bytes para el tamaño mínimo de bloque. El valor por defecto es 0 si no se especifica.

- [opcional] “n” es el número de bloques solicitados (un número entero comprendido entre 0 y 98 304).

  El campo “n” especifica (1) el número de bloques solicitados o (2) cuando se utiliza el campo de mapa de bits (“b”), un límite en el número de bloques que devolverá la solicitud de mapa de bits. Este segundo uso es opcional. Si no está definido, el valor predeterminado es *`DataBlockSize`* 131072/.

- [opcional] “b” es un mapa de bits que representa los bloques que se solicitan.

  Con un mapa de bits, el dispositivo puede solicitar bloques no consecutivos, lo que facilita la gestión de los reintentos tras un error. Consulte Crea un mapa de bits para una solicitud GetStream para obtener instrucciones sobre cómo usar los campos de mapa de bits para especificar qué parte del archivo de transmisión se devolverá en la respuesta `GetStream`. Para este campo, convierta el mapa de bits en una cadena que represente el valor del mapa de bits en notación hexadecimal. El mapa de bits debe tener menos de 12.288 bytes.

> **Important**
>
> Debe especificarse “n” o “b”. Si no se especifica ninguno de ellos, es posible que la solicitud `GetStream` no sea válida cuando el tamaño del archivo sea inferior a 131072 bytes (128 KB).

GetStream respuesta

Una respuesta `GetStream` en JSON tiene el aspecto que se muestra en este ejemplo para cada bloque de datos que se solicite.

```
{
    "c": "1bb8aaa1-5c18-4d21-80c2-0b44fee10380",
    "f": 0,
    "l": 4096,
```

```
    "i": 2,
    "p": "..."
}
```

- “c” es el campo de token del cliente. Esto se devuelve si se especificó en la solicitud `GetStream`. Use el token del cliente para asociar la respuesta a su solicitud.
- “f” es el ID del archivo de transmisión al que pertenece la carga útil del bloque de datos actual.
- “l“ es el tamaño de la carga útil del bloque de datos en bytes.
- “i” es el ID del bloque de datos contenido en la carga útil. Los bloques de datos se enumeran a partir de 0.
- “p” contiene la carga útil del bloque de datos. Este campo es una cadena que representa el valor del bloque de datos en la codificación Base64.

Crea un mapa de bits para una solicitud GetStream

Puede usar el campo de mapa de bits (b) en una `GetStream` solicitud para obtener bloques no consecutivos de un archivo de transmisión. Esto ayuda a los dispositivos con una capacidad de RAM limitada a solucionar los problemas de entrega de la red. Un dispositivo solo puede solicitar aquellos bloques que no se hayan recibido o que no se hayan recibido correctamente. El mapa de bits determina qué bloques del archivo de transmisión se devolverán. Para cada bit, que se establece en 1 en el mapa de bits, se devolverá el bloque correspondiente del archivo de transmisión.

A continuación, se muestra un ejemplo de cómo especificar un mapa de bits y sus campos de soporte en una solicitud `GetStream`. Por ejemplo, desea recibir un archivo de transmisión en fragmentos de 256 bytes (el tamaño del bloque). Imagine que cada bloque de 256 bytes tiene un número que especifica su posición en el archivo, empezando por 0. Así, el bloque 0 es el primer bloque de 256 bytes del archivo, el bloque 1 es el segundo y, así, sucesivamente. Desea solicitar los bloques 20, 21, 24 y 43 del archivo.

Desplazamientos de bloques

Como el primer bloque es el número 20, especifique el desplazamiento (campo o) como 20 para ahorrar espacio en el mapa de bits.

Número de bloques

Para garantizar que el dispositivo no reciba más bloques de los que puede gestionar con recursos de memoria limitados, puede especificar el número máximo de bloques que se deben devolver

en cada mensaje enviado mediante un sistema de entrega de archivos basado en MQTT. Tenga en cuenta que este valor no se tiene en cuenta si el propio mapa de bits especifica menos de este número de bloques o si el tamaño total de los mensajes de respuesta enviados mediante la entrega de archivos basada en MQTT supera el límite de servicio de 128 KB por solicitud `GetStream`.

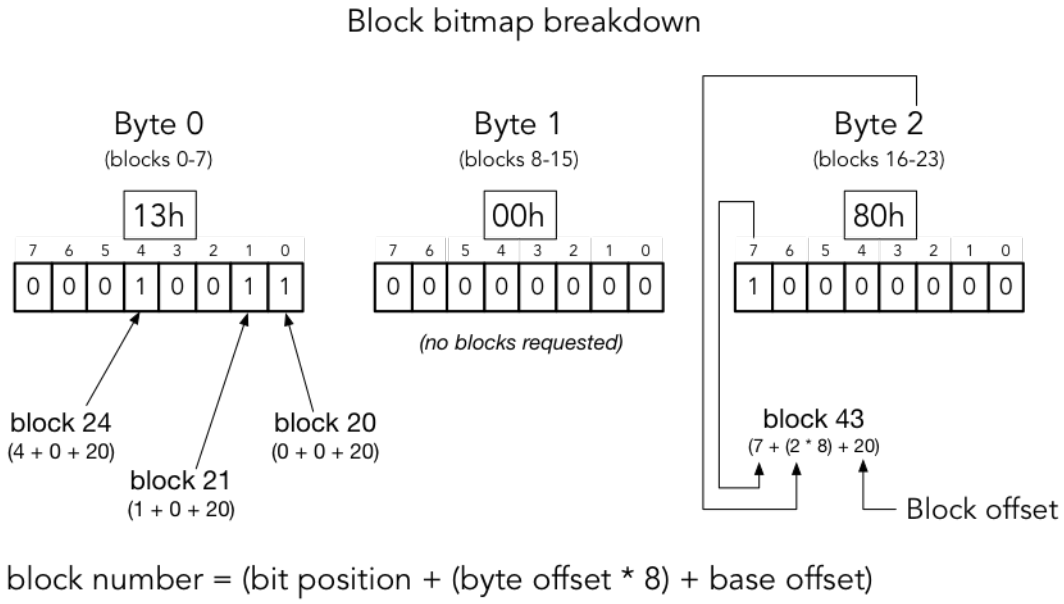
Mapa de bits de bloques

El mapa de bits en sí es una matriz de bytes sin signo expresados en notación hexadecimal e incluidos en la solicitud `GetStream` como una representación en cadena del número. Pero para construir esta cadena, empecemos por pensar en el mapa de bits como una secuencia larga de bits (un número binario). Si un bit de esta secuencia se establece en 1, el bloque correspondiente del archivo de transmisión se devolverá al dispositivo. En nuestro ejemplo, queremos recibir los bloques 20, 21, 24 y 43, por lo que debemos establecer los bits 20, 21, 24 y 43 en nuestro mapa de bits. Podemos usar el desplazamiento del bloque para ahorrar espacio, así que después de restar el desplazamiento de cada número de bloque, queremos establecer los bits 0, 1, 4 y 23, como en el siguiente ejemplo.

```
1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
```

Si se toma un byte (8 bits) a la vez, se escribe convencionalmente como: “0b00010011”, “0b00000000” y “0b10000000”. El bit 0 aparece en nuestra representación binaria al final del primer byte y el bit 23 al principio del último. Esto puede resultar confuso a menos que conozca las convenciones. El primer byte contiene los bits 7-0 (en ese orden), el segundo byte contiene los bits 15-8, el tercer byte contiene los bits 23-16, y así sucesivamente. En notación hexadecimal, se convierte en “0x130080”.

> **Tip**
>
> Puede convertir la notación binaria estándar en hexadecimal. Tome cuatro dígitos binarios a la vez y conviértalos a su equivalente hexadecimal. Por ejemplo, “0001” pasa a ser “1”, “0011” pasa a ser “3” y así sucesivamente.



Al juntar todo esto, el JSON de nuestra solicitud `GetStream` tiene el siguiente aspecto.

```
{
    "c" : "1",
    "s" : 1,
    "l" : 256,
    "f" : 1,
    "o" : 20,
    "n" : 32,
    "b" : "130080"
}
```

- “c” es el campo de token del cliente.
- “s” es la versión de transmisión esperada.
- “l“ es el tamaño de la carga útil del bloque de datos en bytes.
- “f” es el identificador del índice del archivo fuente.
- “o” es el desplazamiento de los bloque.
- “n” es el número de bloques.
- “b” es el mapa de bits de BlockID que falta a partir del desplazamiento. Este valor debe estar codificado con based64.

Gestión de errores derivados de la entrega de archivos AWS IoT basada en MQTT

Una respuesta de error que se envía a un dispositivo para ambos `DescribeStream` y `GetStream` APIs contiene un token de cliente, un código de error y un mensaje de error. Una respuesta de error típica se parece al siguiente ejemplo.

```
{
    "o": "BlockSizeOutOfBounds",
    "m": "The block size is out of bounds",
    "c": "1bb8aaa1-5c18-4d21-80c2-0b44fee10380"
}
```

- “o” es el código de error que indica la razón por la que se ha producido un error. Consulte los códigos de error más adelante en esta sección para obtener más información.
- “`m`” es el mensaje de error que contiene los detalles del error.
- “`c`” es el campo de token del cliente. Esto se puede devolver si se especificó en la solicitud `DescribeStream`. Puede utilizar el testigo del cliente para asociar la respuesta a su solicitud.

  El campo del token del cliente no siempre se incluye en una respuesta de error. Cuando el token de cliente proporcionado en la solicitud no es válido o tiene un formato incorrecto, no se devuelve en la respuesta al error.

> **Note**
>
> Por motivos de compatibilidad con versiones anteriores, es posible que los campos de la respuesta al error no estén abreviados. Por ejemplo, el código de error puede designarse mediante los campos “code” u “o” y el campo token de cliente puede designarse mediante los campos “clientToken” o “c”. Le recomendamos que utilice la forma de abreviatura que se muestra arriba.

InvalidTopic

El tema MQTT del mensaje de transmisión no es válido.

InvalidJson

La solicitud de transmisión no es un documento JSON válido.

InvalidCbor

La solicitud de transmisión no es un documento CBOR válido.

InvalidRequest

Por lo general, la solicitud se identifica como incorrecta. Para más información, consulte el mensaje de error.

Unauthorized

La solicitud no está autorizada para acceder a los archivos de datos de transmisión en el medio de almacenamiento, como Amazon S3. Para más información, consulte el mensaje de error.

BlockSizeOutOfBounds

El tamaño del bloque está fuera de los límites. Consulte la sección Entrega de archivos basada en MQTT” en Service Quotas AWS IoT Core.

OffsetOutOfBounds

El desplazamiento está fuera de los límites. Consulte la sección Entrega de archivos basada en MQTT” en Service Quotas AWS IoT Core.

BlockCountLimitExceeded

El número de bloques de solicitudes está fuera de los límites. Consulte la sección Entrega de archivos basada en MQTT” en Service Quotas AWS IoT Core.

BlockBitmapLimitExceeded

El tamaño del mapa de bits de solicitud está fuera de los límites. Consulte la sección Entrega de archivos basada en MQTT” en Service Quotas AWS IoT Core.

ResourceNotFound

No se encontraron el flujo, los archivos, las versiones de archivos o los bloques solicitados. Consulte el mensaje de error para obtener más información.

VersionMismatch

La versión de transmisión de la solicitud no coincide con la versión de transmisión de la característica de entrega de archivos basada en MQTT. Esto indica que los datos de la

transmisión se han modificado desde que el dispositivo recibió inicialmente la versión de transmisión.

ETagDiscrepancia

El S3 ETag de la transmisión no coincide con el ETag de la última versión del objeto S3.

InternalError

Se ha producido un error interno en la entrega de archivos basada en MQTT.

Un ejemplo de caso de uso en Freertos OTA

El agente OTA de FreeRTOS (over-the-air) utiliza la entrega de archivos AWS IoT basada en MQTT para transferir imágenes de firmware de FreeRTOS a dispositivos FreeRTOS. Para enviar el conjunto de datos inicial a un dispositivo, utiliza el servicio AWS IoT Job para programar un trabajo de actualización OTA en los dispositivos Freertos.

Para una implementación de referencia de un cliente de entrega de archivos basado en MQTT, consulte los códigos del agente OTA de FreeRTOS en la documentación de FreeRTOS.

Asesor de dispositivos

Device Advisor es una capacidad de prueba basada en la nube y totalmente administrada para validar dispositivos IoT durante el desarrollo del software del dispositivo. Device Advisor proporciona pruebas prediseñadas que puede usar para validar los dispositivos de IoT y lograr una conectividad confiable y segura antes de implementar los dispositivos en producción. AWS IoT Core Las pruebas prediseñadas de Device Advisor le ayudarán a validar el software de su dispositivo según las prácticas recomendadas de uso de TLS, MQTT, Device Shadow e IoT Jobs. También puede descargar informes de cualificación firmados para enviarlos a la Red de socios de AWS y obtener la cualificación de su dispositivo para el Catálogo de dispositivos de socios de AWS sin necesidad de enviar su dispositivo y esperar a que lo prueben.

> **Note**
>
> Device Advisor es compatible en las regiones us-east-1, us-west-2, ap-northeast-1 y eu-west-1.
> Device Advisor es compatible con dispositivos y clientes que utilizan los protocolos MQTT y MQTT over WebSocket Secure (WSS) para publicar mensajes y suscribirse a ellos. Todos los protocolos admiten y IPv4 . IPv6
> Device Advisor admite certificados de servidor RSA.

Cualquier dispositivo diseñado para conectarse AWS IoT Core puede utilizar Device Advisor. Puede acceder a Device Advisor desde la AWS IoT consola o mediante el AWS CLI SDK. Cuando esté listo para probar el dispositivo, regístrelo AWS IoT Core y configure el software del dispositivo con el terminal Device Advisor. A continuación, elija las pruebas prediseñadas, configúrelas, ejecute las pruebas en su dispositivo y obtenga los resultados de las pruebas junto con registros detallados o un informe de cualificación.

Device Advisor es un terminal de prueba en la AWS nube. Puede probar sus dispositivos configurándolos para que se conecten al punto de conexión de prueba proporcionado por Device Advisor. Una vez configurado un dispositivo para conectarse al punto final de prueba, puede visitar la consola de Device Advisor o usar el AWS SDK para elegir las pruebas que quiere ejecutar en sus dispositivos. A continuación, Device Advisor administra el ciclo de vida completo de una prueba, incluido el aprovisionamiento de recursos, la programación del proceso de prueba, la administración de la máquina de estados, el registro del comportamiento del dispositivo, el registro de los resultados y el suministro de los resultados finales en forma de informe de la prueba.

Protocolos TLS

El protocolo de seguridad de la capa de transporte (TLS) se utiliza para cifrar datos confidenciales a través de redes inseguras, como internet. El protocolo TLS es el sucesor del protocolo Secure Sockets Layer (SSL).

Device Advisor es compatible con los siguientes protocolos TLS:

- TLS 1.3 (recomendado)
- TLS 1.2

Protocolos, asignaciones de puertos y autenticación

El protocolo de comunicación de dispositivos lo utiliza un dispositivo o un cliente para conectarse al agente de mensajes mediante un punto de conexión de dispositivo. En la siguiente tabla se enumeran los protocolos admitidos por los puntos de conexión de Device Advisor, así como los métodos de autenticación y los puertos que se utilizan.

Protocolos, autenticación y asignaciones de puertos

Protocolo	Operaciones admitidas	Autenticación	Puerto	Nombre del protocolo ALPN
MQTT ha terminado WebSocket	Publicar, suscribirse	Signature Version 4	443	N/A
MQTT	Publicar, suscribirse	Certificado de cliente X.509	8883	`x-amzn-mqtt-ca`
MQTT	Publicar, suscribirse	Certificado de cliente X.509	443	N/A

El capítulo contiene las siguientes secciones:

- Configuración
- Introducción a Device Advisor en la consola
- Flujo de trabajo de Device Advisor
- Flujo de trabajo detallado de la consola de Device Advisor

- Flujo de trabajo de la consola de pruebas de larga duración
- puntos de conexión de VPC de Device Advisor (AWS PrivateLink)
- Casos de prueba de Device Advisor

Configuración

Antes de usar Device Advisor por primera vez, realice las siguientes tareas:

Cree un objeto de &IoT

En primer lugar, cree un objeto IoT y asocie un certificado. Para ver un tutorial sobre cómo crear objetos, consulte Crear un objeto.

Cree un rol de IAM para utilizarlo como su rol de dispositivo

> **Note**
>
> Puede crear rápidamente el rol de dispositivo con la consola de Device Advisor. Para saber cómo configurar su función de dispositivo con la consola de Device Advisor, consulte Introducción a Device Advisor en la consola.

1. Ve a la AWS Identity and Access Management consola e inicia sesión en la que Cuenta de AWS utilizas para realizar las pruebas de Device Advisor.
2. En el panel de navegación izquierdo, elija Políticas.
3. Elija Crear política.
4. En Crear política, haga lo siguiente:

    a. En Servicio, seleccione IoT.

    b. En Acciones, elija una de las siguientes opciones:

    - (Recomendado) Seleccione acciones basadas en la política asociada al objeto o certificado IoT que creó en la sección anterior.
    - Busque las siguientes acciones en el cuadro Filtrar acción y selecciónelas:
        - `Connect`
        - `Publish`

- `Subscribe`
- `Receive`
- `RetainPublish`

c. En Recursos, restrinja los recursos de cliente y tema. Restringir estos recursos es una de las prácticas recomendadas en materia de seguridad. Para restringir los recursos, haga lo siguiente:

i. Seleccione Especificar ARN del recurso de cliente para la acción Conectar.

ii. Seleccione Agregar ARN y, a continuación, realice una de las siguientes acciones:

> **Note**
>
> El clientId es el ID del cliente MQTT que su dispositivo utiliza para interactuar con Device Advisor.

- Rellene los campos Región, accountID y clientID en el editor ARN visual.
- Introduce manualmente los nombres de los recursos de Amazon (ARNs) de los temas de IoT con los que quieres ejecutar tus casos de prueba.

iii. Elija Añadir.

iv. Elija Especificar ARN del recurso del tema para la recepción y una acción más.

v. Seleccione Agregar ARN y, a continuación, realice una de las siguientes acciones:

> **Note**
>
> El nombre del tema es el tema MQTT en el que su dispositivo publica los mensajes.

- Rellene los campos Región, accountID y Nombre del tema en el editor ARN visual.
- Introduce manualmente los temas ARNs de IoT con los que deseas ejecutar tus casos de prueba.

vi. Elija Añadir.

vii. Seleccione Especificar ARN del recurso topicFilter para la acción Suscribirse.

viii. Seleccione Agregar ARN y, a continuación, realice una de las siguientes acciones:

> **Note**
>
> El nombre del tema es el tema MQTT al que su dispositivo se suscribe.

- Rellene los campos Región, accountID y Nombre del tema en el editor ARN visual.
- Introduce manualmente los temas ARNs de IoT con los que deseas ejecutar tus casos de prueba.

ix. Elija Agregar.

5. Elija Siguiente: etiquetas.
6. Elija Siguiente: Revisar.
7. En Revisar política, introduzca un nombre para su política.
8. Elija Crear política.
9. En el panel de navegación izquierdo, seleccione Roles.
10. Elija Crear rol.
11. En Seleccionar entidad de confianza, elija Política de confianza personalizada.
12. Introduzca la siguiente política de confianza en el cuadro Política de confianza personalizada. Para protegerse contra el problema de la sustitución confusa, agregue las claves contextuales de condición global `aws:SourceArn` y `aws:SourceAccount` a la política.

> **Important**
>
> Su `aws:SourceArn` debe ajustarse a `format: arn:aws:iotdeviceadvisor:`*`region:account-id`*`:*`. Asegúrese de que *`region`* coincide con su región de AWS IoT y *`account-id`* con el ID de su cuenta de cliente. Para obtener más información, consulte Prevención de la sustitución confusa entre servicios.

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

```
            "Effect": "Allow",
            "Principal": {
                "Service": "iotdeviceadvisor.amazonaws.com"
        },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "123456789012"
            },
                "ArnLike": {
                    "aws:SourceArn":
 "arn:aws:iotdeviceadvisor:*:123456789012:suitedefinition/*"
            }
        }
        }
    ]
}
```

13. Elija Siguiente.
14. Elija la política que creó en el paso 4.
15. (Opcional) En Establecer límite de permisos, seleccione Utilizar un límite de permisos para controlar los permisos que puede tener el rolo como máximo y, a continuación, seleccione la política que ha creado.
16. Elija Siguiente.
17. Introduzca un nombre de rol y una descripción del rol.
18. Elija Crear rol.

Cree una política administrada de forma personalizada para que un usuario de IAM utilice Device Advisor

1. Vaya a la consola de IAM en https://console.aws.amazon.com/iam/. Si se le solicita, introduzca sus credenciales de AWS para iniciar sesión.
2. En el panel de navegación izquierdo, elija Políticas.
3. Elija Crear política y, a continuación, elija la pestaña JSON.
4. Agregue los permisos necesarios para utilizar Device Advisor. El documento de la política se encuentra en el tema Prácticas recomendadas de seguridad.
5. Elija Revisar la política.

6. Introduzca un Nombre y una Descripción.
7. Elija Crear política.

Cree un usuario de IAM para utilizar Device Advisor

> **Note**
>
> Le recomendamos que cree un usuario de IAM para utilizarlo cuando ejecute las pruebas de Device Advisor. Ejecutar las pruebas de Device Advisor desde un usuario administrador puede plantear riesgos de seguridad y no es recomendable.

1. Diríjase a la consola de IAM en https://console.aws.amazon.com/iam/Si se le solicita, introduzca sus AWS credenciales para iniciar sesión.
2. En el panel de navegación izquierdo, elija Usuarios.
3. Elija Add user (Agregar usuario).
4. Introduzca un nombre de usuario.
5. Los usuarios necesitan acceso programático si quieren interactuar con personas AWS ajenas a. Consola de administración de AWS La forma de conceder el acceso programático depende del tipo de usuario que acceda. AWS

   Para conceder acceso programático a los usuarios, elija una de las siguientes opciones.

¿Qué usuario necesita acceso programático?	Para	Mediante
IAM	(Recomendado) Utilice las credenciales de la consola como credenciales temporales para firmar las solicitudes programáticas dirigidas al AWS CLI AWS SDKs, o. AWS APIs	Siga las instrucciones de la interfaz que desea utilizar: • Para ello AWS CLI, consulte Iniciar sesión para el desarrollo AWS local en la Guía del AWS Command Line Interface usuario.

¿Qué usuario necesita acceso programático?	Para	Mediante
		• Para ello AWS SDKs, consulte Iniciar sesión para el desarrollo AWS local en la Guía de referencia de AWS SDKs and Tools.
Identidad del personal (Usuarios administrados en el IAM Identity Center)	Utilice credenciales temporales para firmar las solicitudes programáticas dirigidas al AWS CLI, AWS SDKs, o AWS APIs.	Siga las instrucciones de la interfaz que desea utilizar: • Para ello AWS CLI, consulte Configuración del AWS CLI uso AWS IAM Identity Center en la Guía del AWS Command Line Interface usuario. • Para AWS SDKs ver las herramientas y AWS APIs, consulte la autenticación del Centro de Identidad de IAM en la Guía de referencia de herramientas AWS SDKs y herramien tas.
IAM	Utilice credenciales temporales para firmar las solicitudes programáticas dirigidas al AWS CLI AWS SDKs, o. AWS APIs	Siga las instrucciones de Uso de credenciales temporales con AWS recursos de la Guía del usuario de IAM.

¿Qué usuario necesita acceso programático?	Para	Mediante
IAM	(No recomendado) Utilice credenciales de larga duración para firmar las solicitudes programáticas dirigidas al AWS CLI AWS SDKs, o. AWS APIs	Siga las instrucciones de la interfaz que desea utilizar: • Para ello AWS CLI, consulte Autenticación con credenciales de usuario de IAM en la Guía del AWS Command Line Interface usuario. • Para obtener AWS SDKs información sobre las herramientas, consulte Autenticarse con credenciales de larga duración en la Guía de referencia de herramientas AWS SDKs y herramien tas. • Para ello AWS APIs, consulte Administrar las claves de acceso para los usuarios de IAM en la Guía del usuario de IAM.

6. Elija Siguiente: permisos.
7. Para dar acceso, agregue permisos a los usuarios, grupos o roles:

   - Usuarios y grupos en AWS IAM Identity Center:

     Cree un conjunto de permisos. Siga las instrucciones de Creación de un conjunto de permisos en la Guía del usuario de AWS IAM Identity Center .

   - Usuarios gestionados en IAM a través de un proveedor de identidades:

Cree un rol para la federación de identidades. Siga las instrucciones descritas en Creación de un rol para un proveedor de identidad de terceros (federación) en la Guía del usuario de IAM.

- Usuarios de IAM:
  - Cree un rol que el usuario pueda aceptar. Siga las instrucciones descritas en Creación de un rol para un usuario de IAM en la Guía del usuario de IAM.
  - (No recomendado) Adjunte una política directamente a un usuario o agregue un usuario a un grupo de usuarios. Siga las instrucciones descritas en Adición de permisos a un usuario (consola) de la Guía del usuario de IAM.

8. Introduzca el nombre de la política administrada de forma personalizada que ha creado en el cuadro de búsqueda. A continuación, marque la casilla Nombre de la política.
9. Elija Siguiente: Etiquetas.
10. Elija Siguiente: Revisar.
11. Seleccione la opción Crear usuario.
12. Seleccione Cerrar.

Device Advisor requiere el acceso a sus AWS recursos (cosas, certificados y puntos de conexión) en su nombre. Su usuario de IAM debe tener los permisos necesarios. Device Advisor también publicará los registros en Amazon CloudWatch si adjuntas la política de permisos necesaria a tu usuario de IAM.

Configuración del dispositivo

Device Advisor utiliza la extensión TLS de indicación de nombre de servidor (SNI) para aplicar configuraciones de TLS. Los dispositivos deben utilizar esta extensión cuando se conecten y pasen un nombre de servidor que sea idéntico al punto de conexión de prueba de Device Advisor.

Device Advisor permite la conexión TLS cuando una prueba está en el estado `Running`. Deniega la conexión TLS antes y después de cada prueba. Por esta razón, le recomendamos que utilice el mecanismo de reintento de conexión del dispositivo para una experiencia de prueba totalmente automatizada con Device Advisor. Puede ejecutar conjuntos de pruebas que incluyan más de un caso de prueba, como conexión TLS, conexión MQTT y publicación MQTT. Si ejecuta varios casos de prueba, le recomendamos que su dispositivo intente conectarse a nuestro punto de conexión de prueba cada cinco segundos. A continuación, puede automatizar la ejecución de varios casos de prueba en secuencia.

> **Note**
>
> Para preparar el software de su dispositivo para las pruebas, le recomendamos que utilice un SDK que pueda conectarse a AWS IoT Core. A continuación, deberá actualizar el SDK con el punto de conexión de prueba de Device Advisor proporcionado para su Cuenta de AWS.

Device Advisor admite dos tipos de puntos de conexión: puntos de conexión de nivel de cuenta y de nivel de dispositivo. Elija el punto de conexión que mejor se adapte a su caso. Para ejecutar simultáneamente varios conjuntos de pruebas para distintos dispositivos, utilice un punto de conexión de nivel de dispositivo.

Ejecute el siguiente comando para obtener el punto de conexión de nivel de dispositivo:

Para los clientes de MQTT que utilicen certificados de cliente X.509:

```
aws iotdeviceadvisor get-endpoint --thing-arn your-thing-arn
```

o

```
aws iotdeviceadvisor get-endpoint --certificate-arn your-certificate-arn
```

Para los WebSocket clientes de MQTT que utilizan la versión 4 de Signature:

```
aws iotdeviceadvisor get-endpoint --device-role-arn your-device-role-arn --
authentication-method SignatureVersion4
```

Para ejecutar un conjunto de pruebas a la vez, elija un punto de conexión de nivel de cuenta. Ejecute el siguiente comando para obtener el punto de conexión de nivel cuenta:

```
aws iotdeviceadvisor get-endpoint
```

Introducción a Device Advisor en la consola

Este tutorial le ayudará a empezar a utilizar AWS IoT Core Device Advisor la consola. Device Advisor ofrece características como las pruebas obligatorias y los informes de cualificación firmados. Puede utilizar estas pruebas e informes para calificar y enumerar los dispositivos en el Catálogo de

dispositivos de socios de AWS, tal y como se detalla en el programa de cualificación de AWS IoT Core.

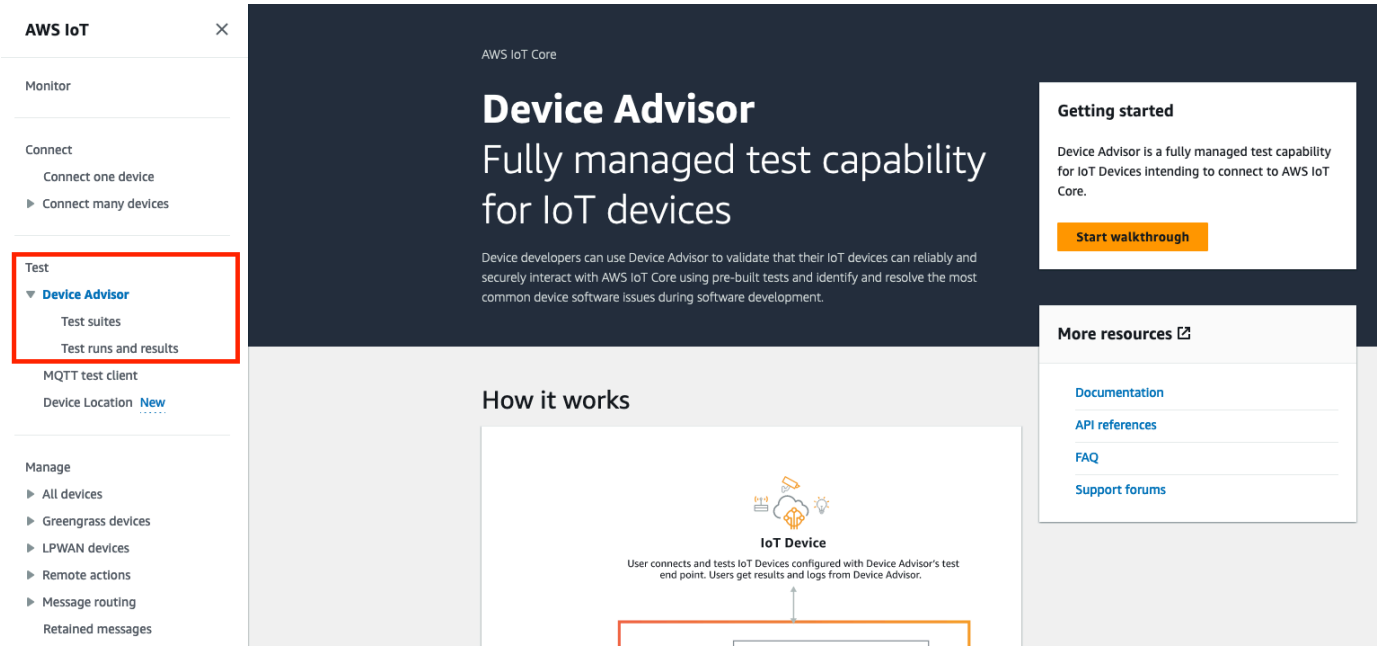
Para obtener más información sobre el uso de Device Advisor, consulte Flujo de trabajo de Device Advisor y Flujo de trabajo detallado de la consola de Device Advisor.

Para completar este tutorial, siga los pasos descritos en Configuración.

> **Note**
>
> Device Advisor es compatible con lo siguiente Regiones de AWS:
>
> - Este de EE. UU. (Norte de Virginia)
> - Oeste de EE. UU. (Oregón)
> - Asia-Pacífico (Tokio)
> - Europa (Irlanda)

Introducción

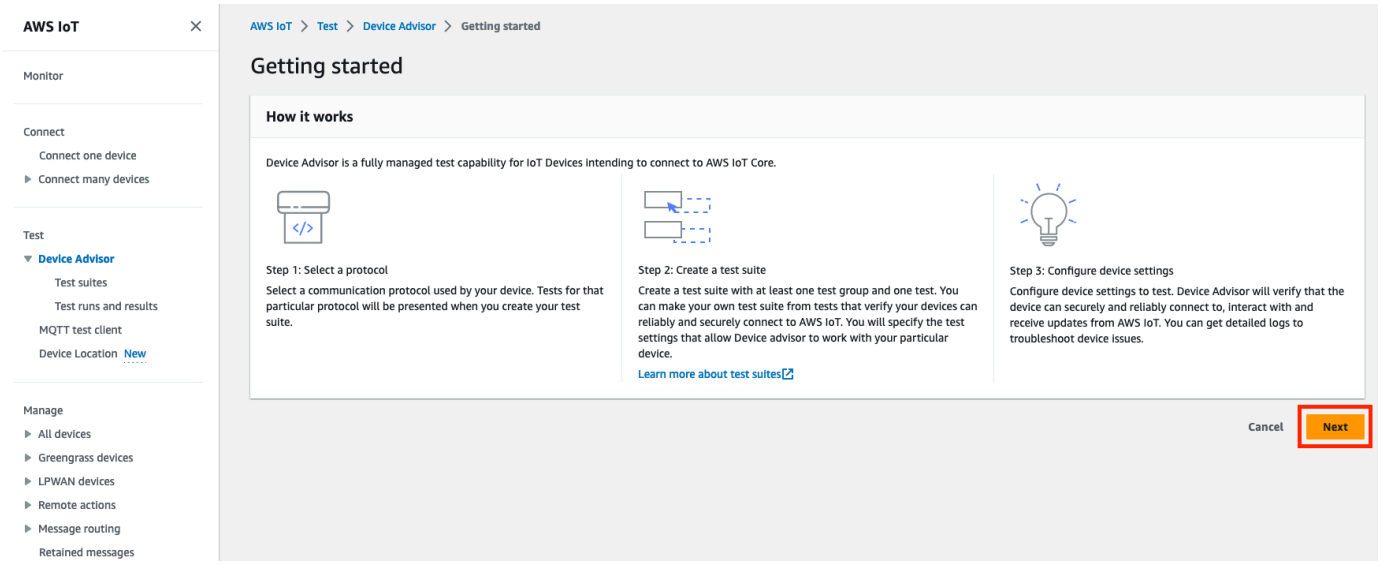
1. En el panel de navegación de la consola de AWS IoT, en Prueba, seleccione Device Advisor. A continuación, elija el botón Iniciar recorrido en la consola.



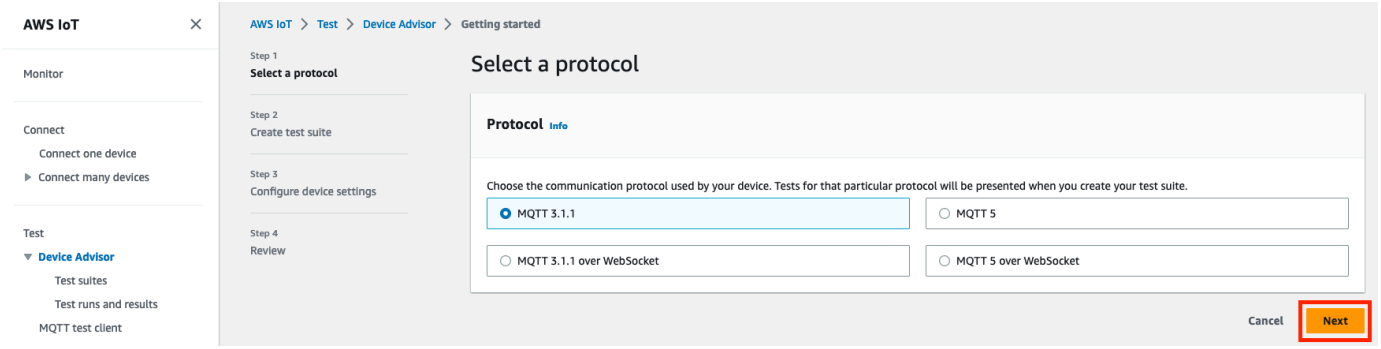
2. La página Introducción a Device Advisor ofrece una visión general de los pasos necesarios para crear un conjunto de pruebas y ejecutarlas con su dispositivo. También puede encontrar aquí el

punto de conexión de prueba de Device Advisor para su cuenta. Debe configurar el firmware o el software del dispositivo utilizado para las pruebas para conectarse a este punto de conexión de prueba.

Para completar este tutorial, primero debe crear un objeto y un certificado. Después de revisar la información en Cómo funciona, elija Siguiente.

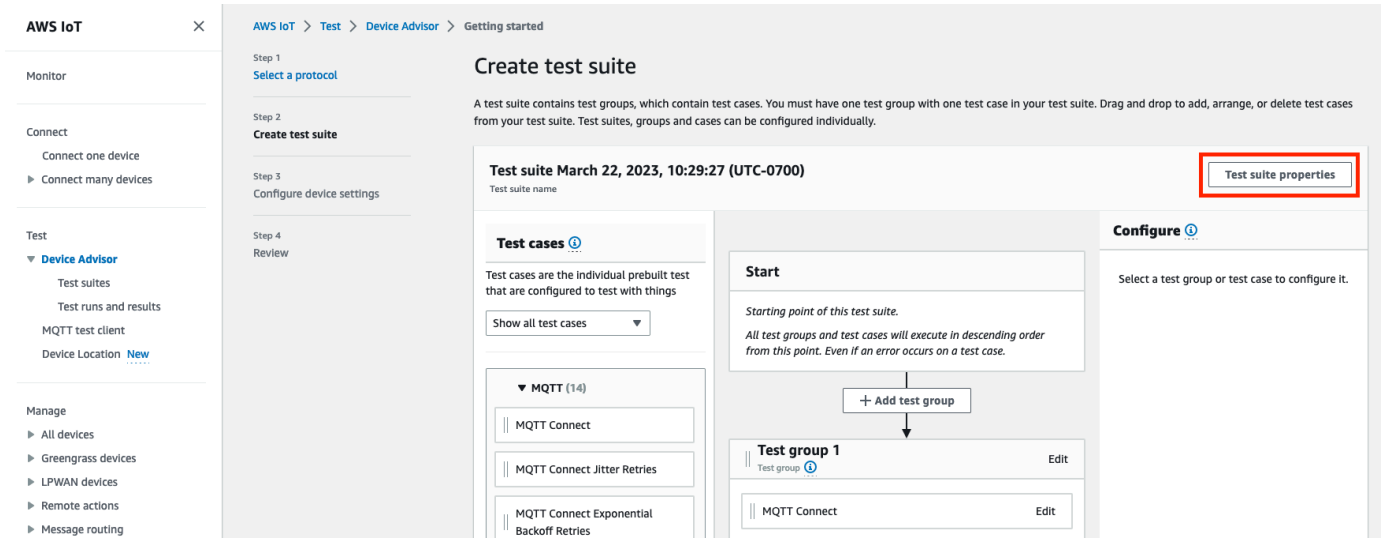


3. En el Paso 1: Seleccionar un protocolo, seleccione un protocolo de entre las opciones de la lista. A continuación, elija Siguiente.



4. En el paso 2, creará y configurará un conjunto de pruebas personalizado. Un conjunto de pruebas personalizado debe tener al menos un grupo de pruebas, y cada grupo de pruebas debe tener al menos un caso de prueba. Hemos agregado el caso de prueba MQTT Connect para que pueda empezar.

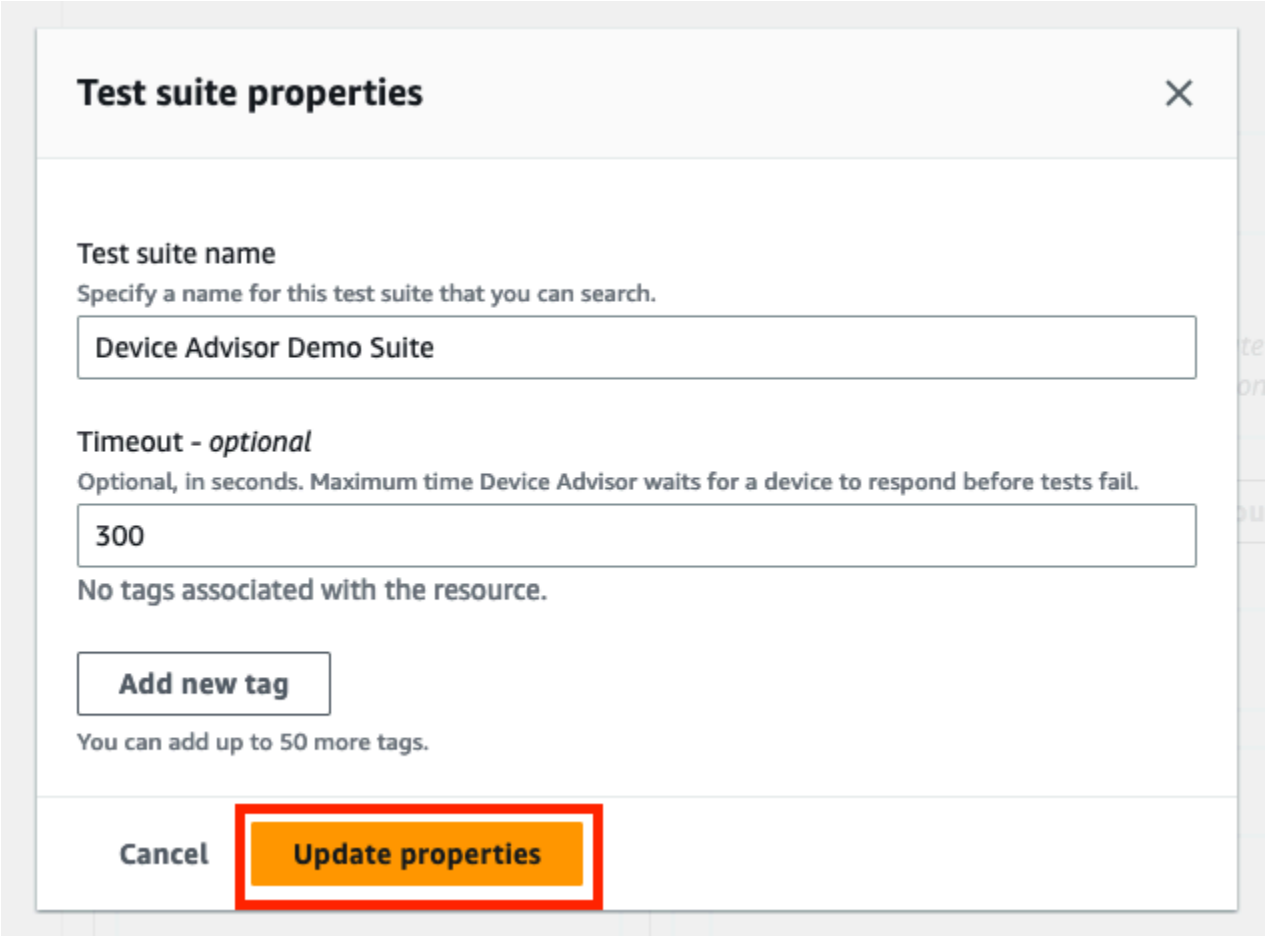
   Seleccione Propiedades del conjunto de pruebas.



Proporcione las propiedades del conjunto de pruebas al crear su conjunto de pruebas. Puede configurar las siguientes propiedades de nivel de conjunto:

- Nombre del conjunto de pruebas: introduzca un nombre para su conjunto de pruebas.
- Tiempo de espera (opcional): el tiempo de espera (en segundos) para cada caso de prueba en el conjunto de pruebas actual. Si no especifica un valor de tiempo de espera, se utilizará el valor predeterminado.
- Etiquetas (opcional): agregue etiquetas al conjunto de pruebas.

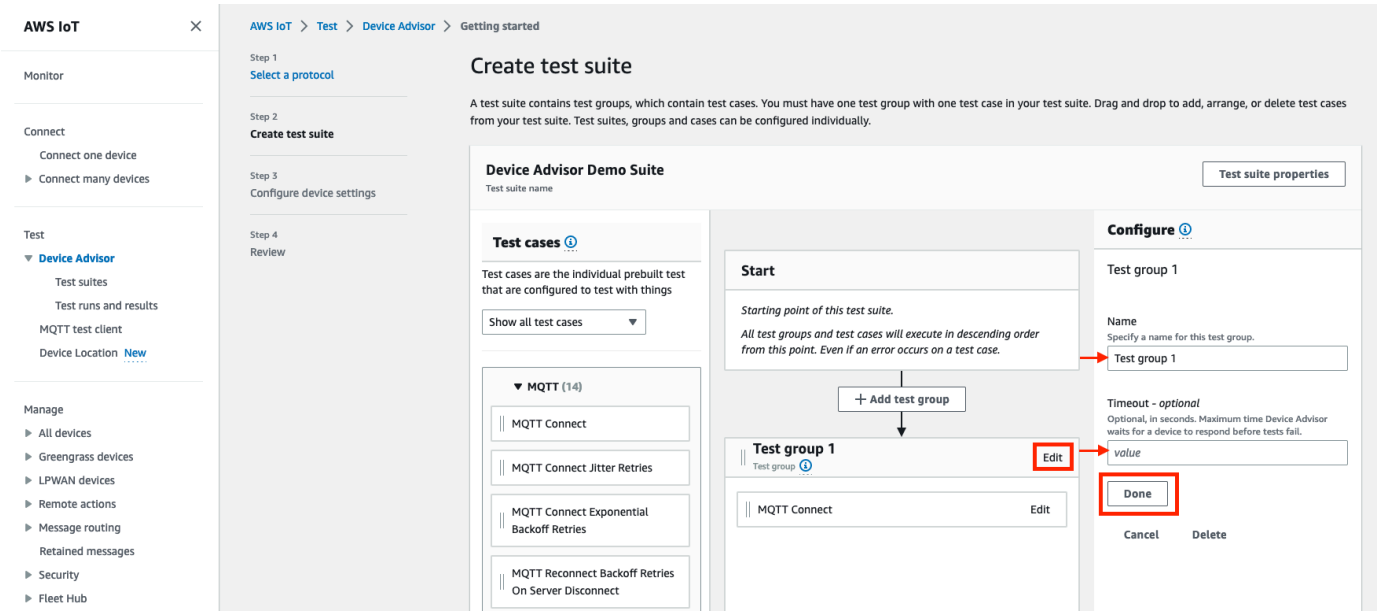
Cuando haya finalizado, seleccione Actualizar propiedades.



5. (Opcional) Para actualizar la configuración del grupo de pruebas, seleccione el botón Editar situado junto al nombre del grupo de pruebas.

   - Nombre: introduzca un nombre personalizado para el grupo del conjunto de pruebas.
   - Tiempo de espera (opcional): el tiempo de espera (en segundos) para cada caso de prueba en el conjunto de pruebas actual. Si no especifica un valor de tiempo de espera, se utilizará el valor predeterminado.

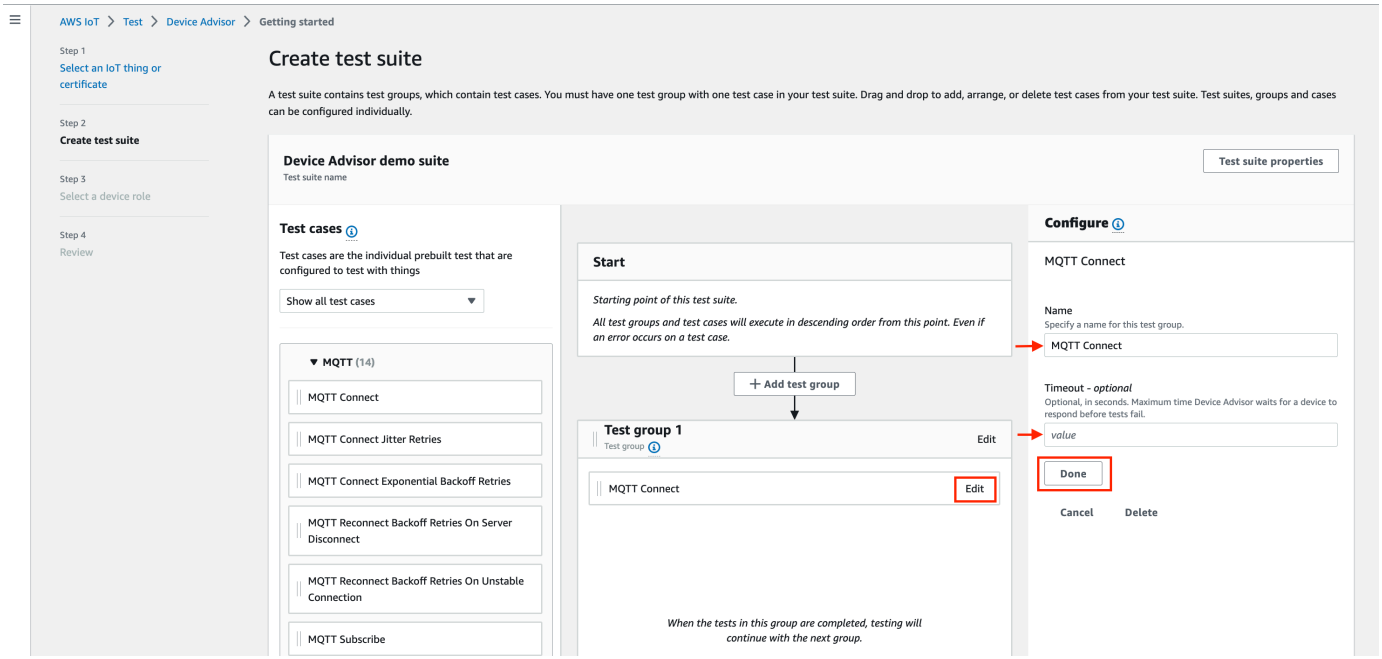
   Cuando haya terminado, elija Hecho para continuar.



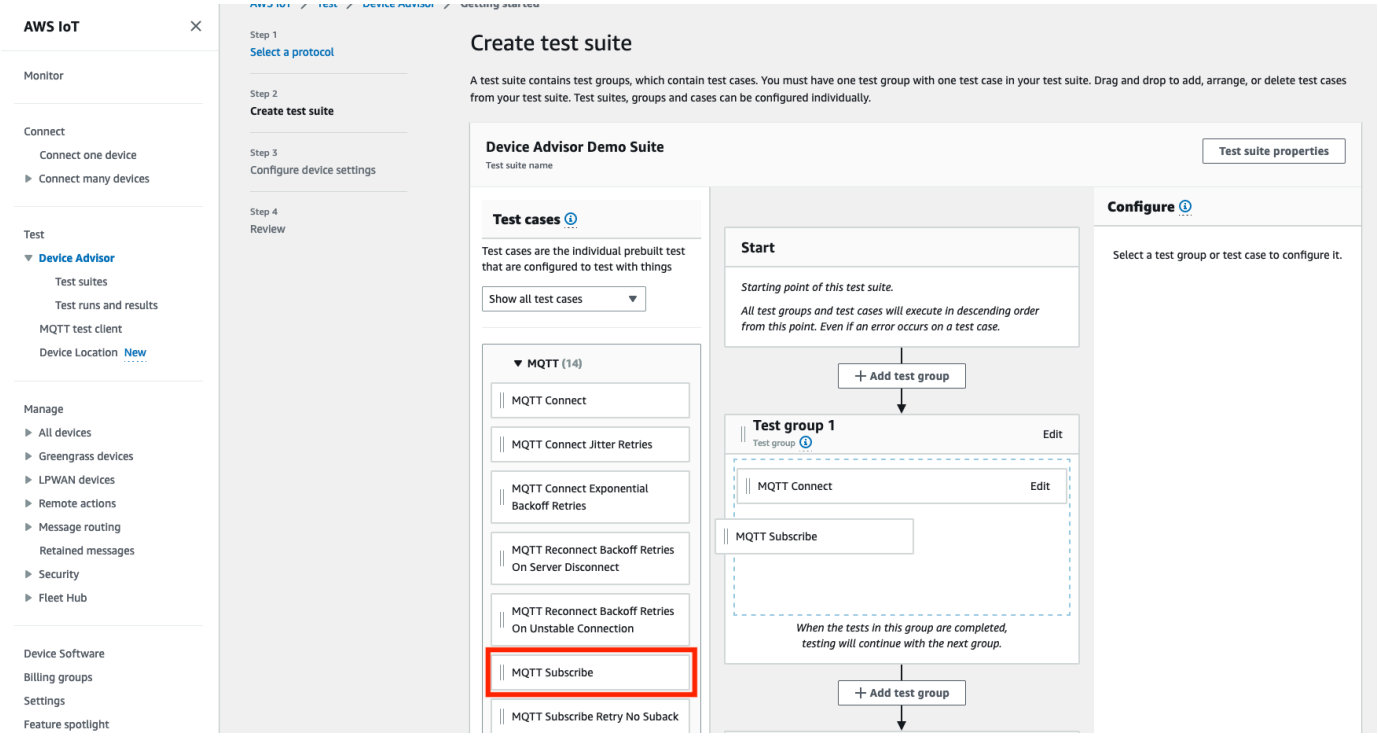
6. (Opcional) Para actualizar la configuración de un caso de prueba, seleccione el botón Editar situado junto al nombre del caso de prueba.

   - Nombre: introduzca un nombre personalizado para el grupo del conjunto de pruebas.
   - Tiempo de espera (opcional): el tiempo de espera (en segundos) para el caso de prueba seleccionado. Si no especifica un valor de tiempo de espera, se utilizará el valor predeterminado.

   Cuando haya terminado, elija Hecho para continuar.

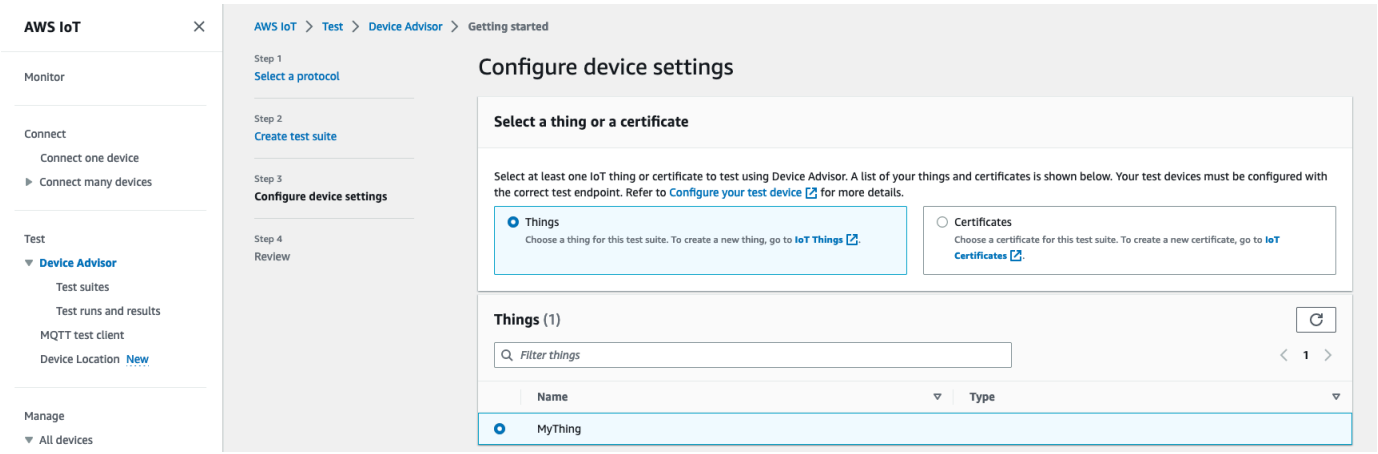


7. (Opcional) Para agregar más grupos de pruebas al conjunto de pruebas, seleccione Agregar grupo de pruebas y, a continuación, siga las instrucciones del paso 5.
8. (Opcional) Para agregar más casos de prueba, arrastre los casos de prueba de la sección Casos de prueba a cualquiera de sus grupos de pruebas.



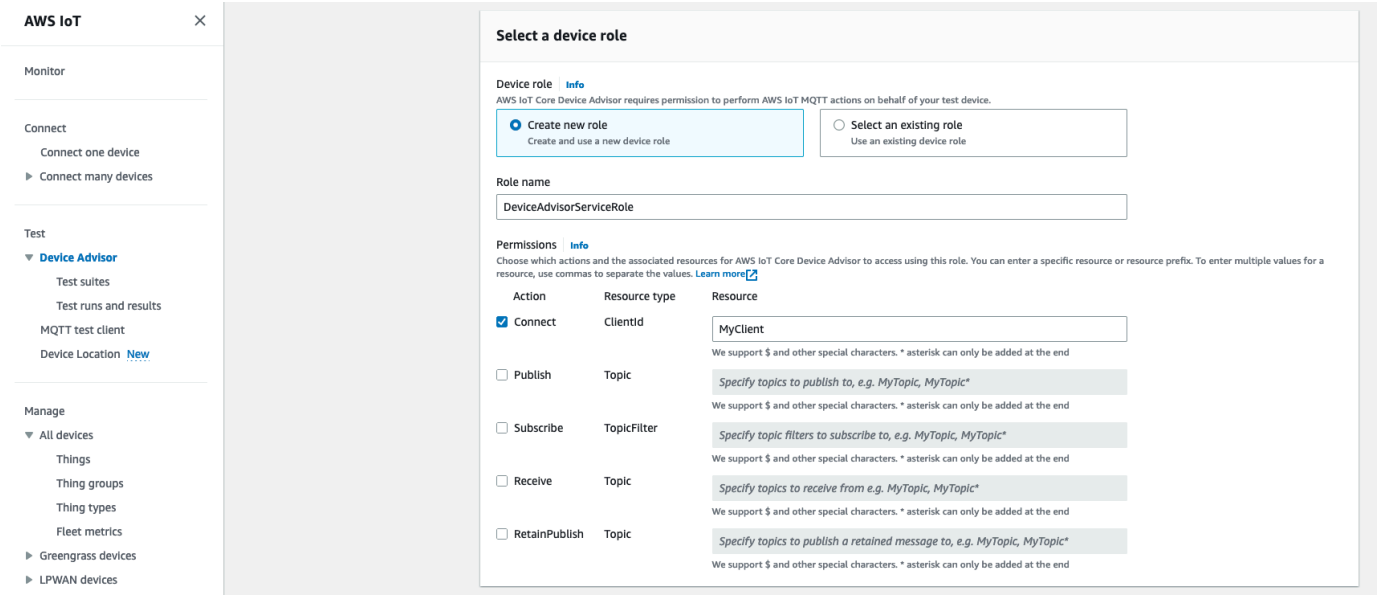
9. Puede cambiar el orden de sus grupos y casos de prueba. Para realizar cambios, arrastre los casos de prueba de la lista hacia arriba o hacia abajo. Device Advisor ejecuta las pruebas en el orden en que usted las enumeró.

   Una vez configurado el conjunto de pruebas, seleccione Siguiente.
10. En el paso 3, seleccione un elemento o certificado para probarlo AWS IoT con Device Advisor. Si no dispone de objetos o certificados, consulte Configuración.

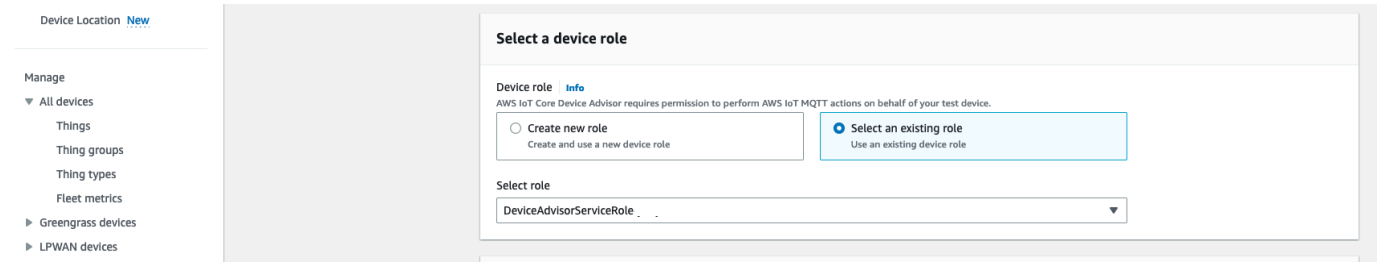


11. Puede configurar una función de dispositivo que Device Advisor utilice para realizar acciones de AWS IoT MQTT en nombre del dispositivo de prueba. Solo para el caso de prueba MQTT Connect, la acción Conectar se selecciona automáticamente. Esto se debe a que el rol de dispositivo requiere este permiso para ejecutar el conjunto de pruebas. Para otros casos de prueba, se seleccionan las acciones correspondientes.

    Proporcione los valores de los recursos para cada una de las acciones seleccionadas. Por ejemplo, para la acción Conectar, proporcione el ID de cliente que utiliza su dispositivo para conectarse al punto de conexión de Device Advisor. Puede proporcionar varios valores separados por comas y anteponer valores con un carácter comodín (*). Por ejemplo, para dar permiso para publicar sobre cualquier tema que empiece por `MyTopic`, introduzca **`MyTopic*`** como valor del recurso.

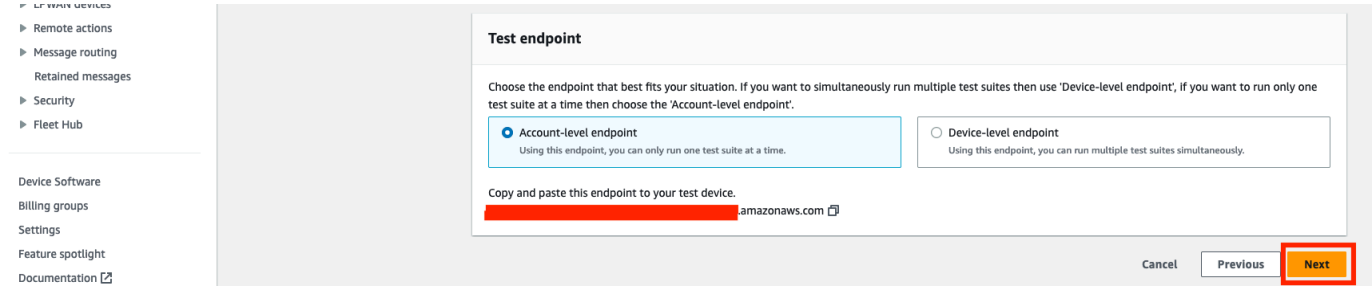


    Para utilizar un rol de dispositivo creado previamente en Configuración, elija Seleccionar un rol existente. A continuación, elija el rol de su dispositivo en Seleccionar rol.



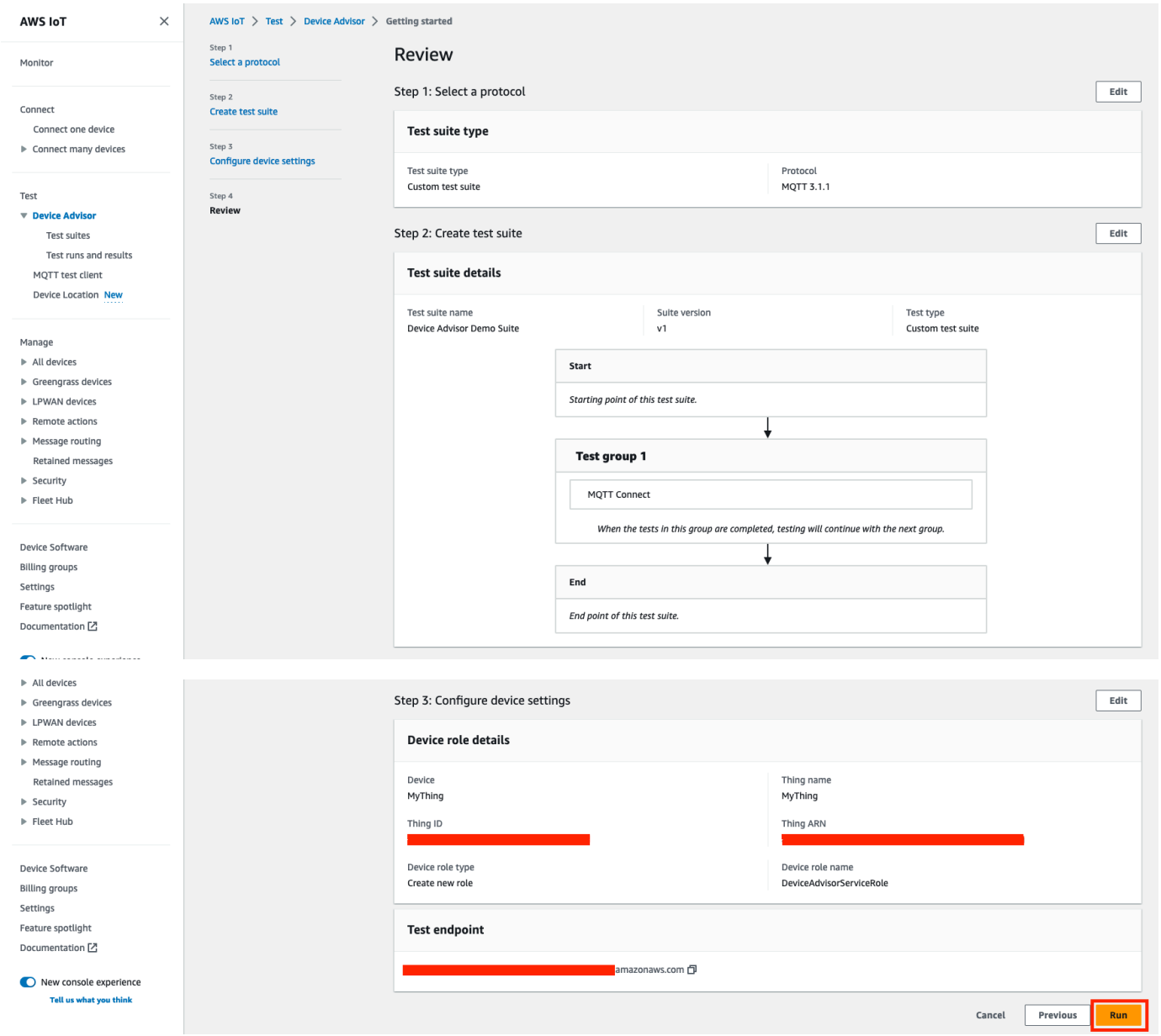
    Configure el rol de su dispositivo con una de las dos opciones proporcionadas y, a continuación, seleccione Siguiente.

12. En la sección Punto de enlace de prueba, seleccione el punto de conexión que mejor se adapte a su caso de uso. Para ejecutar varios conjuntos de pruebas simultáneamente con el mismo Cuenta de AWS, seleccione Dispositivo de punto final. Para ejecutar un conjunto de pruebas a la vez, seleccione Punto de enlace de nivel de cuenta.

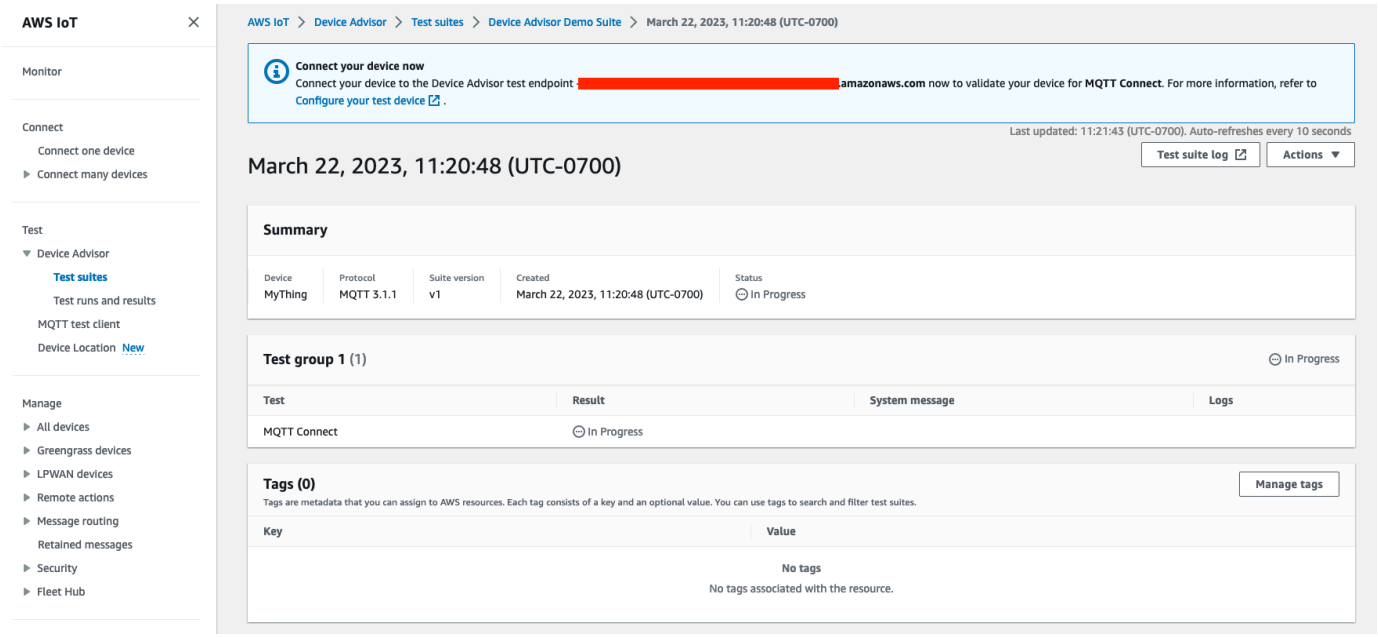


13. El paso 4 muestra un resumen del dispositivo de prueba seleccionado, el punto de conexión de prueba, el conjunto de pruebas y el rol del dispositivo de prueba configurado. Para realizar cambios en una sección, elija el botón Editar de la sección que desee editar. Una vez que haya confirmado la configuración de las pruebas, seleccione Ejecutar para crear el conjunto de pruebas y ejecutarlas.

> **Note**
>
> Para obtener los mejores resultados, puede conectar el dispositivo de prueba seleccionado al punto de conexión de prueba de Device Advisor antes de iniciar la ejecución del conjunto de pruebas. Le recomendamos que disponga de un mecanismo para que su dispositivo intente conectarse a nuestro punto de conexión de prueba cada cinco segundos durante un máximo de uno o dos minutos.



14. En el panel de navegación situado debajo de Prueba, elija Device Advisor y, a continuación, seleccione Ejecuciones de pruebas y resultados. Seleccione la ejecución de un conjunto de pruebas para ver los detalles de su ejecución y los registros.



15. Para acceder a los CloudWatch registros de Amazon de la suite, ejecuta:

    - Elija el registro del conjunto de pruebas para ver los CloudWatch registros de la ejecución del conjunto de pruebas.
    - Elija Registro de casos de prueba para cualquier caso de prueba para ver los CloudWatch registros específicos de cada caso de prueba.

16. Basándose en los resultados de las pruebas, solucione los problemas de su dispositivo hasta que todas las pruebas queden superadas.

Flujo de trabajo de Device Advisor

Este tutorial explica cómo crear un conjunto de pruebas personalizado y ejecutar pruebas con el dispositivo que desea probar en la consola. Una vez finalizadas las pruebas, podrá ver los resultados de las mismas y los registros detallados.

Requisitos previos

Antes de comenzar este tutorial, complete los pasos descritos en Configuración.

Crear una definición de conjunto de pruebas

En primer lugar, instala un AWS SDK.

Sintaxis de **`rootGroup`**

Un grupo raíz es una cadena JSON que especifica qué casos de prueba incluir en su conjunto de pruebas. También especifica cualquier configuración necesaria para esos casos de prueba. Utilice el grupo raíz para estructurar y ordenar su conjunto de pruebas en función de sus necesidades. La jerarquía de un conjunto de pruebas es:

```
test suite # test group(s) # test case(s)
```

Un conjunto de pruebas debe tener al menos un grupo de pruebas, y cada grupo de pruebas debe tener al menos un caso de prueba. Device Advisor ejecuta las pruebas en el orden en que usted define los grupos de pruebas y los casos de prueba.

Cada grupo raíz sigue esta estructura básica:

```
{
    "configuration": {  // for all tests in the test suite
        "": ""
    }
    "tests": [{
        "name": ""
        "configuration": {  // for all sub-groups in this test group
            "": ""
        },
        "tests": [{
            "name": ""
            "configuration": {  // for all test cases in this test group
                "": ""
            },
            "test": {
                "id": ""
                "version": ""
            }
        }]
    }]
}
```

En el grupo raíz, se define el conjunto de pruebas con los parámetros `name`, `configuration` y `tests` que contiene el grupo. El grupo `tests` contiene las definiciones de las pruebas individuales. Usted define cada prueba con `name`, `configuration` y un bloque `test` que define los casos de prueba para esa prueba. Por último, cada caso de prueba se define con `id` y `version`.

Para obtener información acerca de cómo utilizar los campos `"id"` y `"version"` para cada caso de prueba (bloque `test`), consulte Casos de prueba de Device Advisor. Esa sección también contiene información sobre los ajustes disponibles para `configuration`.

El siguiente bloque es un ejemplo de configuración de un grupo raíz. Esta configuración especifica los casos de prueba MQTT Connect (caso deseable) y Reintentos de retroceso exponencial de MQTT Connect, junto con las descripciones de los campos de configuración.

```
{
    "configuration": {},  // Suite-level configuration
    "tests": [            // Group definitions should be provided here
      {
        "name": "My_MQTT_Connect_Group",  // Group definition name
        "configuration": {}               // Group definition-level configuration,
        "tests": [                        // Test case definitions should be provided
 here
        {
            "name": "My_MQTT_Connect_Happy_Case",  // Test case definition name
            "configuration": {
                "EXECUTION_TIMEOUT": 300        // Test case definition-level
 configuration, in seconds
            },
            "test": {
                "id": "MQTT_Connect",              // test case id
                "version": "0.0.0"                 // test case version
            }
        },
        {
            "name": "My_MQTT_Connect_Jitter_Backoff_Retries",  // Test case definition
 name
            "configuration": {
                "EXECUTION_TIMEOUT": 600                // Test case definition-level
 configuration,  in seconds
            },
            "test": {
                "id": "MQTT_Connect_Jitter_Backoff_Retries",  // test case id
                "version": "0.0.0"                                  // test case version
            }
        }]
    }]
}
```

Debe proporcionar la configuración del grupo raíz cuando cree la definición del conjunto de pruebas. Guarde el parámetro `suiteDefinitionId` que se devuelve en el objeto de respuesta. Puede utilizar este ID para recuperar la información de definición de su conjunto de pruebas y ejecutar su conjunto de pruebas.

He aquí un ejemplo de SDK de Java:

```
response = iotDeviceAdvisorClient.createSuiteDefinition(
        CreateSuiteDefinitionRequest.builder()
            .suiteDefinitionConfiguration(SuiteDefinitionConfiguration.builder()
                .suiteDefinitionName("your-suite-definition-name")
                .devices(
                    DeviceUnderTest.builder()
                        .thingArn("your-test-device-thing-arn")
                        .certificateArn("your-test-device-certificate-arn")
                        .deviceRoleArn("your-device-role-arn") //if using SigV4 for
 MQTT over WebSocket
                        .build()
                )
                .rootGroup("your-root-group-configuration")
                .devicePermissionRoleArn("your-device-permission-role-arn")
                .protocol("MqttV3_1_1 || MqttV5 || MqttV3_1_1_OverWebSocket ||
 MqttV5_OverWebSocket")
                .build()
            )
            .build()
)
```

Obtener una definición del conjunto de pruebas

Después de crear la definición del conjunto de pruebas, recibirá el parámetro `suiteDefinitionId` en el objeto de respuesta de la operación de la API `CreateSuiteDefinition`.

Cuando la operación devuelva el parámetro `suiteDefinitionId`, podrá ver nuevos campos `id` dentro de cada grupo y la definición del caso de prueba dentro del grupo raíz. Puede usarlos IDs para ejecutar un subconjunto de la definición de su conjunto de pruebas.

Ejemplo de SDK de Java:

```
response = iotDeviceAdvisorClient.GetSuiteDefinition(
    GetSuiteDefinitionRequest.builder()
        .suiteDefinitionId("your-suite-definition-id")
```

```
        .build()
)
```

Obtener un punto de conexión de prueba

Utilice la operación de la API `GetEndpoint` para obtener el punto de conexión de prueba utilizado por su dispositivo. Seleccione el punto de conexión que mejor se adapte a su prueba. Para ejecutar simultáneamente varios conjuntos de pruebas, utilice el punto de conexión de nivel de dispositivo proporcionando un `thing ARN`, `certificate ARN` o un `device role ARN`. Para ejecutar un único conjunto de pruebas, no proporcione argumentos a la GetEndpoint operación para elegir el punto final a nivel de cuenta.

Ejemplo de SDK:

```
response = iotDeviceAdvisorClient.getEndpoint(GetEndpointRequest.builder()
.certificateArn("your-test-device-certificate-arn")
.thingArn("your-test-device-thing-arn")
.deviceRoleArn("your-device-role-arn") //if using SigV4 for MQTT over WebSocket

.build())
```

Iniciar la ejecución de un conjunto de pruebas

Después de crear una definición del conjunto de pruebas y configurar su dispositivo de prueba para que se conecte a su punto de conexión de prueba de Device Advisor, ejecute su conjunto de pruebas con la API `StartSuiteRun`.

Para los clientes de MQTT, utilice `certificateArn` o `thingArn` para ejecutar el conjunto de pruebas. Ambos están configurados, se utiliza el certificado si pertenece al objeto.

En el caso de MQTT en lugar de WebSocket cliente, utilice `deviceRoleArn` para ejecutar el conjunto de pruebas. Si el rol especificado es diferente del rol especificado en la definición del conjunto de pruebas, el rol especificado anula el rol definido.

Para `.parallelRun()`, utilice `true` si utiliza un punto de conexión de nivel de dispositivo para ejecutar varios conjuntos de pruebas en paralelo utilizando una Cuenta de AWS.

Ejemplo de SDK:

```
response = iotDeviceAdvisorClient.startSuiteRun(StartSuiteRunRequest.builder()
```

```
.suiteDefinitionId("your-suite-definition-id")
.suiteRunConfiguration(SuiteRunConfiguration.builder()
    .primaryDevice(DeviceUnderTest.builder()
        .certificateArn("your-test-device-certificate-arn")
        .thingArn("your-test-device-thing-arn")
        .deviceRoleArn("your-device-role-arn") //if using SigV4 for MQTT over WebSocket

        .build())
    .parallelRun(true | false)
    .build())
.build())
```

Guarde el parámetro `suiteRunId` de la respuesta. Se utilizará para recuperar los resultados de la ejecución de este conjunto de pruebas.

Obtener una ejecución del conjunto de pruebas

Después de iniciar la ejecución de un conjunto de pruebas, puede comprobar su progreso y sus resultados con la API `GetSuiteRun`.

Ejemplo de SDK:

```
// Using the SDK, call the GetSuiteRun API.

response = iotDeviceAdvisorClient.GetSuiteRun(
GetSuiteRunRequest.builder()
    .suiteDefinitionId("your-suite-definition-id")
    .suiteRunId("your-suite-run-id")
.build())
```

Detener la ejecución de un conjunto de pruebas

Para detener la ejecución de un conjunto de pruebas que aún esté en curso, puede llamar a la operación de la API `StopSuiteRun`. Tras llamar a la operación `StopSuiteRun`, el servicio inicia el proceso de limpieza. Mientras el servicio ejecuta el proceso de limpieza, el conjunto de pruebas ejecuta actualizaciones de estado a `Stopping`. El proceso de limpieza puede durar varios minutos. Una vez finalizado el proceso, el estado de ejecución del conjunto de pruebas se actualiza a `Stopped`. Después de que la ejecución de una prueba se haya detenido por completo, puede iniciar otra ejecución del conjunto de pruebas. Puede comprobar periódicamente el estado de ejecución del conjunto utilizando la operación de la API `GetSuiteRun`, como se muestra en la sección anterior.

Ejemplo de SDK:

```
// Using the SDK, call the StopSuiteRun API.

response = iotDeviceAdvisorClient.StopSuiteRun(
StopSuiteRun.builder()
    .suiteDefinitionId("your-suite-definition-id")
    .suiteRunId("your-suite-run-id")
.build())
```

Obtener un informe de cualificación para una ejecución correcta del conjunto de pruebas de cualificación

Si ejecuta un conjunto de pruebas de cualificación que finaliza correctamente, puede recuperar un informe de cualificación con la operación de la API `GetSuiteRunReport`. Utilice este informe de cualificación para cualificar su dispositivo con el programa de cualificación de AWS IoT Core . Para determinar si su conjunto de pruebas es un conjunto de pruebas de cualificación, compruebe si el parámetro `intendedForQualification` está ajustado en `true`. Después de llamar a la operación de la API `GetSuiteRunReport`, puede descargar el informe de la URL devuelta durante un máximo de 90 segundos. Si transcurren más de 90 segundos desde la última vez que llamó a la operación `GetSuiteRunReport`, vuelva a llamarla para recuperar una nueva URL válida.

Ejemplo de SDK:

```
// Using the SDK, call the getSuiteRunReport API.

response = iotDeviceAdvisorClient.getSuiteRunReport(
    GetSuiteRunReportRequest.builder()
        .suiteDefinitionId("your-suite-definition-id")
        .suiteRunId("your-suite-run-id")
        .build()
)
```

Flujo de trabajo detallado de la consola de Device Advisor

En este tutorial, creará un conjunto de pruebas personalizado y ejecutar pruebas con el dispositivo que desea probar en la consola. Una vez finalizadas las pruebas, podrá ver los resultados de las mismas y los registros detallados.

Tutoriales

- Requisitos previos
- Crear una definición de conjunto de pruebas
- Iniciar la ejecución de un conjunto de pruebas
- Detener la ejecución de un conjunto de pruebas (opcional)
- Ver los detalles y los registros de las ejecuciones del conjunto de pruebas
- Descarga un informe AWS IoT de calificación

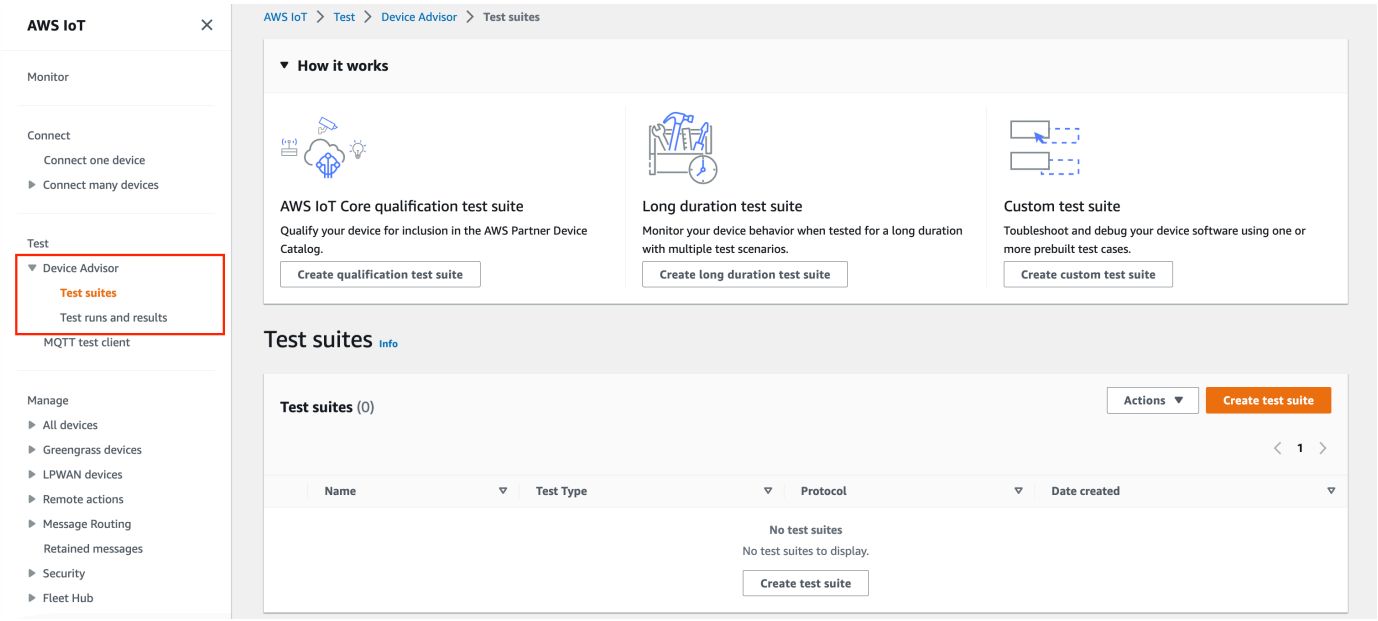
Requisitos previos

Para completar este tutorial, debe crear un objeto y un certificado.

Crear una definición de conjunto de pruebas

Cree un conjunto de pruebas para poder ejecutarlo en sus dispositivos y realizar la verificación.

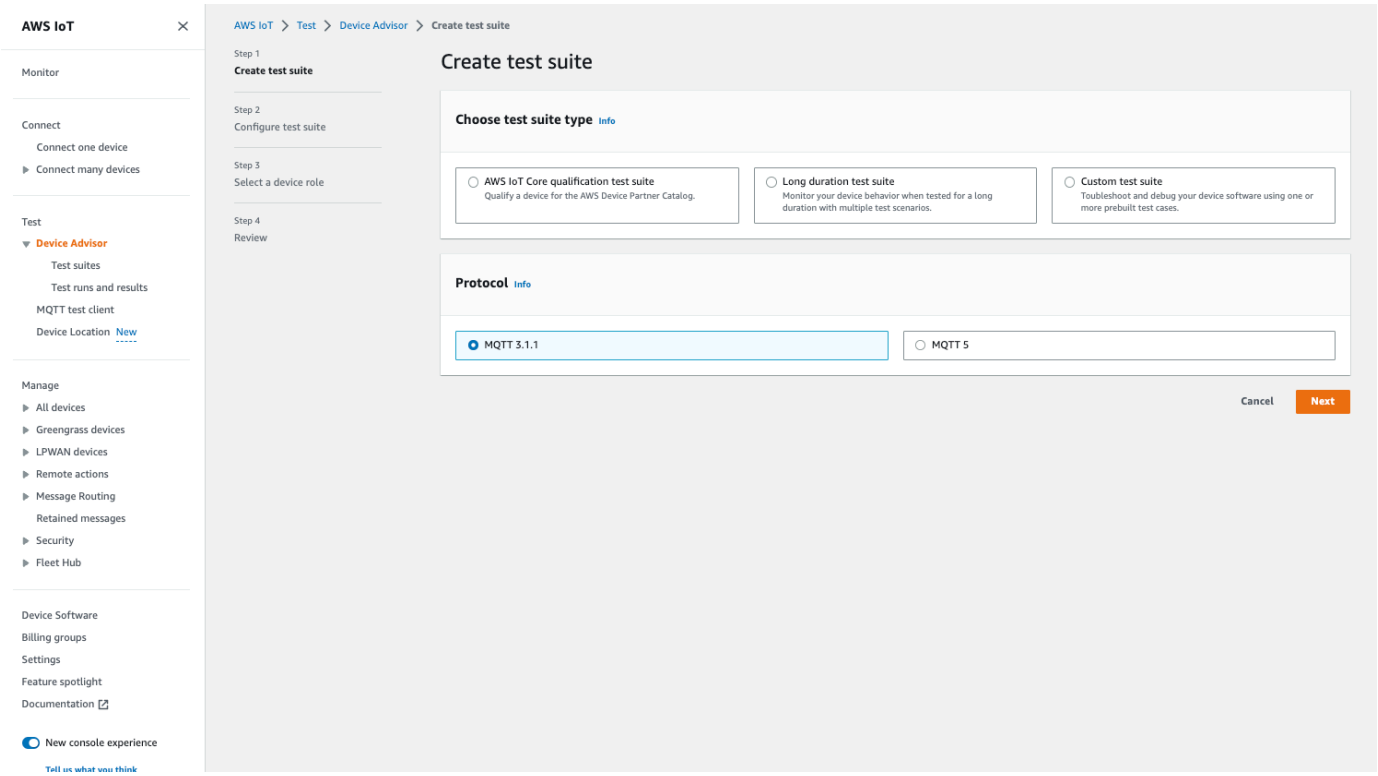
1. En la consola de AWS IoT, en el panel de navegación, expanda Prueba, Device Advisor y elija Conjuntos de pruebas.



   Seleccione Crear conjunto de pruebas.

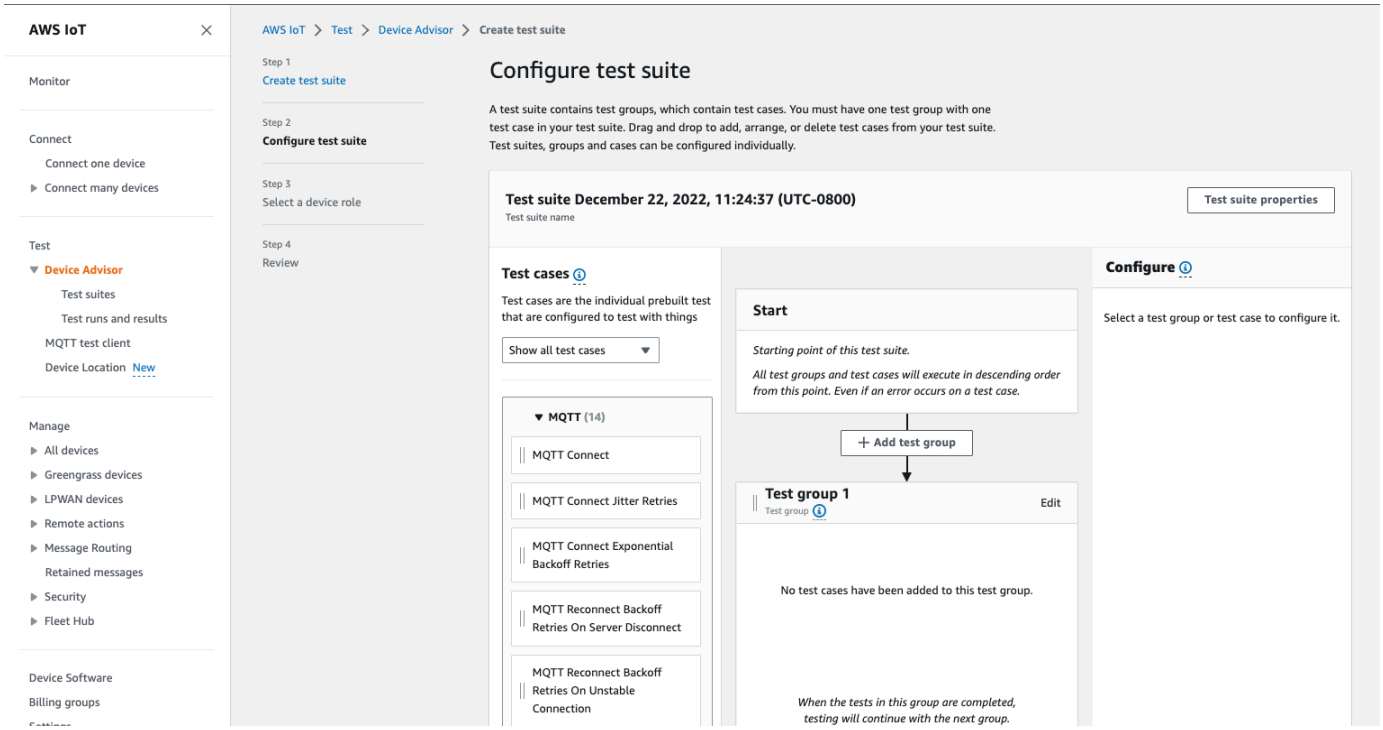
2. Seleccione `Use the AWS Qualification test suite` o `Create a new test suite`.

   Para el protocolo, elija MQTT 3.1.1 o MQTT 5.

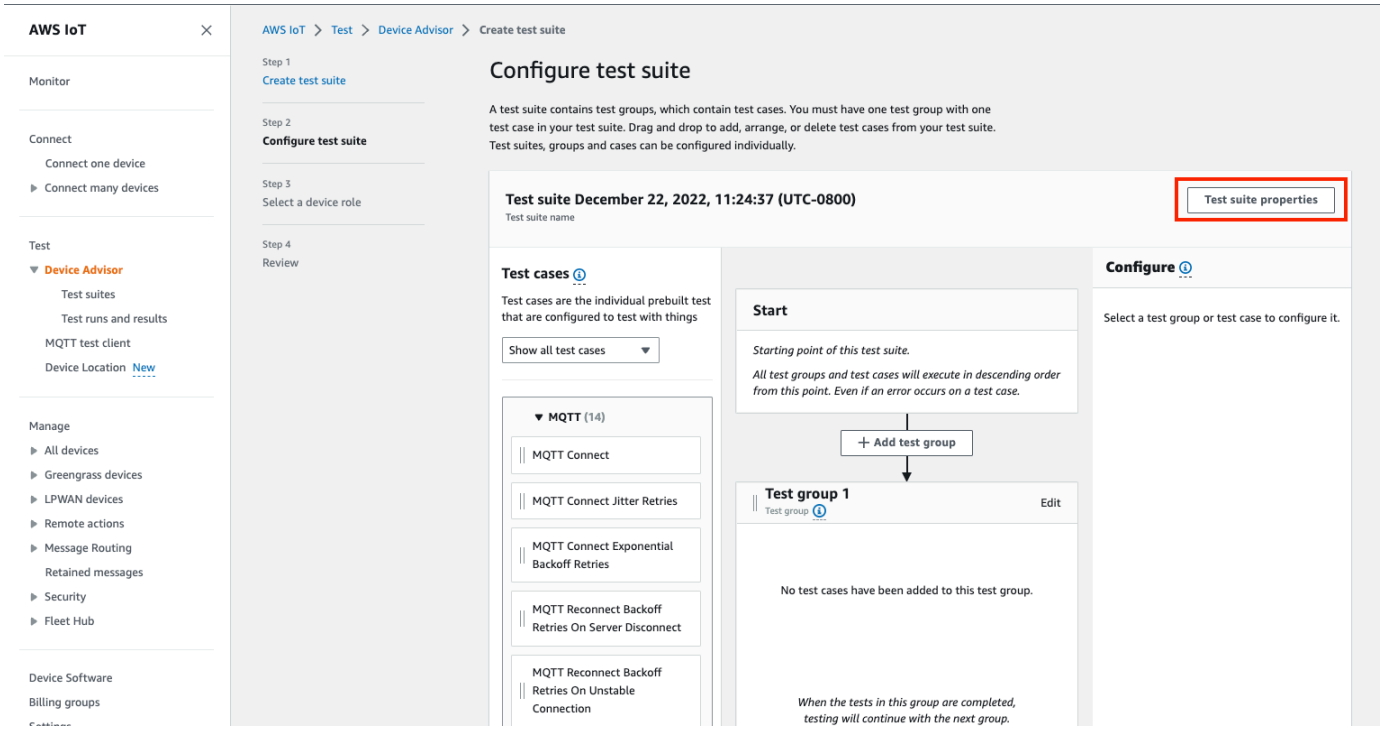


Seleccione si `Use the AWS Qualification test suite` desea reunir los requisitos e incluya su dispositivo en el catálogo de dispositivos AWS asociados. Al elegir esta opción, se preseleccionan los casos de prueba necesarios para que su dispositivo pueda participar en el programa de cualificación de AWS IoT Core . Los grupos de pruebas y los casos de prueba no pueden agregarse ni eliminarse. Aún tendrá que configurar las propiedades del conjunto de pruebas.

Seleccione `Create a new test suite` para crear y configurar un conjunto de pruebas personalizado. Recomendamos empezar con esta opción para las pruebas iniciales y la solución de problemas. Un conjunto de pruebas personalizado debe tener al menos un grupo de pruebas, y cada grupo de pruebas debe tener al menos un caso de prueba. Para el propósito de este tutorial, seleccionaremos esta opción y elegiremos Siguiente.



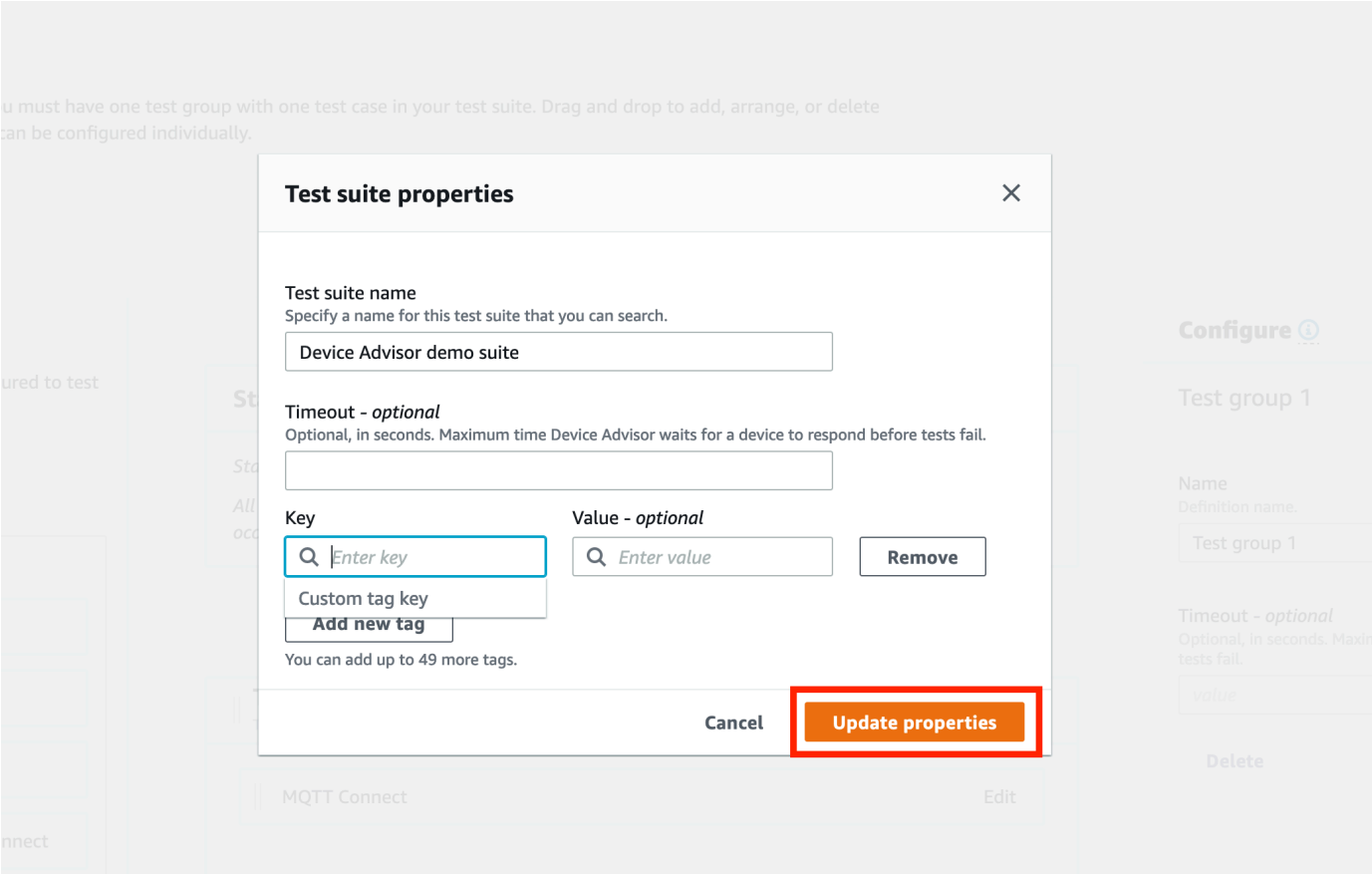
3. Seleccione Propiedades del conjunto de pruebas. Debe crear las propiedades del conjunto de pruebas cuando cree su conjunto de pruebas.



En Propiedades del conjunto de pruebas, complete lo siguiente:

- Nombre del conjunto de pruebas: puede crear el conjunto con un nombre personalizado.

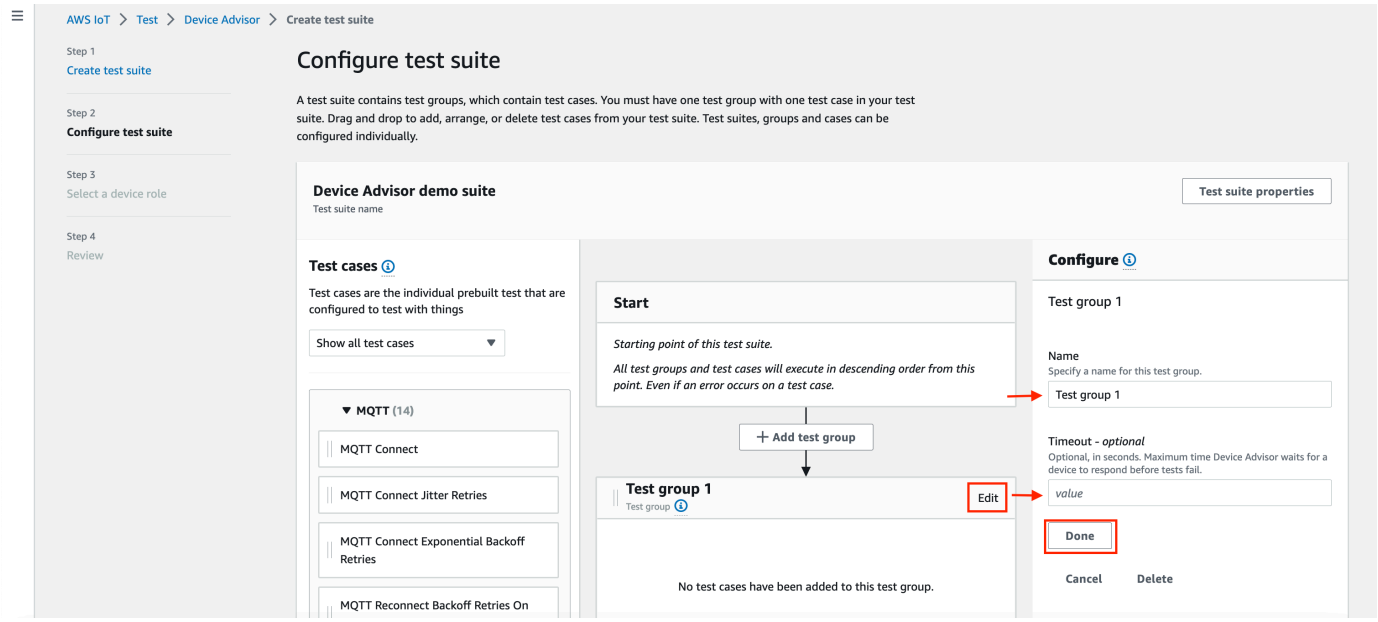
- Tiempo de espera (opcional): el tiempo de espera (en segundos) para cada caso de prueba en el conjunto de pruebas actual. Si no especifica un valor de tiempo de espera, se utilizará el valor predeterminado.
- Etiquetas (opcional): agregue etiquetas al conjunto de pruebas.



Cuando haya finalizado, seleccione Actualizar propiedades.

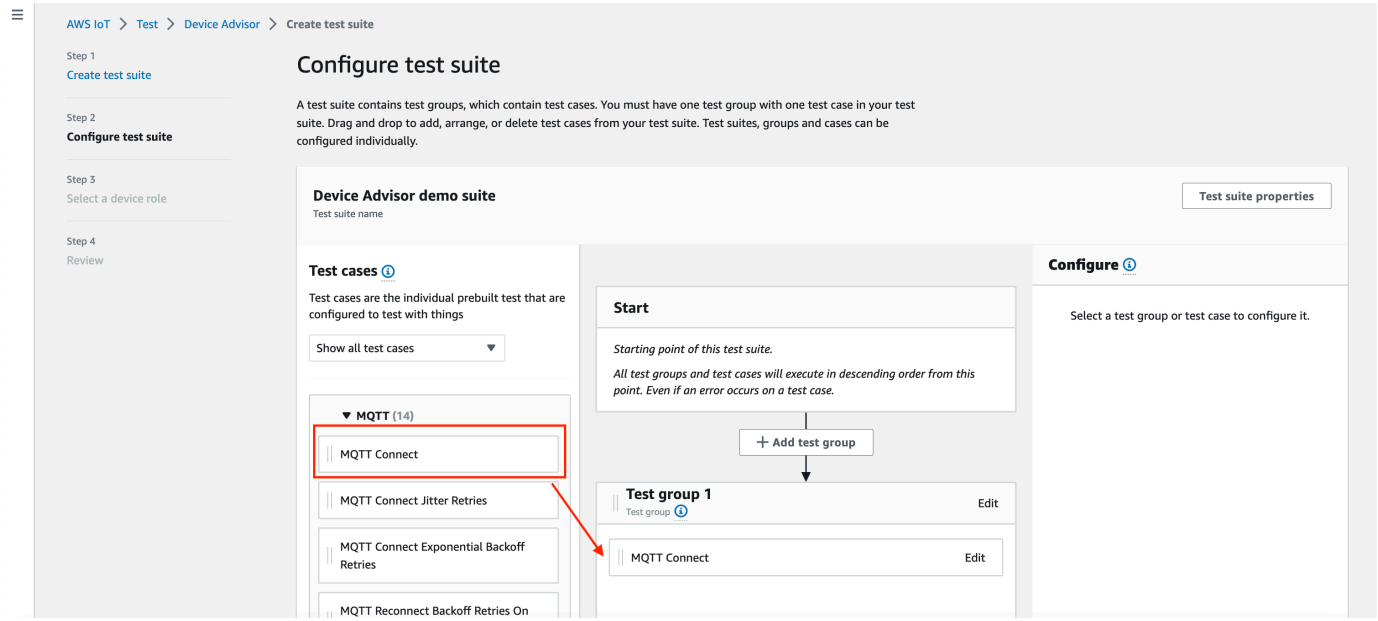
4. Para modificar la configuración a nivel de grupo, en `Test group 1`, seleccione Editar. A continuación, introduzca un nombre para asignar al grupo un nombre personalizado.

   Si lo desea, también puede introducir un valor de tiempo de espera en segundos en el grupo de prueba seleccionado. Si no especifica un valor de tiempo de espera, se utilizará el valor predeterminado.



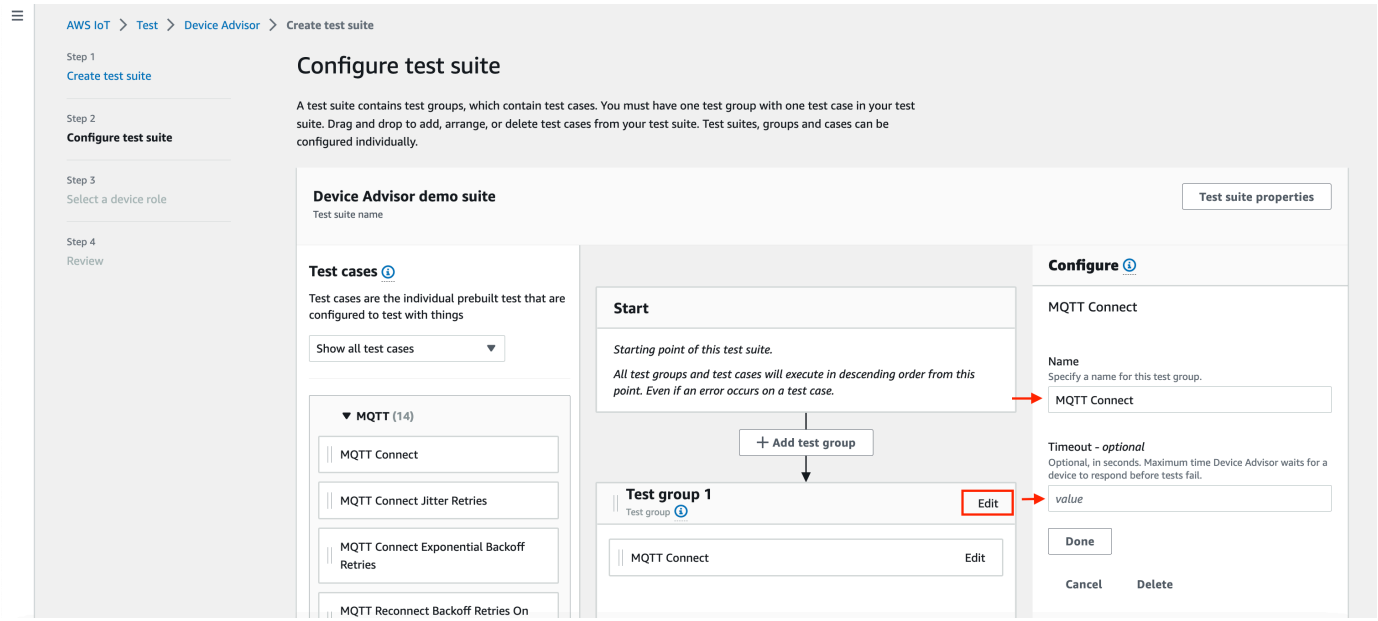
Seleccione Listo.

5. Arrastre uno de los casos de prueba disponibles desde Casos de prueba hasta el grupo de pruebas.



6. Para modificar la configuración de nivel de caso de prueba del caso de prueba que ha agregado a su grupo de pruebas, elija Editar. A continuación, introduzca un nombre para asignar al grupo un nombre personalizado.

Si lo desea, también puede introducir un valor de tiempo de espera en segundos en el grupo de prueba seleccionado. Si no especifica un valor de tiempo de espera, se utilizará el valor predeterminado.



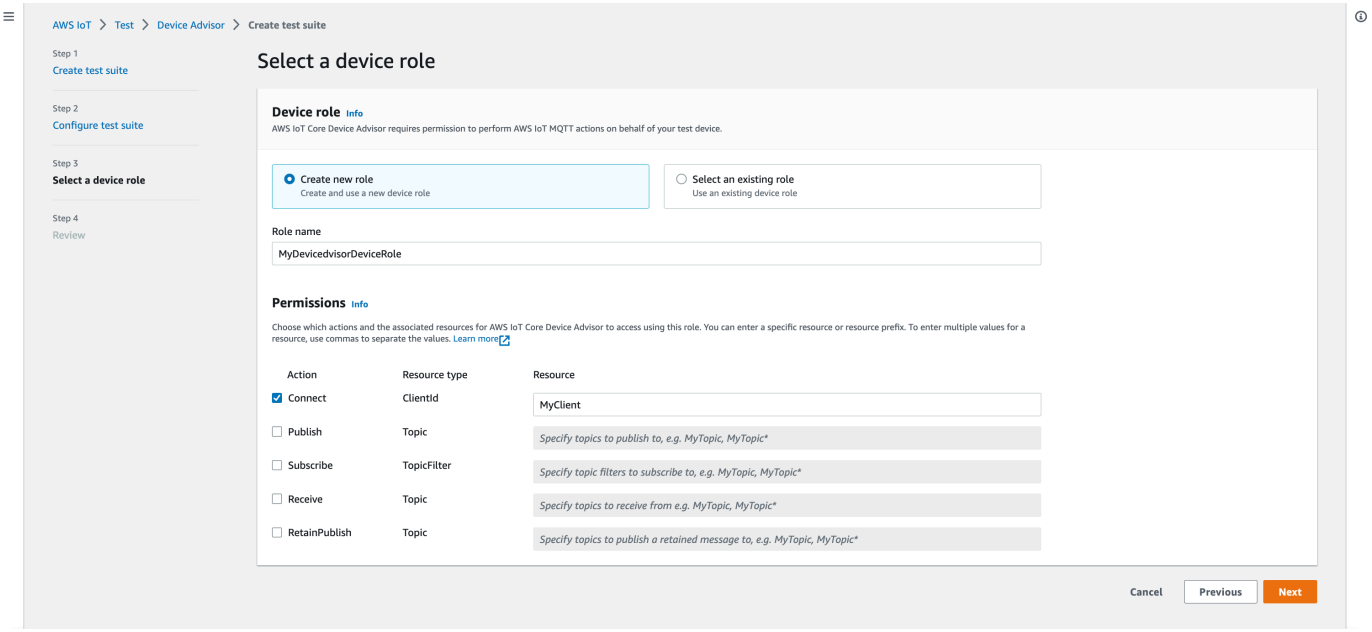
Seleccione Listo.

> **Note**
>
> Para agregar más grupos de pruebas al conjunto de pruebas, elija Agregar grupo de pruebas. Siga los pasos anteriores para crear y configurar más grupos de pruebas o para agregar más casos de prueba a uno o más grupos de pruebas. Los grupos de pruebas y los casos de prueba se pueden reordenar seleccionando y arrastrando un caso de prueba a la posición deseada. Device Advisor ejecuta las pruebas en el orden en que usted define los grupos de pruebas y los casos de prueba.

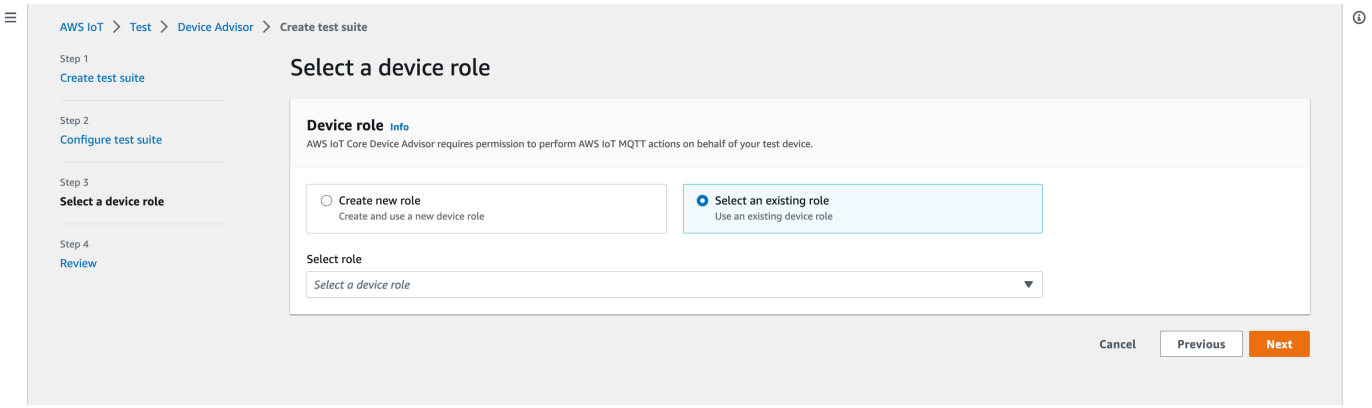
7. Elija Siguiente.
8. En el paso 3, configure una función de dispositivo que Device Advisor utilizará para realizar acciones de AWS IoT MQTT en nombre del dispositivo de prueba.

   Si seleccionó el caso de prueba de MQTT Connect solo en el paso 2, la acción Conectar se marcará automáticamente, ya que el rol del dispositivo requiere ese permiso para ejecutar este conjunto de pruebas. Si ha seleccionado otros casos de prueba, se comprobarán las acciones necesarias correspondientes. Asegúrese de proporcionar los valores de los recursos para cada

una de las acciones. Por ejemplo, para la acción Conectar, proporcione el ID de cliente con el que su dispositivo se conectará al punto de conexión de Device Advisor. Puede proporcionar varios valores mediante comas para separarlos, y también puede proporcionar valores de prefijo con un carácter comodín (*). Por ejemplo, para dar permiso para publicar sobre cualquier tema que empiece por `MyTopic`, puede introducir «`MyTopic*`» como valor del recurso.



Si ya ha creado un rol de dispositivo anteriormente y desea usarlo, seleccione Seleccionar un rol existente y elija su rol de dispositivo en Seleccionar rol.



Configure el rol de su dispositivo con una de las dos opciones proporcionadas y, a continuación, seleccione Siguiente.

9. En el paso 4, asegúrese de que la configuración proporcionada en cada uno de los pasos sea correcta. Para editar la configuración proporcionada para un paso en particular, elija Editar para el paso correspondiente.

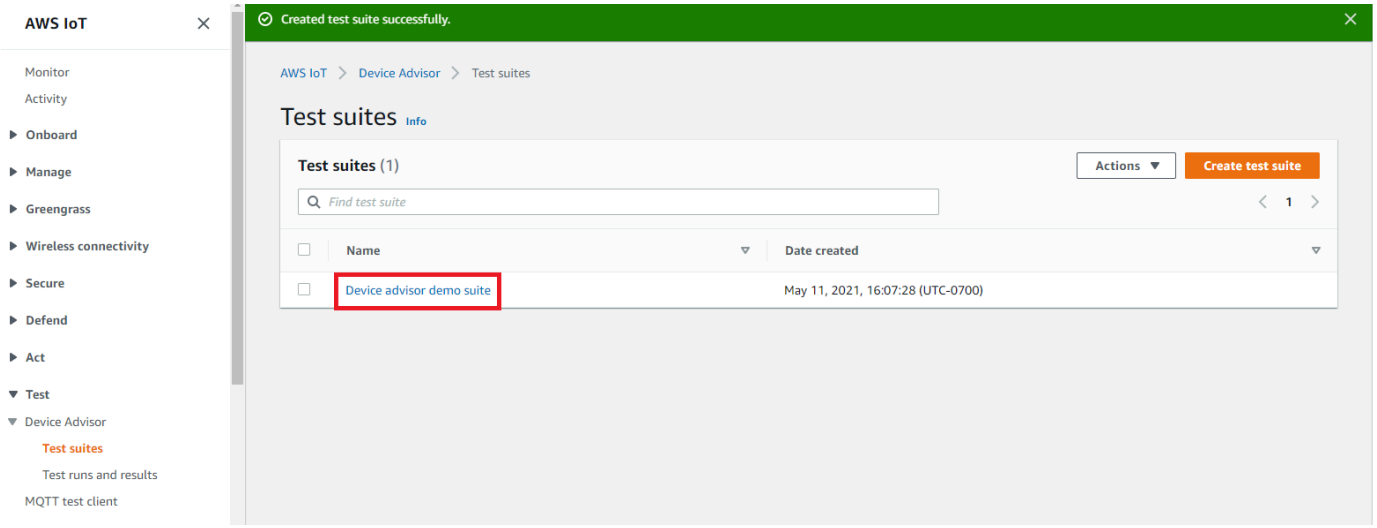
Después de verificar la configuración, elija Crear conjunto de pruebas.

El conjunto de pruebas debe crearse correctamente y se le redirigirá a la página Conjuntos de pruebas, donde podrá ver todos los conjuntos de pruebas que se han creado.

Si no se pudo crear el conjunto de pruebas, asegúrese de que el conjunto de pruebas, los grupos de pruebas, los casos de prueba y el rol del dispositivo se hayan configurado de acuerdo con las instrucciones anteriores.

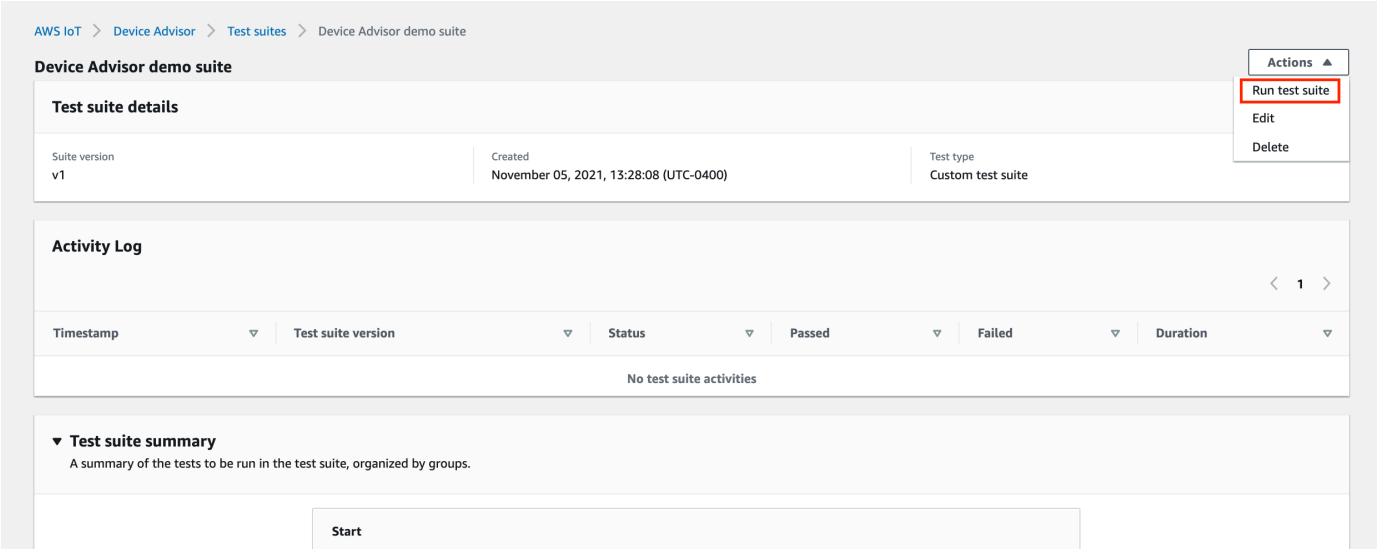
Iniciar la ejecución de un conjunto de pruebas

1. En la consola de AWS IoT, en el panel de navegación, expanda Prueba, Device Advisor y elija Conjuntos de pruebas.
2. Elija el conjunto de pruebas del que desee ver los detalles del conjunto de pruebas.



La página de detalles del conjunto de pruebas muestra toda la información relacionada con el conjunto de pruebas.

3. Seleccione Acciones y, a continuación, Ejecutar conjunto de pruebas.

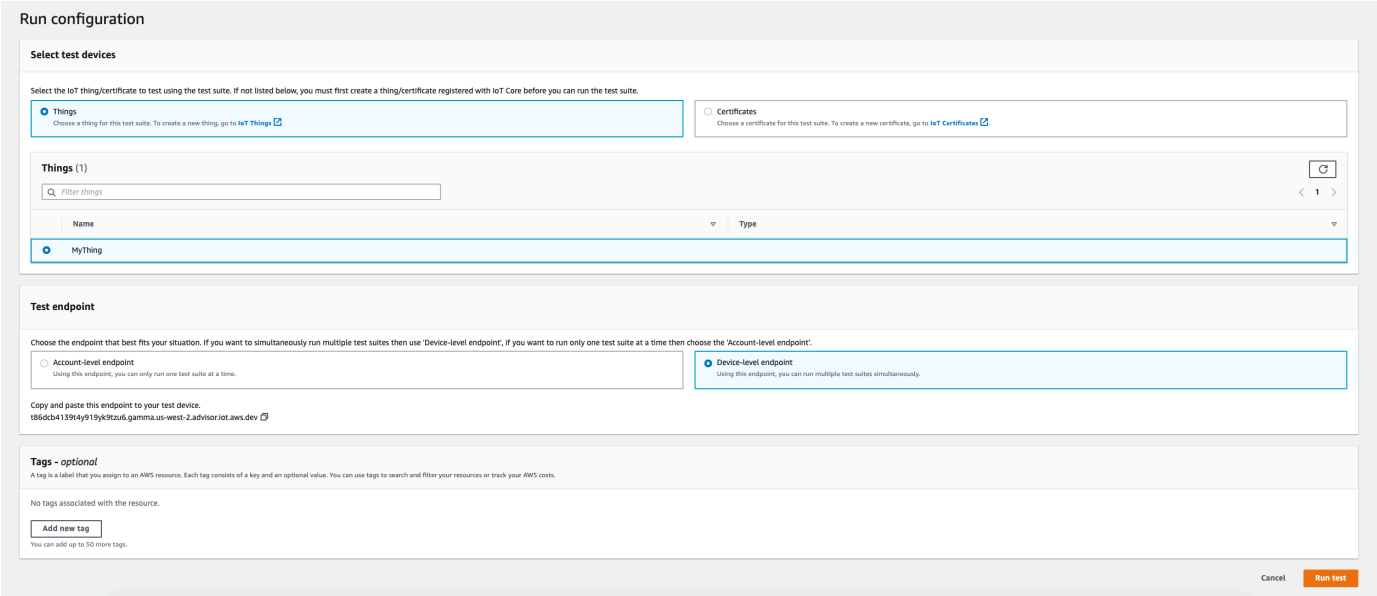


4. En Ejecutar la configuración, tendrá que seleccionar un elemento o AWS IoT un certificado para probarlo con Device Advisor. Si no tiene ningún objeto o certificado existente, cree primero AWS IoT Core los recursos.

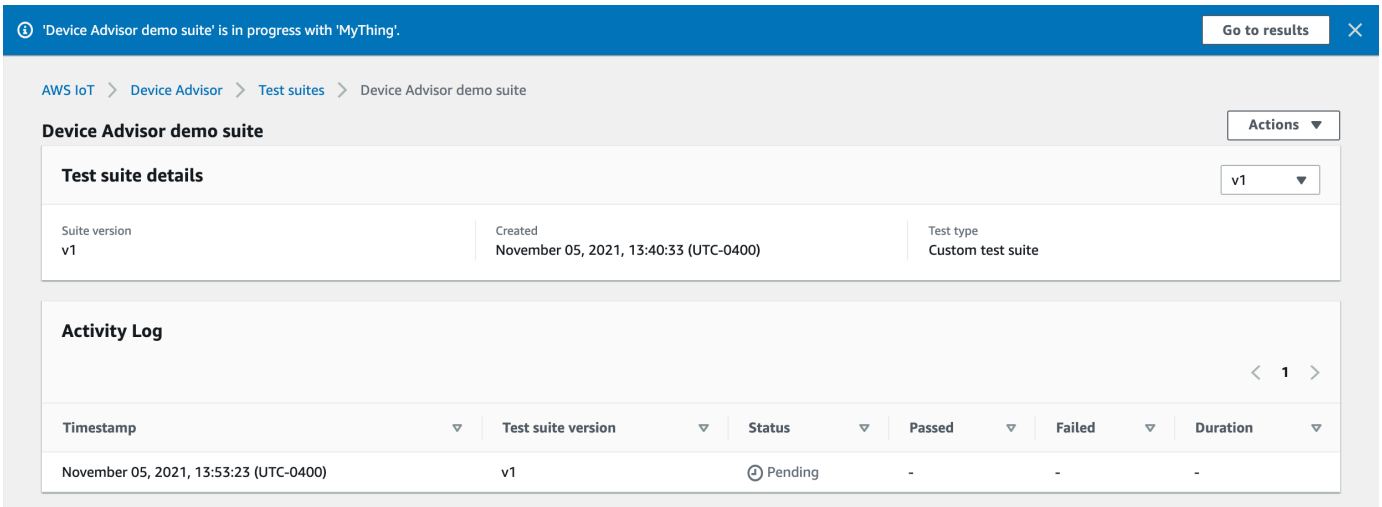
   En la sección Punto de enlace de prueba, seleccione el punto de conexión que mejor se adapte a su caso. Si planea ejecutar varios conjuntos de pruebas simultáneamente con la misma AWS cuenta en el futuro, seleccione Punto final a nivel de dispositivo. En caso contrario, si pretende ejecutar solo un conjunto de pruebas a la vez, seleccione Punto de enlace de nivel de cuenta.

   Configure su dispositivo de prueba con el punto de conexión de prueba de Device Advisor seleccionado.

   Tras seleccionar un objeto o un certificado y elegir un punto de conexión de Device Advisor, elija Ejecutar prueba.

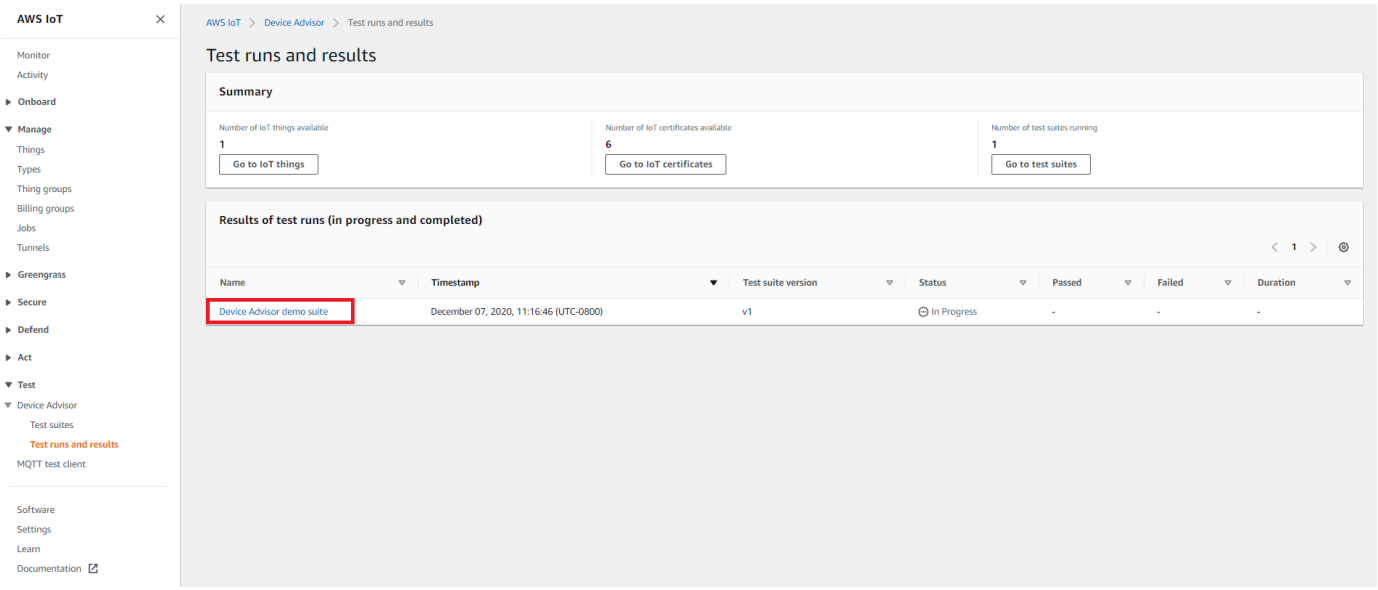


5. Seleccione Ir a los resultados en el banner superior para ver los detalles de la ejecución de la prueba.

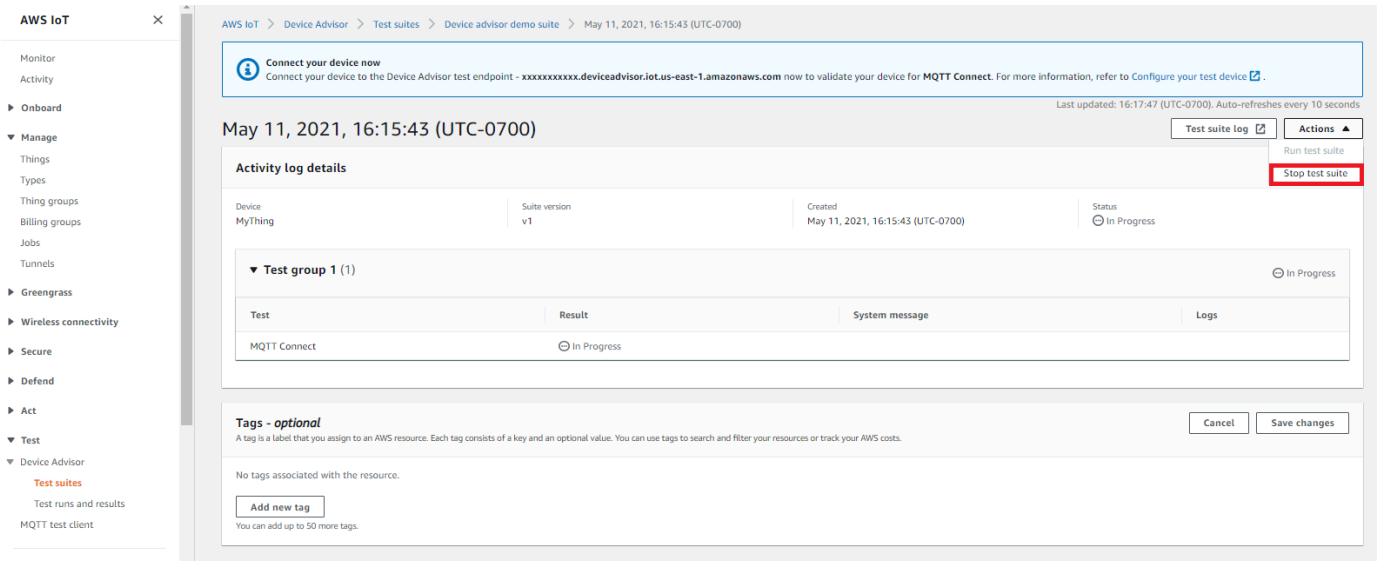


Detener la ejecución de un conjunto de pruebas (opcional)

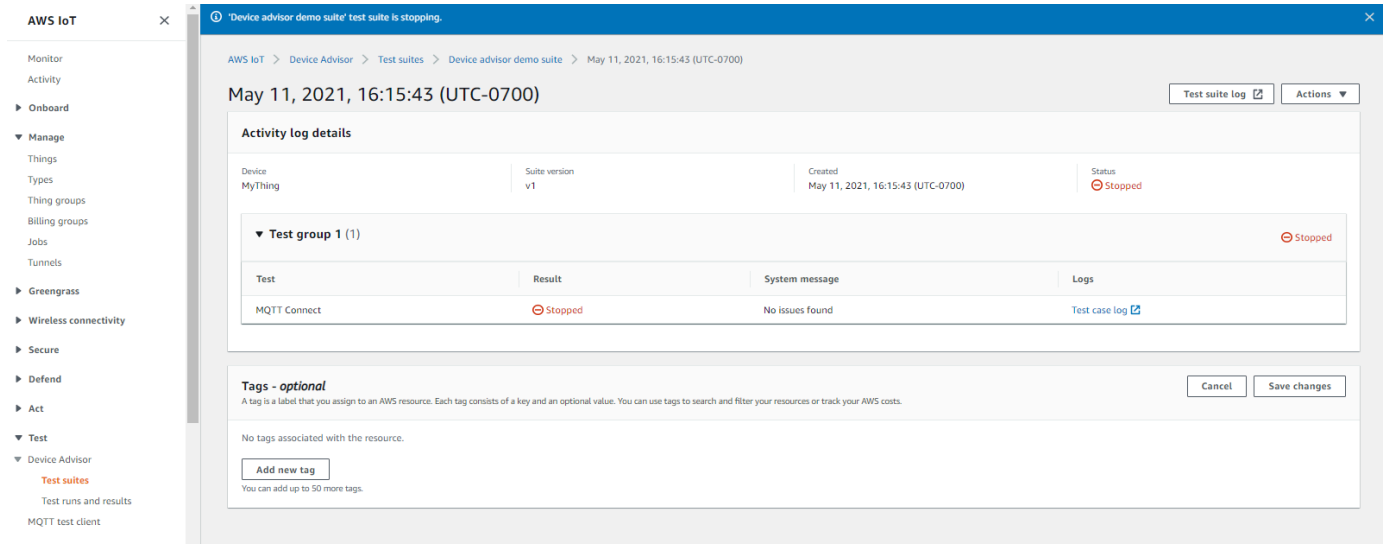
1. En la consola de AWS IoT, en el panel de navegación, expanda Prueba, Device Advisor y elija Ejecuciones de pruebas y resultados.
2. Elija el conjunto de pruebas en curso que quiera detener.



3. Seleccione Acciones y, a continuación, Detener conjunto de pruebas.



4. El proceso de limpieza tardará varios minutos en completarse. Mientras se ejecuta el proceso de limpieza, el estado de ejecución de la prueba será `STOPPING`. Espere a que se complete el proceso de limpieza y a que el estado del conjunto de pruebas cambie al mismo estado `STOPPED` antes de iniciar la ejecución de un nuevo conjunto.

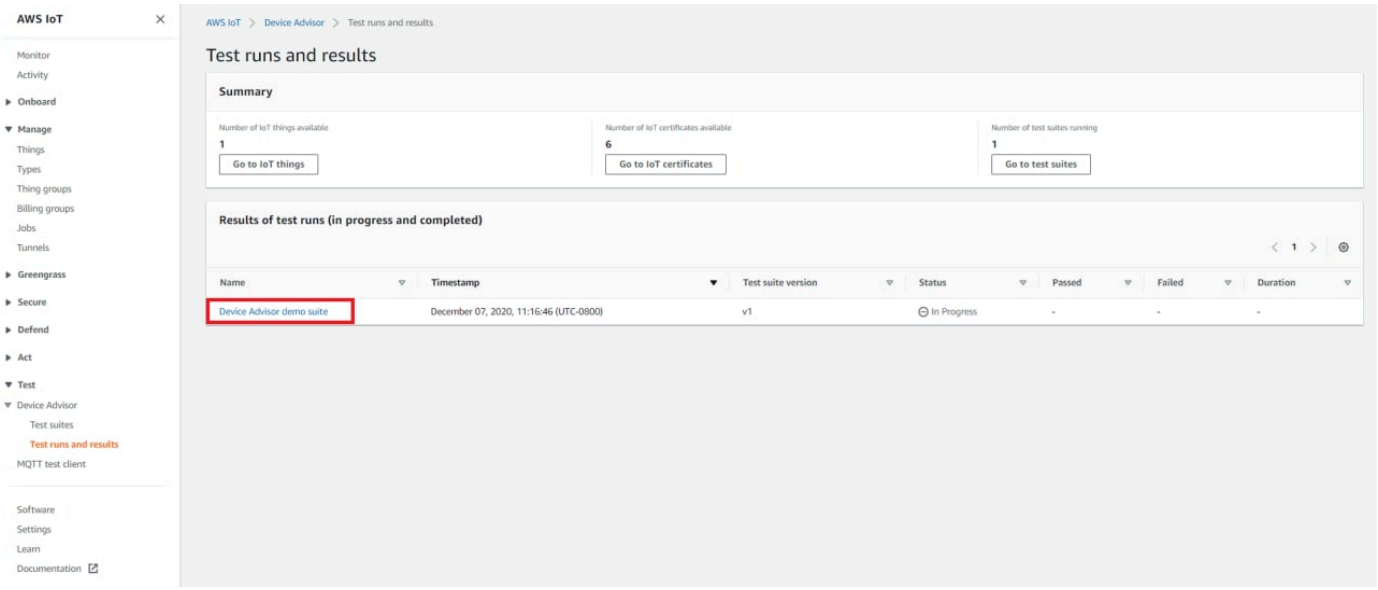


Ver los detalles y los registros de las ejecuciones del conjunto de pruebas

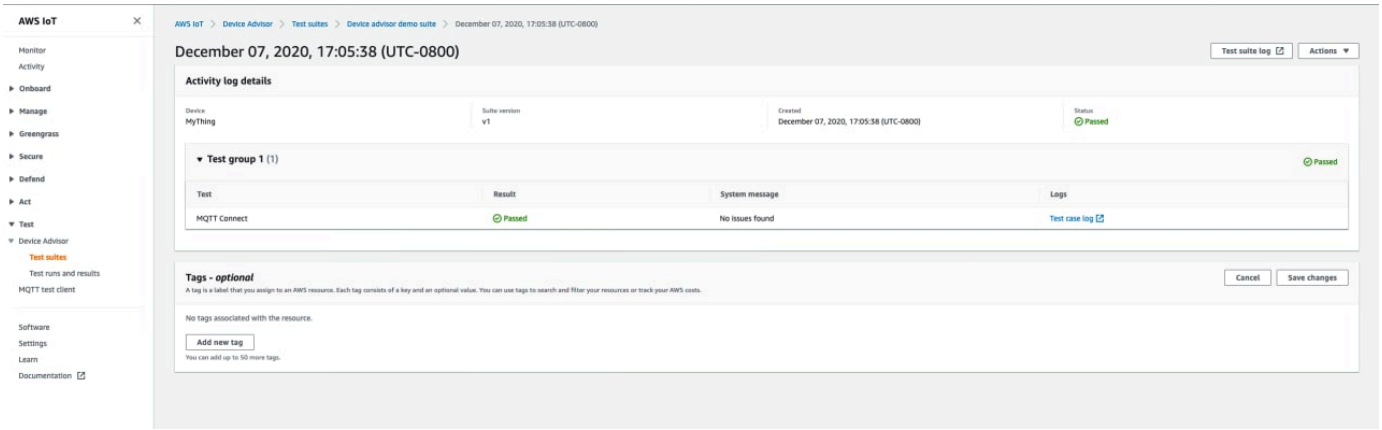
1. En la consola de AWS IoT, en el panel de navegación, expanda Prueba, Device Advisor y elija Ejecuciones de pruebas y resultados.

   Se abre esta página:

   - Número de objetos IoT
   - Número de certificados de IoT
   - Número de conjuntos de pruebas actualmente en ejecución
   - Todas las ejecuciones del conjunto de pruebas que se han creado
2. Elija el conjunto de pruebas del que desee ver los detalles de la ejecución y los registros.



La página de resumen de la ejecución muestra el estado de la ejecución actual del conjunto de pruebas. Esta página se actualiza automáticamente cada 10 segundos. Le recomendamos que disponga de un mecanismo para que su dispositivo intente conectarse a nuestro punto de conexión de prueba cada cinco segundos durante uno o dos minutos. A continuación, puede ejecutar varios casos de prueba en secuencia de forma automatizada.



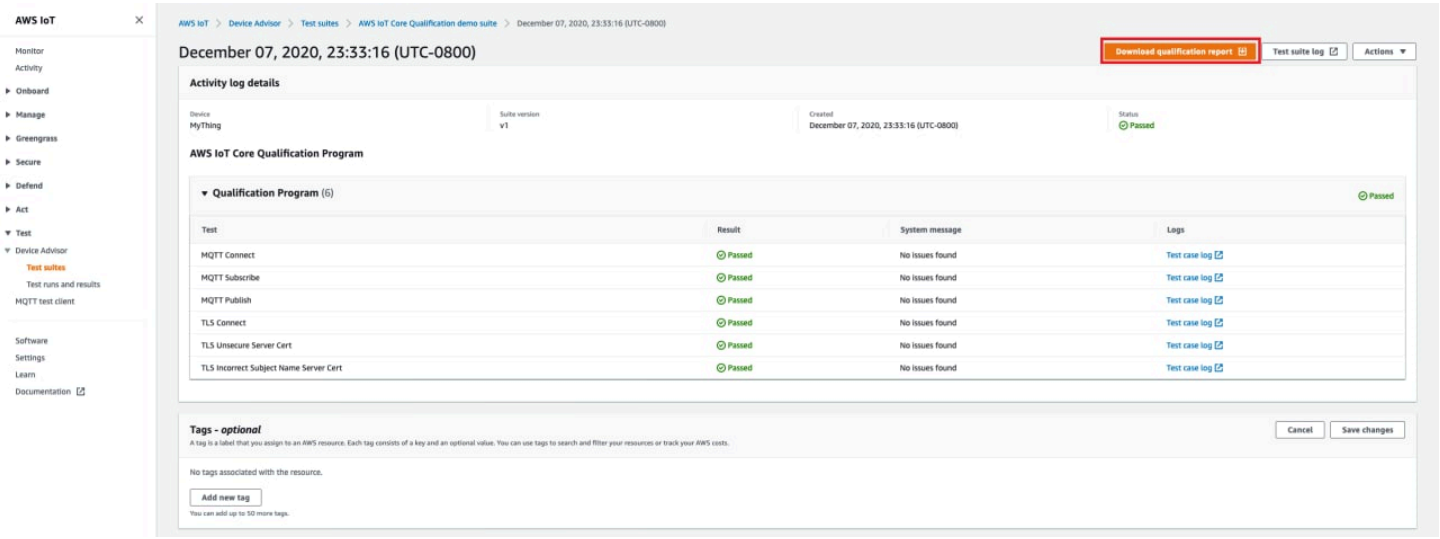
3. Para acceder a los CloudWatch registros de la ejecución del conjunto de pruebas, elija el registro del conjunto de pruebas.

   Para acceder a CloudWatch los registros de cualquier caso de prueba, selecciona Registro de casos de prueba.

4. Basándose en los resultados de las pruebas, solucione los problemas de su dispositivo hasta que todas las pruebas queden superadas.

Descarga un informe AWS IoT de calificación

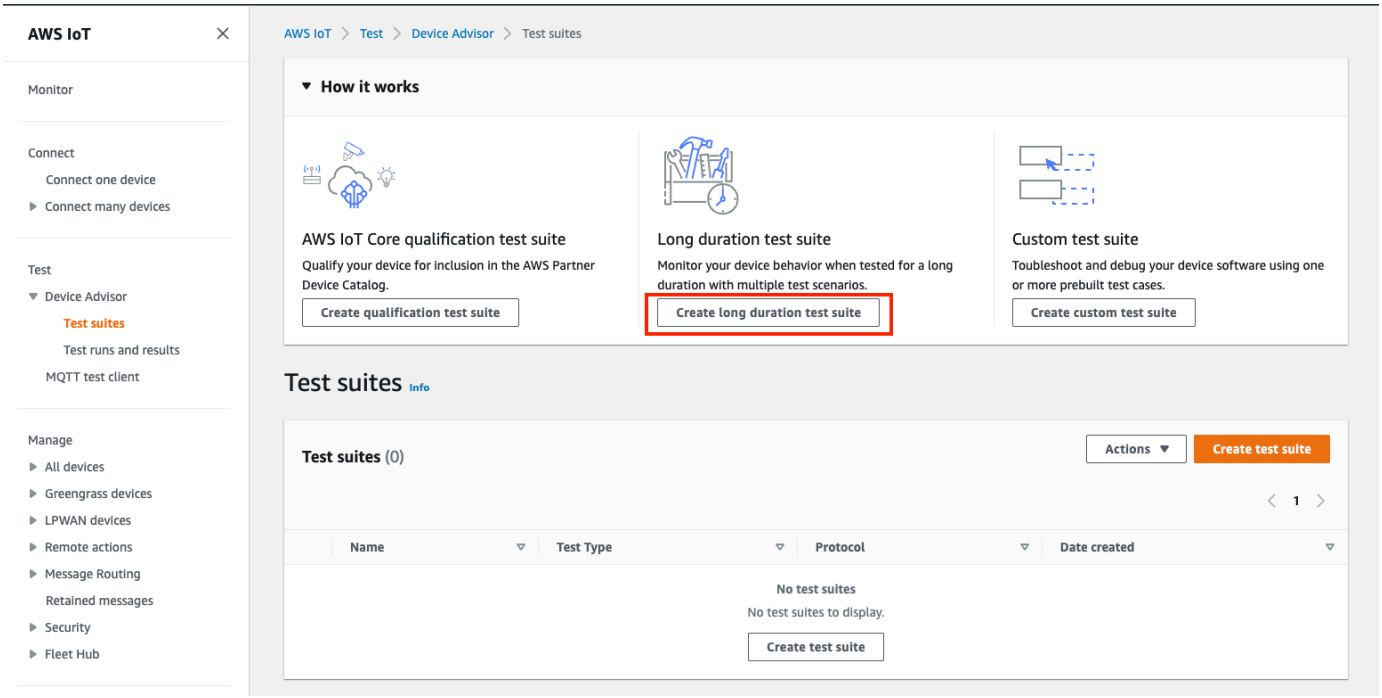
Si ha elegido la opción Utilizar el conjunto de pruebas de AWS IoT cualificación al crear un conjunto de pruebas y ha podido ejecutar un conjunto de pruebas de cualificación, puede descargar un informe de cualificación seleccionando Descargar el informe de cualificación en la página de resumen de la ejecución de las pruebas.



Flujo de trabajo de la consola de pruebas de larga duración

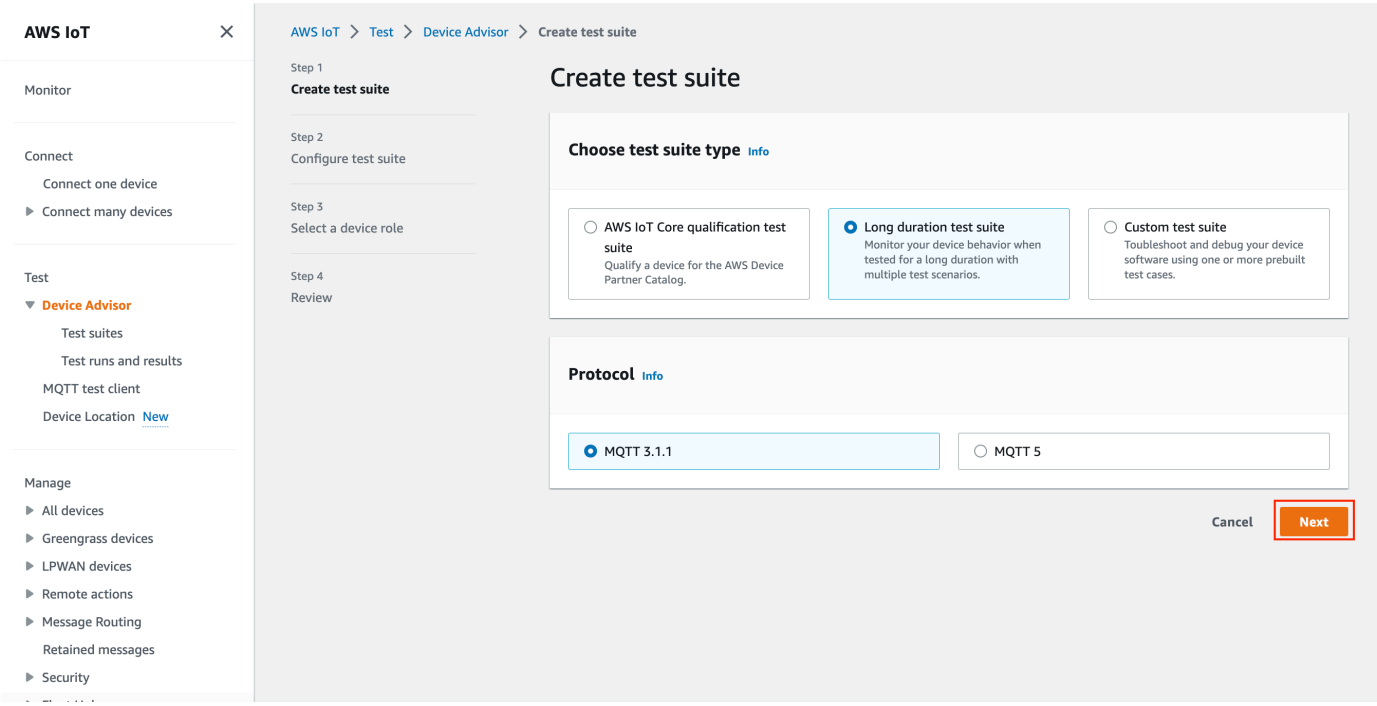
Este tutorial le ayuda a empezar con las pruebas de larga duración en Device Advisor mediante la consola. Para completar el tutorial, siga los pasos que se indican en Configuración.

1. En el panel de navegación de la consola de AWS IoT, expanda Prueba, luego Device Advisor y, por último, Conjuntos de pruebas. En la página, seleccione Crear conjunto de pruebas de larga duración.



2. En la página Crear conjunto de pruebas, seleccione Conjunto de pruebas de larga duración y elija Siguiente.

   Para el protocolo, elija MQTT 3.1.1 o MQTT 5.



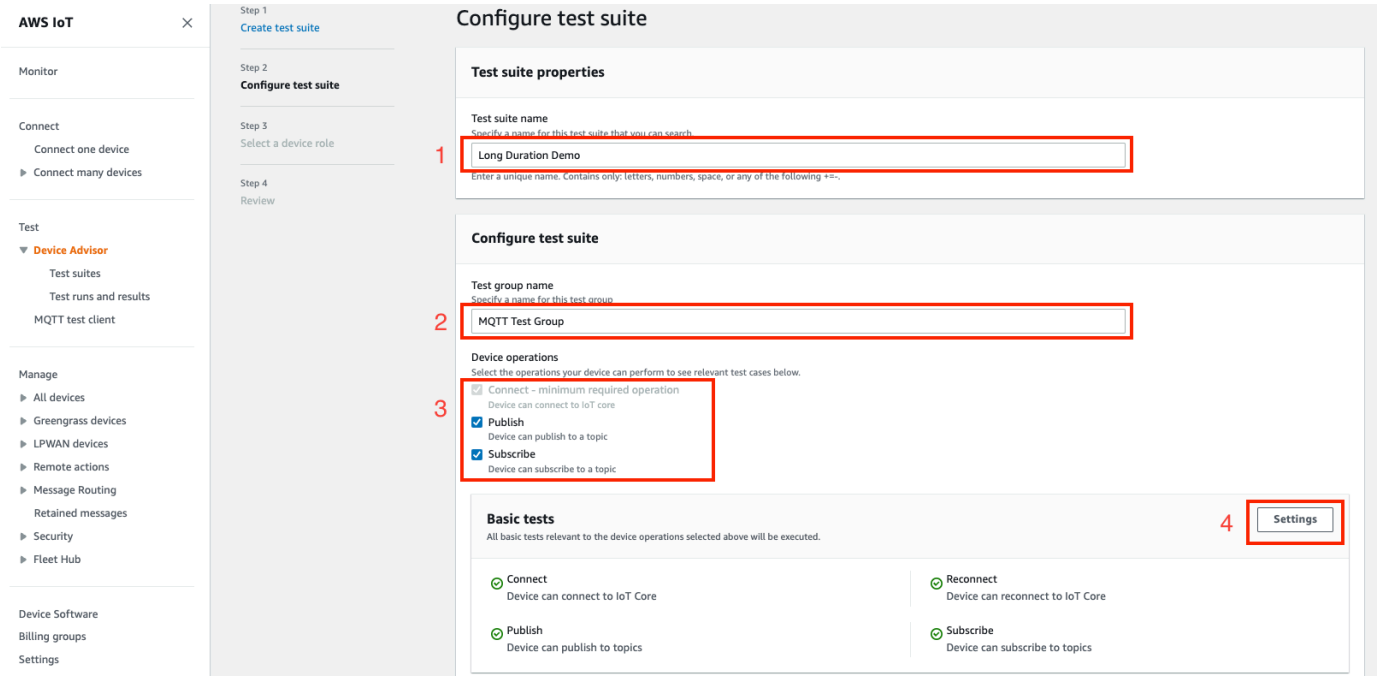
3. Haga lo siguiente en la página Configurar conjunto de pruebas:

   a. Actualice el campo Nombre del conjunto de pruebas.

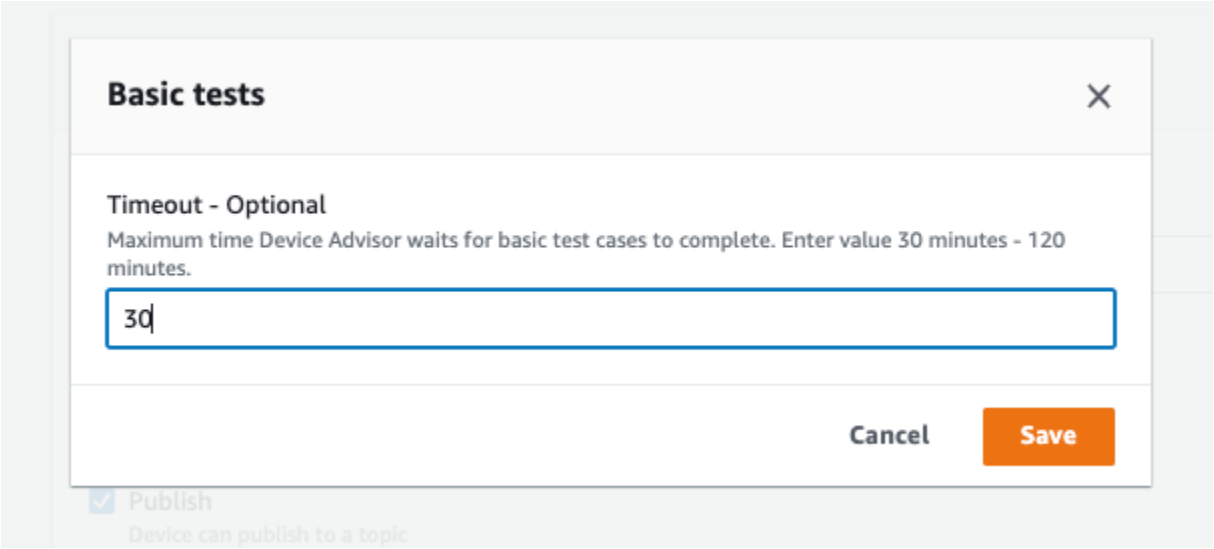
b. Actualice el campo Nombre del grupo de pruebas.

c. Elija las operaciones del dispositivo que el dispositivo puede realizar. Esto seleccionará las pruebas que se van a ejecutar.

d. Seleccione la opción Configuración.



4. (Opcional) Introduzca el tiempo máximo que Device Advisor debe esperar hasta que se completen las pruebas básicas. Seleccione Guardar.



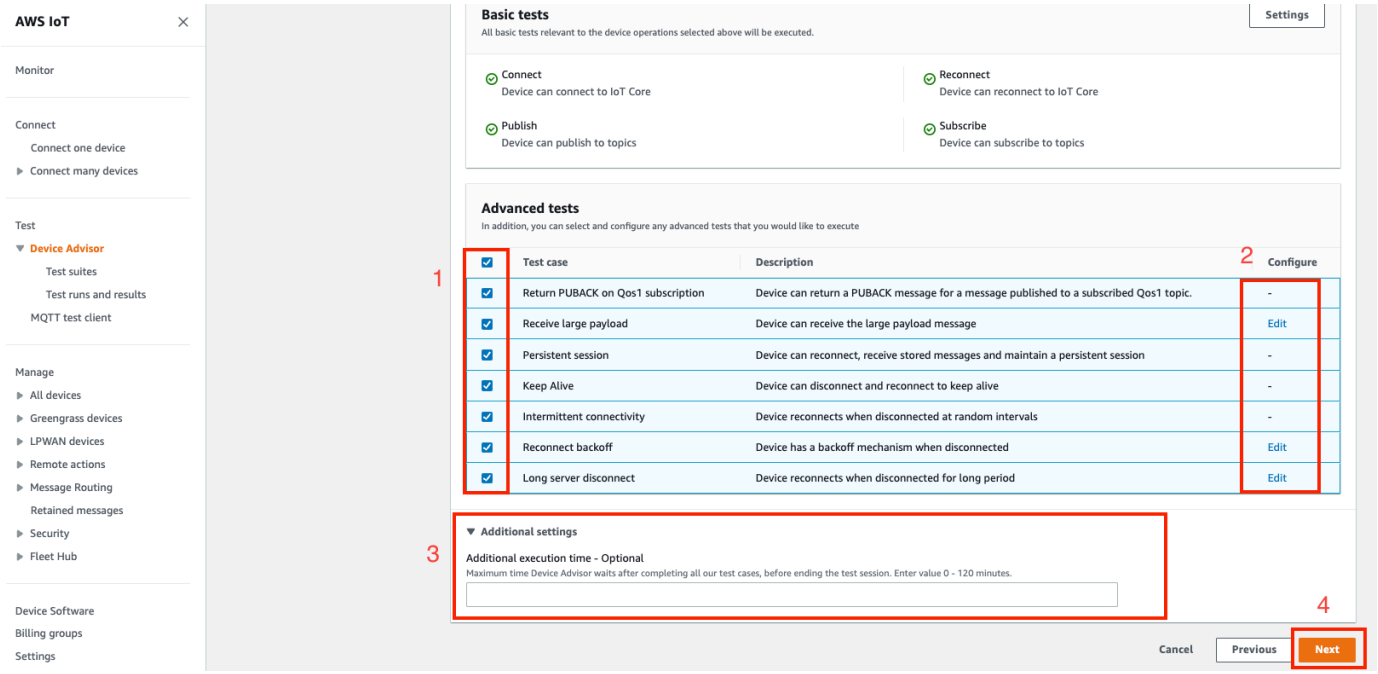
5. Realice lo siguiente en las secciones Pruebas avanzadas y Configuración adicional.

a. Marque o desmarque las pruebas avanzadas que quiera ejecutar como parte de esta prueba.

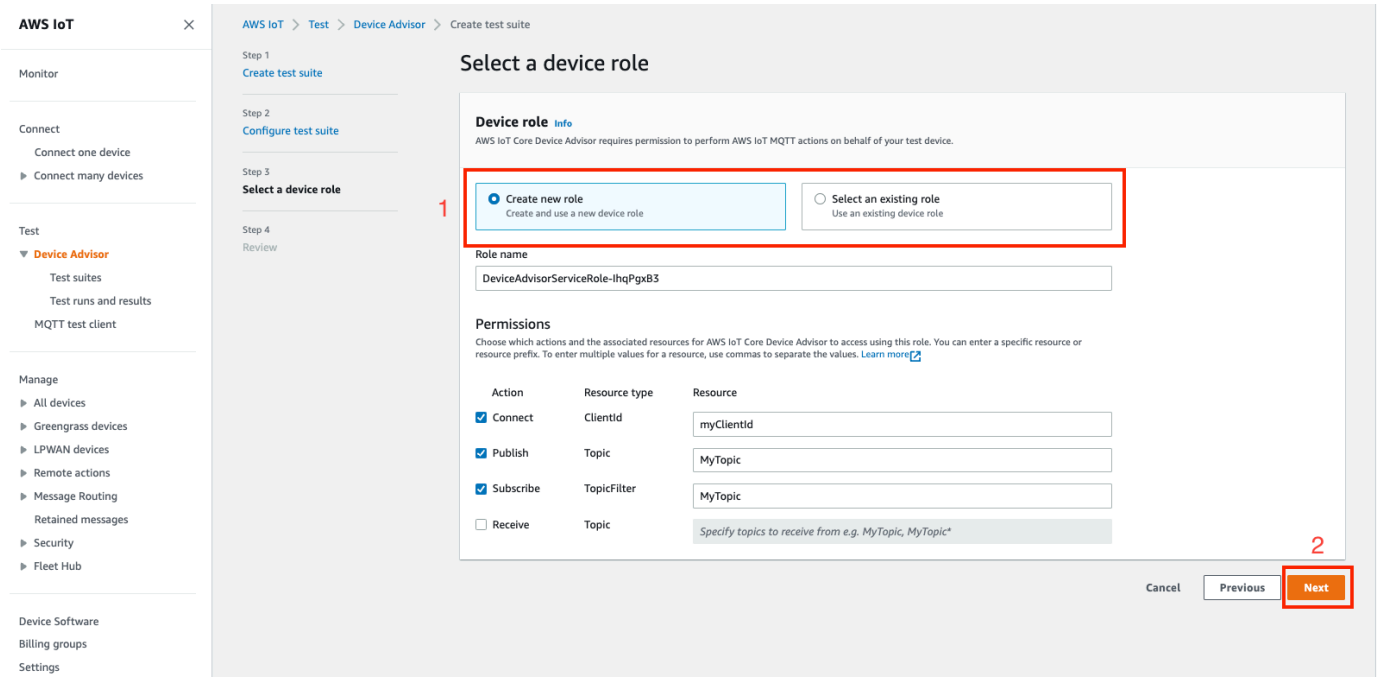
b. Edite las configuraciones de las pruebas cuando proceda.

c. Configure el tiempo de ejecución adicional en la sección Configuración adicional.

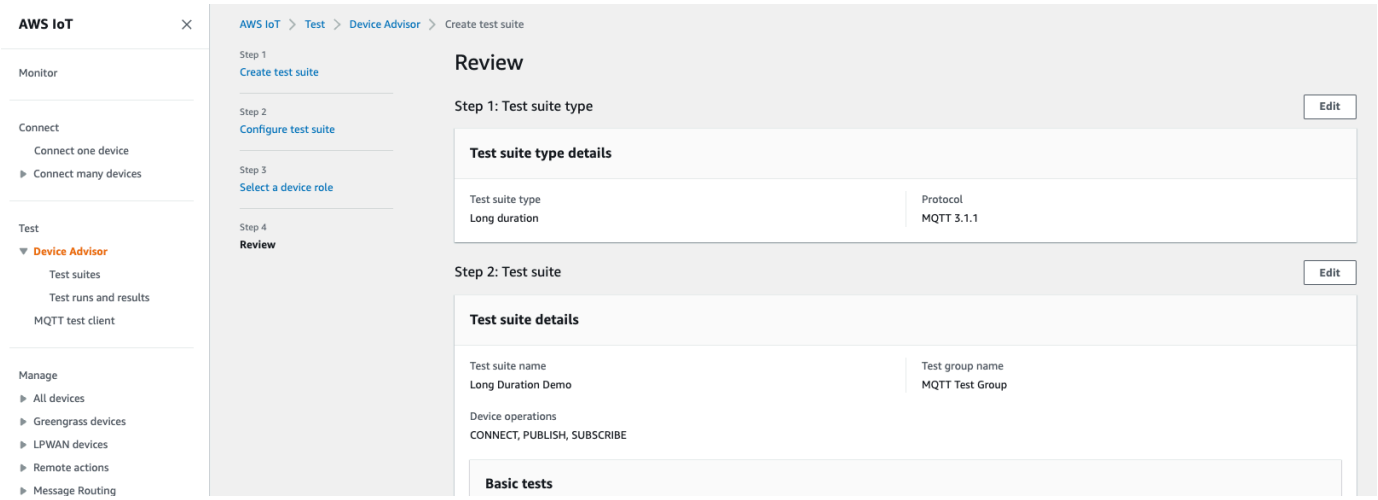
d. Elija Siguiente para ir al siguiente paso.

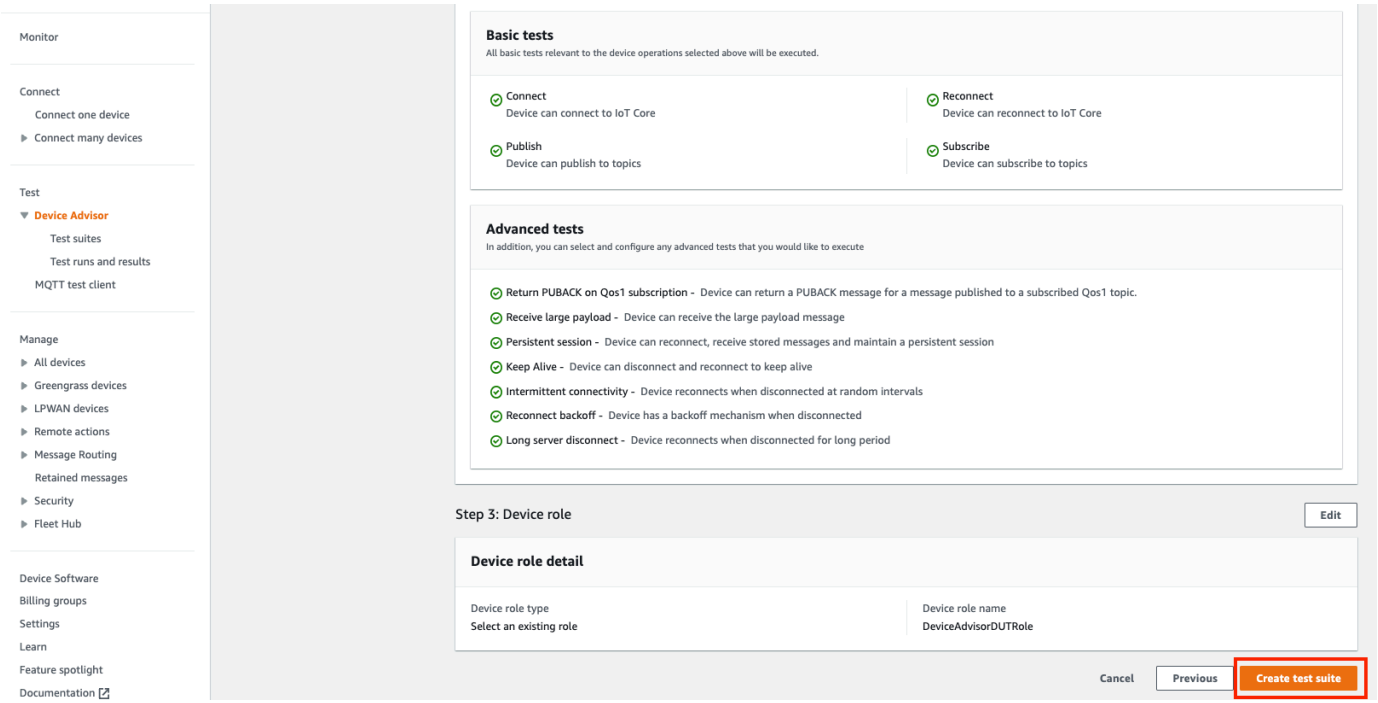


6. En este paso, cree un nuevo rol o seleccione un rol existente. Para obtener más información, consulte Cree un rol de IAM para utilizarlo como su rol de dispositivo.

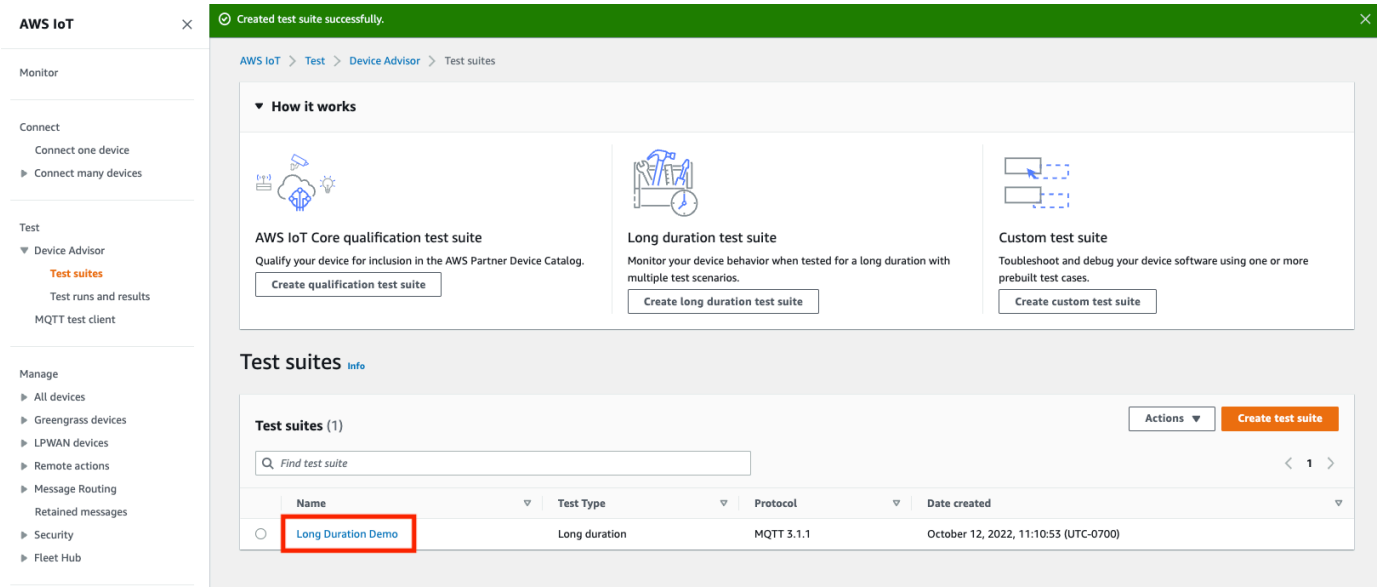


7. Revise todas las configuraciones creadas hasta este paso y seleccione Crear conjunto de pruebas.

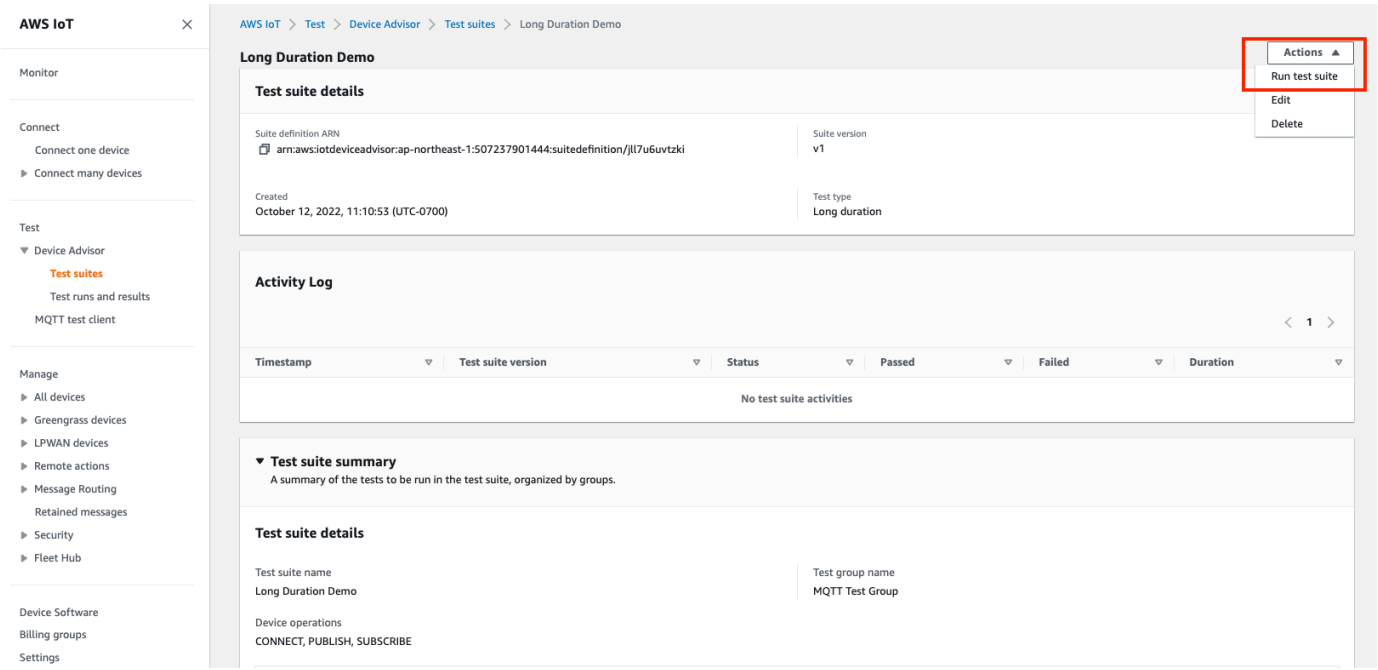




8. El conjunto de pruebas creado se encuentra en la sección Conjuntos de pruebas. Seleccione el conjunto para ver detalles.



9. Para ejecutar el conjunto de pruebas creado, seleccione Acciones y, a continuación, Ejecutar conjunto de pruebas.

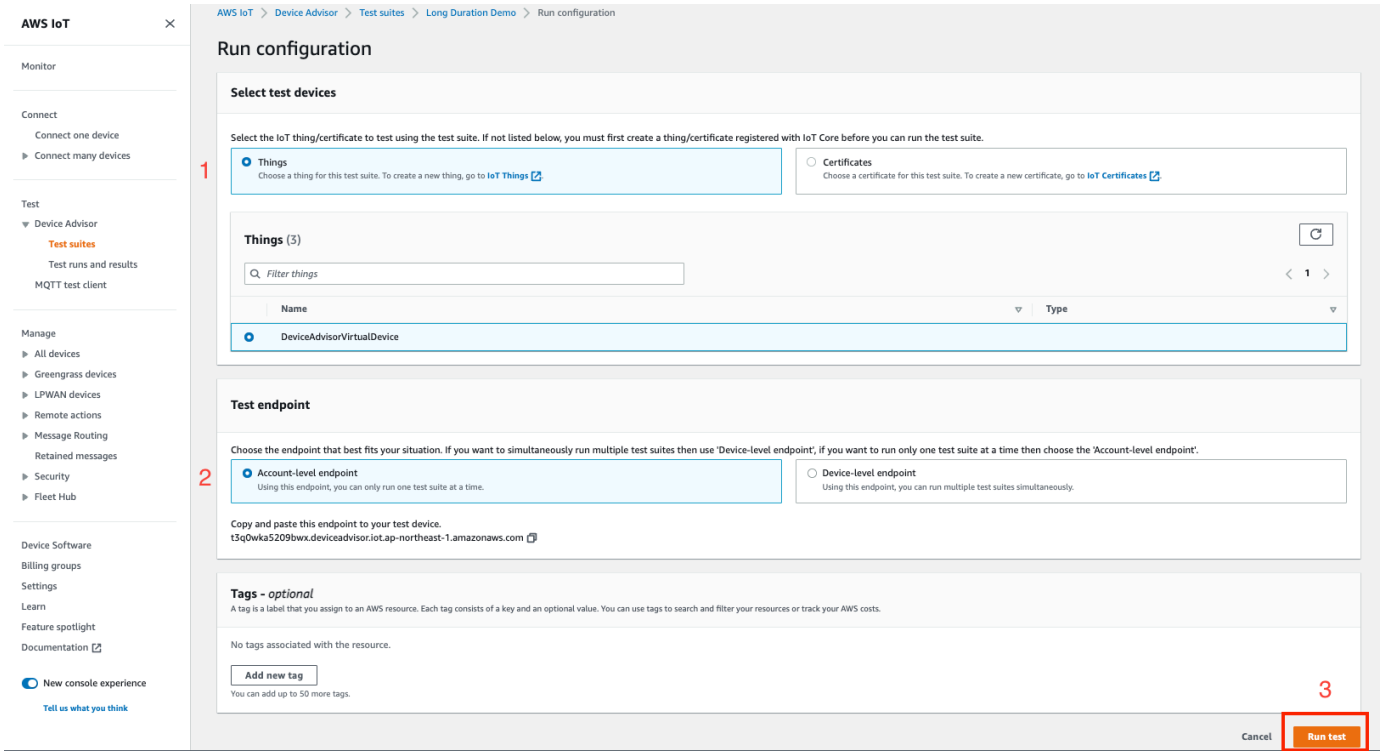


10. Elija las opciones de configuración en la página Ejecutar configuración.

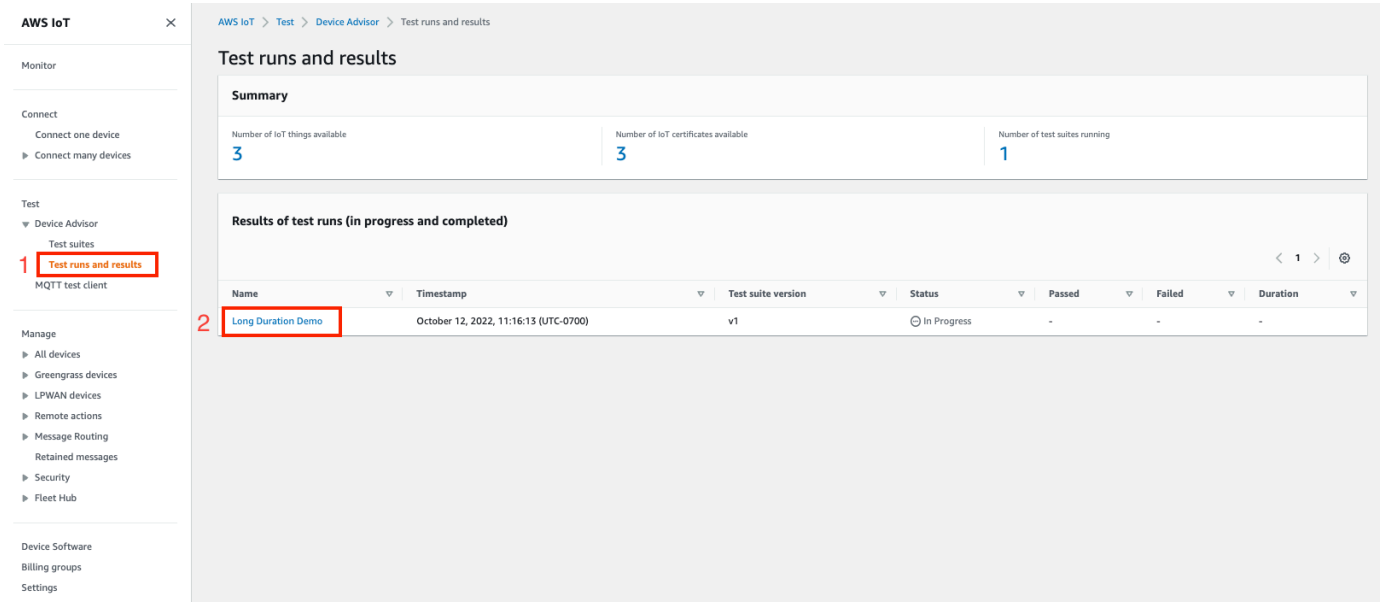
    a. Seleccione los objetos o el certificado en los que desee ejecutar la prueba.

    b. Seleccione el punto de conexión de nivel de cuenta o el punto de conexión de nivel de dispositivo.

    c. Seleccione Ejecutar prueba para ejecutar la prueba.

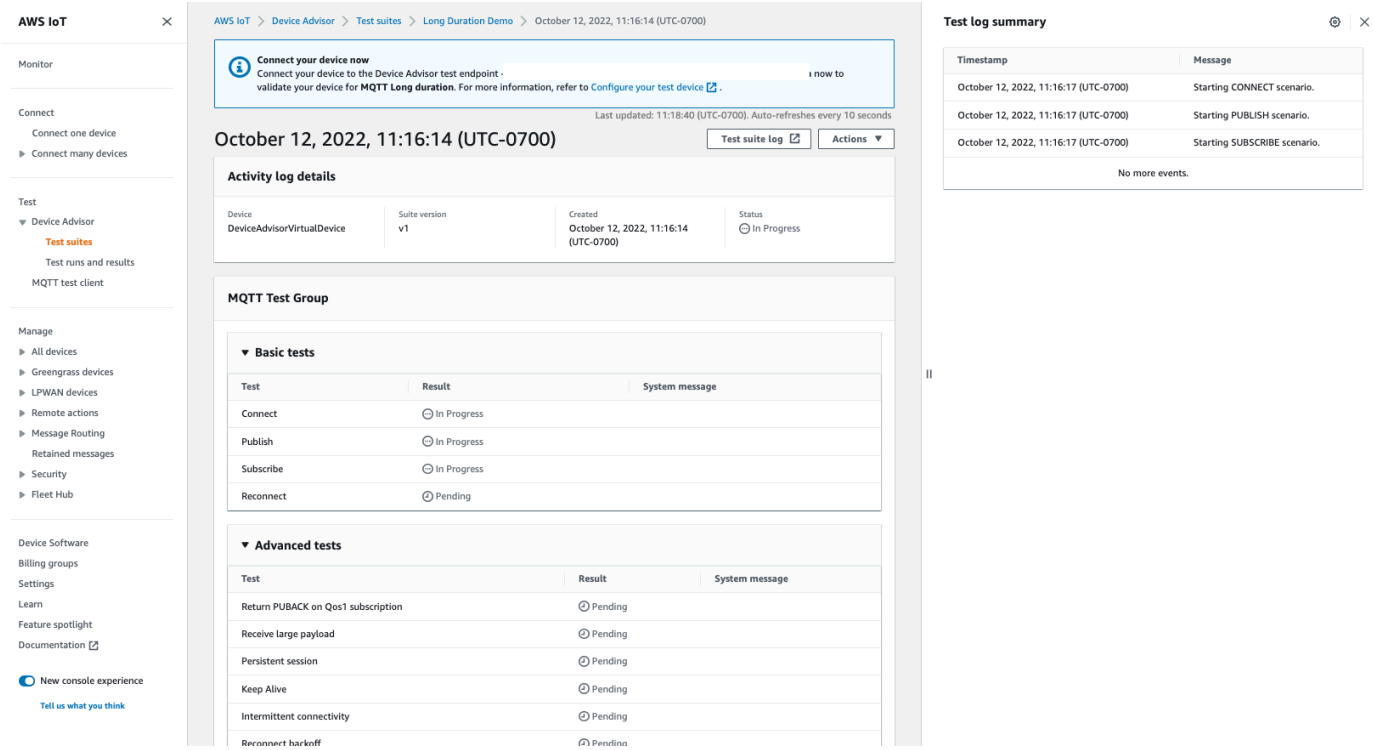


11. Para ver los resultados de la ejecución del conjunto de pruebas, seleccione Ejecuciones de pruebas y resultados en el panel de navegación izquierdo. Elija el conjunto de pruebas que se ejecutó para ver los detalles de los resultados.

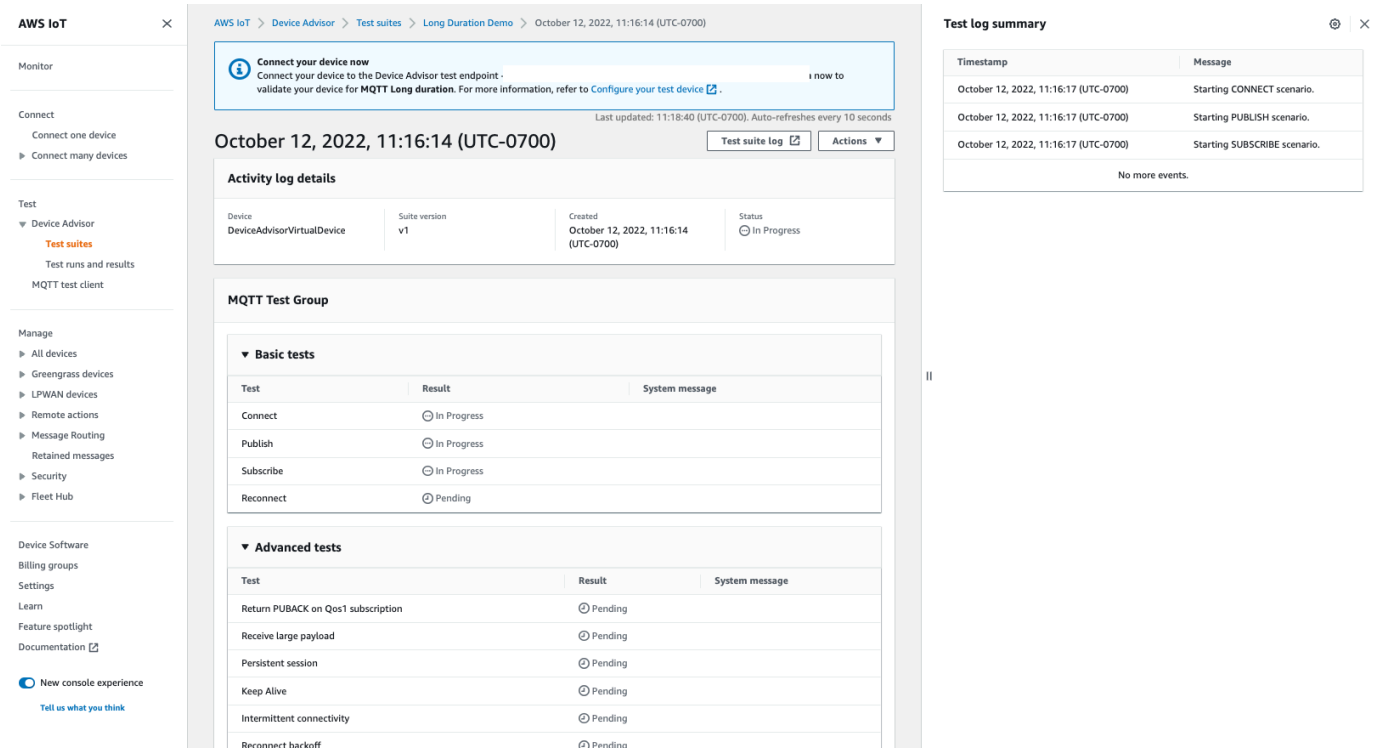


12. El paso anterior abre la página de resumen de la prueba. En esta página se muestran todos los detalles de la ejecución de la prueba. Cuando la consola solicite iniciar la conexión del

dispositivo, conéctelo al punto de conexión proporcionado. El progreso de las pruebas se puede ver en esta página.



13. La prueba de larga duración proporciona un resumen adicional del registro de pruebas en el panel lateral, que muestra todos los eventos importantes que se producen entre el dispositivo y el agente casi en tiempo real. Para ver registros más detallados, elija Registro de casos de prueba.



puntos de conexión de VPC de Device Advisor (AWS PrivateLink)

Puede establecer una conexión privada entre la VPC y el punto final de AWS IoT Core Device Advisor prueba (plano de datos) mediante la creación de un punto final de la VPC de interfaz. Puede usar este punto final para validar los AWS IoT dispositivos y lograr una conectividad fiable y segura AWS IoT Core antes de implementarlos en la producción. Las pruebas prediseñadas de Device Advisor le ayudarán a validar el software de su dispositivo según las prácticas recomendadas de uso de TLS, MQTT, Device Shadow y AWS IoT Jobs.

AWS PrivateLinkpotencia los puntos finales de la interfaz que se utilizan con sus dispositivos de IoT. Este servicio le ayuda a acceder al punto de conexión de prueba de AWS IoT Core Device Advisor de forma privada sin una puerta de enlace de internet, un dispositivo NAT, una conexión VPN o una conexión Direct Connect . Las instancias de su VPC que envían paquetes TCP y MQTT no necesitan direcciones IP públicas para comunicarse con AWS IoT Core Device Advisor los puntos finales de prueba. El tráfico entre tu VPC y AWS IoT Core Device Advisor no sale. Nube de AWS Cualquier comunicación TLS y MQTT entre los dispositivos IoT y los casos de prueba de Device Advisor se mantiene dentro de los recursos que tiene a su disposición en su Cuenta de AWS.

Cada punto de conexión de la interfaz está representado por una o más interfaces de redes elásticas en las subredes.

Para obtener más información sobre el uso de los puntos de conexión de VPC de interfaz, consulte puntos de conexión de VPC de interfaz (AWS PrivateLink) en la Guía del usuario de Amazon VPC.

Consideraciones sobre los puntos AWS IoT Core Device Advisor finales de VPC

Revise las propiedades y limitaciones de los puntos de conexión de interfaz en la Guía del usuario de Amazon VPC antes de configurar los puntos de conexión de VPC de interfaz. Tenga en cuenta lo siguiente antes de continuar:

- AWS IoT Core Device Advisor actualmente admite realizar llamadas al punto final de prueba de Device Advisor (plano de datos) desde su VPC. Un agente de mensajes utiliza las comunicaciones del plano de datos para enviar y recibir datos. Lo hace con la ayuda de paquetes TLS y MQTT. Puntos de conexión de VPC para AWS IoT Core Device Advisor conectar su AWS IoT dispositivo a los puntos de conexión de prueba de Device Advisor. Este punto de conexión de VPC no utiliza las acciones de la API del plano de control. Para crear o ejecutar un conjunto de pruebas u otro plano de control APIs, utilice la consola, un AWS SDK o una interfaz de línea de AWS comandos a través de la Internet pública.
- Los siguientes puntos de conexión de VPC son Regiones de AWS compatibles con: AWS IoT Core Device Advisor
  - Este de EE. UU. (Norte de Virginia)
  - Oeste de EE. UU. (Oregón)
  - Asia-Pacífico (Tokio)
  - Europa (Irlanda)
- Device Advisor admite MQTT con certificados de cliente X.509 y certificados de servidor RSA.
- En este momento, no se admiten políticas de puntos de conexión de VPC.
- Consulte los requisitos previos de los puntos de conexión de VPC para obtener instrucciones sobre cómo crear recursos que conecten los puntos de conexión de VPC. Debe crear una VPC y subredes privadas para usar los puntos de enlace de la AWS IoT Core Device Advisor VPC.
- Hay cuotas en sus recursos. AWS PrivateLink Para obtener más información, consulte Cuotas de AWS PrivateLink.
- Los puntos de conexión de VPC solo admiten tráfico. IPv4

Cree un punto final de VPC de interfaz para AWS IoT Core Device Advisor

Para empezar con los puntos de conexión de VPC, cree un punto de conexión de VPC de interfaz. A continuación, seleccione AWS IoT Core Device Advisor como. Servicio de AWS Si está utilizando el AWS CLI, llame describe-vpc-endpoint-servicespara confirmar que AWS IoT Core Device Advisor está presente en una zona de disponibilidad de su Región de AWS. Confirme que el grupo de seguridad asociado al punto de conexión permita la comunicación mediante el protocolo TCP para el tráfico MQTT y TLS. Por ejemplo, en la región Este de EE. UU. (Norte de Virginia), utilice el siguiente comando:

```
aws ec2 describe-vpc-endpoint-services --service-name com.amazonaws.us-
east-1.deviceadvisor.iot
```

Puede crear un punto final de VPC para AWS IoT Core usar el siguiente nombre de servicio:

- com.amazonaws.region.deviceadvisor.iot

De forma predeterminada, el DNS privado está activado para el punto de conexión. Esto garantiza que el uso del punto de conexión de prueba predeterminado permanezca dentro de sus subredes privadas. Para obtener tu punto de conexión a nivel de cuenta o dispositivo, usa la consola AWS CLI o un AWS SDK. Por ejemplo, si ejecuta get-endpoint en una subred pública o en una conexión pública a internet, puede obtener su punto de conexión y usarlo para conectarse a Device Advisor. Para obtener más información, consulte Acceso a un servicio a través de un punto de conexión de interfaz en la Guía del usuario de Amazon VPC.

Para conectar los clientes MQTT a las interfaces de punto final de la VPC, AWS PrivateLink el servicio crea registros DNS en una zona alojada privada conectada a la VPC. Estos registros de DNS dirigen las solicitudes del dispositivo AWS IoT al punto de conexión de VPC.

Control del acceso a AWS IoT Core Device Advisor más de puntos finales de VPC

Puede restringir el acceso de los dispositivos AWS IoT Core Device Advisor y permitir el acceso solo a través de los puntos finales de la VPC mediante claves de contexto de condición de la VPC. AWS IoT Core admite las siguientes claves de contexto relacionadas con la VPC:

- SourceVpc
- SourceVpce

- VPCSourceIp

> **Note**
>
> AWS IoT Core Device Advisor no admite políticas de puntos finales de VPC en este momento.

La siguiente política otorga permiso para conectarse AWS IoT Core Device Advisor mediante un ID de cliente que coincida con el nombre de la cosa. También publica en cualquier tema con el prefijo del nombre del objeto. La política está condicionada a que el dispositivo se conecte a un punto de conexión de VPC con un ID de punto de conexión de VPC concreto. Esta política deniega los intentos de conexión a su punto de conexión de prueba de AWS IoT Core Device Advisor público.

```
{
"Version":"2012-10-17",
    "Statement": [
        {
"Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"
            ],
            "Condition": {
"StringEquals": {
"aws:SourceVpce": "vpce-1a2b3c4d"
            }
        }

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

```
                "arn:aws:iot:us-east-1:123456789012:topic/
${iot:Connection.Thing.ThingName}/*"
            ]
        }
    ]
}
```

Casos de prueba de Device Advisor

Device Advisor ofrece pruebas prediseñadas en seis categorías.

- TLS
- MQTT
- Sombra
- Ejecución de trabajo
- Permisos y políticas
- Pruebas de larga duración

Casos de prueba de Device Advisor para poder optar al programa de calificación de AWS dispositivos.

Su dispositivo debe superar las siguientes pruebas para cumplir los requisitos del programa de cualificación de dispositivos AWS.

> **Note**
>
> Esta es una lista revisada de las pruebas de cualificación.

- Conexión TLS («Conexión TLS»)
- Certificado de servidor TLS con nombre de asunto incorrecto («Nombre común del asunto (CN)/nombre alternativo del asunto (SAN) incorrecto»)
- Certificado de servidor TLS no seguro («No firmado por una entidad de certificación reconocida»)

- Soporte de dispositivos TLS para AWS IoT conjuntos de cifrado («Soporte de dispositivos TLS para conjuntos de cifrado AWS IoT recomendados»)
- Fragmentos de tamaño máximo de recepción TLS («Fragmentos de tamaño máximo de recepción TLS»)
- Certificado de servidor TLS caducado («Certificado de servidor caducado»)
- Certificado de servidor TLS de gran tamaño («Certificado de servidor TLS de gran tamaño»)
- MQTT Connect («El dispositivo envía CONNECT a AWS IoT Core (Happy case)»)
- Suscripción MQTT («Puede suscribirse (caso deseable)»)
- Publicación MQTT («QoS0 (caso deseable)»)
- Reintentos fluctuantes de conexión MQTT («Reintentos de conexión del dispositivo con retroceso fluctuante - Sin respuesta CONNACK»)

TLS

Utilice estas pruebas para determinar si el protocolo de seguridad de la capa de transporte (TLS) entre sus dispositivos AWS IoT es seguro.

> **Note**
>
> Device Advisor ahora admite TLS 1.3.

Recorrido correcto

Conexión TLS

Valida si el dispositivo que se está probando puede completar el protocolo de enlace TLS con. AWS IoT Esta prueba no valida la implementación de MQTT del dispositivo cliente.

Example Definición de caso de prueba de API:

> **Note**
>
> `EXECUTION_TIMEOUT` tiene un valor predeterminado de 5 minutos. Para obtener los mejores resultados, se recomienda un valor de tiempo de espera de 2 minutos.

```
"tests":[
   {
      "name":"my_tls_connect_test",
      "configuration": {
         // optional:
         "EXECUTION_TIMEOUT":"300",  //in seconds
      },
      "test":{
         "id":"TLS_Connect",
         "version":"0.0.0"
      }
   }
]
```

Example Resultados de los casos de prueba:

- Aprobar: el dispositivo sometido a prueba completó el protocolo de enlace TLS con. AWS IoT
- Aprobar con advertencias: el dispositivo objeto de la prueba completó el protocolo de enlace TLS AWS IoT, pero el dispositivo emitió mensajes de advertencia de TLS. AWS IoT
- Fallo: el dispositivo sometido a prueba no pudo completar el protocolo de enlace TLS debido a un error de protocolo de enlace. AWS IoT

Fragmentos de tamaño máximo de recepción TLS

Este caso de prueba valida que su dispositivo puede recibir y procesar fragmentos de tamaño máximo de TLS. El dispositivo de prueba debe suscribirse a un tema preconfigurado con QoS 1 para recibir una gran carga. Puede personalizar la carga con la `${payload}` de configuración.

Example Definición de caso de prueba de API:

> **Note**
>
> `EXECUTION_TIMEOUT` tiene un valor predeterminado de 5 minutos. Para obtener los mejores resultados, se recomienda un valor de tiempo de espera de 2 minutos.

```
"tests":[
   {
      "name":"TLS Receive Maximum Size Fragments",
      "configuration": {
```

```
        // optional:
        "EXECUTION_TIMEOUT":"300",  //in seconds
        "PAYLOAD_FORMAT":"{"message":"${payload}"}", // A string with a placeholder
 ${payload}, or leave it empty to receive a plain string.
        "TRIGGER_TOPIC": "test_1" // A topic to which a device will subscribe, and
 to which a test case will publish a large payload.
     },
     "test":{
        "id":"TLS_Receive_Maximum_Size_Fragments",
        "version":"0.0.0"
     }
  }
]
```

Conjuntos de cifrado

Soporte de dispositivos TLS para los conjuntos de cifrado AWS IoT recomendados

Valida que las suites de cifrado en el mensaje de saludo del cliente de TLS del dispositivo bajo prueba contenga los conjuntos de cifrado de AWS IoT recomendados. Proporciona información adicional sobre los conjuntos de cifrado compatibles con el dispositivo.

Example Definición de caso de prueba de API:

> **Note**
>
> EXECUTION_TIMEOUT tiene un valor predeterminado de 5 minutos. Se recomienda un valor de tiempo de espera de 2 minutos.

```
"tests":[
  {
     "name":"my_tls_support_aws_iot_cipher_suites_test",
     "configuration": {
        // optional:
        "EXECUTION_TIMEOUT":"300",  // in seconds
     },
     "test":{
        "id":"TLS_Support_AWS_IoT_Cipher_Suites",
        "version":"0.0.0"
     }
```

```
    }
]
```

Example Resultados de los casos de prueba:

- Aprobación: los conjuntos de cifrado del dispositivo que se están probando contienen al menos uno de los conjuntos de AWS IoT cifrado recomendados y no contienen ningún conjunto de cifrado no compatible.
- Superada con advertencias: los conjuntos de cifrado del dispositivo contienen al menos un conjunto de cifrado de AWS IoT , pero:
  1. No contiene ninguno de los conjuntos de cifrado recomendados
  2. Contiene conjuntos de cifrado que no son compatibles con. AWS IoT

  Le sugerimos que compruebe que todos los conjuntos de cifrado no compatibles son seguros.
- Error: el dispositivo que se está probando con los conjuntos de cifrado no contiene ninguno de los conjuntos de cifrado AWS IoT compatibles.

Certificado de servidor de mayor tamaño

Certificado de servidor de mayor tamaño de TLS

Valida en su dispositivo que se pueda completar el protocolo de enlace TLS con AWS IoT cuando recibe y procesa un certificado de servidor de mayor tamaño. El tamaño del certificado de servidor (en bytes) utilizado en esta prueba es 20 veces mayor que el que se utiliza actualmente en el caso de prueba de TLS Connect e IoT Core. Durante este caso de prueba, AWS IoT comprueba el espacio de búfer del dispositivo para TLS. Si el espacio de búfer es lo suficientemente grande, el protocolo de enlace TLS se completa sin errores. Esta prueba no valida la implementación de MQTT del dispositivo. El caso de prueba se produce una vez finalizado el proceso de protocolo de enlace TLS.

Example Definición de caso de prueba de API:

> **Note**
>
> `EXECUTION_TIMEOUT` tiene un valor predeterminado de 5 minutos. Para obtener los mejores resultados, se recomienda un valor de tiempo de espera de 2 minutos. Si este caso de prueba falla pero el caso de prueba de TLS Connect pasa, le recomendamos que aumente el límite de espacio de búfer del dispositivo para TLS. Si aumenta el límite

de espacio de búfer, su dispositivo podrá procesar un certificado de servidor de mayor tamaño en caso de que el tamaño aumente.

```
"tests":[
   {
      "name":"my_tls_large_size_server_cert_test",
      "configuration": {
         // optional:
         "EXECUTION_TIMEOUT":"300",  // in seconds
      },
      "test":{
         "id":"TLS_Large_Size_Server_Cert",
         "version":"0.0.0"
      }
   }
]
```

Example Resultados de los casos de prueba:

- Superada: el dispositivo sometido a prueba completó el protocolo de enlace TLS con AWS IoT.
- Aprobar con advertencias: el dispositivo que se está probando completó el protocolo de enlace TLS AWS IoT, pero hay mensajes de advertencia de TLS provenientes del dispositivo o AWS IoT.
- Fallo: el dispositivo sometido a prueba no pudo completar el protocolo de enlace TLS AWS IoT debido a un error durante el proceso de protocolo de enlace.

Certificado de servidor no seguro de TLS

No firmado por entidad de certificación reconocida

Valida que el dispositivo que se está probando cierra la conexión si se le presenta un certificado de servidor sin una firma válida de la entidad de certificación de ATS. Un dispositivo solo debe conectarse a un punto de conexión que presente un certificado válido.

Example Definición de caso de prueba de API:

> **Note**
>
> EXECUTION_TIMEOUT tiene un valor predeterminado de 5 minutos. Se recomienda un valor de tiempo de espera de 2 minutos.

```
"tests":[
   {
      "name":"my_tls_unsecure_server_cert_test",
      "configuration": {
         // optional:
         "EXECUTION_TIMEOUT":"300",  //in seconds
      },
      "test":{
         "id":"TLS_Unsecure_Server_Cert",
         "version":"0.0.0"
      }
   }
]
```

Example Resultados de los casos de prueba:

- Superada: el dispositivo sometido a prueba cerró la conexión.
- Fallo: el dispositivo objeto de la prueba completó el protocolo de enlace TLS con. AWS IoT

Certificado de servidor TLS con nombre de asunto incorrecto/Nombre común del asunto (CN)/ Nombre alternativo del asunto (SAN) incorrecto

Valida que el dispositivo que se está probando cierra la conexión si se le presenta un certificado de servidor para un nombre de dominio diferente al solicitado.

Example Definición de caso de prueba de API:

> **Note**
>
> EXECUTION_TIMEOUT tiene un valor predeterminado de 5 minutos. Se recomienda un valor de tiempo de espera de 2 minutos.

```
"tests":[
   {
      "name":"my_tls_incorrect_subject_name_cert_test",
      "configuration": {
         // optional:
         "EXECUTION_TIMEOUT":"300",   // in seconds
      },
      "test":{
         "id":"TLS_Incorrect_Subject_Name_Server_Cert",
         "version":"0.0.0"
      }
   }
]
```

Example Resultados de los casos de prueba:

- Superada: el dispositivo sometido a prueba cerró la conexión.
- Fallo: el dispositivo objeto de la prueba completó el protocolo de enlace TLS con. AWS IoT

Certificado de servidor TLS caducado

Certificado de servidor caducado

Valida que el dispositivo sometido a prueba cierre la conexión si se le presenta un certificado de servidor caducado.

Example Definición de caso de prueba de API:

> **Note**
>
> EXECUTION_TIMEOUT tiene un valor predeterminado de 5 minutos. Se recomienda un valor de tiempo de espera de 2 minutos.

```
"tests":[
   {
      "name":"my_tls_expired_cert_test",
      "configuration": {
         // optional:
         "EXECUTION_TIMEOUT":"300",  //in seconds
```

```
      },
      "test":{
         "id":"TLS_Expired_Server_Cert",
         "version":"0.0.0"
      }
   }
]
```

Example Resultados de los casos de prueba:

- Aprobar: el dispositivo que se está probando se niega a completar el protocolo de enlace TLS con él. AWS IoT El dispositivo envía un mensaje de alerta de TLS antes de cerrar la conexión.
- Superada con advertencias: el dispositivo sometido a prueba rechaza completar el protocolo de enlace TLS con AWS IoT. Sin embargo, no envía un mensaje de alerta de TLS antes de cerrar la conexión.
- Fallo: el dispositivo que se está probando completa el protocolo de enlace TLS con. AWS IoT

MQTT

CONNECT, DISCONNECT y RECONNECT

«El dispositivo envía CONNECT a AWS IoT Core (Happy case)»

Valida que el dispositivo sometido a prueba envía una solicitud CONNECT.

Definición de caso de prueba de API:

> **Note**
>
> `EXECUTION_TIMEOUT` tiene un valor predeterminado de 5 minutos. Se recomienda un valor de tiempo de espera de 2 minutos.

```
"tests":[
   {
      "name":"my_mqtt_connect_test",
      "configuration": {
         // optional:
         "EXECUTION_TIMEOUT":"300",   // in seconds
```

```
      },
      "test":{
         "id":"MQTT_Connect",
         "version":"0.0.0"
      }
   }
]
```

«El dispositivo puede devolver PUBACK a un tema arbitrario para QoS1»

Este caso de prueba comprobará si el dispositivo (cliente) puede devolver un mensaje de PUBACK si ha recibido un mensaje de publicación del agente tras suscribirse a un tema con QoS1.

El contenido y el tamaño de la carga son configurables para este caso de prueba. Si el tamaño de la carga está configurado, Device Advisor sobrescribirá el valor del contenido de la carga útil y enviará una carga útil predefinida al dispositivo con el tamaño deseado. El tamaño de la carga útil es un valor entre 0 y 128 y no puede superar los 128 KB. AWS IoT Core rechaza las solicitudes de publicación y conexión de más de 128 KB, tal y como se muestra en la página del agente de mensajes de AWS IoT Core y los límites y cuotas del protocolo.

Definición de caso de prueba de API:

> **Note**
>
> `EXECUTION_TIMEOUT` tiene un valor predeterminado de 5 minutos. Se recomienda un valor de tiempo de espera de 2 minutos. `PAYLOAD_SIZE` se puede configurar con un valor entre 0 y 128 kilobytes. Al definir un tamaño de carga, se anula el contenido de la carga, ya que Device Advisor enviará una carga predefinida con el tamaño indicado al dispositivo.

```
"tests":[
{
        "name":"my_mqtt_client_puback_qos1",
        "configuration": {
            // optional:"TRIGGER_TOPIC": "myTopic",
            "EXECUTION_TIMEOUT":"300", // in seconds
            "PAYLOAD_FOR_PUBLISH_VALIDATION":"custom payload",
            "PAYLOAD_SIZE":"100" // in kilobytes
```

```
        },
        "test": {
            "id": "MQTT_Client_Puback_QoS1",
            "version": "0.0.0"
        }
    }
]
```

«Reintentos de conexión del dispositivo con retroceso fluctuante - Sin respuesta CONNACK»

Valida que el dispositivo que se está probando utiliza el retroceso fluctuante adecuado al volver a conectarse con el agente al menos cinco veces. El agente registra la marca de tiempo de la solicitud CONNECT del dispositivo sometido a prueba, valida los paquetes, hace una pausa sin enviar CONNACK al dispositivo sometido a prueba y espera a que el dispositivo vuelva a enviar la solicitud. Se deja pasar el sexto intento de conexión y se permite que CONNACK regrese al dispositivo objeto de la prueba.

Se vuelve a realizar el proceso anterior. En total, este caso de prueba requiere que el dispositivo se conecte al menos 12 veces. Las marcas de tiempo recopiladas se utilizan para validar que el dispositivo que se está probando utiliza retroceso fluctuante. Si el dispositivo que se está probando tiene un retroceso fluctuante estrictamente exponencial, este caso de prueba superará con advertencias.

Recomendamos implementar el mecanismo de retroceso fluctuante y exponencial en el dispositivo sometido a prueba para superar este caso de prueba.

Definición de caso de prueba de API:

> **Note**
>
> `EXECUTION_TIMEOUT` tiene un valor predeterminado de 5 minutos. Se recomienda un valor de tiempo de espera de 4 minutos.

```
"tests":[
   {
      "name":"my_mqtt_jitter_backoff_retries_test",
      "configuration": {
         // optional:
         "EXECUTION_TIMEOUT":"300",    // in seconds
```

```
      },
      "test":{
         "id":"MQTT_Connect_Jitter_Backoff_Retries",
         "version":"0.0.0"
      }
   }
]
```

«Reintentos de conexión del dispositivo con retroceso exponencial - Sin respuesta CONNACK»

Valida que el dispositivo que se está probando utiliza el retroceso exponencial adecuado al volver a conectarse con el agente al menos cinco veces. El agente registra la marca de tiempo de la solicitud CONNECT del dispositivo sometido a prueba, valida los paquetes, hace una pausa sin enviar un CONNACK al dispositivo cliente y espera a que el dispositivo vuelva a enviar la solicitud. Las marcas de tiempo recopiladas se utilizan para validar que el dispositivo que se está probando utiliza retroceso exponencial.

Recomendamos implementar el mecanismo de retroceso fluctuante y exponencial en el dispositivo sometido a prueba para superar este caso de prueba.

Definición de caso de prueba de API:

> **Note**
>
> `EXECUTION_TIMEOUT` tiene un valor predeterminado de 5 minutos. Se recomienda un valor de tiempo de espera de 4 minutos.

```
"tests":[
   {
      "name":"my_mqtt_exponential_backoff_retries_test",
      "configuration": {
         // optional:
         "EXECUTION_TIMEOUT":"600",  // in seconds
      },
      "test":{
         "id":"MQTT_Connect_Exponential_Backoff_Retries",
         "version":"0.0.0"
      }
   }
]
```

«Reconexión del dispositivo con retroceso fluctuante - Tras la desconexión del servidor»

Valida si un dispositivo sometido a prueba utiliza la fluctuación y el retroceso necesarios al volver a conectarse después de haber sido desconectado del servidor. Device Advisor desconecta el dispositivo del servidor al menos cinco veces y observa el comportamiento del dispositivo al volver a conectarse mediante MQTT. Device Advisor registra la marca de tiempo de la solicitud CONNECT para el dispositivo sometido a prueba, realiza la validación del paquete, hace una pausa sin enviar un CONNACK al dispositivo cliente y espera a que el dispositivo objeto de prueba vuelva a enviar la solicitud. Las marcas de tiempo recopiladas se utilizan para validar que el dispositivo sometido a prueba utiliza fluctuación y retroceso durante la reconexión. Si el dispositivo sometido a prueba tiene un retroceso estrictamente exponencial o no implementa un mecanismo de retroceso fluctuante adecuado, este caso de prueba superará con advertencias. Si el dispositivo que se está probando ha implementado un mecanismo de retroceso lineal o un mecanismo de retroceso constante, la prueba fallará.

Para superar este caso de prueba, le recomendamos que implemente el mecanismo de retroceso y fluctuación exponencial en el dispositivo sometido a prueba.

Definición de caso de prueba de API:

> **Note**
>
> `EXECUTION_TIMEOUT` tiene un valor predeterminado de 5 minutos. Se recomienda un valor de tiempo de espera de 4 minutos.
> El número de intentos de reconexión para validar el retroceso se puede cambiar especificando el parámetro `RECONNECTION_ATTEMPTS`. El número debe estar comprendido entre 5 y 10. El valor predeterminado es 5.

```
"tests":[
   {
      "name":"my_mqtt_reconnect_backoff_retries_on_server_disconnect",
      "configuration":{
         // optional:
         "EXECUTION_TIMEOUT":"300",  // in seconds
         "RECONNECTION_ATTEMPTS": 5
      },
      "test":{
         "id":"MQTT_Reconnect_Backoff_Retries_On_Server_Disconnect",
```

```
         "version":"0.0.0"
      }
   }
]
```

«Reconexión del dispositivo con un retroceso fluctuante - En caso de una conexión inestable»

Valida si un dispositivo sometido a prueba utiliza la fluctuación y el retroceso necesarios al volver a conectarse en una conexión inestable. Device Advisor desconecta el dispositivo del servidor después de cinco conexiones correctas y observa el comportamiento del dispositivo para la reconexión MQTT. Device Advisor registra la marca de tiempo de la solicitud CONNECT para el dispositivo sometido a prueba, realiza la validación del paquete, envía de vuelta CONNACK, se desconecta, registra la marca de tiempo de la desconexión y espera a que el dispositivo objeto de prueba vuelva a enviar la solicitud. Las marcas de tiempo recopiladas se utilizan para validar que el dispositivo sometido a prueba utiliza fluctuación y retroceso durante la reconexión tras conexiones correctas pero inestables. Si el dispositivo sometido a prueba tiene un retroceso estrictamente exponencial o no implementa un mecanismo de retroceso fluctuante adecuado, este caso de prueba superará con advertencias. Si el dispositivo que se está probando ha implementado un mecanismo de retroceso lineal o un mecanismo de retroceso constante, la prueba fallará.

Para superar este caso de prueba, le recomendamos que implemente el mecanismo de retroceso y fluctuación exponencial en el dispositivo sometido a prueba.

Definición de caso de prueba de API:

> **Note**
>
> `EXECUTION_TIMEOUT` tiene un valor predeterminado de 5 minutos. Se recomienda un valor de tiempo de espera de 4 minutos.
> El número de intentos de reconexión para validar el retroceso se puede cambiar especificando el parámetro `RECONNECTION_ATTEMPTS`. El número debe estar comprendido entre 5 y 10. El valor predeterminado es 5.

```
"tests":[
   {
      "name":"my_mqtt_reconnect_backoff_retries_on_unstable_connection",
      "configuration":{
```

```
         // optional:
         "EXECUTION_TIMEOUT":"300",  // in seconds
         "RECONNECTION_ATTEMPTS": 5
      },
      "test":{
         "id":"MQTT_Reconnect_Backoff_Retries_On_Unstable_Connection",
         "version":"0.0.0"
      }
   }
]
```

Publish

«QoS0 (caso deseable)»

Valida que el dispositivo que se está probando publique un mensaje con QoS0 o QoS1. También puede validar el tema del mensaje y la carga especificando el valor del tema y la carga en la configuración de la prueba.

> **Note**
>
> EXECUTION_TIMEOUT tiene un valor predeterminado de 5 minutos. Se recomienda un valor de tiempo de espera de 2 minutos.

```
"tests":[
   {
      "name":"my_mqtt_publish_test",
      "configuration":{
         // optional:
         "EXECUTION_TIMEOUT":"300",  // in seconds
         "TOPIC_FOR_PUBLISH_VALIDATION": "my_TOPIC_FOR_PUBLISH_VALIDATION",
         "PAYLOAD_FOR_PUBLISH_VALIDATION": "my_PAYLOAD_FOR_PUBLISH_VALIDATION",
      },
      "test":{
         "id":"MQTT_Publish",
         "version":"0.0.0"
      }
   }
]
```

«Reintento de publicación de QoS1 - Sin PUBACK»

Valida que el dispositivo sometido a prueba vuelva a publicar un mensaje enviado con QoS1, si el agente no envía PUBACK. También puede validar el tema del mensaje especificando este tema en la configuración de la prueba. El dispositivo cliente no debe desconectarse antes de volver a publicar el mensaje. Esta prueba también valida que el mensaje republicado tenga el mismo identificador de paquete que el original. Durante la ejecución de la prueba, si el dispositivo pierde la conexión y se vuelve a conectar, el caso de prueba se restablecerá sin fallar y el dispositivo tendrá que volver a realizar los pasos del caso de prueba.

Definición de caso de prueba de API:

> **Note**
>
> `EXECUTION_TIMEOUT` tiene un valor predeterminado de 5 minutos. Se recomienda que dure al menos 4 minutos.

```
"tests":[
   {
      "name":"my_mqtt_publish_retry_test",
      "configuration":{
         // optional:
         "EXECUTION_TIMEOUT":"300",  // in seconds
         "TOPIC_FOR_PUBLISH_VALIDATION": "my_TOPIC_FOR_PUBLISH_VALIDATION",
         "PAYLOAD_FOR_PUBLISH_VALIDATION": "my_PAYLOAD_FOR_PUBLISH_VALIDATION",
      },
      "test":{
         "id":"MQTT_Publish_Retry_No_Puback",
         "version":"0.0.0"
      }
   }
]
```

«Publicación de mensajes retenidos»

Valida que el dispositivo sometido a prueba publique un mensaje con `retainFlag` establecido en verdadero. Puede validar el tema y la carga del mensaje estableciendo el valor del tema y la carga en la configuración de la prueba. Si el valor `retainFlag` enviado en el paquete PUBLISH no está establecido en verdadero, el caso de prueba fallará.

Definición de caso de prueba de API:

> **Note**
>
> EXECUTION_TIMEOUT tiene un valor predeterminado de 5 minutos. Se recomienda un valor de tiempo de espera de 2 minutos. Para ejecutar este caso de prueba, agregue la acción `iot:RetainPublish` a su rol de dispositivo.

```
"tests":[
   {
      "name":"my_mqtt_publish_retained_messages_test",
      "configuration":{
         // optional:
         "EXECUTION_TIMEOUT":"300",  // in seconds

 "TOPIC_FOR_PUBLISH_RETAINED_VALIDATION": "my_TOPIC_FOR_PUBLISH_RETAINED_VALIDATION",

 "PAYLOAD_FOR_PUBLISH_RETAINED_VALIDATION": "my_PAYLOAD_FOR_PUBLISH_RETAINED_VALIDATION",
      },
      "test":{
         "id":"MQTT_Publish_Retained_Messages",
         "version":"0.0.0"
      }
   }
]
```

«Publicación con propiedad de usuario»

Valida que el dispositivo sometido a prueba publique un mensaje con la propiedad de usuario correcta. Puede validar la propiedad del usuario configurando el par nombre-valor en la configuración de la prueba. Si la propiedad del usuario no se proporciona o no coincide, el caso de prueba falla.

Definición de caso de prueba de API:

> **Note**
>
> Este es MQTT5 solo un caso de prueba.

EXECUTION_TIMEOUT tiene un valor predeterminado de 5 minutos. Se recomienda un valor de tiempo de espera de 2 minutos.

```
"tests":[
   {
      "name":"my_mqtt_user_property_test",
      "test":{
        "USER_PROPERTIES": [
            {"name": "name1", "value":"value1"},
            {"name": "name2", "value":"value2"}
        ],
        "EXECUTION_TIMEOUT":"300",  // in seconds
      },
      "test":{
         "id":"MQTT_Publish_User_Property",
         "version":"0.0.0"
      }
   }
]
```

Suscribirse

«Puede suscribirse (caso deseable)»

Valida que el dispositivo sometido a prueba esté suscrito a los temas de MQTT. También puede validar el tema al que está suscrito el dispositivo sometido a prueba especificando este tema en la configuración de la prueba.

Definición de caso de prueba de API:

> **Note**
>
> EXECUTION_TIMEOUT tiene un valor predeterminado de 5 minutos. Se recomienda un valor de tiempo de espera de 2 minutos.

```
"tests":[
   {
```

```
      "name":"my_mqtt_subscribe_test",
      "configuration":{
         // optional:
         "EXECUTION_TIMEOUT":"300",  // in seconds
         "TOPIC_LIST_FOR_SUBSCRIPTION_VALIDATION":
["my_TOPIC_FOR_PUBLISH_VALIDATION_a","my_TOPIC_FOR_PUBLISH_VALIDATION_b"]
      },
      "test":{
         "id":"MQTT_Subscribe",
         "version":"0.0.0"
      }
   }
]
```

«Reintento de suscripción - Sin SUBACK»

Valida que el dispositivo sometido a prueba reintenta una suscripción fallida a temas de MQTT. A continuación, el servidor espera y no envía un SUBACK. Si el dispositivo cliente no vuelve a intentar la suscripción, la prueba no se realizará correctamente. El dispositivo cliente debe volver a intentar la suscripción fallida con el mismo identificador de paquete. También puede validar el tema al que está suscrito el dispositivo sometido a prueba especificando este tema en la configuración de la prueba. Durante la ejecución de la prueba, si el dispositivo pierde la conexión y se vuelve a conectar, el caso de prueba se restablecerá sin fallar y el dispositivo tendrá que volver a realizar los pasos del caso de prueba.

Definición de caso de prueba de API:

> **Note**
>
> `EXECUTION_TIMEOUT` tiene un valor predeterminado de 5 minutos. Se recomienda un valor de tiempo de espera de 4 minutos.

```
"tests":[
   {
      "name":"my_mqtt_subscribe_retry_test",
      "configuration":{
         "EXECUTION_TIMEOUT":"300",  // in seconds
         // optional:
         "TOPIC_LIST_FOR_SUBSCRIPTION_VALIDATION":
["myTOPIC_FOR_PUBLISH_VALIDATION_a","my_TOPIC_FOR_PUBLISH_VALIDATION_b"]
```

```
      },
      "test":{
         "id":"MQTT_Subscribe_Retry_No_Suback",
         "version":"0.0.0"
      }
   }
]
```

Keep-Alive

«Matt No Ack PingResp»

Este caso de prueba valida si el dispositivo sometido a prueba se desconecta cuando no recibe una respuesta de ping. Como parte de este caso de prueba, Device Advisor bloquea las respuestas enviadas desde AWS IoT Core las solicitudes de publicación, suscripción y ping. También valida si el dispositivo sometido a prueba desconecta la conexión MQTT.

Definición de caso de prueba de API:

> **Note**
>
> `EXECUTION_TIMEOUT` tiene un valor predeterminado de 5 minutos. Recomendamos un tiempo de espera superior a 1,5 veces el valor `keepAliveTime`.
> El `keepAliveTime` máximo no debe ser superior a 230 segundos para esta prueba.

```
"tests":[
    {
       "name":"Mqtt No Ack PingResp",
       "configuration":
          //optional:
          "EXECUTION_TIMEOUT":"306",   // in seconds
       },
       "test":{
          "id":"MQTT_No_Ack_PingResp",
          "version":"0.0.0"
       }
    }
]
```

Sesión persistente

«Sesión persistente (caso deseable)»

Este caso de prueba valida el comportamiento del dispositivo cuando se desconecta de una sesión persistente. El caso de prueba comprueba si el dispositivo puede volver a conectarse, reanudar las suscripciones a los temas que lo activaron sin tener que volver a suscribirse de forma explícita, recibir los mensajes almacenados en los temas y funcionar según lo previsto durante una sesión persistente. Cuando se supera este caso de prueba, indica que el dispositivo cliente es capaz de mantener una sesión persistente con el AWS IoT Core intermediario de la manera esperada. Para obtener más información sobre las sesiones AWS IoT persistentes, consulte Uso de sesiones persistentes de MQTT.

En este caso de prueba, se espera que el dispositivo cliente envíe un paquete CONNECT con el AWS IoT Core y un indicador de sesión limpia establecido en falso y, a continuación, se suscriba a un tema activador. Tras una suscripción correcta, Device Advisor desconectará el AWS IoT Core dispositivo. Mientras el dispositivo esté desconectado, se almacenará una carga de mensajes de QoS 1 en ese tema. Luego, Device Advisor permitirá que el dispositivo cliente se vuelva a conectar con el punto de conexión de prueba. En este punto, dado que hay una sesión persistente, se espera que el dispositivo cliente reanude sus suscripciones de temas sin enviar ningún paquete SUBSCRIBE adicional y reciba el mensaje de QoS 1 del agente. Tras volver a conectarse, si el dispositivo cliente vuelve a suscribirse al tema de activación mediante el envío de un paquete and/or SUBSCRIBE adicional y el cliente no recibe el mensaje almacenado del tema de activación, el caso de prueba fallará.

Definición de caso de prueba de API:

> **Note**
>
> `EXECUTION_TIMEOUT` tiene un valor predeterminado de 5 minutos. Se recomienda un valor de tiempo de espera de 4 minutos como mínimo. En la primera conexión, el dispositivo cliente debe suscribirse explícitamente a un `TRIGGER_TOPIC` al que no se haya suscrito anteriormente. Para superar el caso de prueba, el dispositivo cliente debe suscribirse correctamente a `TRIGGER_TOPIC` con un QoS 1. Tras volver a conectarse, se espera que el dispositivo cliente comprenda que hay una sesión persistente activa, por lo que debe aceptar el mensaje almacenado enviado por el tema activador y devolver PUBACK para ese mensaje específico.

```
"tests":[
   {
      "name":"my_mqtt_persistent_session_happy_case",
      "configuration":{
         //required:
         "TRIGGER_TOPIC": "myTrigger/topic",
         // optional:
         // if Payload not provided, a string will be stored in the trigger topic to
 be sent back to the client device
         "PAYLOAD": "The message which should be received from AWS IoT Broker after
 re-connecting to a persistent session from the specified trigger topic.",

         "EXECUTION_TIMEOUT":"300" // in seconds
      },
      "test":{
         "id":"MQTT_Persistent_Session_Happy_Case",
         "version":"0.0.0"
      }
   }
]
```

«Sesión persistente - Caducidad de la sesión»

Este caso de prueba ayuda a validar el comportamiento del dispositivo cuando un dispositivo desconectado se vuelve a conectar a una sesión persistente caducada. Cuando finalice la sesión, esperamos que el dispositivo vuelva a suscribirse a los temas a los que estaba suscrito anteriormente mediante el envío explícito de un nuevo paquete SUBSCRIBE.

Durante la primera conexión, esperamos que el dispositivo de prueba se CONECTE con el intermediario de AWS IoT, ya que su `CleanSession` indicador está establecido en falso para iniciar una sesión persistente. A continuación, el dispositivo debería suscribirse a un tema activador. Luego, Device Advisor desconecta AWS IoT Core el dispositivo después de suscribirse correctamente y de iniciar una sesión persistente. Tras la desconexión, AWS IoT Core Device Advisor permite que el dispositivo de prueba se vuelva a conectar con el punto final de prueba. En este punto, cuando el dispositivo de prueba envía otro paquete CONNECT, AWS IoT Core Device Advisor devuelve un paquete CONNACK que indica que la sesión persistente ha caducado. El dispositivo de prueba debe interpretar este paquete correctamente y se espera que vuelva a suscribirse al mismo tema activador cuando finalice la sesión persistente. Si el dispositivo de prueba no vuelve a suscribirse al tema activador, el caso de prueba falla. Para superar la prueba,

el dispositivo debe comprender que la sesión persistente ha finalizado y enviar un nuevo paquete SUBSCRIBE para el mismo tema activador en la segunda conexión.

Si este caso de prueba es válido para un dispositivo de prueba, indica que el dispositivo es capaz de gestionar la reconexión al caducar la sesión persistente de la forma esperada.

Definición de caso de prueba de API:

> **Note**
>
> `EXECUTION_TIMEOUT` tiene un valor predeterminado de 5 minutos. Se recomienda un valor de tiempo de espera de 4 minutos como mínimo. El dispositivo de prueba necesita suscribirse explícitamente a un `TRIGGER_TOPIC`, al que no estaba suscrito anteriormente. Para superar el caso de prueba, el dispositivo de prueba debe enviar un paquete CONNECT con el indicador `CleanSession` establecido en false y suscribirse correctamente a un tema activador con un QoS 1. Tras una conexión correcta, AWS IoT Core Device Advisor desconecta el dispositivo. Tras la desconexión, AWS IoT Core Device Advisor permite que el dispositivo se vuelva a conectar y se espera que el dispositivo vuelva a suscribirse a la misma conexión, `TRIGGER_TOPIC` ya que AWS IoT Core Device Advisor habría terminado la sesión persistente.

```
"tests":[
   {
      "name":"my_expired_persistent_session_test",
      "configuration":{
         //required:
         "TRIGGER_TOPIC": "myTrigger/topic",
         // optional:
         "EXECUTION_TIMEOUT":"300" // in seconds
      },
      "test":{
         "id":"MQTT_Expired_Persistent_Session",
         "version":"0.0.0"
      }
   }
]
```

Sombra

Utilice estas pruebas para comprobar que los dispositivos que se están probando utilizan correctamente el servicio AWS IoT Device Shadow. Para obtener más información, consulte AWS IoT Servicio Device Shadow. Si estos casos de prueba están configurados en su conjunto de pruebas, es necesario proporcionar un objeto al iniciar la ejecución del conjunto.

Por el momento, no WebSocket se admite MQTT over.

Publish

«El dispositivo publica su estado después de conectarse (caso deseable)»

Valida si un dispositivo puede publicar su estado después de conectarse a AWS IoT Core

Definición de caso de prueba de API:

> **Note**
>
> EXECUTION_TIMEOUT tiene un valor predeterminado de 5 minutos. Se recomienda un valor de tiempo de espera de 2 minutos.

```
"tests":[
   {
      "name":"my_shadow_publish_reported_state",
      "configuration": {
         // optional:
         "EXECUTION_TIMEOUT":"300", // in seconds
         "SHADOW_NAME": "SHADOW_NAME",
         "REPORTED_STATE": {
            "STATE_ATTRIBUTE": "STATE_VALUE"
         }
      },
      "test":{
         "id":"Shadow_Publish_Reported_State",
         "version":"0.0.0"
      }
   }
]
```

`REPORTED_STATE` puede proporcionar para una validación adicional del estado de sombra exacto del dispositivo, una vez que se haya conectado. De forma predeterminada, este caso de prueba valida el estado de publicación del dispositivo.

Si no se proporciona *`SHADOW_NAME`*, el caso de prueba busca los mensajes publicados con prefijos de tema del tipo de sombra sin nombre (clásico) de forma predeterminada. Proporcione un nombre de sombra si su dispositivo utiliza el tipo de sombra con nombre. Consulte Uso de sombras en dispositivos para obtener más información.

Actualización

«El dispositivo actualiza el estado notificado al estado deseado (caso deseable)»

Valida si el dispositivo lee todos los mensajes de actualización recibidos y sincroniza el estado del dispositivo para que coincida con las propiedades de estado deseadas. El dispositivo debería publicar el último estado registrado tras la sincronización. Si su dispositivo ya tiene una sombra existente antes de ejecutar la prueba, asegúrese de que el estado deseado configurado para el caso de prueba y el estado registrado existente no coincidan aún. Para identificar los mensajes de actualización de Shadow enviados por Device Advisor, consulte el ClientTokencampo del documento oculto tal y como aparece. `DeviceAdvisorShadowTestCaseSetup`

Definición de caso de prueba de API:

> **Note**
>
> `EXECUTION_TIMEOUT` tiene un valor predeterminado de 5 minutos. Se recomienda un valor de tiempo de espera de 2 minutos.

```
"tests":[
   {
      "name":"my_shadow_update_reported_state",
      "configuration": {
         "DESIRED_STATE": {
            "STATE_ATTRIBUTE": "STATE_VALUE"
         },
         // optional:
         "EXECUTION_TIMEOUT":"300", // in seconds
         "SHADOW_NAME": "SHADOW_NAME"
```

```
      },
      "test":{
         "id":"Shadow_Update_Reported_State",
         "version":"0.0.0"
      }
   }
]
```

`DESIRED_STATE` debe tener al menos un atributo y un valor asociado.

Si no se proporciona `SHADOW_NAME`, el caso de prueba busca los mensajes publicados con prefijos de tema del tipo de sombra sin nombre (clásico) de forma predeterminada. Proporcione un nombre de sombra si su dispositivo utiliza el tipo de sombra con nombre. Consulte Uso de sombras en dispositivos para obtener más información.

Ejecución de trabajo

«El dispositivo puede completar la ejecución de un trabajo»

Este caso de prueba le ayuda a validar si su dispositivo puede recibir actualizaciones mediante AWS IoT Jobs y a publicar el estado de las actualizaciones correctas. Para obtener más información sobre los AWS IoT trabajos, consulte Trabajos.

Para ejecutar correctamente este caso de prueba, hay dos temas de AWS reservados a los que debe asignar el rol de dispositivo. Para suscribirse a los mensajes relacionados con la actividad de trabajos, utilice los temas notify y notify-next. Su función de dispositivo debe permitir la acción PUBLISH en los siguientes temas:

- $aws/things/thingName/jobs/jobId/get
- $aws/things/thingName/jobs/jobId/update

Se recomienda conceder las acciones SUBSCRIBE y RECEIVE para los siguientes temas:

- $aws/things/ ThingName/jobs/get/accepted
- $aws/things/thingName/jobs/jobId/get/rejected
- $aws/things/thingName/jobs/jobId/update/accepted
- $aws/things/thingName/jobs/jobId/update/rejected

Se recomienda conceder la acción SUBSCRIBE para el siguiente tema:

- $aws/things/thingName/jobs/notify-next

Para obtener más información sobre estos temas reservados, consulte los temas reservados de AWS IoT Jobs.

Por el momento, no se admite MQTT over. WebSocket

Definición de caso de prueba de API:

> **Note**
>
> `EXECUTION_TIMEOUT` tiene un valor predeterminado de 5 minutos. Se recomienda un valor de tiempo de espera de 3 minutos. Según el documento de AWS IoT trabajo o la fuente proporcionados, ajuste el valor de tiempo de espera (por ejemplo, si un trabajo tardará mucho en ejecutarse, defina un valor de tiempo de espera más largo para el caso de prueba). Para ejecutar la prueba, se necesita un documento de AWS IoT trabajo válido o un identificador de trabajo ya existente. Un documento de AWS IoT trabajo se puede proporcionar como un documento JSON o un enlace S3. Si se proporciona un documento de trabajo, proporcionar un identificador de trabajo es opcional. Si se proporciona un ID de trabajo, Device Advisor utilizará ese ID al crear el AWS IoT trabajo en su nombre. Si no se proporciona el documento de trabajo, puede proporcionar un ID existente que se encuentre en la misma región en la que está ejecutando el caso de prueba. En este caso, Device Advisor utilizará ese AWS IoT Job mientras ejecuta el caso de prueba.

```
"tests": [
   {
      "name":"my_job_execution",
      "configuration": {
         // optional:
         // Test case will create a job task by using either JOB_DOCUMENT or
 JOB_DOCUMENT_SOURCE.
         // If you manage the job task on your own, leave it empty and provide the
 JOB_JOBID (self-managed job task).
         // JOB_DOCUMENT is a JSON formatted string
         "JOB_DOCUMENT": "{
            \"operation\":\"reboot\",
            \"files\" : {
               \"fileName\" : \"install.py\",
               \"url\" : \"${aws:iot:s3-presigned-url:https://s3.amazonaws.com/
bucket-name/key}\"
```

```
            }
         }",
         // JOB_DOCUMENT_SOURCE is an S3 link to the job document. It will be used
 only if JOB_DOCUMENT is not provided.
         "JOB_DOCUMENT_SOURCE": "https://s3.amazonaws.com/bucket-name/key",
         // JOB_JOBID is mandatory, only if neither document nor document source is
 provided. (Test case needs to know the self-managed job task id).
         "JOB_JOBID": "String",
         // JOB_PRESIGN_ROLE_ARN is used for the presign Url, which will replace the
 placeholder in the JOB_DOCUMENT field
         "JOB_PRESIGN_ROLE_ARN": "String",
         // Presigned Url expiration time. It must be between 60 and 3600 seconds,
 with the default value being 3600.
         "JOB_PRESIGN_EXPIRES_IN_SEC": "Long"
         "EXECUTION_TIMEOUT": "300", // in seconds
      },
      "test": {
         "id": "Job_Execution",
         "version": "0.0.0"
      }
   }
]
```

Para obtener más información sobre la creación y el uso de documentos de trabajo, consulte Documento de trabajo.

Permisos y políticas

Puede utilizar las siguientes pruebas con el fin de determinar si las políticas asociadas a los certificados de sus dispositivos siguen las prácticas estándar recomendadas.

Por el momento, no WebSocket se admite MQTT over.

«Las políticas asociadas a los certificados de dispositivo no contienen caracteres comodín»

Valida si las políticas de permisos asociadas a un dispositivo siguen las prácticas recomendadas y no conceden al dispositivo más permisos de los necesarios.

Definición de caso de prueba de API:

> **Note**
>
> `EXECUTION_TIMEOUT` tiene un valor predeterminado de 1 minuto. Recomendamos establecer un tiempo de espera de al menos 30 segundos.

```
"tests":[
   {
        "name":"my_security_device_policies",
        "configuration": {
            // optional:
            "EXECUTION_TIMEOUT":"60"    // in seconds
        },
        "test": {
            "id": "Security_Device_Policies",
            "version": "0.0.0"
        }
    }
]
```

Pruebas de larga duración

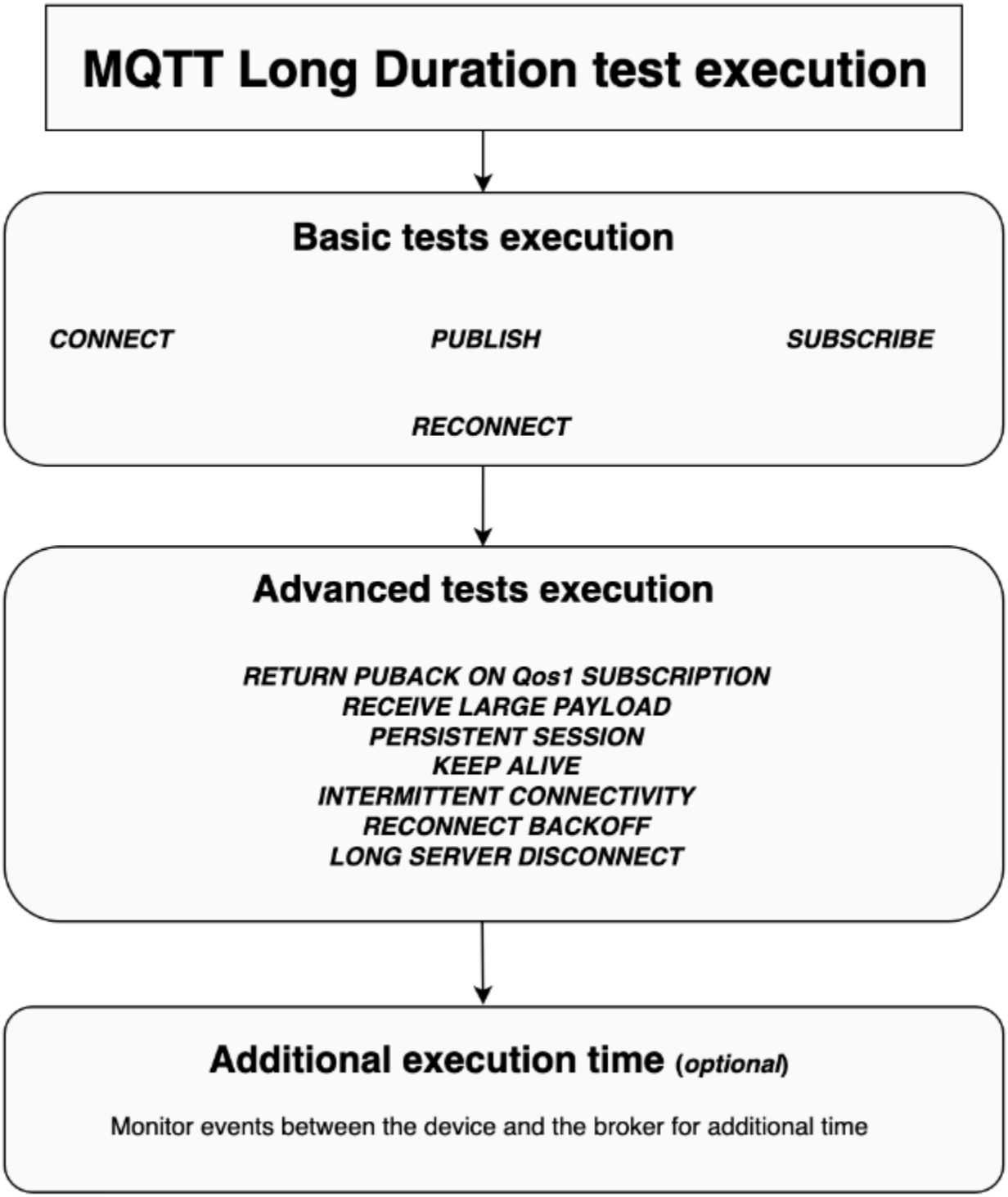
Las pruebas de larga duración son un nuevo conjunto de pruebas que monitoriza el comportamiento de un dispositivo cuando funciona durante períodos de tiempo más prolongados. En comparación con la ejecución de pruebas individuales que se centran en comportamientos específicos de un dispositivo, la prueba de larga duración examina el comportamiento del dispositivo en una variedad de escenarios del mundo real a lo largo de su vida útil. Device Advisor organiza las pruebas en el orden más eficiente posible. La prueba genera resultados y registros, incluido un registro resumido con métricas útiles sobre el rendimiento del dispositivo durante la prueba.

Caso de prueba MQTT de larga duración

En el caso de la prueba de larga duración de MQTT, el comportamiento del dispositivo se observa inicialmente en situaciones de casos deseables, como MQTT Connect, Subscribe, Publish y Reconnect. A continuación, se observa el dispositivo en varias situaciones de fallo complejas, como la interrupción de la reconexión MQTT, la desconexión prolongada del servidor y la conectividad intermitente.

Flujo de ejecución de casos de prueba de larga duración en MQTT

La ejecución de un caso de prueba de larga duración en MQTT consta de tres fases:



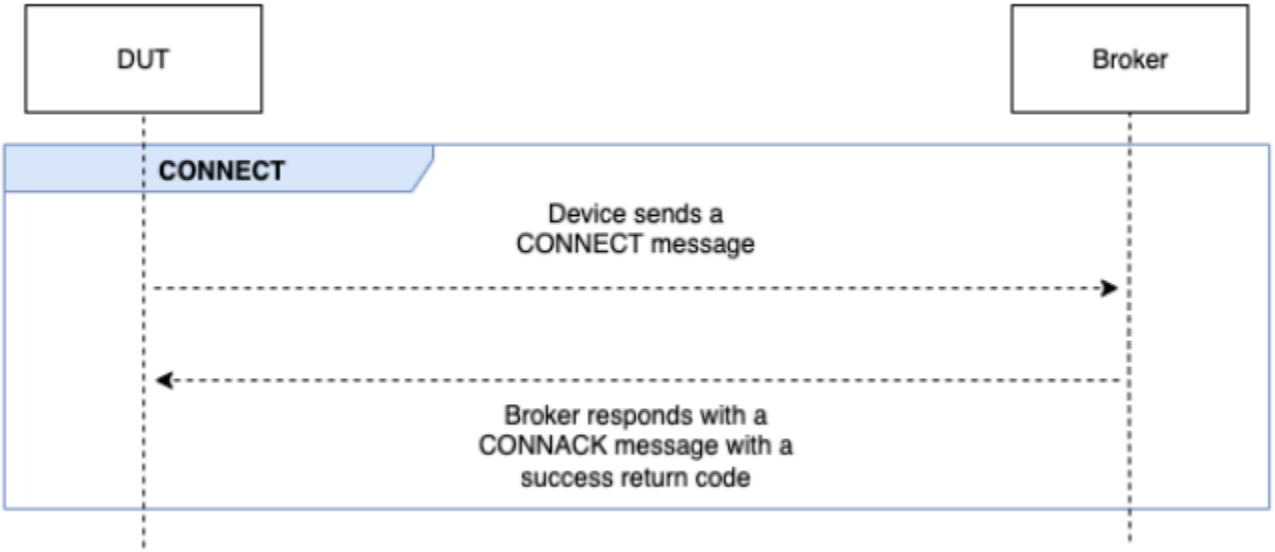
Ejecución de pruebas básicas

En esta fase, el caso de prueba ejecuta pruebas sencillas en paralelo. La prueba valida si el dispositivo tiene las operaciones seleccionadas en la configuración.

El conjunto de pruebas básicas puede incluir lo siguiente, en función de las operaciones seleccionadas:

CONNECT

Este escenario valida si el dispositivo puede establecer una conexión correcta con el agente.

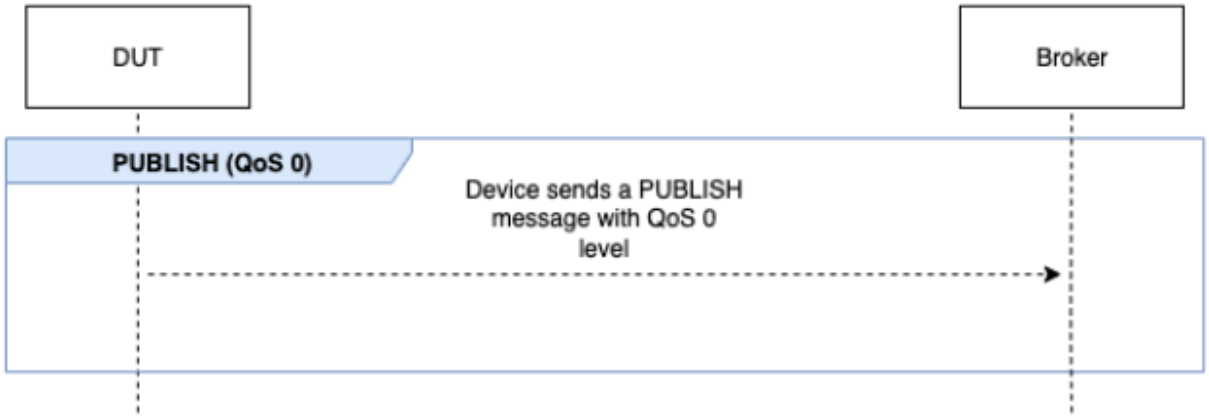


PUBLISH

Este escenario valida si el dispositivo publica correctamente con el agente.

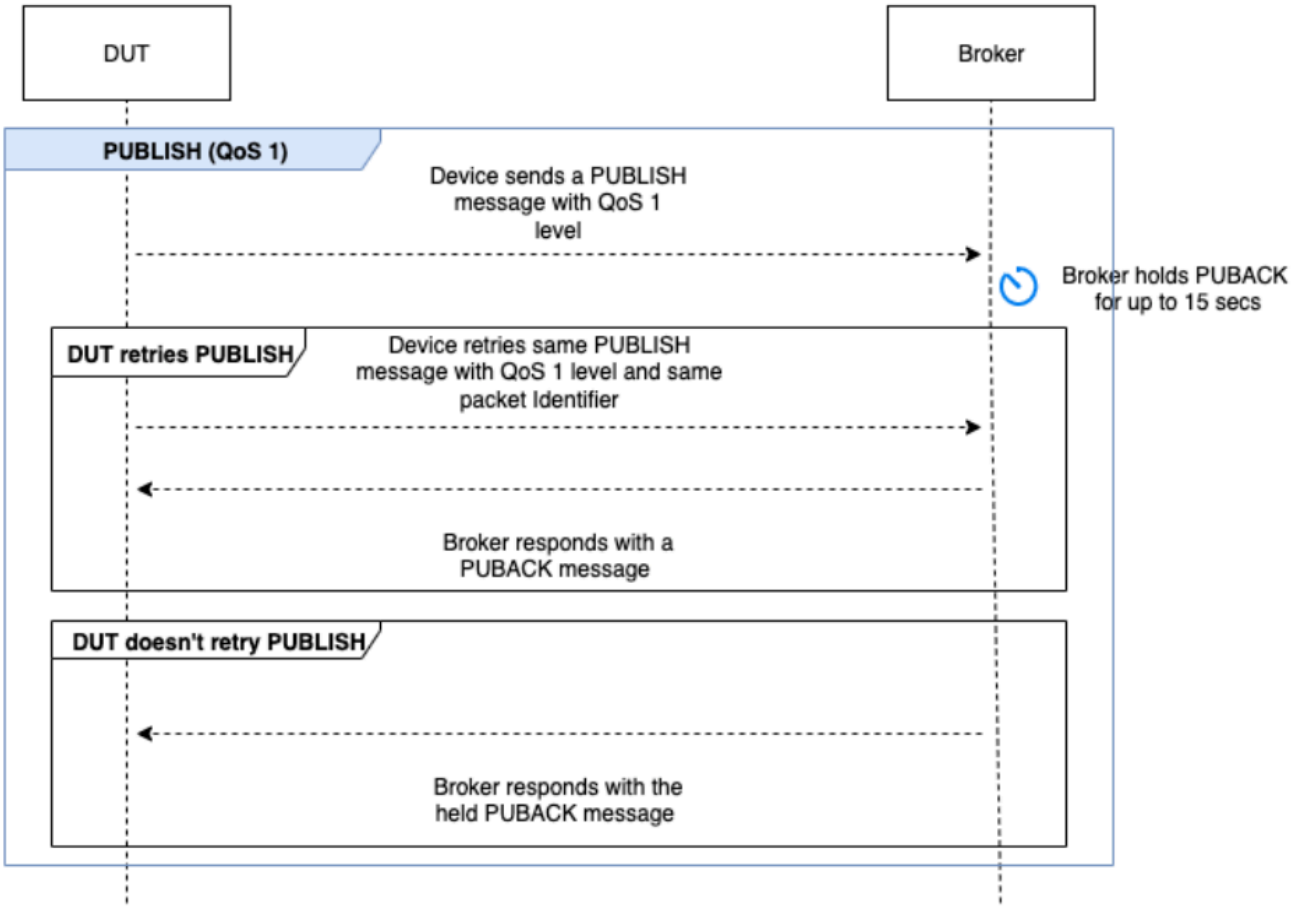
QoS 0

Este caso de prueba valida si el dispositivo envía correctamente un mensaje `PUBLISH` al agente durante una publicación con QoS 0. La prueba no espera a que el dispositivo reciba el mensaje `PUBACK`.



QoS 1

En este caso de prueba, se espera que el dispositivo envíe dos mensajes `PUBLISH` al agente con QoS 1. Tras el primer mensaje `PUBLISH`, el agente espera hasta 15 segundos antes de responder. El dispositivo debe volver a intentar enviar el mensaje `PUBLISH` original con el mismo identificador de paquete dentro del intervalo de 15 segundos. Si lo hace, el agente responde con un mensaje `PUBACK` y la prueba se valida. Si el dispositivo no vuelve a intentar el `PUBLISH`, se envía el `PUBACK` original al dispositivo y la prueba se marca como superada con advertencias, junto con un mensaje del sistema. Durante la ejecución de la prueba, si el dispositivo pierde la conexión y se vuelve a conectar, el escenario de prueba se restablecerá sin fallar y el dispositivo tendrá que volver a realizar los pasos del escenario de prueba.

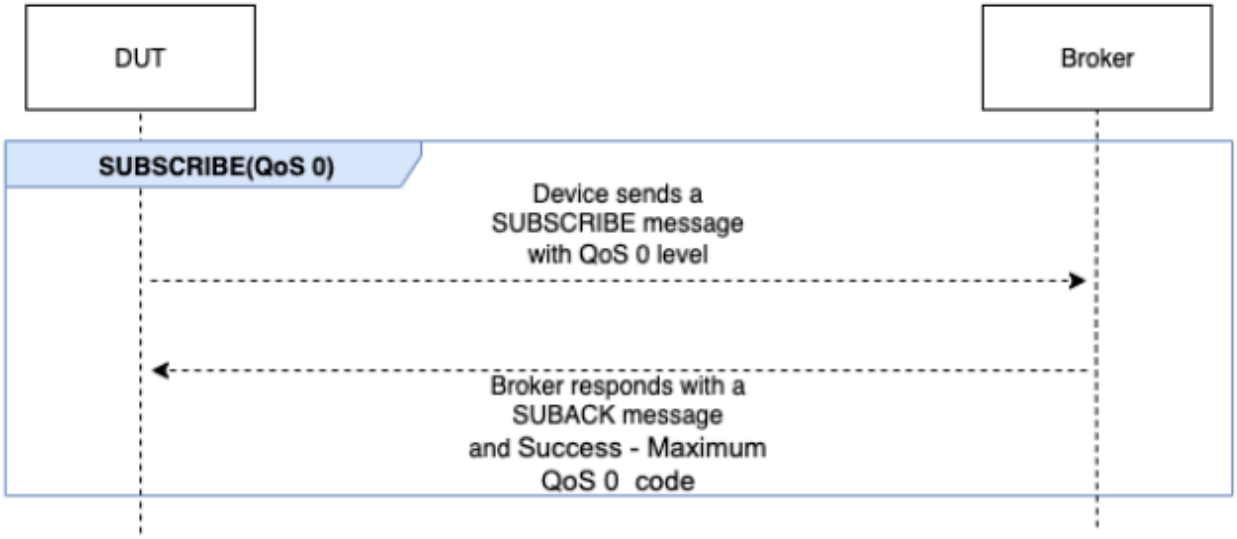


SUBSCRIBE

Este escenario valida si el dispositivo se suscribe correctamente con el agente.

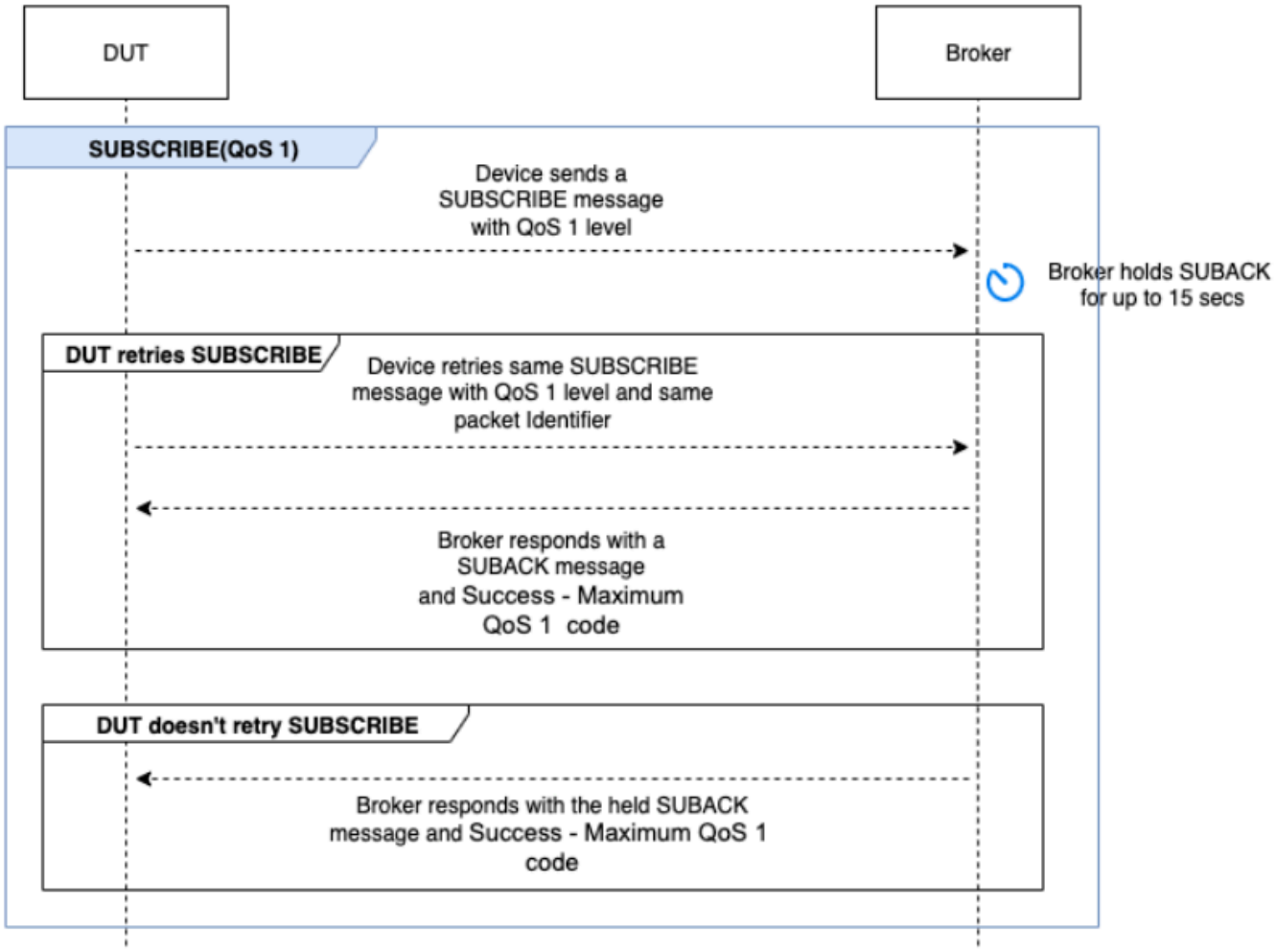
QoS 0

Este caso de prueba valida si el dispositivo envía correctamente un mensaje `SUBSCRIBE` al agente durante una suscripción con QoS 0. La prueba no espera a que el dispositivo reciba un mensaje SUBACK.



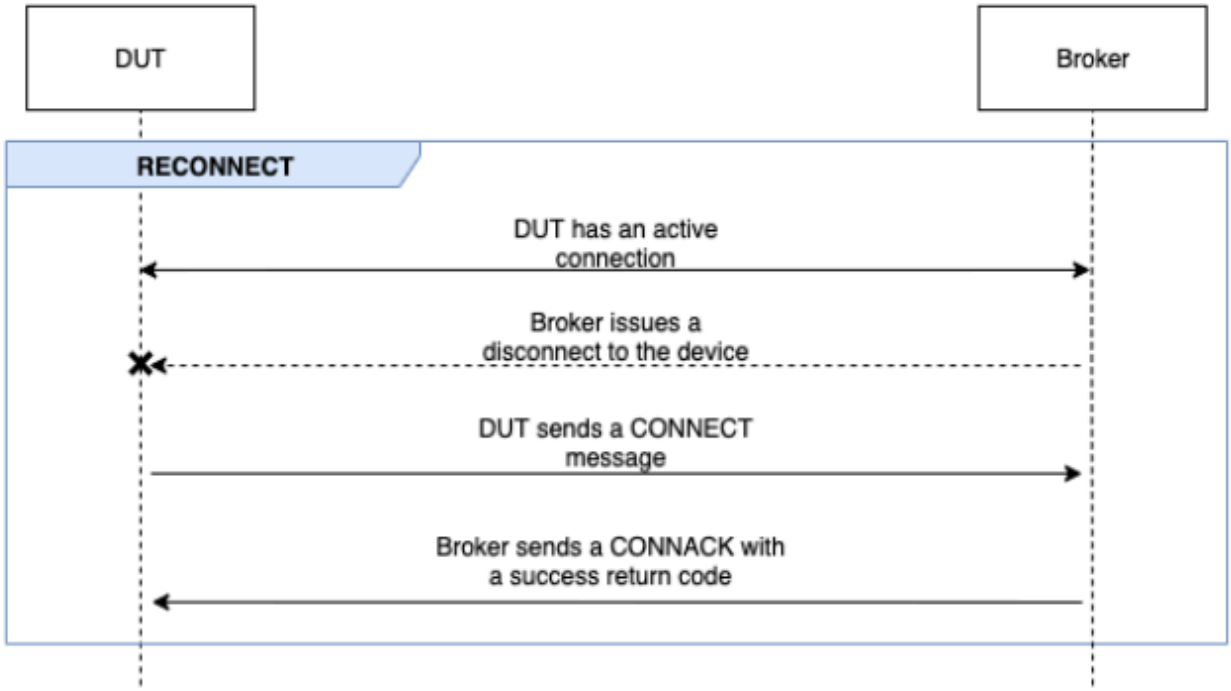
QoS 1

En este caso de prueba, se espera que el dispositivo envíe dos mensajes `SUBSCRIBE` al agente con QoS 1. Tras el primer mensaje `SUBSCRIBE`, el agente espera hasta 15 segundos antes de responder. El dispositivo debe volver a intentar enviar el mensaje `SUBSCRIBE` original con el mismo identificador de paquete dentro del intervalo de 15 segundos. Si lo hace, el agente responde con un mensaje `SUBACK` y la prueba se valida. Si el dispositivo no vuelve a intentar el `SUBSCRIBE`, se envía el `SUBACK` original al dispositivo y la prueba se marca como superada con advertencias, junto con un mensaje del sistema. Durante la ejecución de la prueba, si el dispositivo pierde la conexión y se vuelve a conectar, el escenario de prueba se restablecerá sin fallar y el dispositivo tendrá que volver a realizar los pasos del escenario de prueba.



RECONNECT

Este escenario valida si el dispositivo se vuelve a conectar correctamente con el agente después de desconectarse de una conexión correcta. Device Advisor no desconectará el dispositivo si se conectó más de una vez anteriormente durante el conjunto de pruebas. En su lugar, marcará la prueba como superada.



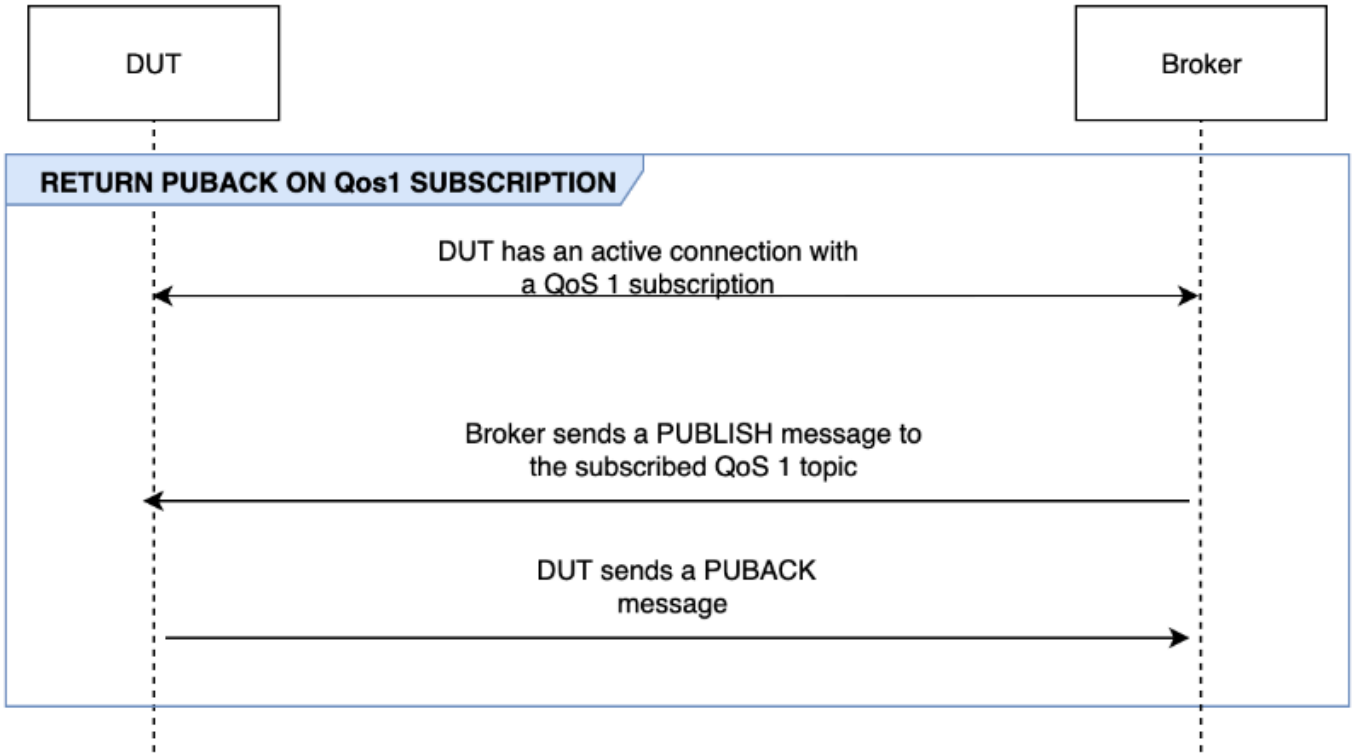
Ejecución de pruebas avanzadas

En esta fase, el caso de prueba ejecuta pruebas más complejas en serie para validar si el dispositivo sigue las prácticas recomendadas. Estas pruebas avanzadas están disponibles para su selección y pueden excluirse si no son necesarias. Cada prueba avanzada tiene su propio valor de tiempo de espera en función de lo que exija la situación.

RETURN PUBACK ON QoS 1 SUBSCRIPTION

> **Note**
>
> Seleccione este escenario únicamente si su dispositivo es capaz de realizar suscripciones de QoS 1.

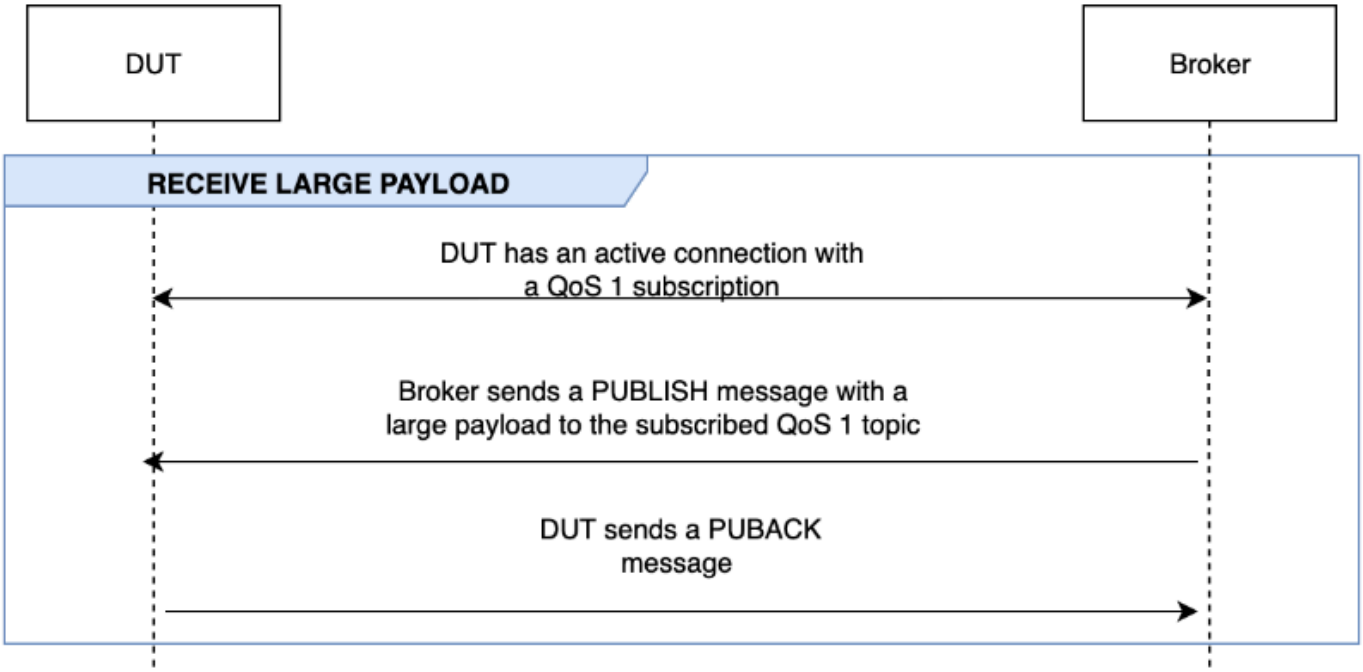
Este escenario se valida si, cuando el dispositivo se suscribe a un tema y recibe un mensaje `PUBLISH` del agente, devuelve un mensaje `PUBACK`.



RECEIVE LARGE PAYLOAD

> **Note**
>
> Seleccione este escenario únicamente si su dispositivo es capaz de realizar suscripciones de QoS 1.

Este escenario valida si el dispositivo responde con un mensaje `PUBACK` después de recibir un mensaje `PUBLISH` del agente sobre un tema de QoS 1 con una gran carga. El formato de la carga esperada se puede configurar mediante la opción `LONG_PAYLOAD_FORMAT`.



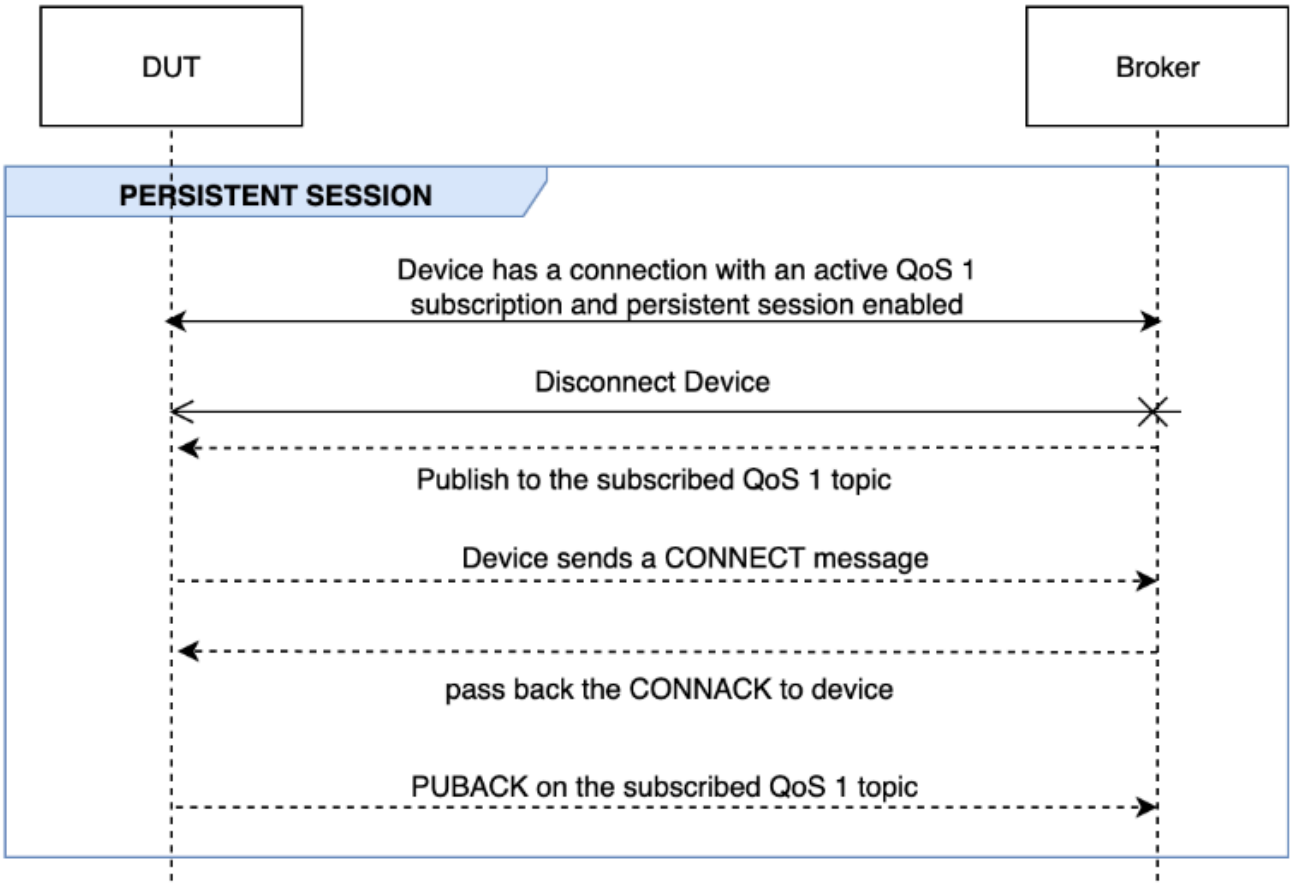
PERSISTENT SESSION

> **Note**
>
> Solo seleccione este escenario únicamente si su dispositivo es capaz de realizar suscripciones de QoS 1 y puede mantener una sesión persistente.

Este escenario valida el comportamiento del dispositivo a la hora de mantener sesiones persistentes. La prueba valida cuándo se cumplen las siguientes condiciones:

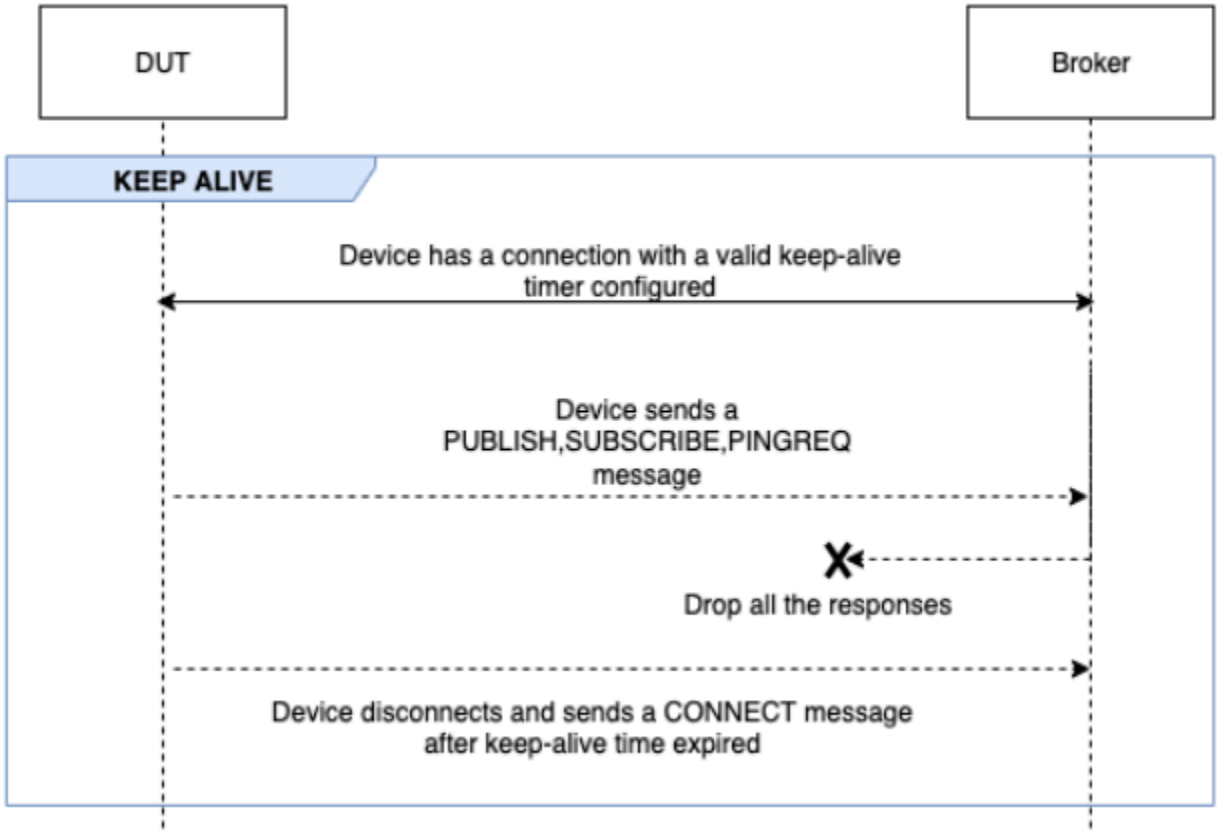
- El dispositivo se conecta al agente con una suscripción de QoS 1 activa y sesiones persistentes habilitadas.
- El dispositivo se desconecta correctamente del agente durante la sesión.
- El dispositivo se vuelve a conectar al agente y reanuda las suscripciones a los temas que lo activaron sin volver a suscribirse explícitamente a esos temas.
- El dispositivo recibe correctamente los mensajes almacenados por el agente sobre los temas a los que se ha suscrito y funciona según lo previsto.

Para obtener más información sobre las sesiones AWS IoT persistentes, consulte Uso de sesiones persistentes de MQTT.



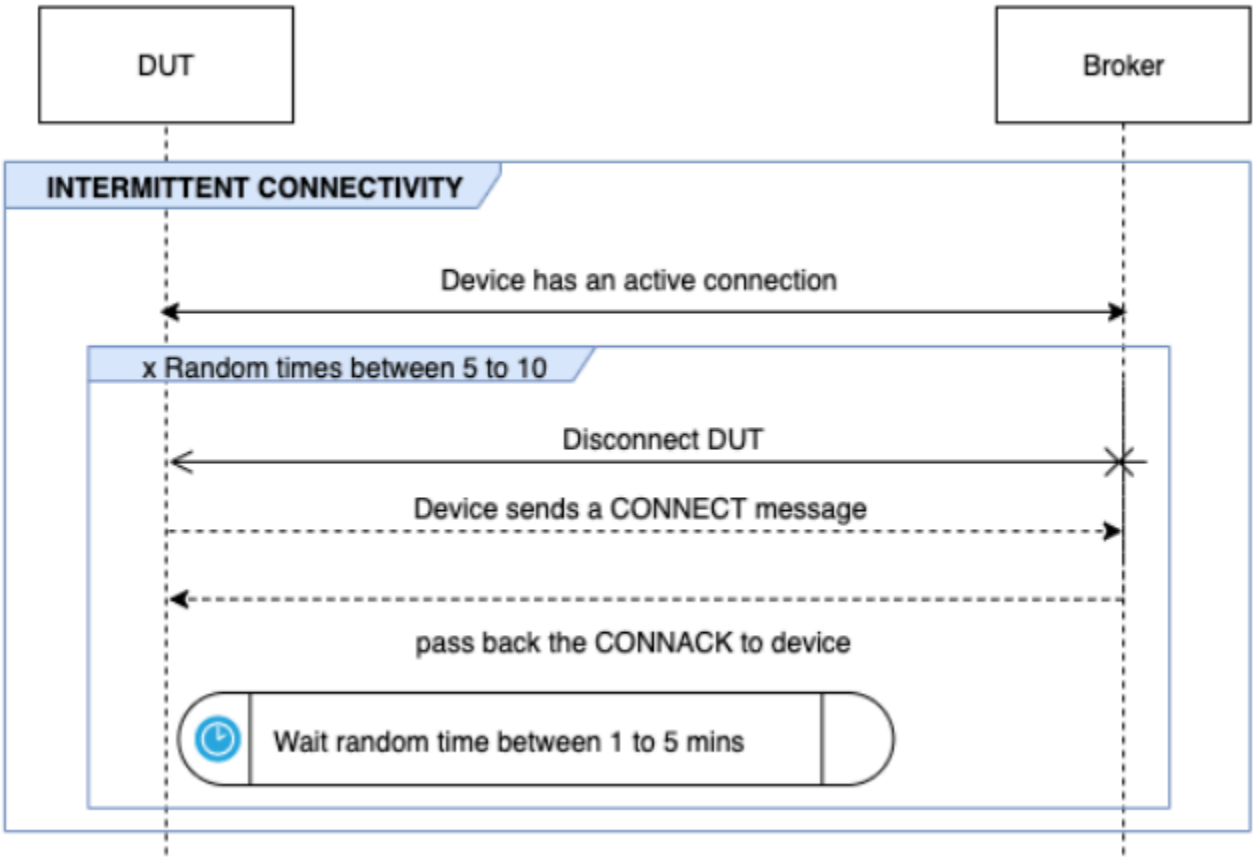
KEEP-ALIVE

Este escenario valida si el dispositivo se desconecta correctamente después de no recibir una respuesta de ping del agente. La conexión debe tener configurado un temporizador Keep-Alive válido. Como parte de esta prueba, el agente bloquea todas las respuestas enviadas para los mensajes `PUBLISH`, `SUBSCRIBE` y `PINGREQ`. También valida si el dispositivo sometido a prueba desconecta la conexión MQTT.



INTERMITTENT CONNECTIVITY

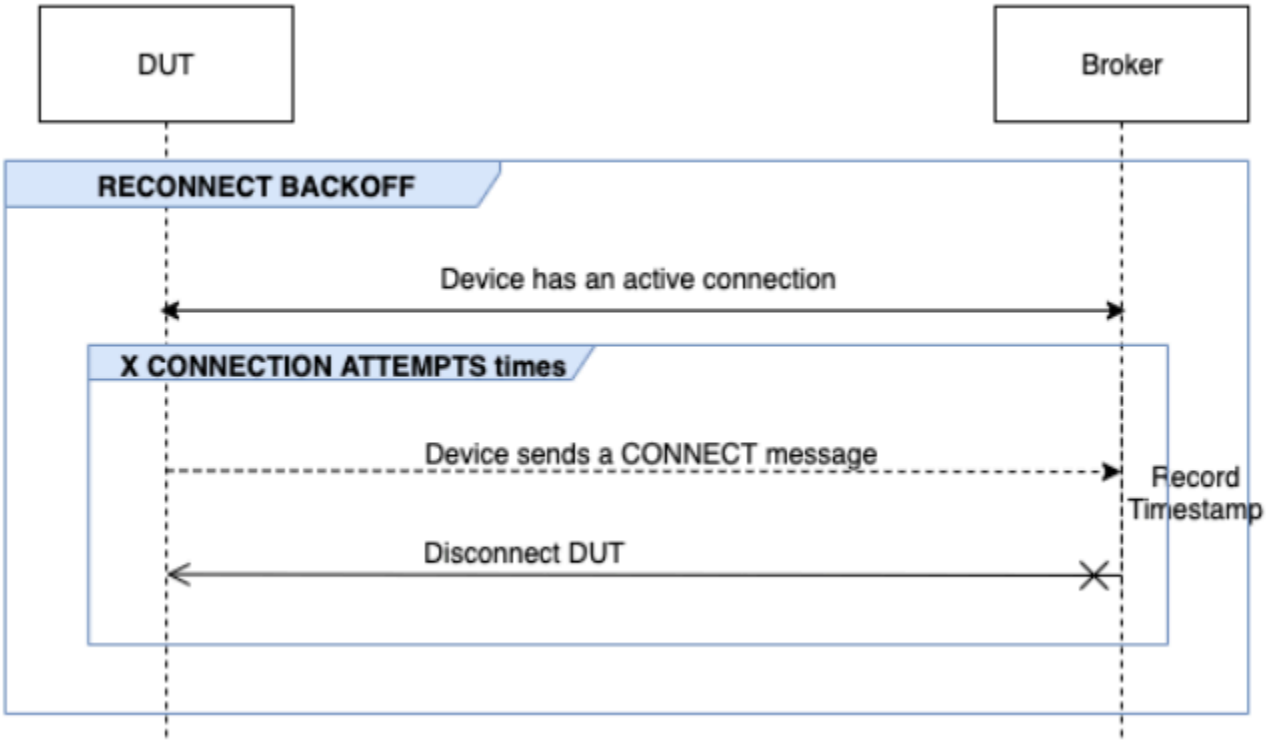
Este escenario valida si el dispositivo puede volver a conectarse al agente después de que el corredor desconecte el dispositivo a intervalos aleatorios durante un período de tiempo aleatorio.



RECONNECT BACKOFF

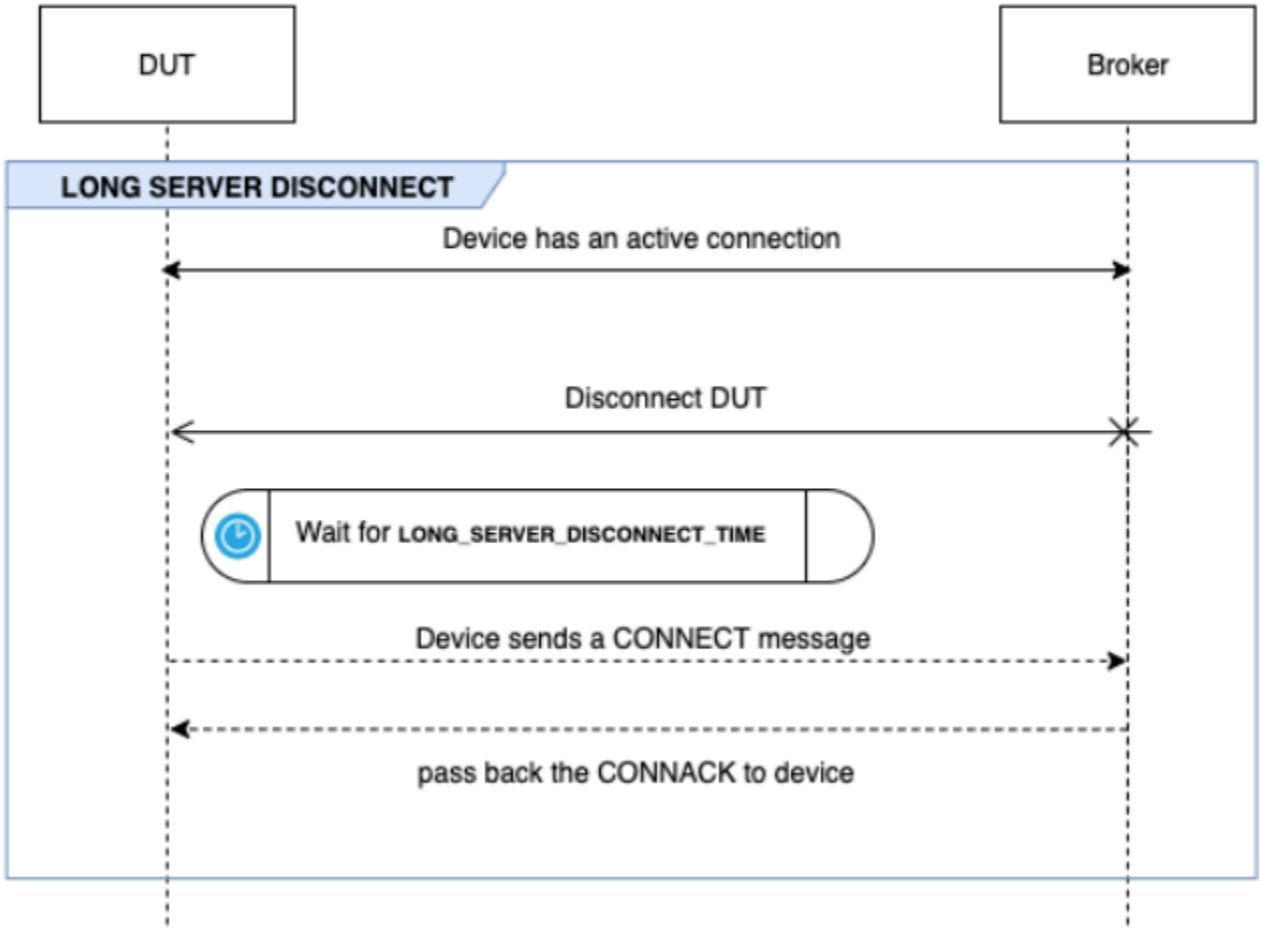
Este escenario valida si el dispositivo tiene implementado un mecanismo de inactividad cuando el agente se desconecta de él varias veces. Device Advisor indica que el tipo de retroceso es exponencial, fluctuante, lineal o constante. El número de intentos de retroceso se puede configurar mediante la opción `BACKOFF_CONNECTION_ATTEMPTS`. El valor predeterminado es 5. El valor se puede configurar entre 5 y 10.

Para superar esta prueba, le recomendamos que implemente el mecanismo de retroceso y fluctuación exponencial en el dispositivo sometido a prueba.



LONG SERVER DISCONNECT

Este escenario valida si el dispositivo puede volver a conectarse correctamente después de que el agente lo desconecte durante un período prolongado (hasta 120 minutos). El tiempo de desconexión del servidor se puede configurar mediante la opción `LONG_SERVER_DISCONNECT_TIME`. El valor predeterminado es 120 minutos. Este valor se puede configurar de 30 a 120 minutos.



Tiempo de ejecución adicional

El tiempo de ejecución adicional es el tiempo que espera la prueba después de completar todas las pruebas anteriores y antes de finalizar el caso de prueba. Los clientes utilizan este período de tiempo adicional para monitorizar y registrar todas las comunicaciones entre el dispositivo y el agente. El tiempo de ejecución adicional se puede configurar mediante la opción `ADDITIONAL_EXECUTION_TIME`. De forma predeterminada, esta opción está establecida en 0 minutos y puede ser de 0 a 120 minutos.

Opciones de configuración de las pruebas de larga duración de MQTT

Todas las opciones de configuración proporcionadas para la prueba de larga duración de MQTT son opcionales. Están disponibles las siguientes opciones:

OPERACIONES

La lista de operaciones que realiza el dispositivo, como `CONNECT`, `PUBLISH` y `SUBSCRIBE`. El caso de prueba ejecuta escenarios basados en las operaciones especificadas. Se supone que las operaciones que no se especifican son válidas.

```
{
"OPERATIONS": ["PUBLISH", "SUBSCRIBE"]
//by default the test assumes device can CONNECT
}
```

ESCENARIOS

En función de las operaciones seleccionadas, el caso de prueba ejecuta escenarios para validar el comportamiento del dispositivo. Existen dos tipos de escenarios:

- Los escenarios básicos son pruebas sencillas que validan si el dispositivo puede realizar las operaciones seleccionadas anteriormente como parte de la configuración. Se preseleccionan en función de las operaciones especificadas en la configuración. No se requiere ninguna entrada adicional en la configuración.
- Los escenarios avanzados son escenarios más complejos que se utilizan en el dispositivo para validar si el dispositivo sigue las prácticas recomendadas en condiciones reales. Son opcionales y se pueden transferir como una serie de escenarios a la entrada de configuración del conjunto de pruebas.

```
{
    "SCENARIOS": [      // list of advanced scenarios
                "PUBACK_QOS_1",
                "RECEIVE_LARGE_PAYLOAD",
                "PERSISTENT_SESSION",
                "KEEP_ALIVE",
                "INTERMITTENT_CONNECTIVITY",
                "RECONNECT_BACK_OFF",
                "LONG_SERVER_DISCONNECT"
    ]
}
```

BASIC_TESTS_EXECUTION_TIME_OUT:

El tiempo máximo que esperará el caso de prueba hasta que se completen todas las pruebas básicas. El valor predeterminado es 60 minutos. Este valor se puede configurar de 30 a 120 minutos.

LONG_SERVER_DISCONNECT_TIME:

El tiempo que tardó el caso de prueba en desconectar y volver a conectar el dispositivo durante la prueba de desconexión prolongada del servidor. El valor predeterminado es 60 minutos. Este valor se puede configurar de 30 a 120 minutos.

ADDITIONAL_EXECUTION_TIME:

La configuración de esta opción proporciona un período de tiempo después de que se hayan completado todas las pruebas para monitorizar los eventos entre el dispositivo y el agente. El valor predeterminado es 0 minutos. Este valor se puede configurar de 0 a 120 minutos.

BACKOFF_CONNECTION_ATTEMPTS:

Esta opción configura el número de veces que el caso de prueba desconecta el dispositivo. Esto se utiliza en la prueba de retroceso de reconexión. El valor predeterminado es 5 intentos. Este valor se puede configurar de 5 a 10.

LONG_PAYLOAD_FORMAT:

El formato de la carga del mensaje que el dispositivo espera cuando el caso de prueba se publique en un tema de QoS 1 suscrito por el dispositivo.

Definición de caso de prueba de API:

```
{
"tests":[
   {
      "name":"my_mqtt_long_duration_test",
      "configuration": {
         // optional
         "OPERATIONS": ["PUBLISH", "SUBSCRIBE"],
         "SCENARIOS": [
            "LONG_SERVER_DISCONNECT",
            "RECONNECT_BACK_OFF",
            "KEEP_ALIVE",
            "RECEIVE_LARGE_PAYLOAD",
```

```
            "INTERMITTENT_CONNECTIVITY",
            "PERSISTENT_SESSION",
         ],
         "BASIC_TESTS_EXECUTION_TIMEOUT": 60, // in minutes (60 minutes by default)
         "LONG_SERVER_DISCONNECT_TIME": 60,   // in minutes (120 minutes by default)
         "ADDITIONAL_EXECUTION_TIME": 60,     // in minutes (0 minutes by default)
         "BACKOFF_CONNECTION_ATTEMPTS": "5",
         "LONG_PAYLOAD_FORMAT":"{"message":"${payload}"}"
      },
      "test":{
         "id":"MQTT_Long_Duration",
         "version":"0.0.0"
      }
   }
 ]
}
```

Registro resumido de caso de prueba MQTT de larga duración

El caso de prueba de larga duración de MQTT dura más que los casos de prueba normales. Se proporciona un registro resumido independiente, en el que se enumeran los eventos importantes, como las conexiones de los dispositivos, la publicación y la suscripción durante la ejecución. Los detalles incluyen lo que se ha probado, lo que no se ha probado y lo que ha fallado. Al final del registro, la prueba incluye un resumen de todos los eventos ocurridos durante la ejecución del caso de prueba. Esto incluye:

- El temporizador Keep-Alive configurado en el dispositivo.
- Indicador de sesión persistente configurado en el dispositivo.
- El número de conexiones del dispositivo durante la ejecución de la prueba.
- El tipo de retroceso de reconexión del dispositivo, si está validado para la prueba de retroceso de reconexión.
- Los temas en los que el dispositivo publicó durante el caso de prueba.
- Los temas a los que el dispositivo se suscribió durante el caso de prueba.

AWS IoT Core Ubicación del dispositivo

Antes de utilizar la función de ubicación del AWS IoT Core dispositivo, consulta los términos y condiciones de esta función. Tenga en cuenta que AWS puede transmitir los parámetros de su solicitud de búsqueda de geolocalización, como los datos de ubicación utilizados para realizar las búsquedas y otra información, al proveedor de datos externo que elija, que puede estar fuera del Región de AWS que está utilizando actualmente. El proveedor externo y el solucionador que se van a utilizar se eligen en función de la carga útil de entrada recibida. Para obtener más informaci ón, consulte los Términos del servicio de AWS.

Utilice la ubicación del AWS IoT Core dispositivo para probar la ubicación de sus dispositivos de IoT mediante solucionadores de terceros. Los solucionadores son algoritmos proporcionados por proveedores externos que resuelven los datos de medición y estiman la ubicación del dispositivo. Al identificar la ubicación de sus dispositivos, puede rastrearlos y depurarlos sobre el terreno para solucionar cualquier problema.

Los datos de medición recopilados de varios orígenes se resuelven y la información de geolocalización se presenta como una carga de GeoJSON. El formato GeoJSON es un formato que se utiliza para codificar estructuras de datos geográficos. La carga útil contiene las coordenadas de latitud y longitud de la ubicación del dispositivo, que se basan en el sistema de coordenadas del Sistema Geodésico Mundial (). WGS84

Temas

- Tipos de mediciones y solucionadores
- Cómo funciona la ubicación del AWS IoT Core dispositivo
- Cómo utilizar la ubicación del dispositivo AWS IoT Core
- Resolver la ubicación de los dispositivos IoT
- Resolución de la ubicación del dispositivo mediante los temas de ubicación AWS IoT Core del dispositivo (MQTT).
- Solucionadores de ubicación y carga útil del dispositivo

Tipos de mediciones y solucionadores

AWS IoT Core Device Location colabora con proveedores externos para resolver los datos de medición y proporcionar una ubicación estimada del dispositivo. En la siguiente tabla se muestran los tipos de medición y los solucionadores de ubicación de terceros, así como información sobre los dispositivos compatibles. Para obtener información sobre los dispositivos LoRa WAN y cómo configurar la ubicación de los dispositivos para ellos, consulte Configurar la posición de los recursos LoRa WAN.

> **Note**
>
> Los dispositivos IoT generales y los dispositivos Sidewalk pueden usar los temas de MQTT sobre la ubicación del dispositivo para obtener la información de ubicación. Para los tipos de medición de direcciones IP, móviles y Wi-Fi, si los dispositivos publican los datos de medición en los temas reservados en el formato GeoJSON definido, la ubicación del AWS IoT Core dispositivo puede resolver la ubicación del dispositivo. Para el tipo de medición GNSS, el dispositivo debe tener el LR11xx chip para escanear los datos de medición y obtener la información de ubicación resuelta mediante el solucionador GNSS. Para obtener información sobre cómo obtener información de ubicación para los dispositivos LoRa WAN, consulte Configuración de la posición de los recursos LoRa WAN en la AWS IoT documentación sobre redes inalámbricas.

Tipos de mediciones y solucionadores

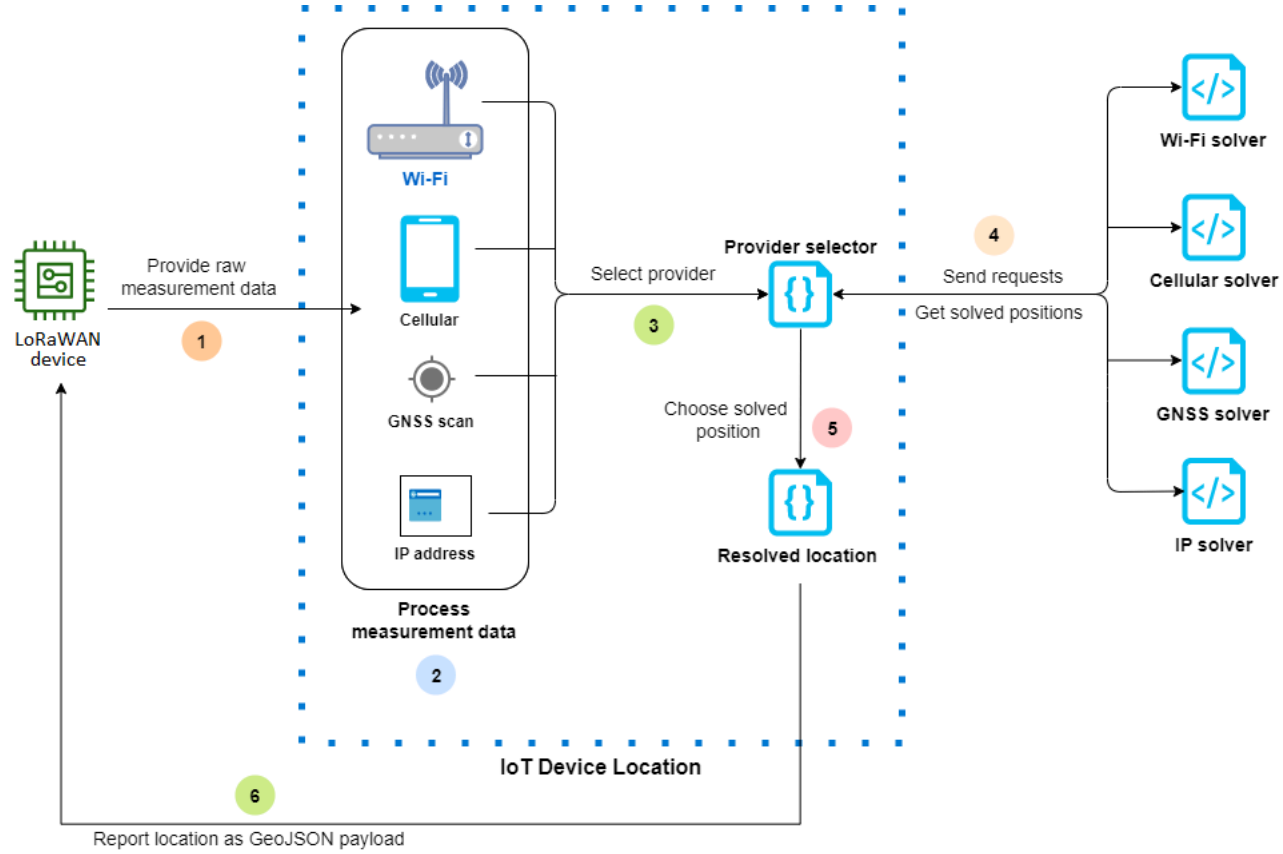
Tipo de medición	Solucionadores de terceros	Dispositivos compatibles
Puntos de acceso Wi-Fi	Solucionador basado en Wi-Fi	Dispositivos de IoT generales, LoRa WAN y dispositivos Amazon Sidewalk
Torres de radio de telefonía móvil: datos GSM, LTE, CDMA, SCDMA, WCMDA y TD-SCDMA	Solucionador basado en dispositivos móviles	Dispositivos de IoT generales, LoRa WAN y dispositivos Amazon Sidewalk
Dirección IP	Solucionador de búsqueda inversa de IP	Cualquier dispositivo de IoT que se conecte a través de TCP/IP

Tipo de medición	Solucionadores de terceros	Dispositivos compatibles
Datos de escaneo GNSS (mensajes NAV)	Solucionador GNSS	Dispositivos de IoT generales, LoRa WAN y dispositivos Amazon Sidewalk
Bluetooth de bajo consumo (BLE)	Solucionador basado en BLE	Dispositivos Amazon Sidewalk

Para obtener más información sobre los solucionadores de ubicación y ejemplos que muestran la carga del dispositivo para los distintos tipos de medición, consulte Solucionadores de ubicación y carga útil del dispositivo.

Cómo funciona la ubicación del AWS IoT Core dispositivo

El siguiente diagrama muestra cómo la ubicación del AWS IoT Core dispositivo recopila los datos de medición y resuelve la información de ubicación de los dispositivos.



Los siguientes pasos muestran cómo funciona la ubicación del AWS IoT Core dispositivo.

1. Recibir los datos de medición

   Los datos de medición sin procesar relacionados con la ubicación del dispositivo se envían primero desde el dispositivo. Los datos de medición se especifican como una carga JSON.

2. Procesamiento de datos de medición

   Los datos de medición se procesan y la ubicación del AWS IoT Core dispositivo elige los datos de medición que se van a utilizar, que pueden ser datos de Wi-Fi, datos móviles, escaneos GNSS o direcciones IP.

3. Elección del solucionador

   El solucionador de terceros se elige en función de los datos de medición. Por ejemplo, si los datos de medición contienen información sobre Wi-Fi y direcciones IP, elija el solucionador de Wi-Fi y el solucionador de búsqueda inversa de IP.

4. Obtención de la ubicación resuelta

   Se envía una solicitud de API a los proveedores del solucionador para solicitar que resuelvan la ubicación. AWS IoT Core A continuación, Device Location obtiene la información de geolocalización estimada de los solucionadores.

5. Elección de la ubicación resuelta

   Se comparan la información de ubicación resuelta y su precisión, y AWS IoT Core Device Location selecciona los resultados de geolocalización con la mayor precisión.

6. Información sobre la ubicación de salida

   La información de geolocalización se le envía como carga de GeoJSON. La carga útil contiene las coordenadas WGS84 geográficas, la información de precisión, los niveles de confianza y la marca temporal en la que se obtuvo la ubicación resuelta.

Cómo utilizar la ubicación del dispositivo AWS IoT Core

Los siguientes pasos muestran cómo usar la ubicación AWS IoT Core del dispositivo.

1. Proporcione datos de medición

   Especifique los datos de medición sin procesar relacionados con la ubicación del dispositivo como carga de JSON. Para recuperar los datos de medición de la carga útil, vaya a los registros del dispositivo o utilice CloudWatch Registros y copie la información de los datos de la carga útil. La carga de JSON debe contener uno o varios tipos de medición de datos. Para ver ejemplos que muestran el formato de carga de varios solucionadores, consulte Solucionadores de ubicación y carga útil del dispositivo.

2. Resolver la información de ubicación

   Utilice la página de ubicación del dispositivo en la AWS IoT consola o la operación de la GetPositionEstimateAPI, transfiera los datos de medición de la carga útil y resuelva la ubicación del dispositivo. AWS IoT Core A continuación, Device Location selecciona el solucionador con la mayor precisión e informa de la ubicación del dispositivo. Para obtener más información, consulte Resolver la ubicación de los dispositivos IoT.

3. Copiar la información de ubicación

   Compruebe la información de geolocalización que resolvió AWS IoT Core Device Location y que se notificó como carga útil de GeoJSON. Puede copiar la carga útil para utilizarla con sus aplicaciones y otras aplicaciones. Servicio de AWS Por ejemplo, puedes enviar tus datos de ubicación geográfica a Amazon Location Service mediante la acción de la Ubicación AWS IoT regla.

Los siguientes temas muestran cómo utilizar la ubicación AWS IoT Core del dispositivo y ejemplos de la carga útil de la ubicación del dispositivo.

- Resolver la ubicación de los dispositivos IoT
- Solucionadores de ubicación y carga útil del dispositivo

Resolver la ubicación de los dispositivos IoT

Use la ubicación del AWS IoT Core dispositivo para decodificar los datos de medición de sus dispositivos y resuelva la ubicación del dispositivo con soluciones de terceros. La ubicación resuelta se genera como una carga de GeoJSON con las coordenadas geográficas y la información de precisión. Puede resolver la ubicación de su dispositivo desde la AWS IoT consola, la API AWS IoT inalámbrica o. AWS CLI

Temas

- Resolver la ubicación del dispositivo (consola)
- Resolver la ubicación del dispositivo (API)
- Solución de errores al resolver la ubicación

Resolver la ubicación del dispositivo (consola)

Para resolver la ubicación del dispositivo (consola)

1. Ve a la página de ubicación del dispositivo en la AWS IoT consola.
2. Obtenga los datos de medición de la carga útil de los registros de su dispositivo o de CloudWatch los registros e introdúzcalos en la sección Posición de resolución mediante carga útil.

   El siguiente código muestra un ejemplo de carga de JSON. La carga contiene datos de medición de telefonía móvil y Wi-Fi. Si la carga contiene tipos adicionales de datos de medición, se utilizará el solucionador con la mayor precisión. Para obtener más información y ejemplos de cargas, consulte the section called “Solucionadores de ubicación y carga útil del dispositivo”.

   > **Note**
   >
   > La carga de JSON debe contener al menos un tipo de datos de medición.

```
{
    "Timestamp": 1664313161,
    "Ip":{
        "IpAddress": "54.240.198.35"
    },
    "WiFiAccessPoints": [{
        "MacAddress": "A0:EC:F9:1E:32:C1",
        "Rss": -77
    }],
    "CellTowers": {
        "Gsm": [{
            "Mcc": 262,
            "Mnc": 1,
            "Lac": 5126,
            "GeranCid": 16504,
```

```
            "GsmLocalId": {
                "Bsic": 6,
                "Bcch": 82
            },
            "GsmTimingAdvance": 1,
            "RxLevel": -110,
            "GsmNmr": [{
                "Bsic": 7,
                "Bcch": 85,
                "RxLevel": -100,
                "GlobalIdentity": {
                    "Lac": 1,
                    "GeranCid": 1
                }
            }]
        }],
        "Wcdma": [{
            "Mcc": 262,
            "Mnc": 7,
            "Lac": 65535,
            "UtranCid": 14674663,
            "WcdmaNmr": [{
                    "Uarfcndl": 10786,
                    "UtranCid": 14674663,
                    "Psc": 149
                },
                {
                    "Uarfcndl": 10762,
                    "UtranCid": 14674663,
                    "Psc": 211
                }
            ]
        }],
        "Lte": [{
            "Mcc": 262,
            "Mnc": 2,
            "EutranCid": 2898945,
            "Rsrp": -50,
            "Rsrq": -5,
            "LteNmr": [{
                    "Earfcn": 6300,
                    "Pci": 237,
                    "Rsrp": -60,
                    "Rsrq": -6,
```

```
                    "EutranCid": 2898945
                },
                {
                    "Earfcn": 6300,
                    "Pci": 442,
                    "Rsrp": -70,
                    "Rsrq": -7,
                    "EutranCid": 2898945
                }
            ]
        }]
    }
}
```

3. Para resolver la información de ubicación, elija Resolver.

   La información de ubicación es de tipo blob y se devuelve como una carga que utiliza el formato GeoJSON, que es un formato que se utiliza para codificar estructuras de datos geográficos. La carga contiene:

   - Las coordenadas WGS84 geográficas, que incluyen la información de latitud y longitud. También puede incluir información sobre la altitud.
   - El tipo de información de ubicación notificada, por ejemplo Punto. Un tipo de ubicación de punto representa la ubicación como WGS84 latitud y longitud, codificadas como un punto GeoJSON.
   - La información de precisión horizontal y vertical, que indica la diferencia en metros entre la información de ubicación estimada por los solucionadores y la ubicación real del dispositivo.
   - El nivel de confianza, que indica la incertidumbre en la respuesta a la estimación de ubicación. El valor predeterminado es 0,68, lo que indica una probabilidad del 68 % de que la ubicación real del dispositivo esté dentro del radio de incertidumbre de la ubicación estimada.
   - La ciudad, el estado, el país y el código postal donde se encuentra el dispositivo. Esta información solo se mostrará cuando se utilice el solucionador de búsqueda inversa de IP.
   - La información de marca temporal, que corresponde a la fecha y la hora a las que se ha resuelto la ubicación. Utiliza el formato de marca temporal de Unix.

   El siguiente código muestra un ejemplo de carga de GeoJSON devuelta al resolver la ubicación.

> **Note**
>
> Si la ubicación del AWS IoT Core dispositivo informa de errores al intentar resolver la ubicación, puedes solucionar los errores y resolver la ubicación. Para obtener más información, consulte Solución de errores al resolver la ubicación.


```
{
    "coordinates": [
        13.376076698303223,
        52.51823043823242
    ],
    "type": "Point",
    "properties": {
        "verticalAccuracy": 45,
        "verticalConfidenceLevel": 0.68,
        "horizontalAccuracy": 303,
        "horizontalConfidenceLevel": 0.68,
        "country": "USA",
        "state": "CA",
        "city": "Sunnyvalue",
        "postalCode": "91234",
        "timestamp": "2022-11-18T12:23:58.189Z"
    }
}
```


4. Ve a la sección Ubicación de los recursos y verifica la información de geolocalización proporcionada por AWS IoT Core la ubicación del dispositivo. Puede copiar la carga útil para utilizarla con otras aplicaciones y aplicaciones. Servicio de AWS Por ejemplo, puede utilizar la Ubicación para enviar sus datos de ubicación geográfica a Amazon Location Service.

Resolver la ubicación del dispositivo (API)

Para resolver la ubicación del dispositivo mediante la API AWS IoT inalámbrica, utilice la operación GetPositionEstimateAPI o el comando get-position-estimateCLI. Especifique los datos de medición de la carga como entrada y ejecute la operación de la API para resolver la ubicación del dispositivo.

> **Note**
>
> La operación de la API `GetPositionEstimate` no almacena información sobre ningún dispositivo o estado, y no se puede utilizar para recuperar datos de ubicación históricos. Realiza una operación única que resuelve los datos de medición y genera la ubicación estimada. Para recuperar la información de ubicación, debe especificar la información de carga cada vez que realice esta operación de la API.

En el siguiente comando se muestra un ejemplo de cómo hacer esto con esta operación de la API.

> **Note**
>
> Al ejecutar el comando de la CLI `get-position-estimate`, debe especificar el archivo JSON de salida como primera entrada. Este archivo JSON almacenará la información de ubicación estimada obtenida como respuesta de la CLI en formato GeoJSON. Por ejemplo, el siguiente comando almacena la información de ubicación en el *`locationout.json`* archivo.

```
aws iotwireless get-position-estimate locationout.json \
    --ip IpAddress=""54.240.198.35"" \
    --wi-fi-access-points \
        MacAddress="A0:EC:F9:1E:32:C1",Rss=-75 \
        MacAddress="A0:EC:F9:15:72:5E",Rss=-67
```

Este ejemplo incluye los puntos de acceso Wi-Fi y la dirección IP como tipos de medición. AWS IoT Core Device Location elige entre el solucionador de Wi-Fi y el solucionador de búsqueda inversa de IP, y selecciona el solucionador con mayor precisión.

La ubicación resuelta es una carga que utiliza el formato GeoJSON, que es un formato que se utiliza para codificar estructuras de datos geográficos. A continuación, se guarda en el archivo. *`locationout.json`* La carga útil contiene las coordenadas de WGS84 latitud y longitud, la información de precisión y nivel de confianza, el tipo de datos de ubicación y la marca de tiempo en la que se resolvió la ubicación.

```
{
    "coordinates": [
        13.37704086303711,
```

```
        52.51865005493164
    ],
    "type": "Point",
    "properties": {
        "verticalAccuracy": 707,
        "verticalConfidenceLevel": 0.68,
        "horizontalAccuracy": 389,
        "horizontalConfidenceLevel": 0.68,
        "country": "USA",
        "state": "CA",
        "city": "Sunnyvalue",
        "postalCode": "91234",
        "timestamp": "2022-11-18T14:03:57.391Z"
    }
}
```

Solución de errores al resolver la ubicación

Al intentar resolver la ubicación, es posible que aparezca alguno de los siguientes códigos de error. AWS IoT Core La ubicación del dispositivo puede generar un error al utilizar la operación de la `GetPositionEstimate` API o bien hacer referencia al número de línea correspondiente al error en la AWS IoT consola.

- Error 400

  Este error indica que la ubicación del dispositivo no puede validar el formato JSON de la carga útil AWS IoT Core del dispositivo. El error puede producirse por las siguientes razones:

  - Los datos de medición de JSON tienen un formato incorrecto.
  - La carga contiene solo la información de la marca temporal.
  - Los parámetros de los datos de medición, como la dirección IP, no son válidos.

  Para resolver este error, compruebe si su JSON tiene el formato correcto y si contiene datos de uno o más tipos de medición como entrada. Si la dirección IP no es válida, para obtener información sobre cómo puede proporcionar una dirección IP válida para resolver el error, consulte Solucionador de búsqueda inversa de IP.

- Error 403

  Este error indica que no tienes los permisos para realizar la operación de la API o para usar la AWS IoT consola para recuperar la ubicación del dispositivo. Para resolver este error, compruebe que dispone de los permisos necesarios para realizar esta acción. Este error puede producirse

si la Consola de administración de AWS sesión o el token de AWS CLI sesión han caducado. Para resolver este error, actualiza el token de sesión para usarlo o cierra sesión Consola de administración de AWS y AWS CLI, a continuación, inicia sesión con tus credenciales.

- Error 404

  Este error indica que AWS IoT Core Device Location no encontró ni resolvió ninguna información de ubicación. El error puede deberse a casos como la insuficiencia de datos en la entrada de datos de medición. Por ejemplo:

  - La dirección MAC o la información de la torre de telefonía móvil no son suficientes.
  - La dirección IP no está disponible para buscar y recuperar la ubicación.
  - La carga de GNSS no es suficiente.

  Para resolver el error en estos casos, compruebe si los datos de medición contienen suficiente información necesaria para determinar la ubicación del dispositivo.

- Error 500

  Este error indica que se produjo una excepción del servidor interno cuando AWS IoT Core Device Location intentó resolver la ubicación. Para intentar corregir este error, actualice la sesión y vuelva a intentar enviar los datos de medición que se van a resolver.

Resolución de la ubicación del dispositivo mediante los temas de ubicación AWS IoT Core del dispositivo (MQTT).

Puede utilizar los temas de MQTT reservados para obtener la información más reciente sobre la ubicación de sus dispositivos con la función de ubicación del AWS IoT Core dispositivo.

Formato de los temas de MQTT de ubicación de dispositivos

Los temas reservados para la ubicación de AWS IoT Core dispositivos utilizan el siguiente prefijo:

`$aws/device_location/{customer_device_id}/`

Para crear un tema completo, primero reemplace *`customer_device_id`* por el identificador único que utiliza para identificar el dispositivo. Le recomendamos que especifique el`WirelessDeviceId`, por ejemplo, para los dispositivos LoRa WAN y Sidewalk*`thingName`*, y si el dispositivo está registrado como algo. AWS IoT A continuación, agregue el tema al código auxiliar del tema, como

`get_position_estimate` o `get_position_estimate/accepted`, tal y como se muestra en la siguiente sección.

> **Note**
>
> *`{customer_device_id}`* solo puede contener letras, números y guiones. Al suscribirse a temas de ubicación de dispositivos, solo puede usar el signo más (+) como carácter comodín. Por ejemplo, puede usar el comodín + para que *`{customer_device_id}`* obtenga la información de ubicación de sus dispositivos. Cuando se suscribas al tema de `$aws/device_location/+/get_position_estimate/accepted`, se publicará un mensaje con la información de ubicación de los dispositivos que coincidan con cualquier ID de dispositivo si se ha resuelto correctamente.

Los siguientes son los temas reservados que se utilizan para interactuar con la ubicación AWS IoT Core del dispositivo.

Temas de MQTT de ubicación de dispositivos

Topic	Operaciones permitidas	Description (Descripción)
$aws/device_location/ / get_position_estimate *customer_device_id*	Publish	Un dispositivo publica sobre este tema para que los datos de medición sin procesar escaneados se resuelvan según la ubicación del dispositivo. AWS IoT Core
$aws/device_location/ / get_position_estimate/ accepted *customer_ device_id*	Suscribirse	AWS IoT Core La ubicación del dispositi vo publica la información de ubicación en este tema cuando resuelve correctamente la ubicación del dispositivo.
$aws/device_location/ / get_position_estimate/ rejected *customer_ device_id*	Suscribirse	AWS IoT Core La ubicación del dispositivo publica la información de error en este tema cuando no logra resolver la ubicación del dispositivo.

Política de los temas de MQTT de ubicación de dispositivos

Para recibir mensajes relacionados con la ubicación del dispositivo, el dispositivo debe utilizar una política que le permita conectarse a la puerta de enlace del AWS IoT dispositivo y suscribirse a los temas de MQTT.

A continuación se muestra un ejemplo de la política necesaria para recibir mensajes de los distintos temas.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/device_location/
device-123/get_position_estimate"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/device_location/
device-123/get_position_estimate/accepted",
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/device_location/
device-123/get_position_estimate/rejected"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/
device_location/device-123/get_position_estimate/accepted",
```

```
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/
device_location/device-123/get_position_estimate/rejected"
            ]
        }
    ]
}
```

Temas y carga de Device Location

A continuación, se muestran los temas sobre la ubicación de los AWS IoT Core dispositivos, el formato de la carga útil de sus mensajes y un ejemplo de política para cada tema.

Temas

- /get_position_estimate
- /get_position_estimate/accepted
- /get_position_estimate/rejected

/get_position_estimate

Publique un mensaje sobre este tema para obtener los datos de medición sin procesar del dispositivo para resolverlos por ubicación del AWS IoT Core dispositivo.

```
$aws/device_location/customer_device_id/get_position_estimate
```

AWS IoT Core La ubicación del dispositivo responde publicando en una /get_position_estimate/accepted u otra/get_position_estimate/rejected.

> **Note**
>
> El mensaje publicado en este tema debe ser una carga de JSON válida. Si el mensaje de entrada no está en un formato JSON válido, no recibirá ninguna respuesta. Para obtener más información, consulte Carga útil del mensaje.

Carga útil de mensaje

El formato de carga útil del mensaje sigue una estructura similar a la del cuerpo de la solicitud de operación de la API AWS IoT inalámbrica, `GetPositionEstimate`. Contiene:

- Una cadena `Timestamp` opcional, que corresponde a la fecha y la hora en que se resolvió la ubicación. La cadena `Timestamp` puede tener una longitud mínima de 1 y máxima de 10.
- Una cadena `MessageId` opcional, que se puede utilizar para asignar la solicitud a la respuesta. Si especifica esta cadena, el mensaje publicado en los temas `get_position_estimate/accepted` o `get_position_estimate/rejected` contendrá este `MessageId`. La cadena `MessageID` puede tener una longitud mínima de 1 y máxima de 256.
- Los datos de medición del dispositivo que contiene uno o más de los siguientes tipos de medición:
  - `WiFiAccessPoint`
  - `CellTowers`
  - `IpAddress`
  - `Gnss`

A continuación se muestra un ejemplo de la carga del mensaje.

```
{
    "Timestamp": "1664313161",
    "MessageId": "ABCD1",
    "WiFiAccessPoints":  [
        {
            "MacAddress": "A0:EC:F9:1E:32:C1",
            "Rss": -66
        }
    ],
    "Ip":{
        "IpAddress": "54.192.168.0"
    },
    "Gnss":{
      "Payload":"8295A614A2029517F4F77C0A7823B161A6FC57E25183D96535E3689783F6CA48",
      "CaptureTime":1354393948
    }
}
```

Ejemplo de política de

A continuación, mostramos un ejemplo de la política requerida:

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

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/device_location/
device-123/get_position_estimate"
            ]
        }
    ]
}
```

/get_position_estimate/accepted

AWS IoT Core Device Location publica una respuesta a este tema al devolver la información de ubicación resuelta del dispositivo. La información de ubicación se devuelve en formato GeoJSON.

```
$aws/device_location/customer_device_id/get_position_estimate/accepted
```

A continuación se muestra la carga del mensaje y un ejemplo de política.

Carga útil del mensaje

A continuación, se muestra un ejemplo de la carga del mensaje en formato GeoJSON. Si especificó una `MessageId` en sus datos de medición sin procesar y AWS IoT Core Device Location resolvió la información de ubicación correctamente, la carga útil del mensaje devolverá la misma `MessageId` información.

```
{
    "coordinates": [
        13.37704086303711,
        52.51865005493164
    ],
    "type": "Point",
    "properties": {
        "verticalAccuracy": 707,
        "verticalConfidenceLevel": 0.68,
        "horizontalAccuracy": 389,
        "horizontalConfidenceLevel": 0.68,
```

```
        "country": "USA",
        "state": "CA",
        "city": "Sunnyvalue",
        "postalCode": "91234",
        "timestamp": "2022-11-18T14:03:57.391Z",
        "messageId": "ABCD1"
    }
}
```

Ejemplo de política de

A continuación, mostramos un ejemplo de la política requerida:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/
device_location/device-123/get_position_estimate/accepted"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/device_location/
device-123/get_position_estimate/accepted"
            ]
        }
    ]
}
```

/get_position_estimate/rejected

AWS IoT Core La ubicación del dispositivo publica una respuesta de error sobre este tema cuando no logra resolver la ubicación del dispositivo.

```
$aws/device_location/customer_device_id/get_position_estimate/rejected
```

A continuación se muestra la carga del mensaje y un ejemplo de política. Para obtener información acerca de los errores, consulte Solución de errores al resolver la ubicación.

Carga útil del mensaje

A continuación se muestra un ejemplo de la carga útil del mensaje que proporciona el código y el mensaje de error, que indican por qué AWS IoT Core Device Location no pudo resolver la información de ubicación. Si especificó una `MessageId` al proporcionar los datos de medición sin procesar y la ubicación del AWS IoT Core dispositivo no pudo resolver la información de ubicación, se devolverá la misma `MessageId` información en la carga útil del mensaje.

```
{
    "errorCode": 500,
    "errorMessage":"Internal server error",
    "messageId": "ABCD1"
}
```

Ejemplo de política de

A continuación, mostramos un ejemplo de la política requerida:

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Subscribe"
      ],
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/device_location/
device-123/get_position_estimate/rejected"
```

```
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/device_location/device-123/
get_position_estimate/rejected"
      ]
    }
  ]
}
```

Solucionadores de ubicación y carga útil del dispositivo

Los solucionadores de ubicación son algoritmos que se pueden usar para resolver la ubicación de sus dispositivos de IoT. AWS IoT Core La ubicación del dispositivo admite los siguientes solucionadores de ubicaciones. Verá ejemplos del formato de carga JSON para estos tipos de mediciones, los dispositivos compatibles con el solucionador y cómo se resuelve la ubicación.

Para resolver la ubicación del dispositivo, especifique uno o más de estos tipos de datos de medición. Se devolverá una única ubicación resuelta para todos los datos de medición combinados.

Temas

- Solucionador basado en Wi-Fi
- Solucionador basado en dispositivos móviles
- Solucionador de búsqueda inversa de IP
- Solucionador GNSS
- Solucionador BLE para dispositivos compatibles con Amazon Sidewalk

Solucionador basado en Wi-Fi

Utilice el solucionador basado en Wi-Fi para resolver la ubicación utilizando la información escaneada de los puntos de acceso Wi-Fi. El solucionador es compatible con la tecnología WLAN y se puede usar para calcular la ubicación del dispositivo para dispositivos IoT generales que se

conectan a través de TCP/IP, dispositivos inalámbricos LoRa WAN y dispositivos habilitados para Amazon Sidewalk.

Los dispositivos LoRa WAN deben tener el chipset LoRa Edge, que puede decodificar la información de escaneo Wi-Fi entrante. LoRa Edge es una plataforma de consumo ultrabajo que integra un LoRa transceptor de largo alcance, un escáner GNSS multiconstelación y un escáner MAC inalámbrico pasivo destinado a aplicaciones de geolocalización. Cuando se recibe un mensaje de enlace ascendente del dispositivo, los datos del escaneo de Wi-Fi se envían a la ubicación del AWS IoT Core dispositivo y la ubicación se estima en función de los resultados del escaneo de Wi-Fi. Luego, la información descodificada se pasa al solucionador basado en Wi-Fi para recuperar la información de ubicación.

Para obtener más información sobre cómo resolver los datos de ubicación de los dispositivos compatibles con Amazon Sidewalk, consulta la guía para AWS IoT Core para Amazon Sidewalk desarrolladores.

Ejemplo de carga de un solucionador basado en Wi-Fi

El siguiente código muestra un ejemplo de la carga JSON del dispositivo que contiene los datos de medición. Cuando AWS IoT Core Device Location recibe estos datos como entrada, envía una solicitud HTTP al proveedor del solucionador para resolver la información de ubicación. Para recuperar la información, especifique los valores de la dirección MAC y el RSS (intensidad de la señal recibida). Para ello, proporcione la carga útil JSON con este formato o utilice el parámetro WiFiAccessPointsobject de la operación de la GetPositionEstimateAPI.

```
{
    "Timestamp": 1664313161,    // optional
    "WiFiAccessPoints": [
        {
            "MacAddress": "A0:EC:F9:1E:32:C1",  // required
            "Rss": -75                          // required
        }
    ]
}
```

Solucionador basado en dispositivos móviles

Puede utilizar el solucionador basado en dispositivos móviles para resolver la ubicación mediante los datos de medición obtenidos de las torres de radio de telefonía móvil. El solucionador es compatible

con las siguientes tecnologías. Se obtiene una única información de ubicación resuelta, incluso si incluye datos de medición de alguna o todas estas tecnologías.

- GSM
- CDMA
- WCDMA
- TD-SCDMA
- LTE

Ejemplos de carga de un solucionador basado en dispositivos móviles

El siguiente código muestra ejemplos de la carga de JSON del dispositivo que contiene los datos de medición móviles. Cuando AWS IoT Core Device Location recibe estos datos como entrada, envía una solicitud HTTP al proveedor del solucionador para resolver la información de ubicación. Para recuperar la información, debe proporcionar la carga útil JSON con este formato en la consola o especificar valores para el CellTowersparámetro de la operación de la GetPositionEstimateAPI. Puede proporcionar los datos de medición especificando los valores de los parámetros mediante una o todas estas tecnologías móviles.

LTE (evolución a largo plazo)

Al utilizar estos datos de medición, debe especificar información como la red y el código de país de la red móvil, así como parámetros adicionales opcionales, incluida la información sobre el identificador local. El siguiente código muestra un ejemplo del formato de la carga. Para obtener información sobre estos parámetros, consulte Objeto LTE.

```
{
    "Timestamp": 1664313161,          // optional
    "CellTowers": {
        "Lte": [
        {
          "Mcc": int,                  // required
          "Mnc": int,                  // required
          "EutranCid": int,            // required. Make sure that you use int for
 EutranCid.
          "Tac": int,                  // optional
          "LteLocalId": {              // optional
              "Pci": int,              // required
```

```
            "Earfcn": int,            // required
        },
        "LteTimingAdvance": int,      // optional
        "Rsrp": int,                  // optional
        "Rsrq": float,                // optional
        "NrCapable": boolean,         // optional
        "LteNmr": [                   // optional
             {
                 "Pci": int,          // required
                 "Earfcn": int,       // required
                 "EutranCid": int,    // required
                 "Rsrp": int,         // optional
                 "Rsrq": float        // optional
             }
          ]
       }
     ]
  }
}
```

GSM (Sistema global de comunicaciones móviles)

Cuando utilice estos datos de medición, deberá especificar información como la red y el código de país de la red móvil, la información de la estación base y parámetros adicionales opcionales. El siguiente código muestra un ejemplo del formato de la carga. Para obtener información sobre estos parámetros, consulte Objeto GSM.

```
{
    "Timestamp": 1664313161,          // optional
    "CellTowers": {
        "Gsm": [
        {
          "Mcc": int,                 // required
          "Mnc": int,                 // required
          "Lac": int,                 // required
          "GeranCid": int,            // required
          "GsmLocalId": {             // optional
              "Bsic": int,            // required
              "Bcch": int,            // required
          },
          "GsmTimingAdvance": int,    // optional
          "RxLevel": int,             // optional
          "GsmNmr": [                 // optional
```

```
            {
                "Bsic": int,            // required
                "Bcch": int,            // required
                "RxLevel": int,         // optional
                "GlobalIdentity": {
                    "Lac": int,         // required
                    "GeranCid": int     // required
                }
             }
          ]
       }
    ]
}
```

CDMA (Acceso múltiple por división de código)

Cuando utilice estos datos de medición, deberá especificar información como la potencia de la señal y la información de identificación, la información de la estación base y parámetros adicionales opcionales. El siguiente código muestra un ejemplo del formato de la carga. Para obtener información sobre estos parámetros, consulte Objeto CDMA.

```
{
    "Timestamp": 1664313161,                // optional
    "CellTowers": {
        "Cdma": [
        {
            "SystemId": int,                // required
            "NetworkId": int,               // required
            "BaseStationId": int,           // required
            "RegistrationZone": int,        // optional
            "CdmaLocalId": {                // optional
              "PnOffset": int,              // required
              "CdmaChannel": int,           // required
            },
            "PilotPower": int,              // optional
            "BaseLat": float,               // optional
            "BaseLng": float,               // optional
            "CdmaNmr": [                    // optional
                {
                    "PnOffset": int,        // required
                    "CdmaChannel": int,     // required
                    "PilotPower": int,      // optional
                    "BaseStationId": int    // optional
```

```
                }
            ]
        }
    ]
  }
}
```

WCDMA (Acceso múltiple por división de código de banda ancha)

Cuando utilice estos datos de medición, deberá especificar información como la red y el código del país, la potencia de la señal y la información de identificación, la información de la estación base y parámetros adicionales opcionales. El siguiente código muestra un ejemplo del formato de la carga. Para obtener información sobre estos parámetros, consulte Objeto CDMA.

```
{
    "Timestamp": 1664313161,           // optional
    "CellTowers": {
        "Wcdma": [
        {
          "Mcc": int,                   // required
          "Mnc": int,                   // required
          "UtranCid": int,              // required
          "Lac": int,                   // optional
          "WcdmaLocalId": {             // optional
              "Uarfcndl": int,          // required
              "Psc": int,               // required
          },
          "Rscp": int,                  // optional
          "Pathloss": int,              // optional
          "WcdmaNmr": [                 // optional
                {
                  "Uarfcndl": int,      // required
                  "Psc": int,           // required
                  "UtranCid": int,      // required
                  "Rscp": int,          // optional
                  "Pathloss": int,      // optional
                }
            ]
          }
      ]
    }
}
```

TD-SCDMA (Acceso múltiple por división de código síncrono por división de tiempo)

Cuando utilice estos datos de medición, deberá especificar información como la red y el código del país, la potencia de la señal y la información de identificación, la información de la estación base y parámetros adicionales opcionales. El siguiente código muestra un ejemplo del formato de la carga. Para obtener información sobre estos parámetros, consulte Objeto CDMA.

```
{
    "Timestamp": 1664313161,           // optional
    "CellTowers": {
        "Tdscdma": [
        {
          "Mcc": int,                   // required
          "Mnc": int,                   // required
          "UtranCid": int,              // required
          "Lac": int,                   // optional
          "TdscdmaLocalId": {           // optional
              "Uarfcn": int,            // required
              "CellParams": int,        // required
          },
          "TdscdmaTimingAdvance": int,  // optional
          "Rscp": int,                  // optional
          "Pathloss": int,              // optional
          "TdscdmaNmr": [               // optional
                {
                  "Uarfcn": int,        // required
                  "CellParams": int,    // required
                  "UtranCid": int,      // optional
                  "Rscp": int,          // optional
                  "Pathloss": int,      // optional
                }
             ]
         }
      ]
   }
}
```

Solucionador de búsqueda inversa de IP

Puede utilizar el solucionador de búsqueda inversa de IP para resolver la ubicación utilizando la dirección IP como entrada. El solucionador puede obtener la información de ubicación de los dispositivos aprovisionados. AWS IoT Especifique la información de la dirección IP mediante

un formato que sea el patrón IPv6 estándar IPv4 o el patrón comprimido IPv6 hexadecimal. A continuación, obtendrá la estimación de la ubicación resuelta, que incluye información adicional, como la ciudad y el país donde se encuentra el dispositivo.

> **Note**
>
> Al utilizar la búsqueda inversa de IP, aceptas no utilizarla con el fin de identificar o localizar un hogar o una dirección postal específicos.

Ejemplo de carga de un solucionador de búsqueda inversa de IP

El siguiente código muestra un ejemplo de la carga JSON del dispositivo que contiene los datos de medición. Cuando AWS IoT Core Device Location recibe la información de la dirección IP de los datos de medición, busca esta información en la base de datos del proveedor del solucionador, que luego se utiliza para resolver la información de ubicación. Para recuperar la información, proporcione la carga útil JSON con este formato o especifique valores para el parámetro IP de la operación de la GetPositionEstimateAPI.

> **Note**
>
> Cuando se utiliza este solucionador, además de las coordenadas, también se indica la ciudad, el estado, el país y el código postal en los que se encuentra el dispositivo. Para ver un ejemplo, consulta Resolver la ubicación del dispositivo (consola).

```
{
    "Timestamp": 1664313161,
    "Ip":{
        "IpAddress":"54.240.198.35"
      }
}
```

Solucionador GNSS

Utilice el solucionador GNSS (Sistema global de navegación por satélite) para recuperar la ubicación del dispositivo utilizando la información contenida en los mensajes de los resultados del escaneo GNSS o en los mensajes NAV. Si lo desea, puede proporcionar información de asistencia GNSS

adicional, lo que reduce el número de variables que el solucionador debe utilizar para buscar señales. Al proporcionar esta información de asistencia, que incluye la posición, la altitud y la información sobre el tiempo de captura y la precisión, el solucionador puede identificar fácilmente los satélites a la vista y calcular la ubicación del dispositivo.

Este solucionador se puede usar con dispositivos LoRa WAN y dispositivos habilitados para Amazon Sidewalk, así como con otros dispositivos que se hayan aprovisionado con dispositivos de AWS IoT que se hayan aprovisionado. AWS IoT Para los dispositivos IoT generales, si los dispositivos admiten la estimación de ubicación mediante GNSS, cuando se recibe la información de escaneo GNSS del dispositivo, los transceptores resuelven la información de ubicación. En el caso de los dispositivos LoRa WAN, los dispositivos deben tener el chipset Edge. LoRa Cuando se recibe un mensaje de enlace ascendente desde el dispositivo, los datos del escaneo GNSS se envían a la LoRa WAN o a AWS IoT AWS IoT Amazon Sidewalk, y la ubicación se estima en función de los resultados del escaneo de los transceptores.

Ejemplo de carga útil de un solucionador GNSS

El siguiente código muestra un ejemplo de la carga JSON del dispositivo que contiene los datos de medición. Cuando AWS IoT Core Device Location recibe la información de escaneo GNSS que contiene la carga útil incluida en los datos de medición, utiliza los transceptores y cualquier información de asistencia adicional incluida para buscar señales y resolver la información de ubicación. Para recuperar la información, proporcione la carga útil JSON con este formato o especifique valores para el parámetro GNSS de la operación de la API. GetPositionEstimate

> **Note**
>
> Para que la ubicación del AWS IoT Core dispositivo pueda resolver la ubicación del dispositivo, debe eliminar el byte de destino de la carga útil.

```
{
    "Timestamp": 1664313161,                // optional
    "Gnss": {
        "AssistAltitude": number,           // optional
        "AssistPosition": [ number ],       // optional
        "CaptureTime": number,              // optional
        "CaptureTimeAccuracy": number,      // optional
        "Payload": "string",                // required
        "Use2DSolver": boolean              // optional
```

```
    }
}
```

Solucionador BLE para dispositivos compatibles con Amazon Sidewalk

> **Note**
>
> En el caso de la ubicación basada en Bluetooth Low Energy, AWS IoT devuelve las coordenadas de ubicación en función de la ubicación aproximada de los Sidewalk Gateways cercanos que estén conectados a Amazon Sidewalk y tengan habilitada la función de búsqueda de comunidades. Los datos de ubicación de Gateway son AWS contenido y se le proporcionan únicamente con el fin de ayudarlo a localizar los dispositivos que están conectados a Amazon Sidewalk, y solo debe usar los datos para ese propósito. Solo debe usar los datos de ubicación y acceder a ellos a través de la interfaz y la funcionalidad que generalmente ponemos a su disposición, y no debe intentar volver a identificar, aplicar ingeniería inversa o volver a mapear los datos de ubicación de Gateway que le proporcionemos.

Los dispositivos finales de Amazon Sidewalk que se conecten a través de ellos AWS IoT Core para Amazon Sidewalk pueden resolver sus datos de ubicación mediante mensajes de enlace ascendente BLE, Wi-Fi o GNSS publicados por el dispositivo final de Amazon Sidewalk. Para obtener más información, consulte la guía para desarrolladores.AWS IoT Core para Amazon Sidewalk

Mensajes de los eventos

Esta sección contiene información sobre los mensajes que se publican AWS IoT cuando se actualizan o cambian cosas o trabajos. Para obtener información sobre el AWS IoT Events servicio que le permite crear detectores para monitorear sus dispositivos en busca de fallas o cambios en el funcionamiento y activar acciones cuando se produzcan, consulte AWS IoT Events.

Cómo se generan los mensajes de eventos

AWS IoT publica mensajes de eventos cuando se producen determinados eventos. Por ejemplo, el registro genera eventos cuando se agregan, actualizan o eliminan objetos. Cada evento provoca que se envíe un único mensaje de evento. Los mensajes de evento se publican a través de MQTT con una carga JSON. El contenido de la carga depende del tipo de evento.

> **Note**
>
> Se garantiza que los mensajes de eventos se publican una vez. Es posible que se publiquen más de una. No se garantiza el orden de los mensajes de eventos.

Política de recepción de mensajes de eventos

Para recibir mensajes de eventos, su dispositivo debe utilizar una política adecuada que le permita conectarse a la puerta de enlace del AWS IoT dispositivo y suscribirse a los temas de los eventos de MQTT. También debe suscribirse a los filtros de temas adecuados.

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:us-east-1:123456789012:/$aws/events/*"
            ]
        }
    ]
}
```

Habilite los eventos para AWS IoT

Para que los suscriptores de los temas reservados puedan recibir mensajes, debes habilitar los mensajes de eventos desde Consola de administración de AWS o mediante la API o la CLI. Para obtener información sobre los mensajes de eventos que administran las distintas opciones, consulte la tabla de opciones de configuración de AWS IoT eventos.

- Para habilitar los mensajes de eventos, vaya a la pestaña Configuración de la AWS IoT consola y, a continuación, en la sección Mensajes basados en eventos, elija Administrar eventos. Puede especificar los eventos que desea administrar.
- Para controlar qué tipos de eventos se publican mediante la API o la CLI, llame a la UpdateEventConfigurationsAPI o utilice el comando update-event-configurations CLI. Por ejemplo:

```
aws iot update-event-configurations --event-configurations "{\"THING\":{\"Enabled\":
 true}}"
```

> **Note**
>
> Todas las comillas (") van precedidas de barras diagonales invertidas (\).

Puede obtener la configuración de eventos actual llamando a la DescribeEventConfigurationsAPI o mediante el comando describe-event-configurations CLI. Por ejemplo:

```
aws iot describe-event-configurations
```

Tabla de opciones de configuración de eventos de AWS IoT

Categoría de evento (AWS IoT Consola: configuración: mensajes basados en eventos)	Valor de la clave **`eventConfigurations`** (AWS CLI/API)	Tema del mensaje del evento
(Solo se puede configurar mediante la AWS CLI/API)	`CA_CERTIFICATE`	`$aws/events/certificates/registered/` *`caCertificateId`*
(Solo se puede configurar mediante la AWS CLI/API)	`CERTIFICATE`	`$aws/events/presence/connected/` *`clientId`*
(Solo se puede configurar mediante la AWS CLI/API)	`CERTIFICATE`	`$aws/events/presence/disconnected/` *`clientId`*
(Solo se puede configurar mediante la AWS CLI/API)	`CERTIFICATE`	`$aws/events/subscriptions/subscribed/` *`clientId`*
(Solo se puede configurar mediante la AWS CLI/API)	`CERTIFICATE`	`$aws/events/subscriptions/unsubscribed/` *`clientId`*
Trabajo completado, cancelado	`JOB`	`$aws/events/job/`*`jobID`*`/canceled`
Trabajo completado, cancelado	`JOB`	`$aws/events/job/`*`jobID`*`/cancellation_in_progress`
Trabajo completado, cancelado	`JOB`	`$aws/events/job/`*`jobID`*`/completed`
Trabajo completado, cancelado	`JOB`	`$aws/events/job/`*`jobID`*`/deleted`

Categoría de evento (AWS IoT Consola: configuración: mensajes basados en eventos)	Valor de la clave **`eventConfigurations`** (AWS CLI/API)	Tema del mensaje del evento
Trabajo completado, cancelado	`JOB`	`$aws/events/job/`*`jobID`*`/deletion_in_progress`
Ejecución del trabajo: correcta, fallida, rechazada, cancelad, eliminada	`JOB_EXECUTION`	`$aws/events/jobExecution/ `*`jobID`*`/canceled`
Ejecución del trabajo: correcta, fallida, rechazada, cancelad, eliminada	`JOB_EXECUTION`	`$aws/events/jobExecution/ `*`jobID`*`/deleted`
Ejecución del trabajo: correcta, fallida, rechazada, cancelad, eliminada	`JOB_EXECUTION`	`$aws/events/jobExecution/ `*`jobID`*`/failed`
Ejecución del trabajo: correcta, fallida, rechazada, cancelad, eliminada	`JOB_EXECUTION`	`$aws/events/jobExecution/ `*`jobID`*`/rejected`
Ejecución del trabajo: correcta, fallida, rechazada, cancelad, eliminada	`JOB_EXECUTION`	`$aws/events/jobExecution/ `*`jobID`*`/removed`
Ejecución del trabajo: correcta, fallida, rechazada, cancelad, eliminada	`JOB_EXECUTION`	`$aws/events/jobExecution/ `*`jobID`*`/succeeded`
Ejecución del trabajo: correcta, fallida, rechazada, cancelad, eliminada	`JOB_EXECUTION`	`$aws/events/jobExecution/ `*`jobID`*`/timed_out`

Categoría de evento (AWS IoT Consola: configuración: mensajes basados en eventos)	Valor de la clave **`eventConfigurations`** (AWS CLI/API)	Tema del mensaje del evento
Objeto: creado, actualizado, eliminado	`THING`	`$aws/events/thing/` *`thingName`* `/created`
Objeto: creado, actualizado, eliminado	`THING`	`$aws/events/thing/` *`thingName`* `/updated`
Objeto: creado, actualizado, eliminado	`THING`	`$aws/events/thing/` *`thingName`* `/deleted`
Grupo de objetos: agregado, eliminado	`THING_GROUP`	`$aws/events/thingGroup/` *`thingGroupName`* `/created`
Grupo de objetos: agregado, eliminado	`THING_GROUP`	`$aws/events/thingGroup/` *`thingGroupName`* `/updated`
Grupo de objetos: agregado, eliminado	`THING_GROUP`	`$aws/events/thingGroup/` *`thingGroupName`* `/deleted`
Jerarquía de grupos de objetos: agregada, eliminada	`THING_GROUP_HIERARCHY`	`$aws/events/thingGroupHierarchy/thingGroup/` *`parentThingGroupName`* `/childThingGroup/` *`childThingGroupName`* `/added`

Categoría de evento (AWS IoT Consola: configuración: mensajes basados en eventos)	Valor de la clave **`eventConfigurations`** (AWS CLI/API)	Tema del mensaje del evento
Jerarquía de grupos de objetos: agregada, eliminada	`THING_GROUP_HIERARCHY`	`$aws/events/thingGroupHierarchy/thingGroup/` *`parentThingGroupName`* `/childThingGroup/` *`childThingGroupName`* `/removed`
Pertenencia a un grupo de objetos: agregada, eliminada	`THING_GROUP_MEMBERSHIP`	`$aws/events/thingGroupMembership/thingGroup/` *`thingGroupName`* `/thing/` *`thingName`* `/added`
Pertenencia a un grupo de objetos: agregada, eliminada	`THING_GROUP_MEMBERSHIP`	`$aws/events/thingGroupMembership/thingGroup/` *`thingGroupName`* `/thing/` *`thingName`* `/removed`
Tipo de objeto: creado, actualizado, eliminado	`THING_TYPE`	`$aws/events/thingType/` *`thingTypeName`* `/created`
Tipo de objeto: creado, actualizado, eliminado	`THING_TYPE`	`$aws/events/thingType/` *`thingTypeName`* `/updated`
Tipo de objeto: creado, actualizado, eliminado	`THING_TYPE`	`$aws/events/thingType/` *`thingTypeName`* `/deleted`

Categoría de evento (AWS IoT Consola: configuración: mensajes basados en eventos)	Valor de la clave **`eventConfigurations`** (AWS CLI/API)	Tema del mensaje del evento

Asociación de tipo de objeto: agregada, eliminada	`THING_TYPE_ASSOCIATION`	`$aws/events/thingTypeAssociation/thing/` *`thingName`* `/thingType/` *`thingTypeName`* `/added`
		
		`$aws/events/thingTypeAssociation/thing/` *`thingName`* `/thingType/` *`thingTypeName`* `/removed`

Eventos de registro

El registro puede publicar mensajes de eventos cuando se crean, actualizan o eliminan objetos, tipos de objetos y grupos de objetos. Estos eventos, sin embargo, no están disponibles de forma predeterminada. Para obtener información sobre cómo activar estos eventos, consulte Habilite los eventos para AWS IoT.

El registro puede proporcionar los siguientes tipos de eventos:

- Eventos de objeto
- Eventos de tipo de objeto
- Eventos de grupo de objetos

Eventos de objeto

¿Cosa Created/Updated/Deleted

El registro publica los siguientes mensajes de eventos cuando se crean, actualizan o eliminan objetos:

- `$aws/events/thing/thingName/created`
- `$aws/events/thing/thingName/updated`
- `$aws/events/thing/thingName/deleted`

Los mensajes contienen la siguiente carga de ejemplo:

```
{
    "eventType" : "THING_EVENT",
    "eventId" : "f5ae9b94-8b8e-4d8e-8c8f-b3266dd89853",
    "timestamp" : 1234567890123,
    "operation" : "CREATED|UPDATED|DELETED",
    "accountId" : "123456789012",
    "thingId" : "b604f69c-aa9a-4d4a-829e-c480e958a0b5",
    "thingName" : "MyThing",
    "versionNumber" : 1,
    "thingTypeName" : null,
    "attributes": {
                "attribute3": "value3",
                "attribute1": "value1",
                "attribute2": "value2"
    }
}
```

Las cargas contienen los siguientes atributos:

eventType

Se establece en "THING_EVENT".

eventId

Un ID de evento exclusivo (cadena).

timestamp

La marca de tiempo UNIX de cuándo se produjo el evento.

operación

La operación en la que se activó el evento. Los valores válidos son:

- CREATED
- UPDATED

- DELETED

accountId

Tu Cuenta de AWS carné de identidad.

thingId

El ID de la cosa que se crea, actualiza o elimina.

thingName

El nombre de la cosa que se crea, actualiza o elimina.

versionNumber

La versión de la cosa que se crea, actualiza o elimina. Este valor se establece en 1 cuando se crea un objeto. Aumenta en 1 cada vez que se actualiza el objeto.

thingTypeName

El tipo de objeto asociado al objeto, si existiera. De lo contrario, `null`.

attributes

Un conjunto de pares nombre-valor asociados al objeto.

Eventos de tipo de objeto

Eventos relacionados con el tipo de objeto:

- Tipo de cosa Created/Updated/Deprecated/Undeprecated/Deleted
- Tipo de objeto asociado o desasociado de un objeto

Tipo de cosa Created/Updated/Deprecated/Undeprecated/Deleted

El registro publica los siguientes mensajes de eventos cuando los tipos de objetos se crean, se actualizan, se dan de baja o a los que se les quita la marca de obsoleto:

- `$aws/events/thingType/`*`thingTypeName`*`/created`
- `$aws/events/thingType/`*`thingTypeName`*`/updated`
- `$aws/events/thingType/`*`thingTypeName`*`/deleted`

El mensaje contiene la siguiente carga de ejemplo:

```
{
    "eventType" : "THING_TYPE_EVENT",
    "eventId" : "8827376c-4b05-49a3-9b3b-733729df7ed5",
    "timestamp" : 1234567890123,
    "operation" : "CREATED|UPDATED|DELETED",
    "accountId" : "123456789012",
    "thingTypeId" : "c530ae83-32aa-4592-94d3-da29879d1aac",
    "thingTypeName" : "MyThingType",
    "isDeprecated" : false|true,
    "deprecationDate" : null,
    "searchableAttributes" : [ "attribute1", "attribute2", "attribute3" ],
    "propagatingAttributes": [
        {
     "userPropertyKey": "key",
     "thingAttribute": "model"
        },
        {
     "userPropertyKey": "key",
     "connectionAttribute": "iot:ClientId"
        }
    ],
    "description" : "My thing type"
}
```

Las cargas contienen los siguientes atributos:

eventType

Se establece en "THING_TYPE_EVENT".

eventId

Un ID de evento exclusivo (cadena).

timestamp

La marca de tiempo UNIX de cuándo se produjo el evento.

operación

La operación en la que se activó el evento. Los valores válidos son:

- CREATED

- UPDATED
- DELETED

accountId

Tu Cuenta de AWS carné de identidad.

thingTypeId

El ID del tipo de objeto que se crea, se actualiza, se da de baja o se elimina.

thingTypeName

El nombre del tipo de objeto que se crea, se actualiza, se da de baja o se elimina.

isDeprecated

`true` si el tipo de objeto está descartado. De lo contrario, `false`.

deprecationDate

La marca de tiempo UNIX para cuando el tipo de objeto está descartado.

searchableAttributes

Un conjunto de pares nombre-valor asociados con el tipo de objeto que puede utilizarse para realizar búsquedas.

propagatingAttributes

Una lista de atributos de propagación. Un atributo de propagación puede incluir un atributo de objeto, un atributo de conexión y una clave de propiedad de usuario. Para obtener más información, consulte Agregación de atributos de propagación para enriquecer los mensajes.

description

Una descripción del tipo de objeto.

Tipo de objeto asociado o desasociado de un objeto

El registro publica los siguientes mensajes de eventos cuando se asocia o desasocia un tipo de objeto a un objeto.

- `$aws/events/thingTypeAssociation/thing/`*`thingName`*`/thingType/`*`typeName`*`/added`

- `$aws/events/thingTypeAssociation/thing/thingName/thingType/typeName/removed`

A continuación, se muestra un ejemplo de carga de `added`. Las cargas de los mensajes `removed` son similares.

```
{
    "eventId" : "87f8e095-531c-47b3-aab5-5171364d138d",
    "eventType" : "THING_TYPE_ASSOCIATION_EVENT",
    "operation" : "ADDED",
    "thingId" : "b604f69c-aa9a-4d4a-829e-c480e958a0b5",
    "thingName": "myThing",
    "thingTypeName" : "MyThingType",
    "timestamp" : 1234567890123,
}
```

Las cargas contienen los siguientes atributos:

eventId

Un ID de evento exclusivo (cadena).

eventType

Se establece en "THING_TYPE_ASSOCIATION_EVENT".

operación

La operación en la que se activó el evento. Los valores válidos son:

- ADDED
- REMOVED

thingId

El ID de la cosa cuya asociación de tipo ha cambiado.

thingName

El nombre de la cosa cuya asociación de tipo ha cambiado.

thingTypeName

El tipo de objeto asociado o desasociado de la cosa.

timestamp

La marca de tiempo UNIX de cuándo se produjo el evento.

Eventos de grupo de objetos

Eventos relacionados con el grupo de objetos:

- Grupo de cosas Created/Updated/Deleted
- Objeto agregado o eliminado de un grupo de objetos
- Grupo de objetos agregado o eliminado de un grupo de objetos

Grupo de cosas Created/Updated/Deleted

El registro publica los siguientes mensajes de eventos cuando se crea, actualiza o elimina un grupo de objetos.

- `$aws/events/thingGroup/`*`groupName`*`/created`
- `$aws/events/thingGroup/`*`groupName`*`/updated`
- `$aws/events/thingGroup/`*`groupName`*`/deleted`

A continuación, se muestra un ejemplo de carga de `updated`. Las cargas de los mensajes `created` y `deleted` son similares.

```
{
  "eventType": "THING_GROUP_EVENT",
  "eventId": "8b9ea8626aeaa1e42100f3f32b975899",
  "timestamp": 1603995417409,
  "operation": "UPDATED",
  "accountId": "571EXAMPLE833",
  "thingGroupId": "8757eec8-bb37-4cca-a6fa-403b003d139f",
  "thingGroupName": "Tg_level5",
  "versionNumber": 3,
  "parentGroupName": "Tg_level4",
  "parentGroupId": "5fce366a-7875-4c0e-870b-79d8d1dce119",
  "description": "New description for Tg_level5",
  "rootToParentThingGroups": [
    {
      "groupArn": "arn:aws:iot:us-west-2:571EXAMPLE833:thinggroup/TgTopLevel",
```

```
      "groupId": "36aa0482-f80d-4e13-9bff-1c0a75c055f6"
    },
    {
      "groupArn": "arn:aws:iot:us-west-2:571EXAMPLE833:thinggroup/Tg_level1",
      "groupId": "bc1643e1-5a85-4eac-b45a-92509cbe2a77"
    },
    {
      "groupArn": "arn:aws:iot:us-west-2:571EXAMPLE833:thinggroup/Tg_level2",
      "groupId": "0476f3d2-9beb-48bb-ae2c-ea8bd6458158"
    },
    {
      "groupArn": "arn:aws:iot:us-west-2:571EXAMPLE833:thinggroup/Tg_level3",
      "groupId": "1d9d4ffe-a6b0-48d6-9de6-2e54d1eae78f"
    },
    {
      "groupArn": "arn:aws:iot:us-west-2:571EXAMPLE833:thinggroup/Tg_level4",
      "groupId": "5fce366a-7875-4c0e-870b-79d8d1dce119"
    }
  ],
  "attributes": {
    "attribute1": "value1",
    "attribute3": "value3",
    "attribute2": "value2"
  },
  "dynamicGroupMappingId": null
}
```

Las cargas contienen los siguientes atributos:

eventType

Se establece en "THING_GROUP_EVENT".

eventId

Un ID de evento exclusivo (cadena).

timestamp

La marca de tiempo UNIX de cuándo se produjo el evento.

operación

La operación en la que se activó el evento. Los valores válidos son:

- CREATED

- UPDATED
- DELETED

accountId

Tu Cuenta de AWS carné de identidad.

thingGroupId

El ID del grupo de objetos que se crea, actualiza o elimina.

thingGroupName

El nombre del grupo de objetos que se crea, actualiza o elimina.

versionNumber

La versión del grupo de objetos. Este valor se establece en 1 cuando se crea un grupo de objetos. Aumenta en 1 cada vez que se actualiza el grupo de objetos.

parentGroupName

El nombre del grupo principal de objetos, si existe.

parentGroupId

El ID del grupo principal de objetos, si existe.

description

Una descripción del grupo de objetos.

rootToParentThingGroups

Una matriz de información acerca del grupo principal de objetos. Hay un elemento por cada grupo principal de objetos, empezando por el grupo de objetos raíz y continuando hasta el grupo principal. Cada entrada contiene los `groupArn` y `groupId` del grupo de objetos.

attributes

Un conjunto de pares nombre-valor asociados al grupo de objetos.

Objeto agregado o eliminado de un grupo de objetos

El registro publica los siguientes mensajes de eventos cuando se agrega un objeto a un grupo de objetos o se elimina de este.

- `$aws/events/thingGroupMembership/thingGroup/thingGroupName/thing/thingName/added`
- `$aws/events/thingGroupMembership/thingGroup/thingGroupName/thing/thingName/removed`

Los mensajes contienen la siguiente carga de ejemplo:

```
{
    "eventType" : "THING_GROUP_MEMBERSHIP_EVENT",
    "eventId" : "d684bd5f-6f6e-48e1-950c-766ac7f02fd1",
    "timestamp" : 1234567890123,
    "operation" : "ADDED|REMOVED",
    "accountId" : "123456789012",
    "groupArn" : "arn:aws:iot:ap-northeast-2:123456789012:thinggroup/
MyChildThingGroup",
    "groupId" : "06838589-373f-4312-b1f2-53f2192291c4",
    "thingArn" : "arn:aws:iot:ap-northeast-2:123456789012:thing/MyThing",
    "thingId" : "b604f69c-aa9a-4d4a-829e-c480e958a0b5",
    "membershipId" : "8505ebf8-4d32-4286-80e9-c23a4a16bbd8"
}
```

Las cargas contienen los siguientes atributos:

eventType

Se establece en "THING_GROUP_MEMBERSHIP_EVENT".

eventId

El ID del evento.

timestamp

La marca de tiempo UNIX de cuándo se produjo el evento.

operación

`ADDED` cuando se agrega un objeto a un grupo de objetos. `REMOVED` cuando se elimina un objeto de un grupo de objetos.

accountId

Tu Cuenta de AWS carné de identidad.

groupArn

El ARN del grupo de objetos.

groupId

El ID del grupo.

thingArn

El ARN de la cosa que se agregó o quitó del grupo de objetos.

thingId

El ID de la cosa que se agregó o quitó del grupo de objetos.

membershipId

Un ID que representa la relación entre objeto y el grupo de objetos. Este valor se genera cuando agrega un objeto a un grupo de objetos.

Grupo de objetos agregado o eliminado de un grupo de objetos

El registro publica los siguientes mensajes de eventos cuando se agrega un grupo de objetos a otro grupo de objetos o se elimina de este.

- `$aws/events/thingGroupHierarchy/thingGroup/`*`parentThingGroupName`*`/childThingGroup/`*`childThingGroupName`*`/added`
- `$aws/events/thingGroupHierarchy/thingGroup/`*`parentThingGroupName`*`/childThingGroup/`*`childThingGroupName`*`/removed`

El mensaje contiene la siguiente carga de ejemplo:

```
{
    "eventType" : "THING_GROUP_HIERARCHY_EVENT",
    "eventId" : "264192c7-b573-46ef-ab7b-489fcd47da41",
    "timestamp" : 1234567890123,
    "operation" : "ADDED|REMOVED",
    "accountId" : "123456789012",
    "thingGroupId" : "8f82a106-6b1d-4331-8984-a84db5f6f8cb",
    "thingGroupName" : "MyRootThingGroup",
    "childGroupId" : "06838589-373f-4312-b1f2-53f2192291c4",
    "childGroupName" : "MyChildThingGroup"
```

```
}
```

Las cargas contienen los siguientes atributos:

eventType

Se establece en "THING_GROUP_HIERARCHY_EVENT".

eventId

El ID del evento.

timestamp

La marca de tiempo UNIX de cuándo se produjo el evento.

operación

`ADDED` cuando se agrega un objeto a un grupo de objetos. `REMOVED` cuando se elimina un objeto de un grupo de objetos.

accountId

Tu Cuenta de AWS carné de identidad.

thingGroupId

El ID del grupo principal de objetos.

thingGroupName

El nombre del grupo principal de objetos.

childGroupId

El ID del grupo secundario de objetos.

childGroupName

El nombre del grupo secundario de objetos.

Eventos de trabajos

El servicio AWS IoT Jobs publica en temas reservados del protocolo MQTT cuando los trabajos están pendientes, completados o cancelados, y cuando un dispositivo informa de que los trabajos

se han ejecutado correctamente o no. Los dispositivos o las aplicaciones de administración y monitorización pueden realizar un seguimiento del estado de los trabajos mediante la suscripción a estos temas.

Cómo habilitar los eventos de trabajos

Los mensajes de respuesta del servicio AWS IoT Jobs no pasan por el intermediario de mensajes y otros clientes o reglas no pueden suscribirse a ellos. Para suscribirse a los mensajes relacionados con la actividad laboral, utilice los temas `notify` y `notify-next`. Para obtener más información acerca de los temas de trabajos, consulte Temas de trabajos.

Para recibir notificaciones de las actualizaciones de los trabajos, habilite estos eventos de trabajos mediante la API o la CLI Consola de administración de AWS, o mediante la misma. Para obtener más información, consulte Habilite los eventos para AWS IoT.

Cómo funcionan los eventos de trabajo

Dado que cancelar o eliminar un trabajo puede llevar un tiempo, se envían dos mensajes para indicar el comienzo y el final de una solicitud. Por ejemplo, cuando se inicia una solicitud de cancelación, se envía un mensaje al tema `$aws/events/job/jobID/cancellation_in_progress`. Cuando finaliza una solicitud de cancelación, se envía un mensaje al tema `$aws/events/job/jobID/canceled`.

En las solicitudes de eliminación de trabajos se lleva a cabo un proceso parecido. Las aplicaciones de administración y monitorización pueden suscribirse a estos temas y hacer un seguimiento del estado de los trabajos. Para obtener más información acerca de cómo publicar en temas de MQTT y suscribirse a ellos, consulte the section called “Protocolos de comunicación de dispositivos”.

Tipos de eventos de trabajo

A continuación, se muestran los diferentes tipos de eventos de trabajo:

Job Completed/Canceled/Deleted

El servicio AWS IoT Jobs publica un mensaje sobre un tema de MQTT cuando se completa, cancela o elimina un trabajo o cuando la cancelación o eliminación está en curso:

- `$aws/events/job/`*`jobID`*`/completed`
- `$aws/events/job/`*`jobID`*`/canceled`
- `$aws/events/job/`*`jobID`*`/deleted`

- `$aws/events/job/jobID/cancellation_in_progress`
- `$aws/events/job/jobID/deletion_in_progress`

El mensaje `completed` contiene la siguiente carga de ejemplo:

```
{
  "eventType": "JOB",
  "eventId": "7364ffd1-8b65-4824-85d5-6c14686c97c6",
  "timestamp": 1234567890,
  "operation": "completed",
  "jobId": "27450507-bf6f-4012-92af-bb8a1c8c4484",
  "status": "COMPLETED",
  "targetSelection": "SNAPSHOT|CONTINUOUS",
  "targets": [
    "arn:aws:iot:us-east-1:123456789012:thing/a39f6f91-70cf-4bd2-a381-9c66df1a80d0",
    "arn:aws:iot:us-east-1:123456789012:thinggroup/2fc4c0a4-6e45-4525-
a238-0fe8d3dd21bb"
  ],
  "description": "My Job Description",
  "completedAt": 1234567890123,
  "createdAt": 1234567890123,
  "lastUpdatedAt": 1234567890123,
  "jobProcessDetails": {
    "numberOfCanceledThings": 0,
    "numberOfRejectedThings": 0,
    "numberOfFailedThings": 0,
    "numberOfRemovedThings": 0,
    "numberOfSucceededThings": 3
  }
}
```

El mensaje `canceled` contiene la siguiente carga de ejemplo:

```
{
  "eventType": "JOB",
  "eventId": "568d2ade-2e9c-46e6-a115-18afa1286b06",
  "timestamp": 1234567890,
  "operation": "canceled",
  "jobId": "4d2a531a-da2e-47bb-8b9e-ff5adcd53ef0",
  "status": "CANCELED",
  "targetSelection": "SNAPSHOT|CONTINUOUS",
  "targets": [
```

```
    "arn:aws:iot:us-east-1:123456789012:thing/Thing0-947b9c0c-ff10-4a80-b4b3-
cd33d0145a0f",
    "arn:aws:iot:us-east-1:123456789012:thinggroup/
ThingGroup1-95c644d5-1621-41a6-9aa5-ad2de581d18f"
  ],
  "description": "My job description",
  "createdAt": 1234567890123,
  "lastUpdatedAt": 1234567890123
}
```

El mensaje `deleted` contiene la siguiente carga de ejemplo:

```
{
      "eventType": "JOB",
      "eventId": "568d2ade-2e9c-46e6-a115-18afa1286b06",
      "timestamp": 1234567890,
      "operation": "deleted",
      "jobId": "4d2a531a-da2e-47bb-8b9e-ff5adcd53ef0",
      "status": "DELETED",
      "targetSelection": "SNAPSHOT|CONTINUOUS",
      "targets": [
        "arn:aws:iot:us-east-1:123456789012:thing/Thing0-947b9c0c-ff10-4a80-b4b3-
cd33d0145a0f",
        "arn:aws:iot:us-east-1:123456789012:thinggroup/
ThingGroup1-95c644d5-1621-41a6-9aa5-ad2de581d18f"
      ],
      "description": "My job description",
      "createdAt": 1234567890123,
      "lastUpdatedAt": 1234567890123,
      "comment": "Comment for this operation"
    }
```

El mensaje `cancellation_in_progress` contiene la siguiente carga de ejemplo:

```
{
      "eventType": "JOB",
      "eventId": "568d2ade-2e9c-46e6-a115-18afa1286b06",
      "timestamp": 1234567890,
      "operation": "cancellation_in_progress",
      "jobId": "4d2a531a-da2e-47bb-8b9e-ff5adcd53ef0",
      "status": "CANCELLATION_IN_PROGRESS",
      "targetSelection": "SNAPSHOT|CONTINUOUS",
      "targets": [
```

```
        "arn:aws:iot:us-east-1:123456789012:thing/Thing0-947b9c0c-ff10-4a80-b4b3-
cd33d0145a0f",
        "arn:aws:iot:us-east-1:123456789012:thinggroup/
ThingGroup1-95c644d5-1621-41a6-9aa5-ad2de581d18f"
      ],
      "description": "My job description",
      "createdAt": 1234567890123,
      "lastUpdatedAt": 1234567890123,
      "comment": "Comment for this operation"
    }
```

El mensaje `deletion_in_progress` contiene la siguiente carga de ejemplo:

```
{
      "eventType": "JOB",
      "eventId": "568d2ade-2e9c-46e6-a115-18afa1286b06",
      "timestamp": 1234567890,
      "operation": "deletion_in_progress",
      "jobId": "4d2a531a-da2e-47bb-8b9e-ff5adcd53ef0",
      "status": "DELETION_IN_PROGRESS",
      "targetSelection": "SNAPSHOT|CONTINUOUS",
      "targets": [
        "arn:aws:iot:us-east-1:123456789012:thing/Thing0-947b9c0c-ff10-4a80-b4b3-
cd33d0145a0f",
        "arn:aws:iot:us-east-1:123456789012:thinggroup/
ThingGroup1-95c644d5-1621-41a6-9aa5-ad2de581d18f"
      ],
      "description": "My job description",
      "createdAt": 1234567890123,
      "lastUpdatedAt": 1234567890123,
      "comment": "Comment for this operation"
    }
```

Estado final de ejecución de trabajo

El servicio AWS IoT Jobs publica un mensaje cuando un dispositivo actualiza la ejecución de un trabajo al estado terminal:

- `$aws/events/jobExecution/`*`jobID`*`/succeeded`
- `$aws/events/jobExecution/`*`jobID`*`/failed`
- `$aws/events/jobExecution/`*`jobID`*`/rejected`
- `$aws/events/jobExecution/`*`jobID`*`/canceled`

- `$aws/events/jobExecution/jobID/timed_out`
- `$aws/events/jobExecution/jobID/removed`
- `$aws/events/jobExecution/jobID/deleted`

El mensaje contiene la siguiente carga de ejemplo:

```
{
  "eventType": "JOB_EXECUTION",
  "eventId": "cca89fa5-8a7f-4ced-8c20-5e653afb3572",
  "timestamp": 1234567890,
  "operation": "succeeded|failed|rejected|canceled|removed|timed_out",
  "jobId": "154b39e5-60b0-48a4-9b73-f6f8dd032d27",
  "thingArn": "arn:aws:iot:us-east-1:123456789012:myThing/6d639fbc-8f85-4a90-924d-
a2867f8366a7",
  "status": "SUCCEEDED|FAILED|REJECTED|CANCELED|REMOVED|TIMED_OUT",
  "statusDetails": {
    "key": "value"
  }
}
```

Eventos del ciclo de vida

AWS IoT puede publicar eventos del ciclo de vida sobre los temas de MQTT. Estos eventos están disponibles de forma predeterminada y no se pueden desactivar.

> **Note**
>
> Es posible que los mensajes de ciclo de vida se envíen de forma desordenada. Puede que reciba mensajes duplicados.
> `thingName` solo se incluirá si el cliente se conecta mediante la característica objeto exclusivo.

En este tema:

- Eventos de conexión/desconexión
- Evento de error de intento de conexión
- Eventos de suscripción/cancelación de suscripción

Eventos de conexión/desconexión

> **Note**
>
> Con la indexación de flotas de AWS IoT Device Management, puede buscar cosas, ejecutar consultas agregadas y crear grupos dinámicos basados en eventos de cosas Connect/Disconnect . Para obtener más información, consulte Indexación de flotas.

AWS IoT publica un mensaje sobre los siguientes temas de MQTT cuando un cliente se conecta o se desconecta:

- `$aws/events/presence/connected/`*`clientId`*: un cliente se ha conectado al agente de mensajes.
- `$aws/events/presence/disconnected/`*`clientId`*: un cliente se ha desconectado del agente de mensajes.

La siguiente es una lista de los elementos de JSON que se incluyen en los connection/disconnection mensajes publicados en el `$aws/events/presence/connected/`*`clientId`* tema.

clientId

El ID del cliente que se conecta o se desconecta.

> **Note**
>
> Los clientes IDs que contienen # o + no reciben eventos del ciclo de vida.

thingName

El nombre del objeto de IoT. `thingName` solo se incluirá si el cliente se conecta mediante la característica objeto exclusivo.

clientInitiatedDisconnect

True si el cliente inició la desconexión. De lo contrario, devuelve false. Sólo se encuentra en mensajes de desconexión.

disconnectReason

La razón por la que el cliente se está desconectando. Sólo se encuentra en mensajes de desconexión. La siguiente tabla contiene valores válidos e indica si el agente de mensajes enviará mensajes Last Will and Testament (LWT) cuando se produzca la desconexión.

Motivo de desconexión	Description (Descripción)	El agente de mensajes enviará los mensajes LWT
`AUTH_ERROR`	El cliente no pudo autenticarse o la autorización devolvió un error.	Sí[1]
`CLIENT_INITIATED_DISCONNECT`	El cliente indica que se desconectará. El cliente puede hacerlo enviando un paquete de `DISCONNECT` control MQTT o un paquete `Close frame` si el cliente utiliza una WebSocket conexión.	No
`CLIENT_ERROR`	El cliente hizo algo mal que provocó su desconexión. Por ejemplo, un cliente se desconectará por enviar más de un paquete `CONNECT` de MQTT en la misma conexión o si el cliente intenta publicar con una carga útil que supera el límite de carga útil.	Sí
`CONNECTION_LOST`	La conexión cliente-servidor está cortada. Esto puede ocurrir durante un período de alta latencia de red o cuando se pierde la conexión a Internet.	Sí
`DUPLICATE_CLIENTID`	El cliente está utilizando un ID de cliente que ya está en uso. En este caso, el cliente que ya está conectado se desconectará con esta razón de desconexión.	Sí
`FORBIDDEN_ACCESS`	No se permite la conexión del cliente. Por ejemplo, un cliente con una dirección IP denegada no podrá conectarse.	Sí[1]

Motivo de desconexión	Description (Descripción)	El agente de mensajes enviará los mensajes LWT
`MQTT_KEEP_ALIVE_TIMEOUT`	Si no hay comunicación cliente-servidor para 1,5 veces el tiempo de mantenimiento del cliente, el cliente se desconecta.	Sí
`SERVER_ERROR`	Desconectado debido a problemas inesperados del servidor.	Sí
`SERVER_INITIATED_DISCONNECT`	El servidor desconecta de forma intencionada un cliente por razones operativas.	Sí
`API_INITIATED_DISCONNECT`	El cliente se desconectó mediante la API `DeleteConnection` .	Sí[2]
`THROTTLED`	El cliente se desconecta por exceder una limitación controlada.	Sí
`WEBSOCKET_TTL_EXPIRATION`	El cliente está desconectado porque a WebSocket lleva más tiempo conectado que su time-to-live valor.	Sí
`CUSTOMAUTH_TTL_EXPIRATION`	El cliente está desconectado porque lleva más tiempo conectado que el time-to-live valor de su autorizador personalizado.	Sí

[1]Si el dispositivo tiene una conexión activa antes de recibir este error.

[2]Para evitar los mensajes de Last Will and Testament (LWT), establezca `preventWillMessage=true` para anular el comportamiento de envío de LWT predeterminado de la API `DeleteConnection`.

eventType

El tipo de evento. Los valores válidos son `connected` o `disconnected`.

ipAddress

La dirección IP del cliente que se conecta. Puede estar en IPv6 formato IPv4 o. Sólo se encuentra en los mensajes de conexión.

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 a nivel mundial AWS IoT que existe durante toda la sesión.

timestamp

Una aproximación de cuándo se produjo el evento.

versionNumber

El número de versión del evento del ciclo de vida. Se trata de un valor entero largo que aumenta de forma monótona para cada conexión de un ID de cliente. El número de versión puede utilizarlo un suscriptor para deducir el orden de los eventos del ciclo de vida.

> **Note**
>
> Los mensajes de conexión y desconexión de una conexión de cliente tienen el mismo número de versión.
> El número de versión podría saltarse algunos valores y no se garantiza que se vaya a incrementar de forma coherente en 1 para cada evento.
> Si un cliente no se conecta durante aproximadamente una hora, el número de versión se restablece a 0. En el caso de las sesiones persistentes, el número de versión se restablece a 0 después de que un cliente haya estado desconectado durante más tiempo del tiempo configurado time-to-live (TTL) para la sesión persistente.

Un mensaje de conexión tiene la siguiente estructura.

```
{
    "clientId": "186b5",
    "thingName": "exampleThing",
```

```
    "timestamp": 1573002230757,
    "eventType": "connected",
    "sessionIdentifier": "00000000-0000-0000-0000-000000000000",
    "principalIdentifier": "12345678901234567890123456789012",
    "ipAddress": "192.0.2.0",
    "versionNumber": 0
}
```

Un mensaje de desconexión tiene la siguiente estructura.

```
{
    "clientId": "186b5",
    "thingName": "exampleThing",
    "timestamp": 1573002340451,
    "eventType": "disconnected",
    "sessionIdentifier": "00000000-0000-0000-0000-000000000000",
    "principalIdentifier": "12345678901234567890123456789012",
    "clientInitiatedDisconnect": true,
    "disconnectReason": "CLIENT_INITIATED_DISCONNECT",
    "versionNumber": 0
}
```

Gestión de desconexiones del cliente

La práctica recomendada consiste siempre en tener implementado un estado de espera para los eventos del ciclo de vida, incluidos los mensajes Last Will and Testament (LWT). Cuando se recibe un mensaje de desconexión, el código debe esperar un periodo de tiempo y verificar que un dispositivo sigue sin conexión antes de tomar cualquier medida. Una forma de hacerlo consiste en utilizar colas con retraso de SQS. Cuando un cliente recibe un evento de ciclo de vida o un mensaje LWT, se puede poner en cola un mensaje (por ejemplo, durante 5 segundos). Cuando dicho mensaje está disponible y se procesa (por parte de Lambda u otro servicio), primero se puede comprobar si el dispositivo sigue sin conexión antes de tomar otras medidas.

Evento de error de intento de conexión

AWS IoT publica un mensaje sobre el siguiente tema de MQTT cuando un cliente no está autorizado a conectarse o cuando se ha configurado un testamento y un testamento y el cliente no está autorizado a publicar en ese tema de último testamento.

```
$aws/events/presence/connect_failed/clientId
```

A continuación, se muestra una lista de elementos JSON que se incluyen en los mensajes de autorización de conexión publicados en el tema de `$aws/events/presence/connect_failed/`*`clientId`*.

clientId

El ID de cliente que intentó conectarse y que no pudo hacerlo.

> **Note**
>
> Los clientes IDs que contienen # o + no reciben eventos del ciclo de vida.

thingName

El nombre del objeto de IoT. `thingName` solo se incluirá si el cliente se conecta mediante la característica objeto exclusivo.

timestamp

Una aproximación de cuándo se produjo el evento.

eventType

El tipo de evento. El valor válido es `connect_failed`.

connectFailureReason

El motivo por el que se produce un error en la conexión. El valor válido es `AUTHORIZATION_FAILED`.

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 a nivel mundial AWS IoT que existe durante toda la sesión.

ipAddress

La dirección IP del cliente que se conecta. Puede estar en IPv6 formato IPv4 o. Sólo se encuentra en los mensajes de conexión.

Un mensaje de error de conexión tiene la siguiente estructura.

```
{
    "clientId": "186b5",
    "thingName": "exampleThing",
    "timestamp": 1460065214626,
    "eventType": "connect_failed",
    "connectFailureReason": "AUTHORIZATION_FAILED",
    "principalIdentifier": "123456789012345678901234567890123456789012",
    "sessionIdentifier": "00000000-0000-0000-0000-000000000000",
    "ipAddress" : "192.0.2.0"
}
```

Eventos de suscripción/cancelación de suscripción

AWS IoT publica un mensaje sobre el siguiente tema de MQTT cuando un cliente se suscribe o cancela la suscripción a un tema de MQTT:

```
$aws/events/subscriptions/subscribed/clientId
```

o

```
$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",
    "thingName": "exampleThing",
    "timestamp": 1460065214626,
    "eventType": "subscribed" | "unsubscribed",
    "sessionIdentifier": "00000000-0000-0000-0000-000000000000",
    "principalIdentifier": "123456789012345678901234567890123456789012",
    "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 clientes IDs que contienen # o + no reciben eventos del ciclo de vida.

thingName

El nombre del objeto de IoT. `thingName` solo se incluirá si el cliente se conecta mediante la característica objeto exclusivo.

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 a nivel mundial AWS IoT que existe durante toda la sesión.

timestamp

Una aproximación de cuándo se produjo el evento.

temas

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.

Solución de problemas AWS IoT

Ayúdenos a mejorar este tema

Explíquenos cómo mejorarlo

La siguiente información puede ayudarle a solucionar problemas comunes en AWS IoT.

Tareas

- AWS IoT Core guía de solución de problemas
- AWS IoT Device Management guía de solución de problemas
- AWS IoT Guía de solución de problemas de Device Advisor
- AWS IoT errores

AWS IoT Core guía de solución de problemas

Ayúdenos a mejorar este tema

Explíquenos cómo mejorarlo

Esta es la sección de solución de problemas de AWS IoT Core.

Temas

- Diagnóstico de problemas de conectividad
- Diagnóstico de problemas de las reglas
- Diagnóstico de problemas relacionados con las sombras
- Diagnosticar problemas con acciones del flujo de entrada de Salesforce IoT
- Diagnóstico de los límites de flujo
- Resolución para las desconexiones en una flota de dispositivos

Diagnóstico de problemas de conectividad

Ayúdenos a mejorar este tema

Explíquenos cómo mejorarlo

Una conexión correcta a AWS IoT requiere:

- Una conexión válida.
- Un certificado válido y activo.
- Una política que permita la conexión y el funcionamiento deseados.

Connection

¿Cómo puedo encontrar el punto de conexión correcto?

- El valor de `endpointAddress` devuelto por `aws iot describe-endpoint --endpoint-type iot:Data-ATS`

  o

- El valor de `domainName` devuelto por `aws iot describe-domain-configuration --domain-configuration-name "domain_configuration_name"`

¿Cómo puedo encontrar el valor correcto de indicación de nombre de servidor (SNI)?

El valor de SNI correcto es la `endpointAddress` devuelta por el describe-endpoint o el `domainName` devueltos por los comandos de describe-domain-configuration. Es la misma dirección que el punto de conexión del paso anterior. Al conectar los dispositivos a AWS IoT Core, los clientes pueden enviar la extensión de indicación del nombre del servidor (SNI), que no es obligatoria pero sí muy recomendable. Para usar características como el registro de varias cuentas, los dominios personalizados y los puntos de conexión de VPC, debe usar la extensión SNI. Para obtener más información, consulte Seguridad en el transporte en AWS IoT.

¿Cómo puedo solucionar un problema de conectividad que no parece terminar?

Puede usar AWS Device Advisor como ayuda para solucionar problemas. Las pruebas prediseñadas de Device Advisor le ayudarán a validar el software de su dispositivo de acuerdo con procedimientos recomendados de uso de TLS, MQTT, AWS IoT Device Shadow e AWS IoT Jobs.

Este es un enlace al contenido disponible de Device Advisor.

Autenticación

Los dispositivos deben estar autenticados para poder conectarse a AWS IoT los puntos finales. En el caso de los dispositivos que se utilizan Certificados de cliente X.509 para la autenticación, los certificados deben estar registrados AWS IoT y activos.

¿Cómo autentican mis dispositivos los puntos AWS IoT finales?

Añada el certificado de AWS IoT CA al almacén de confianza de su cliente. Consulte la documentación sobre la autenticación de servidores en AWS IoT Core y siga los enlaces para descargar el certificado de CA correspondiente.

¿Qué se comprueba cuando se conecta un dispositivo AWS IoT?

Cuando un dispositivo intenta conectarse a AWS IoT, ocurre lo siguiente:

1. AWS IoT comprueba si hay un certificado válido y un valor de indicación del nombre del servidor (SNI).
2. AWS IoT comprueba que el certificado utilizado está registrado en la AWS IoT cuenta y que se ha activado.
3. Cuando un dispositivo intenta realizar alguna acción AWS IoT, como suscribirse o publicar un mensaje, se comprueba la política adjunta al certificado con el que se conectó para confirmar que el dispositivo está autorizado a realizar esa acción.

¿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.aws-region.amazonaws.com:8443 -
CAfile CA.pem -cert cert.pem -key privateKey.pem
```

Para obtener más información sobre el uso de `openssl s_client`, consulte Documentación de OpenSSL s_client.

¿Cómo compruebo el estado de un certificado?

- Mostrar la lista de certificados

  Si no conoce el ID del certificado, puede ver el estado de todos los certificados con el comando `aws iot list-certificates`.

- Mostrar los detalles de un certificado

  Si conoce el ID del certificado, este comando le muestra información más detallada sobre el mismo.

  ```
  aws iot describe-certificate --certificate-id "certificateId"
  ```

- Revise el certificado en la AWS IoT consola

  En la consola de AWS IoT, en el menú de la izquierda, seleccione Seguridad y, a continuación, Certificados.

  En la lista, seleccione el certificado que está usando para conectarse a fin de abrir la página de detalles.

  En la página de detalles del certificado, puede ver el estado actual.

  El estado del certificado se puede cambiar desde el menú Acciones, situado en la esquina superior derecha de la página de detalles.

Autorización

AWS IoT los recursos AWS IoT Core políticas se utilizan para autorizar a esos recursos a realizar acciones. Para que se autorice una acción, los AWS IoT recursos especificados deben tener un documento de política adjunto que otorgue el permiso para realizar esa acción.

He recibido una respuesta PUBNACK o SUBNACK del agente. ¿Qué tengo que hacer?

Asegúrese de que haya una política adjunta al certificado que va a utilizar para llamar AWS IoT. De forma predeterminada, se deniegan todas las publish/subscribe operaciones.

Asegúrese de que la política adjunta autorice las acciones que intenta llevar a cabo.

Asegúrese de que la política adjunta autorice los recursos que intentarán llevar a cabo las acciones autorizadas.

Tengo una entrada AUTHORIZATION_FAILURE en mis registros.

Asegúrese de que haya una política adjunta al certificado que va a utilizar para llamar AWS IoT. De forma predeterminada, se deniegan todas las publish/subscribe operaciones.

Asegúrese de que la política adjunta autorice las acciones que intenta llevar a cabo.

Asegúrese de que la política adjunta autorice los recursos que intentarán llevar a cabo las acciones autorizadas.

¿Cómo puedo saber lo que autoriza la política?

En la consola de AWS IoT, en el menú de la izquierda, seleccione Seguridad y, a continuación, Certificados.

En la lista, seleccione el certificado que está usando para conectarse a fin de abrir la página de detalles.

En la página de detalles del certificado, puede ver el estado actual.

En el menú de la izquierda de la página de detalles del certificado, seleccione Políticas para ver las políticas adjuntas al certificado.

Seleccione la política que desee para ver su página de detalles.

En la página de detalles de la política, revise el documento de la política para ver qué cosas autoriza.

Seleccione Editar el documento de política para hacer cambios en el documento de la política.

Seguridad e identidad

Al proporcionar los certificados de servidor para la configuración de dominio AWS IoT personalizada, los certificados tienen un máximo de cuatro nombres de dominio.

Para obtener más información, consulte Puntos de conexión y cuotas de AWS IoT Core.

Diagnóstico de problemas de las reglas

Ayúdenos a mejorar este tema

Explíquenos cómo mejorarlo

En esta sección, se describen algunos de los aspectos que se deben comprobar cuando hay problemas con alguna regla.

Configuración de CloudWatch registros para la solución de problemas

La mejor forma de depurar los problemas que tenga con las reglas es utilizar CloudWatch los registros. Al activar CloudWatch los registros AWS IoT, puede ver qué reglas se activan y si se han aplicado correctamente o no. También obtiene información sobre si las condiciones de la cláusula WHERE coinciden. Para obtener más información, consulte Supervise AWS IoT mediante registros CloudWatch.

El problema más habitual de las reglas es la autorización. Los registros muestran si su función no está autorizada para desempeñarse AssumeRole en el recurso. A continuación hay un log de ejemplo generado por el registro detallado:

```
{
    "timestamp": "2017-12-09 22:49:17.954",
    "logLevel": "ERROR",
    "traceId": "ff563525-6469-506a-e141-78d40375fc4e",
    "accountId": "123456789012",
    "status": "Failure",
    "eventType": "RuleExecution",
    "clientId": "iotconsole-123456789012-3",
    "topicName": "test-topic",
    "ruleName": "rule1",
    "ruleAction": "DynamoAction",
    "resources": {
        "ItemHashKeyField": "id",
        "Table": "trashbin",
        "Operation": "Insert",
        "ItemHashKeyValue": "id",
        "IsPayloadJSON": "true"
    },
    "principalId": "ABCDEFG1234567ABCD890:outis",
    "details": "User: arn:aws:sts::123456789012:assumed-role/dynamo-
testbin/5aUMInJH is not authorized to perform: dynamodb:PutItem on
 resource: arn:aws:dynamodb:us-east-1:123456789012:table/testbin (Service:
 AmazonDynamoDBv2; Status Code: 400; Error Code: AccessDeniedException; Request ID:
 AKQJ987654321AKQJ123456789AKQJ987654321AKQJ987654321)"
}
```

A continuación hay un log de ejemplo similar generado por el registro global:

```
2017-12-09 22:49:17.954 TRACEID:ff562535-6964-506a-e141-78d40375fc4e
PRINCIPALID:ABCDEFG1234567ABCD890:outis [ERROR] EVENT:DynamoActionFailure
```

```
TOPICNAME:test-topic CLIENTID:iotconsole-123456789012-3
MESSAGE:Dynamo Insert record failed. The error received was User:
 arn:aws:sts::123456789012:assumed-role/dynamo-testbin/5aUMInJI is not authorized to
 perform: dynamodb:PutItem on resource: arn:aws:dynamodb:us-east-1:123456789012:table/
testbin
(Service: AmazonDynamoDBv2; Status Code: 400; Error Code: AccessDeniedException;
 Request ID: AKQJ987654321AKQJ987654321AKQJ987654321AKQJ987654321).
Message arrived on: test-topic, Action: dynamo, Table: trashbin, HashKeyField: id,
 HashKeyValue: id, RangeKeyField: None, RangeKeyValue: 123456789012
No newer events found at the moment. Retry.
```

Para obtener más información, consulte the section called “Visualización AWS IoT de los registros en la CloudWatch consola”.

Diagnóstico de servicios externos

El usuario final controla los servicios externos. Antes de ejecutar una regla, asegúrese de que los servicios externos que ha vinculado a dicha regla estén configurados y tengan suficientes unidades de rendimiento y capacidad para su aplicación.

Diagnóstico de problemas de SQL

Si la consulta SQL no devuelve los datos esperados:

- Revise los registros para ver si hay mensajes de error.
- Confirme que la sintaxis de SQL coincide con la del documento JSON del mensaje.

  Revise los nombres de objetos y propiedades utilizados en la consulta y compárelos con los utilizados en el documento JSON de la carga de mensajes del tema. Para obtener más información sobre el formato de JSON en consultas SQL, vaya a Extensiones JSON.
- Comprueba si los nombres de objetos o propiedades de JSON incluyen caracteres reservados o numéricos.

  Para obtener más información sobre los caracteres reservados en las referencias a objetos JSON dentro de consultas SQL, vaya a Extensiones JSON.

Diagnóstico de problemas relacionados con las sombras

Ayúdenos a mejorar este tema

Explíquenos cómo mejorarlo

Diagnóstico de sombras

Problema	Directrices para solucionar problemas
El documento de la sombra de un dispositivo se rechaza con `Invalid JSON document`.	Si no está familiarizado con JSON, modifique los ejemplos proporcionados en esta guía y adáptelos a sus necesidades. Para obtener más información, consulte Ejemplos de documento de sombra.
He enviado un JSON correcto, pero no se almacena o solo se almacena parcialmente en el documento de sombra del dispositivo.	Compruebe que ha seguido las directrices de formato JSON. Solo se almacenarán los campos JSON de las secciones `desired` y `reported`. 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 del dispositivo supera el tamaño máximo permitido.	La sombra del dispositivo admite únicamente 8 KB de datos. Intente acortar los nombres de los campos que están dentro del documento JSON o cree más sombras generando más objetos. Un dispositivo puede tener un número ilimitado de things/shadows asociados. El único requisito es que cada nombre de objeto debe ser único en la cuenta.
Cuando recibo la sombra de un dispositivo, esta supera los 8 KB. ¿Por qué pasa esto?	Al recibirlos, el AWS IoT servicio añade metadatos a la sombra del dispositivo. El servicio incluye estos datos en su respuesta, pero no cuentan para el límite de 8 KB. Para calcular el límite, solo se tienen en cuenta los

Problema	Directrices para solucionar problemas
	datos de estado `desired` y `reported` del documento de estado enviado a la sombra del dispositivo.
Mi solicitud se ha rechazado porque la versión es incorrecta. ¿Qué tengo que hacer?	Realice una operación GET para sincroniz arse con la última versión del documento de estado. Al usar MQTT, suscríbase al tema / update/accepted para recibir notificaciones sobre cambios de estado y la última versión del documento JSON.
La marca de tiempo está desajustada en varios segundos.	La marca de tiempo de los campos individua les y de todo el documento JSON se actualiza cuando el AWS IoT servicio recibe el documento o cuando se publica el documento estatal en el. /mensajeupdate/accepted and ./ update/delta. Los mensajes pueden retrasars e en la red, lo que puede provocar una demora de varios segundos en la marca de tiempo.
Mi dispositivo puede publicar en los temas de sombra correspondientes y suscribirse a ellos, pero cuando intento actualizar el documento de sombra mediante la API de REST de HTTP, recibo un mensaje HTTP 403.	Asegúrese de haber creado políticas en IAM para permitir el acceso a estos temas y a la acción correspondiente (UPDATE/GET/ DELETE) para las credenciales que utiliza. Las políticas de IAM y las de certificado son independientes.
Otros problemas.	El servicio Device Shadow registra los errores en CloudWatch Logs. Para identificar los problemas de configuración y del dispositivo, habilite CloudWatch los registros y consulte los registros para obtener información sobre la depuración.

Diagnosticar problemas con acciones del flujo de entrada de Salesforce IoT

Ayúdenos a mejorar este tema

Explíquenos cómo mejorarlo

Registro de seguimiento de ejecución

¿Cómo puedo ver el registro de seguimiento de ejecución de una acción de Salesforce?

Consulte la sección Supervise AWS IoT mediante registros CloudWatch. Una vez que haya activado los registros, podrá ver el seguimiento de la 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?

Vea los registros generados por la ejecución de la acción de Salesforce en CloudWatch los registros. Si lo ve`Action executed successfully`, significa que el motor de AWS IoT reglas recibió la confirmación de Salesforce IoT de que el mensaje se envió correctamente al flujo de entrada objetivo.

Si tiene problemas con la plataforma de Salesforce IoT, póngase en contacto con la ayuda de Salesforce IoT.

¿Qué hago si los mensajes no se han enviado correctamente a un flujo de entrada de Salesforce IoT?

Vea los registros generados por la ejecución de la acción de Salesforce en CloudWatch Logs. En función de la entrada de registro, 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 si la carga que envía presenta 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 recibe un error que no aparece aquí, póngase en contacto con AWS IoT Support.

Diagnóstico de los límites de flujo

Resolución para el error “Se ha superado el límite de flujo de su cuenta AWS

Si aparece `"Error: You have exceeded the limit for the number of streams in your AWS account."`, puede limpiar las secuencias no utilizadas de su cuenta en lugar de solicitar un aumento del límite.

Para limpiar una transmisión no utilizada que creaste con el SDK AWS CLI o el SDK:

```
aws iot delete-stream –stream-id value
```

Para obtener más detalles, consulte delete-stream.

> **Note**
>
> Puedes usar el `list-streams` comando para buscar la transmisión IDs.

Resolución para las desconexiones en una flota de dispositivos

Ayúdenos a mejorar este tema

Explíquenos cómo mejorarlo

AWS IoT las desconexiones de la flota de dispositivos pueden ocurrir por varias razones. En este artículo se explica cómo diagnosticar el motivo de una desconexión y cómo gestionar las desconexiones causadas por un mantenimiento regular del AWS IoT servicio o por un límite de regulación.

Para diagnosticar el motivo de la desconexión

Puede registrar el grupo de registros de AWSIotLogsV2 CloudWatchpara identificar el motivo de la desconexión en el `disconnectReason` campo de la entrada del registro.

También puede utilizar la función de eventos AWS IoT del ciclo de vida para identificar el motivo de la desconexión. Si te has suscrito al evento de desconexión del ciclo de vida (`$aws/events/presence/disconnected/`*`clientId`*), recibirás una notificación AWS IoT cuando se produzca la desconexión. Puede identificar el motivo de la desconexión en el campo `disconnectReason` de la notificación.

Para obtener más información, consulta las entradas de CloudWatch AWS IoT registro y los eventos del ciclo de vida.

Para solucionar problemas de desconexiones debidas al AWS IoT mantenimiento del servicio

Las desconexiones causadas por el mantenimiento AWS IoT del servicio se registran como un evento AWS IoT del ciclo `SERVER_INITIATED_DISCONNECT` de vida y. CloudWatch Para gestionar estas desconexiones, ajusta la configuración del lado del cliente para asegurarte de que tus dispositivos se puedan volver a conectar automáticamente a la plataforma. AWS IoT

Para solucionar problemas de desconexión debidos a una limitación

Las desconexiones provocadas por un límite de regulación se registran como un evento del ciclo de vida y. `THROTTLED` AWS IoT CloudWatch Para gestionar estas desconexiones, puede solicitar aumentos de límite del agente de mensajes cuando aumente el número de dispositivos.

Para obtener más información, consulte el agente de mensajes de AWS IoT.

AWS IoT Device Management guía de solución de problemas

Ayúdenos a mejorar este tema

Explíquenos cómo mejorarlo

Esta es la sección de solución de problemas de AWS IoT Device Management.

Temas

- AWS IoT Solución de problemas de trabajos
- Resolución de problemas de la indexación de flotas
- AWS IoT Solución de problemas del catálogo de paquetes de software de administración de dispositivos

AWS IoT Solución de problemas de trabajos

Esta es la sección de solución de problemas de AWS IoT Jobs.

¿Cómo puedo localizar un punto final AWS IoT de Jobs?

¿Cómo puedo localizar el punto final del plano de control de AWS IoT Jobs?

AWS IoT Jobs admite las operaciones de la API del plano de control mediante el protocolo HTTPS. Compruebe que se ha conectado al punto de conexión correcto del plano de control mediante el protocolo HTTPS.

Para obtener una lista de puntos finales AWS específicos de una región, consulte AWS IoT Núcleo: puntos finales del plano de control.

Para obtener una lista de los puntos de conexión del plano de control de AWS IoT Jobs que cumplen con la norma FIPS, consulte Puntos de enlace de FIPS por servicio.

> **Note**
>
> AWS IoT Trabaja y AWS IoT Core comparte los mismos puntos finales específicos de la región. AWS

¿Cómo puedo localizar el punto final del plano de datos AWS IoT de Jobs?

AWS IoT Jobs admite las operaciones de la API del plano de datos mediante los protocolos HTTPS y MQTT. Compruebe que se ha conectado al punto de conexión correcto del plano de datos mediante los protocolos HTTPS o MQTT.

- Protocolo HTTPS
  - Utilice el comando describe-endpoint de la CLI, como se ve a continuación, o la API de REST `DescribeEndpoint`. Para el tipo de punto de conexión, utilice `iot:Jobs`.

    ```
    aws iot describe-endpoint --endpoint-type iot:Jobs
    ```
- Protocolo MQTT
  - Utilice el comando describe-endpoint de la CLI, como se ve a continuación, o la API de REST `DescribeEndpoint`. Para el tipo de punto de conexión, utilice `iot:Data-ATS`.

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

Para obtener una lista de los puntos de conexión del plano de datos de AWS IoT Jobs que cumplen con la norma FIPS, consulte Puntos de enlace de FIPS por servicio.

¿Cómo superviso la actividad AWS IoT de Jobs y proporciono métricas?

La supervisión de la actividad de AWS IoT Jobs con Amazon CloudWatch proporciona visibilidad en tiempo real de las operaciones de AWS IoT Jobs en curso y ayuda a controlar los costes mediante CloudWatch alarmas a través de AWS IoT Rules. Debe configurar el registro antes de poder monitorear la actividad de AWS IoT Jobs y configurar CloudWatch las alarmas. Para obtener más información sobre la configuración del registro, consulte Configure el AWS IoT registro.

Para obtener más información sobre Amazon CloudWatch y sobre cómo configurar el permiso a través de un rol de usuario de IAM para usar CloudWatch los recursos, consulta Gestión de identidad y acceso para Amazon CloudWatch.

¿Cómo configuro las métricas y el monitoreo de AWS IoT Jobs con Amazon CloudWatch?

Para configurar el AWS IoT registro, sigue los pasos que se describen en Configurar el AWS IoT registro. AWS IoT la configuración del registro se puede realizar en la Consola de administración de AWS AWS CLI, o API. AWS IoT La configuración del registro para grupos de cosas específicos debe realizarse únicamente en la API AWS CLI o.

La sección AWS IoT de métricas de Jobs contiene las métricas AWS IoT de Jobs utilizadas para monitorear la actividad AWS IoT de Jobs. En él se explica cómo ver las métricas en Consola de administración de AWS y AWS CLI.

Además, puede configurar CloudWatch alarmas para que le avisen de métricas específicas que desee supervisar de cerca. Para obtener información sobre la configuración de alarmas, consulta Cómo usar CloudWatch las alarmas de Amazon.

Resolución de problemas en un solo dispositivo y en flotas de dispositivos

La ejecución de un trabajo mantiene un estado **QUEUED** de forma indefinida

Cuando la ejecución de un trabajo con un estado `QUEUED` no pasa al siguiente estado lógico, como `IN_PROGRESS`, `FAILED` o `TIMED_OUT`, el motivo puede ser una de las siguientes situaciones:

- Revisa la actividad de tu dispositivo en los CloudWatch registros ubicados en la CloudWatch consola. Para obtener más información, consulta Supervisar el AWS IoT uso de CloudWatch registros.
- Es posible que el rol de IAM asociado al trabajo y a la posterior ejecución del mismo no tengan los permisos correctos que aparecen en las declaraciones de la política de IAM adjuntas a ese rol de IAM. Utilice la API `describe-job` para identificar el rol de IAM vinculado a ese trabajo y a su posterior ejecución, y revise la política de IAM para comprobar si los permisos son correctos. Cuando se hayan actualizado las declaraciones de permisos de la política, debería poder ejecutar el comando de API `AssumeRole` en el recurso.

No se ha creado una ejecución de trabajo para mi objeto o grupo de objetos

Cuando un trabajo actualiza su estado a `IN_PROGRESS`, este empezará a distribuir el documento de trabajo a todos los dispositivos del grupo de destino. Esta actualización del estado creará una ejecución de trabajo para cada dispositivo de destino. Si no se ha creado una ejecución de trabajo para alguno de los dispositivos de destino, consulte los siguientes pasos:

- ¿Apunta el trabajo directamente al `thing`?, ¿tiene el trabajo un estado `IN_PROGRESS`?, ¿es un trabajo simultáneo? Si se cumplen las tres condiciones, el trabajo sigue enviando ejecuciones a todos los dispositivos del grupo de destino, y ese `thing` específico aún no ha recibido su ejecución.

- Revise los dispositivos de su grupo objetivo para ver el trabajo y el estado del trabajo en la consola AWS de administración o utilice el comando de la `describe-job`API.
- Utilice el comando de API `describe-job` para comprobar si el trabajo tiene la propiedad `IsConcurrent` establecida en true o false. Para obtener más información, consulte Job limits.

- El trabajo no apunta directamente al `thing`.
  - Si el `Thing` se ha añadido a un `ThingGroup` y el trabajo apuntaba al `ThingGroup`, compruebe que el `Thing` sea parte del `ThingGroup`.
  - Si se trata de un trabajo del tipo captura, tiene un estado `IN_PROGRESS` y es simultáneo, el trabajo sigue enviando ejecuciones de trabajo a todos los dispositivos del grupo de destino, y ese `Thing` específico aún no ha recibido su ejecución de trabajo.
  - Si se trata de un trabajo del tipo continuo, tiene un estado `IN_PROGRESS` y es simultáneo, el trabajo sigue enviando ejecuciones de trabajo a todos los dispositivos del grupo de destino, y ese `Thing` específico aún no ha recibido su ejecución de trabajo. En el caso de trabajos del tipo continuo, también puede eliminar el `Thing` del `ThingGroup` y, luego, añadir de nuevo el `Thing` al `ThingGroup`.
  - Si el trabajo es un trabajo instantáneo con un estado de estado `IN_PROGRESS` y no es simultáneo, es probable que AWS IoT Jobs no reconozca la relación de `ThingGroup` pertenencia `Thing` o pertenencia. Se recomienda añadir varios segundos de tiempo de espera después de la llamada a `AddThingToThingGroup` antes de crear el `Job`. Como alternativa, puede cambiar la selección de objetivos a `Continuous`, lo que hará que el servicio se encargue de cubrir el evento de asociación de pertenencia de `Thing` y `ThingGroup` retrasados.

El nuevo trabajo no se ejecuta debido a un error **`LimitedExceededException`**

Si hay un error al crear un trabajo y la respuesta de error es `LimitedExceededException`, llame a la API `list-jobs` y revise todos los trabajos con `isConcurrent=true` para determinar si está dentro del límite de simultaneidad de trabajos. Consulte Job limits para obtener información adicional sobre los trabajos simultáneos. Para ver los límites de simultaneidad de trabajos y solicitar un aumento en esos límites, consulte AWS IoT Device Management jobs limits and quotas.

Límite en el tamaño del documento de trabajo

El tamaño del documento de trabajo está limitado por el tamaño de la carga de MQTT. Si necesita un documento de trabajo de más de 32 kB (kilobytes) o 32 000 B (bytes), cree y almacene el documento de trabajo en Amazon S3 y añada una URL de objeto de Amazon S3 en el campo `documentSource` de la API `CreateJob` o mediante la AWS CLI. Para el Consola de administración de AWS, añada una URL de objeto de Amazon S3 en el cuadro de texto URL de Amazon S3 al crear un trabajo.

- Consola de administración de AWS crear documentación de trabajo: cree y gestione trabajos mediante el Consola de administración de AWS
- AWS CLI crear documentación de trabajo: cree y gestione trabajos mediante el AWS CLI
- `CreateJob`Documentación de la API: CreateJob

El mensaje MQTT del lado del dispositivo solicita limitación

Si recibe un código de error 400 `ThrottlingException`, el mensaje MQTT del lado del dispositivo ha fallado debido a que ha alcanzado el límite de solicitudes simultáneas del lado del dispositivo. Consulte AWS IoT Device Management jobs limits and quotas para obtener más información sobre las limitaciones y ver si es posible ajustarlas.

Error de tiempo de espera de la conexión

El código de error 400 `RequestExpired` indica un fallo de conexión debido a valores de espera bajos o de alta latencia en el lado del cliente.

- Consulte Testing connectivity with your device data endpoint para obtener información sobre cómo probar la conexión entre el lado del cliente y el lado del servidor.

Comando de API no válido

Confirme que se ha introducido el comando de API correcto para evitar que aparezca un mensaje de error diciendo que el comando de API no es válido. Consulte la documentación de referencia de las API de AWS IoT para obtener una lista completa de todos los comandos de API en AWS IoT .

Error de conexión en el lado del servicio

Un código de error 503 `ServiceUnavailable` indica que el error se origina en el servidor.

- Consulte Panel de AWS Health (todos los AWS servicios) para ver el estado actual de todos los AWS servicios.
- Consulte Panel de AWS Health (personal Cuenta de AWS) para ver el estado actual de su información personal Cuenta de AWS.

Resolución de problemas de la indexación de flotas

Solución de problemas de consultas de agregación en el servicio de indexación de flotas

Si tiene errores de falta de coincidencia de tipos, puede utilizar CloudWatch los registros para solucionar el problema. CloudWatch Los registros deben estar habilitados antes de que el servicio de indexación de flotas los escriba. Para obtener más información, consulte Supervise AWS IoT mediante registros CloudWatch.

Para hacer consultas de agregación en campos no administrados, debe especificar un campo que usted haya definido en el argumento `customFields` transferido a `UpdateIndexingConfiguration` o update-indexing-configuration. Si el valor del campo no coincide con el tipo de datos del campo configurado, este valor se omite al realizar una consulta de agregación.

Si un campo no se puede indexar porque el tipo no coincide, el servicio de indexación de flotas envía un registro de errores a Logs. CloudWatch El registro de errores contiene el nombre del campo, el valor que no se pudo convertir y el nombre de objeto del dispositivo. A continuación, se muestra un ejemplo de registro de error.

```
{
  "timestamp": "2017-02-20 20:31:22.932",
  "logLevel": "ERROR",
  "traceId": "79738924-1025-3a00-a669-7bec69f7f07a",
  "accountId": "000000000000",
  "status": "SucceededWithIssues",
  "eventType": "IndexingCustomFieldFailed",
  "thingName": "thing0",
  "failedCustomFields": [
    {
      "Name": "attributeName1",
      "Value": "apple",
      "ExpectedType": "String"
```

```
    },
    {
      "Name": "attributeName2",
      "Value": "2",
      "ExpectedType": "Boolean"
    }
  ]
}
```

Si un dispositivo se ha desconectado durante aproximadamente una hora, el valor `timestamp` del estado de conectividad podría no aparecer. En el caso de las sesiones persistentes, es posible que falte el valor después de que un cliente haya estado desconectado durante más tiempo del tiempo configurado time-to-live (TTL) para la sesión persistente. Los datos de estado de conectividad solo se indexan para las conexiones donde el ID de cliente tiene un nombre de objeto coincidente. (El ID de cliente es el valor que se utiliza para conectar un dispositivo a) AWS IoT Core.

Resolución de problemas en la configuración de la indexación de flotas

No se puede cambiar a versiones inferiores en la configuración de la indexación de flotas

No se puede cambiar la configuración de la indexación de flotas a una versión inferior cuando quiere eliminar los orígenes de datos asociados a una métrica de flotas o a un grupo dinámico.

Por ejemplo, si la configuración de indexación tiene datos de registro, datos de sombra y datos de conectividad y existe una métrica de flotas con la consulta `thingName:TempSensor* AND shadow.desired.temperature>80`, actualizar la configuración de indexación para incluir solo los datos de registro generará un error.

No se admite la modificación de los campos personalizados utilizados por las métricas de flota existentes.

No se puede actualizar la configuración de la indexación debido a que las métricas de flota o los grupos dinámicos son incompatibles

Si no puede actualizar la configuración de indexación porque las métricas de flota o los grupos dinámicos son incompatibles, elimine las métricas de flota o los grupos dinámicos incompatibles antes de actualizar la configuración de indexación.

Solución de problemas de la indexación de ubicaciones y geoconsultas

Para solucionar errores tipográficos no coincidentes en la indexación de ubicaciones y las consultas geográficas, puede habilitar los registros. CloudWatch Para obtener más información sobre cómo supervisar el AWS IoT uso CloudWatch, sigue la guía. step-by-step

Al indexar los datos de ubicación mediante geoconsultas, los campos de ubicación que especifique en `geoLocations` deben coincidir con los campos de ubicación que pase a `UpdateIndexingConfiguration`. Si hay una discrepancia, la indexación de la flota envía un error de tipo no coincidente a. CloudWatch El registro de errores contiene el nombre del campo, el valor que no se pudo convertir y el nombre de objeto del dispositivo.

A continuación, se muestra un ejemplo de registro de error.

```
{
"timestamp": "2023-11-09 01:39:43.466",
    "logLevel": "ERROR",
    "traceId": "79738924-1025-3a00-a669-7bec69f7f07a",
    "accountId": "123456789012",
    "status": "Failure",
    "eventType": "IndexingGeoLocationFieldFailed",
    "thingName": "thing0",
    "failedGeolocationFields": [
        {
"Name": "attributeName1",
            "Value": "apple",
            "ExpectedType": "Geopoint"
        }
    ],
    "reason": "failed to index the field because it could not be converted to one of
 the expected geoLocation formats."
}
```

Para obtener más información, consulte Indexación de datos de ubicación.

Resolución de problemas en las métricas de flotas

No se pueden ver los puntos de datos en CloudWatch

Si puedes crear una métrica de flota pero no puedes ver los puntos de datos CloudWatch, es probable que no tengas nada que cumpla con los criterios de la cadena de consulta.

Consulte este comando de ejemplo para ver cómo crear una métrica de flotas:

```
aws iot create-fleet-metric --metric-name "example_FM" --query-string
 "thingName:TempSensor* AND attributes.temperature>80" --period 60 --aggregation-field
 "attributes.temperature" --aggregation-type name=Statistics,values=count
```

Si no tiene un objeto que cumpla con los criterios de la cadena de consulta `--query-string "thingName:TempSensor* AND attributes.temperature>80"`:

- Con `values=count` ella, podrás crear una métrica de flota y habrá puntos de datos para mostrarlos CloudWatch. Los puntos de datos del valor `count` son siempre 0.
- Con `values` otras `count` opciones, podrás crear una métrica de flota, pero no verás la métrica de la flota ni habrá puntos de datos que mostrar CloudWatch. CloudWatch

AWS IoT Solución de problemas del catálogo de paquetes de software de administración de dispositivos

Esta es la sección de solución de problemas del catálogo de paquetes de software de administración de AWS IoT dispositivos.

Mensajes de error de la resolución de problemas general

En esta sección se enumeran los errores más comunes que se producen a lo largo del ciclo de vida de la versión del paquete de software.

Errores **`HeadBucket`**

Los siguientes mensajes de error aparecen al llamar a la operación de la API `HeadBucket` o al comando de la CLI `head-bucket` para validar el bucket de Amazon S3 utilizado para cargar archivos durante la implementación de un trabajo.

Para obtener más información sobre el uso de un bucket de Amazon S3 para cargar archivos durante la implementación de un trabajo, consulte URL prefirmada para la carga de archivos.

`InvalidRoleException`

```
    "Permission denied when attempting to use role %s to access bucket %s."
```

`InvalidRequestException`

```
    "Cross region S3 bucket is not supported for presigned url upload placeholder"
```

```
InvalidRequestException
    "S3 bucket in job document presigned url upload placeholder not found"
```

```
InvalidRequestException
    "Given S3 bucket name is invalid."
```

```
InvalidRequestException
    "Provided S3 bucket is not valid: %s. Error: %s"
```

Amazon S3 GetObject

El siguiente mensaje de error aparece cuando se proporciona un argumento no válido, lo que provoca un error en la operación de la API `GetObject` de Amazon S3.

```
InvalidRequestException
    "Provided argument for presigned url is invalid"
```

Compatibilidad con el identificador de versión de Amazon S3

Cuando solicite acceso a un bucket de Amazon S3 mediante el control de versiones, asegúrese de incluir su `versionId` pues, de lo contrario, podría aparecer el siguiente error.

Para obtener más información sobre los buckets de Amazon S3 que utilizan el control de versiones, consulte Uso del control de versiones en buckets de Amazon S3.

```
InvalidRequestException
    "VersionId not found when attempting to access s3 url"
```

Marcadores de posición dentro de una URL prefirmada para cargar archivos

Los siguientes mensajes de error aparecen cuando hay problemas con un marcador de posición dentro de una URL prefirmada que se utiliza para cargar archivos a un bucket de Amazon S3 de destino durante la implementación de un trabajo. Para obtener más información sobre el uso de un bucket de Amazon S3 para cargar archivos durante la implementación de un trabajo y el significado de un marcador de posición local, consulte URL prefirmada para la carga de archivos.

El siguiente mensaje de error aparece cuando no se reconoce el marcador de posición local.

```
InvalidJobDocumentException
    "Undefined placeholder, ${...}, inside of presign url upload parameter"
```

El siguiente mensaje de error aparece cuando se intenta utilizar el marcador de posición local en una URL prefirmada que no está destinada para la carga de archivos.

```
InvalidJobDocumentException
    "Local placeholder, ${...}, is only valid inside of presign url upload"
```

URL de Amazon S3 anidada incorrectamente

El siguiente mensaje de error aparece cuando la URL de Amazon S3 está anidada incorrectamente dentro de otro marcador de posición.

```
InvalidJobDocumentException
    "${aws:%s[...]} should not be the second layer pattern."
```

Anidación de artefactos de la versión del paquete

El siguiente mensaje de error aparece cuando la URL prefirmada con el artefacto de la versión del paquete está anidada incorrectamente dentro de otro marcador de posición.

```
InvalidJobDocumentException
    "${aws:iot:package:[...]:artifact:s3-presigned-url} cannot be nested inside another
 placeholder."
```

Falta el artefacto de la versión del paquete

El siguiente mensaje de error aparece cuando no se encuentra el artefacto de la versión del paquete al que se hace referencia.

```
InvalidJobDocumentException
    "Package %s version %s does not have an associated artifact to generate an S3
 presigned url."
```

Marcadores de posición de la versión y el paquete de software

El siguiente mensaje de error aparece cuando el marcador de posición del documento de trabajo para el paquete de software y la versión del paquete no se resuelven con los valores válidos deseados para la implementación del trabajo porque se hace referencia a varios paquetes de software y versiones de paquetes en el parámetro `destinationPackageVersions` o en la pestaña ARN de la página de detalles de Versión del paquete.

```
InvalidJobDocumentException
    "Cannot resolve empty package name and version name given multiple elements in
 destination package versions."
```

Uso de un paquete de software y una versión de paquete vacíos

El siguiente mensaje de error aparece cuando intenta utilizar un paquete vacío o una versión de paquete sin la otra en un documento de trabajo.

```
InvalidJobDocumentException
    "Empty package name and version name have to be used in pair."
```

Uso nulo en el documento de trabajo

El siguiente mensaje de error aparece al intentar especificar `$null` como versión de paquete en el documento de trabajo. `$null` solo se puede utilizar dentro del parámetro `destinationPackageVersions` cuando se utiliza la operación de la API `CreateJob`.

```
InvalidJobDocumentException
    "$null is not allowed to be referenced as a package version in job documents."
```

Todos los atributos en una versión de paquete

El siguiente mensaje de error aparece al intentar utilizar todos los atributos de una versión de paquete y rodearlos de texto o marcadores de posición adicionales.

Para obtener más información sobre el uso de todos los atributos en una versión de paquete de software, consulte Parámetros de sustitución de trabajos AWS IoT.

```
InvalidJobDocumentException
    "The package version attribute placeholder for all attributes has to be a
 json value by itself and not appended with other strings or nested with other
 placeholders."
```

Límite de marcadores de posición locales en la URL prefirmada para la carga de archivos

El siguiente mensaje de error aparece cuando se supera el límite de marcadores de posición locales utilizados en una URL prefirmada para cargar archivos durante la implementación de un trabajo.

Para obtener más información sobre el uso de una URL prefirmada para cargar archivos durante la implementación de un trabajo, consulte URL prefirmada para la carga de archivos.

```
InvalidJobDocumentException
    "The occurrence of local placeholder %s within S3 presigned url upload placeholder
 exceeds limit of %d."
```

Marcadores de posición locales en un bucket de Amazon S3

El siguiente mensaje de error aparece cuando intenta colocar una URL de marcador de posición local en el nombre del bucket de Amazon S3 para un marcador de posición de URL prefirmada que se utiliza para cargar archivos durante la implementación de un trabajo.

Para obtener más información sobre el uso de una URL prefirmada para cargar archivos durante la implementación de un trabajo, consulte URL prefirmada para la carga de archivos.

```
InvalidJobDocumentException
    "S3 bucket name in presigned url upload is not allowed to contain any placeholders"
```

Corchetes de apertura y cierre

El siguiente mensaje de error aparece al añadir un parámetro o marcador de posición a un documento de trabajo sin la llave de cierre “}”.

```
InvalidJobDocumentException
    "One or more parameters or placeholders are not terminated."
```

Rol de IAM con una URL prefirmada de Amazon S3

El siguiente mensaje de error aparece cuando intenta utilizar una URL prefirmada de Amazon S3 en un documento de trabajo sin un rol de IAM.

Para obtener más información sobre Amazon S3 presigned URLs, consulte Trabajar con URLs presigned.

```
InvalidRequestException
    "presignedUrlConfig role ARN is required to generate an S3 presigned url in job
 document."
```

Rol de IAM con una URL prefirmada de Amazon S3 para un artefacto de versión de paquete

El siguiente mensaje de error aparece cuando intenta utilizar una URL prefirmada de Amazon S3 que representa a un artefacto de versión de paquete en un documento de trabajo sin un rol de IAM.

```
InvalidRequestException
    "presignedUrlConfig role ARN is required to generate an S3 presigned url in job
 document for package %s version %s artifact."
```

Mensajes de error en la lista de materiales de software

En esta sección se enumeran los errores más comunes asociados a una lista de materiales de software (SBOM) vinculada a una versión de paquete.

Validación de entrada para la solicitud de asociación de SBOM

Al utilizar la operación de la API `AssociateSbomWithPackageVersion`, aparece el siguiente mensaje de error y el parámetro `s3Location` es nulo.

```
InvalidRequestException "Associate request needs to include SBOM reference"
```

Para obtener más información sobre el funcionamiento de la `AssociateSbomWithPackageVersion` API, consulte. AssociateSbomWithPackageVersion

Errores de validación de la SBOM

En esta sección se enumeran los errores más comunes que se producen durante la validación inicial de la lista de materiales de software (SBOM) cuando se asocia a una versión de paquete de software.

Al utilizar la operación de la API `AssociateSbomWithPackageVersion` y `bucket`, aparece el siguiente mensaje de error y el parámetro `s3Location` es nulo.

```
InvalidRequestException "S3 bucket name for SBOM cannot be null"
```

El siguiente mensaje de error aparece cuando la cadena en `bucket` del parámetro `s3Location` de la operación de la API `AssociateSbomWithPackageVersion` es demasiado larga.

```
InvalidRequestException "S3 bucket name for SBOM is illegal. String length exceeds
 limit"
```

El siguiente mensaje de error aparece cuando el parámetro `key` es nulo.

```
InvalidRequestException "S3 key name for SBOM cannot be null"
```

El siguiente mensaje de error aparece cuando la cadena en `key` del parámetro `s3Location` de la operación de la API `AssociateSbomWithPackageVersion` es demasiado larga.

```
InvalidRequestException "S3 key name for SBOM is illegal. String length exceeds limit"
```

El siguiente mensaje de error aparece cuando la cadena en `version` del parámetro `s3Location` de la operación de la API `AssociateSbomWithPackageVersion` es nulo.

```
InvalidRequestException "S3 object version for SBOM cannot be null"
```

El siguiente mensaje de error aparece cuando la cadena en `version` del parámetro `s3Location` de la operación de la API `AssociateSbomWithPackageVersion` es demasiado larga.

```
InvalidRequestException "S3 object version for SBOM is illegal. String length exceeds
 limit"
```

El siguiente mensaje de error aparece cuando el tamaño del archivo zip de la SBOM almacenado en el bucket de Amazon S3 es demasiado grande.

```
InvalidRequestException "S3 object file size exceeds limit"
```

El siguiente mensaje de error aparece cuando utiliza la operación de la API `AssociateSbomWithPackageVersion` y el número actual de validaciones de la SBOM en curso ya ha alcanzado el límite máximo.

```
LimitExceededException "Too many ongoing SBOM validation workflows. Please wait and
 retry"
```

Problemas de acceso con el archivo SBOM en el bucket de Amazon S3

El siguiente mensaje de error aparece cuando otra entidad no puede acceder al bucket de Amazon S3 porque el bucket no existe o no se han otorgado los permisos adecuados para acceder al bucket.

Para obtener más información acerca de los permisos necesarios para acceder al bucket de Amazon S3, consulte Almacenamiento de la lista de materiales de software.

```
InvalidRequestException "SBOM not accessible by the service. Please make sure the
 bucket exists and S3 permission is granted."
```

El siguiente mensaje de error aparece cuando otra entidad no puede acceder al archivo zip de la SBOM del parámetro `key` debido a que el bucket de Amazon S3 no existe o no se han otorgado los permisos adecuados para acceder al contenido almacenado en el bucket de Amazon S3.

```
InvalidRequestException "SBOM not accessible by the service. Please make sure the key
 exists and S3 permission is granted."
```

El siguiente mensaje de error aparece cuando otra entidad no puede acceder al bucket de Amazon S3 porque el bucket, la clave y el ID de versión no existen o no se han otorgado los permisos adecuados para acceder al bucket de Amazon S3. Además, este mensaje de error puede aparecer si los permisos concedidos no son suficientes para acceder al archivo zip de la SBOM en el bucket de Amazon S3.

```
InvalidRequestException "SBOM not accessible by the service. Please make sure the
 bucket/key/version exists and S3 permission is granted."
```

El siguiente mensaje de error aparece cuando otra entidad no puede acceder al bucket de Amazon S3 debido a que el bucket se encuentra en otra región.

```
InvalidRequestException "Cross-region S3 bucket for %s is not supported."
```

El siguiente mensaje de error aparece cuando otra entidad no puede acceder al bucket de Amazon S3 debido a que los parámetros `bucket`, `key` o `version` están mal escritos al utilizar la operación de la API `AssociateSbomWithPackageVersion`.

```
InvalidRequestException "Please make sure SBOM S3 bucket name/key length/version is
 valid"
```

AWS IoT Guía de solución de problemas de Device Advisor

Ayúdenos a mejorar este tema

Explíquenos cómo mejorarlo

General

P: ¿Puedo ejecutar varios conjuntos de pruebas en paralelo?

R: Sí. Ahora, Device Advisor admite la ejecución de varios conjuntos de pruebas en diferentes dispositivos mediante un punto de conexión en el nivel del dispositivo. Si utiliza el punto de conexión en el nivel de la cuenta, puede ejecutar un conjunto a la vez, ya que hay un punto de conexión de Device Advisor disponible por cuenta. Para obtener más información, consulte Configure your device.

P: He visto en mi dispositivo que Device Advisor ha rechazado la conexión TLS. ¿Eso es algo normal?

R: Sí. Device Advisor deniega la conexión TLS antes y después de cada ejecución de prueba. Recomendamos que los usuarios implementen un mecanismo de reintento del dispositivo para disfrutar de una experiencia de prueba totalmente automatizada con Device Advisor. Si ejecuta un conjunto de pruebas con más de un caso de prueba (por ejemplo, conexión TLS, conexión MQTT y publicación MQTT), le recomendamos que diseñe un mecanismo para su dispositivo. El mecanismo puede intentar conectarse a nuestro punto de conexión de prueba cada cinco segundos durante uno o dos minutos. De este modo, puede ejecutar varios casos de prueba en secuencia de forma automatizada.

P: ¿Puedo obtener un historial de todas las llamadas a la API de Device Advisor hechas en mi cuenta a fin de realizar tareas de análisis de seguridad y resolución de problemas operativos?

R: Sí. Para recibir un historial de las llamadas a la API de Device Advisor realizadas en su cuenta, solo tiene que activar la consola de AWS IoT administración y filtrar el origen del evento`iotdeviceadvisor.amazonaws.com`. CloudTrail

P: ¿Cómo puedo ver los inicios de sesión de Device Advisor CloudWatch?

R: Los registros generados durante la ejecución de un conjunto de pruebas se cargan CloudWatch si agrega la política requerida (por ejemplo CloudWatchFullAccess) a su función de

servicio (consulteConfiguración). Si hay al menos un caso de prueba en el conjunto de pruebas, se crea un grupo de registros "aws/iot/deviceadvisor/$testSuiteId" con dos flujos de registros. Una secuencia es el «$testRunId» e incluye registros de las acciones realizadas antes y después de ejecutar los casos de prueba en el conjunto de pruebas, como los pasos de configuración y limpieza. El otro flujo de registro es «$ suiteRunId _$»testRunId, que es específico de la ejecución de un conjunto de pruebas. Los eventos enviados desde los dispositivos se AWS IoT Core registrarán en este flujo de registro.

P: ¿Para qué sirve el rol de permisos del dispositivo?

R: Device Advisor se interpone entre su dispositivo de prueba y AWS IoT Core simula escenarios de prueba. Acepta las conexiones y los mensajes de sus dispositivos de prueba y los reenvía a AWS IoT Core ; para ello, asume el rol de permiso de su dispositivo e inicia una conexión en su nombre. Es importante asegurarse de que los permisos de función del dispositivo sean los mismos que los del certificado que utiliza para ejecutar las pruebas. AWS IoT las políticas de certificación no se aplican cuando Device Advisor inicia una conexión AWS IoT Core en tu nombre mediante la función de permisos del dispositivo. Sin embargo, se aplican los permisos del rol de permisos del dispositivo que usted establezca.

P: ¿En qué regiones es compatible Device Advisor?

R: Device Advisor es compatible en las regiones us-east-1, us-west-2, ap-northeast-1 y eu-west-1.

P: ¿Por qué veo resultados incoherentes?

R: Una de las principales causas de los resultados incoherentes es haber establecido el `EXECUTION_TIMEOUT` de una prueba en un valor demasiado bajo. Para obtener más información sobre los valores de `EXECUTION_TIMEOUT` recomendados y predeterminados, consulte Device Advisor test cases.

P: ¿Qué protocolo MQTT admite Device Advisor?

R: Device Advisor es compatible con la versión 3.1.1 de MQTT con certificados de cliente X509.

P: ¿Qué sucede si mi caso de prueba falla y aparece un mensaje de que se ha agotado el tiempo de espera de la ejecución aunque haya intentado conectar mi dispositivo al punto de conexión de prueba?

R: Valide todos los pasos de la sección Cree un rol de IAM para usarlo como rol de dispositivo. Si la prueba sigue dando error, es posible que el dispositivo no esté enviando la extensión de indicación de nombre de servidor (SNI) correcta, algo necesario para que Device Advisor

funcione. El valor de SNI correcto es la dirección del punto final que se devuelve al seguir la sección Configure su dispositivo. AWS IoT también requiere que los dispositivos envíen la extensión de indicación del nombre del servidor (SNI) al protocolo Transport Layer Security (TLS). Para obtener más información, consulte Seguridad del transporte en. AWS IoT

P: Mi conexión MQTT falla debido al error «libaws-c-mqtt: AWS_ERROR _MQTT_UNEXPECTED_HANGUP» (o) la conexión MQTT de mi dispositivo se desconecta automáticamente del terminal de Device Advisor. ¿Cómo se puede resolver este error?

R: Este código de error y las desconexiones inesperadas pueden deberse a muchos factores, pero lo más probable es que estén relacionados con el rol de dispositivo asociado a este. Los siguientes puntos de verificación (en orden de prioridad) resolverán este problema.

- El rol de dispositivo asociado al dispositivo debe tener los permisos de IAM mínimos necesarios para ejecutar las pruebas. Device Advisor utilizará la función de dispositivo adjunta para realizar acciones de AWS IoT MQTT en nombre del dispositivo de prueba. Si no se dispone de los permisos necesarios, aparecerá el error `AWS_ERROR_MQTT_UNEXPECTED_HANGUP` o habrá desconexiones inesperadas mientras el dispositivo intenta conectarse al punto de conexión de Device Advisor. Por ejemplo, si ha seleccionado ejecutar el caso de prueba MQTT Publish, las acciones Connect y Publish deben incluirse en el rol con el tema ClientId y el correspondiente (puede proporcionar varios valores mediante comas para separar los valores, y puede proporcionar valores de prefijo con un carácter comodín (*). Por ejemplo, para dar permiso para publicar sobre cualquier tema que empiece por `TestTopic`, puede ofrecer “`TestTopic*`“ como el valor del recurso. Aquí tiene algunos ejemplos de políticas.
- Los valores definidos en el rol de dispositivo para los tipos de recursos no coinciden con los valores reales utilizados en el código. Por ejemplo: en el código del dispositivo ClientId se define una discrepancia entre el rol y el código que ClientId se utiliza realmente. Los valores ClientId, como el tema, y TopicFilter deben ser idénticos en la función y el código del dispositivo.
- El certificado de dispositivo adjunto al dispositivo debe estar activo y tener una política asociada, con los permisos de acción necesarios para los recursos. Tenga en cuenta que la política de certificados de dispositivos concede o deniega el acceso a AWS IoT los recursos y a las operaciones del plano de AWS IoT Core datos. Device Advisor requiere que tenga un certificado de dispositivo activo adjunto a su dispositivo que otorgue los permisos de acción utilizados durante un caso de prueba.

AWS IoT errores

Ayúdenos a mejorar este tema

Explíquenos cómo mejorarlo

En esta sección se enumeran los códigos de error enviados por AWS IoT.

Códigos de error del agente de mensajes

Código de error	Descripción del error
400	Solicitud errónea.
401	Sin autorización.
403	prohibido.
426	Se requiere una actualización.
503	Servicio no disponible.

Identidad y códigos de error de seguridad

Código de error	Descripción del error
401	Sin autorización.

Códigos de error de sombra de dispositivo

Código de error	Descripción del error
400	Solicitud errónea.
401	Sin autorización.
403	prohibido.
404	No encontrado.

Código de error	Descripción del error
409	Conflicto.
413	Solicitud demasiado grande.
422	No se pudo procesar una solicitud.
429	Demasiadas solicitudes.
500	Error interno.
503	Servicio no disponible.

SDK de dispositivos, SDK para móviles y cliente de dispositivo de AWS IoT

Esta página resume los SDK de dispositivos de AWS IoT, las bibliotecas de código abierto, las guías para desarrolladores, las aplicaciones de muestra y las guías de portabilidad para ayudarlo a crear soluciones de IoT innovadoras con AWS IoT y las plataformas de hardware que elija.

Estos SDK son para su uso en el dispositivo de IoT. Si está desarrollando una aplicación de IoT para usarla en un dispositivo móvil, consulte los AWS Mobile SDK. Si está desarrollando una aplicación de IoT o un programa del lado del servidor, consulte los AWS SDKs.

SDK de dispositivos de 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.

> **Note**
>
> Los SDK de dispositivos de AWS IoT han lanzado un cliente MQTT 5. Los SDK de dispositivos de AWS IoT no admiten el uso de TLS 1.3 en macOS.

Estos SDK le ayudan a conectar los dispositivos de IoT a AWS IoT mediante los protocolos MQTT y WSS.

C++

SDK de dispositivosAWS IoT para C++

El SDK de dispositivos de AWS IoT para C++ permite a los desarrolladores compilar aplicaciones conectadas mediante AWS y las API de AWS IoT. En concreto, este SDK se diseñó para los dispositivos que no tienen limitación de recursos y requieren características avanzadas, como la puesta en cola de mensajes, la compatibilidad con varios procesos y las características de idioma más actualizadas. Para obtener más información, consulte los siguientes temas:

- AWS IoT SDK de dispositivo para C++ v2 en GitHub

- AWS IoT Archivo Léame del SDK de dispositivos de para C++ v
- AWS IoT Ejemplos de SDK de dispositivos de para C++ v
- AWS IoT Documentación de la API del SDK de dispositivos para C++ v

Python

SDK de dispositivos de AWS IoT para Python

El SDK de dispositivos AWS IoT 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 proporcionados por AWS IoT y con otros servicios de AWS, como AWS Lambda, Kinesis, Amazon S3, etc.

- SDK de dispositivos de AWS IoT para Python v2 en GitHub
- AWS IoT Archivo Readme del SDK de dispositivos para Python v
- AWS IoT Ejemplos del SDK de dispositivos para Python v
- AWS IoT Documentación de la API del SDK de dispositivos para Python v

JavaScript

SDK de dispositivos de AWS IoT 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 SDK de dispositivos para JavaScript v2 en GitHub
- AWS IoT Archivo Readme del SDK de dispositivos para JavaScript v
- AWS IoT Ejemplos del SDK de dispositivos para JavaScript v
- AWS IoT Documentación de la API del SDK de dispositivos para JavaScript v

Java

SDK de dispositivos AWS IoT para Java

El SDK de dispositivos AWS IoT 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 las sombras. Puede tener acceso a las sombras mediante los métodos GET, UPDATE y DELETE de HTTP. El SDK es también compatible con un modelo de acceso a sombras simplificado, lo que permite a los desarrolladores intercambiar datos con las sombras utilizando únicamente métodos getter y setter, sin tener que serializar ni deserializar documentos JSON.

> **Note**
>
> El SDK de dispositivos de AWS IoT para Java v2 ahora es compatible con el desarrollo de Android. Para obtener más información, consulte AWS IoT Device SDK for Android.

Para obtener más información, consulte los siguientes temas:

- AWS IoT SDK de dispositivos para Java v2 en GitHub
- AWS IoT Archivo Readme del SDK de dispositivos para Java v
- AWS IoT Ejemplos del SDK de dispositivos para Java v
- AWS IoT Documentación de la API del SDK de dispositivos para Java v

Swift

SDK para dispositivos con AWS IoT para Swift

El SDK para dispositivos con AWS IoT para Swift permite a los desarrolladores de Swift crear aplicaciones de AWS IoT para las plataformas macOS, iOS y tvOS de Apple y Linux mediante el protocolo MQTT 5.

Para obtener más información, consulte los siguientes temas:

- AWS IoT Device SDK for Swift on GitHub
- AWS IoT Device SDK for Swift Readme
- AWS IoT Device SDK for Swift Samples

SDK de dispositivos de AWS IoT para Embedded C

> **Note**
>
> Este SDK está diseñado para que lo utilicen desarrolladores de software incrustado con experiencia.

El AWS IoT Device SDK para Embedded C (C-SDK) es un conjunto de archivos de origen C con licencia de código abierto del MIT, que se puede utilizar en aplicaciones integradas para establecer conexiones seguras con dispositivos IoT en AWS IoT Core. Incluye un cliente MQTT, un analizador JSON y Device Shadow de AWS IoT, Jobs de AWS IoT, Fleet Provisioning de AWS IoT y bibliotecas de AWS IoT Device Defender. Este SDK se distribuye como código fuente y puede integrarse en el firmware cliente junto con código de aplicación y un sistema operativo (OS) de su elección.

AWS IoT Device SDK para Embedded C generalmente se dirige a dispositivos con limitaciones de recursos que requieren un tiempo de ejecución optimizado del lenguaje C. Puede usar el SDK en cualquier sistema operativo y alojarlo en cualquier tipo de procesador (por ejemplo, MCU y MPU).

Para obtener más información, consulte los siguientes temas:

- AWS IoT SDK de dispositivos para Embedded C en GitHub
- AWS IoT Achivo Readme del SDK de dispositivos para Embedded C Readme
- AWS IoT Ejemplos de SDK de dispositivos de para Embedded C

AWS Mobile SDK

Los SDK para móviles de AWS proporcionan a los desarrolladores de aplicaciones móviles soporte específico de cada plataforma para las API de los servicios de AWS IoT Core, la comunicación de dispositivos IoT mediante MQTT y las API de otros servicios de AWS.

Android

AWS Mobile SDK para Android

El AWS Mobile SDK para Android contiene una biblioteca, ejemplos y documentación para que los desarrolladores creen aplicaciones móviles conectadas mediante AWS. Este SDK también admite las comunicaciones entre dispositivos mediante MQTT y las llamadas a las API de los servicios de AWS IoT Core. Para obtener más información, consulte los siguientes temas:

- AWS Mobile SDK para Android en GitHub
- AWS Mobile SDK para Android Archivo Readme del
- AWS Mobile SDK para Android Ejemplos del
- AWS Mobile SDK para Android Referencia de la API del
- Documentación de referencia de la clase AWSIoTClient

iOS

AWS Mobile SDK for iOS

El AWS Mobile SDK for iOS es un kit de desarrollo de software de código abierto, distribuido con licencia de Apache Open Source. El AWS Mobile SDK for iOS contiene una biblioteca, ejemplos de código y documentación para ayudar a los desarrolladores a crear aplicaciones móviles conectadas mediante AWS. Este SDK también admite las comunicaciones entre dispositivos mediante MQTT y las llamadas a las API de los servicios de AWS IoT Core. Para obtener más información, consulte los siguientes temas:

- AWS Mobile SDK for iOS en GitHub
- AWS Mobile SDK for iOS Archivo Readme del
- AWS Mobile SDK for iOS Ejemplos del
- Los documentos de referencia de la clase AWSIoT están en el AWS Mobile SDK for iOS

Cliente de dispositivo de AWS IoT

El cliente de dispositivo de AWS IoT proporciona un código para ayudar al dispositivo a conectarse a AWS IoT, realizar tareas de aprovisionamiento de flotas, respaldar las políticas de seguridad del dispositivo, conectarse mediante túneles seguros y procesar trabajos en el dispositivo. Puede instalar este software en el dispositivo para gestionar dichas tareas rutinarias y así poder centrarse en su solución específica.

> **Note**
>
> El cliente de dispositivo de AWS IoT funciona con dispositivos de IoT basados en microprocesadores con procesadores x86_64 o ARM y sistemas operativos Linux comunes.

C++

AWS IoT Cliente de dispositivo de

Para obtener más información acerca del cliente de dispositivo de AWS IoT en C++, consulte lo siguiente:

- AWS IoT Cliente de dispositivo de en código fuente C++ en GitHub
- AWS IoT Archivo Readme del cliente de dispositivo de en C

Versiones anteriores de los SDK de dispositivos de AWS IoT

Se trata de versiones anteriores de los SDK de dispositivos de AWS IoT que se han sustituido por las versiones más recientes indicadas anteriormente. Estos SDK solo reciben actualizaciones de mantenimiento y seguridad. No se actualizarán para incluir nuevas características y no se deben usar en nuevos proyectos.

- AWS IoT C++ Device SDK en GitHub
- AWS IoT C++ Device SDK Readme
- AWS IoT Device SDK for Python v1 on GitHub
- AWS IoT Device SDK for Python v1 Readme
- AWS IoT Device SDK for Java on GitHub
- AWS IoT Device SDK for Java Readme
- AWS IoT Device SDK for JavaScript on GitHub
- AWS IoT Device SDK for JavaScript Readme
- Arduino Yún SDK on GitHub
- Léame de Arduino Yún SDK

Ejemplos de código para AWS IoT usar AWS SDKs

Los siguientes ejemplos de código muestran cómo usarlo AWS IoT con un kit de desarrollo de AWS software (SDK).

Los conceptos básicos son ejemplos de código que muestran cómo realizar las operaciones esenciales dentro de un servicio.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

Para obtener una lista completa de las guías para desarrolladores del AWS SDK y ejemplos de código, consulteSe utiliza AWS IoT con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Ejemplos de código

- Ejemplos básicos de AWS IoT uso AWS SDKs
  - Hola AWS IoT
  - Aprenda los conceptos básicos del AWS IoT uso de un AWS SDK
  - Acciones de AWS IoT uso AWS SDKs
    - Úselo AttachThingPrincipal con un AWS SDK o CLI
    - Úselo CreateKeysAndCertificate con un AWS SDK o CLI
    - Úselo CreateThing con un AWS SDK o CLI
    - Úselo CreateTopicRule con un AWS SDK o CLI
    - Úselo DeleteCertificate con un AWS SDK o CLI
    - Úselo DeleteThing con un AWS SDK o CLI
    - Úselo DeleteTopicRule con un AWS SDK o CLI
    - Úselo DescribeEndpoint con un AWS SDK o CLI
    - Úselo DescribeThing con un AWS SDK o CLI
    - Úselo DetachThingPrincipal con un AWS SDK o CLI
    - Úselo ListCertificates con un AWS SDK o CLI
    - Úselo ListThings con un AWS SDK o CLI
    - Úselo SearchIndex con un AWS SDK o CLI

- Úselo UpdateIndexingConfiguration con un AWS SDK o CLI
- Úselo UpdateThing con un AWS SDK o CLI

Ejemplos básicos de AWS IoT uso AWS SDKs

Los siguientes ejemplos de código muestran cómo utilizar los conceptos básicos de AWS IoT with AWS SDKs.

Ejemplos

- Hola AWS IoT
- Aprenda los conceptos básicos del AWS IoT uso de un AWS SDK
- Acciones de AWS IoT uso AWS SDKs
  - Úselo AttachThingPrincipal con un AWS SDK o CLI
  - Úselo CreateKeysAndCertificate con un AWS SDK o CLI
  - Úselo CreateThing con un AWS SDK o CLI
  - Úselo CreateTopicRule con un AWS SDK o CLI
  - Úselo DeleteCertificate con un AWS SDK o CLI
  - Úselo DeleteThing con un AWS SDK o CLI
  - Úselo DeleteTopicRule con un AWS SDK o CLI
  - Úselo DescribeEndpoint con un AWS SDK o CLI
  - Úselo DescribeThing con un AWS SDK o CLI
  - Úselo DetachThingPrincipal con un AWS SDK o CLI
  - Úselo ListCertificates con un AWS SDK o CLI
  - Úselo ListThings con un AWS SDK o CLI
  - Úselo SearchIndex con un AWS SDK o CLI
  - Úselo UpdateIndexingConfiguration con un AWS SDK o CLI
  - Úselo UpdateThing con un AWS SDK o CLI

Hola AWS IoT

En los siguientes ejemplos de código se muestra cómo empezar a utilizar AWS IoT.

.NET

SDK for .NET (v4)

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
/// <summary>
/// Hello AWS IoT example.
/// </summary>
public class HelloIoT
{
    /// <summary>
    /// Main method to run the Hello IoT example.
    /// </summary>
    /// <param name="args">Command line arguments.</param>
    /// <returns>A Task object.</returns>
    public static async Task Main(string[] args)
    {
        var iotClient = new AmazonIoTClient();

        try
        {
            Console.WriteLine("Hello AWS IoT! Let's list your IoT Things:");
            Console.WriteLine(new string('-', 80));

            // Use pages of 10.
            var request = new ListThingsRequest()
            {
                MaxResults = 10
            };
            var response = await iotClient.ListThingsAsync(request);

            // Since there is not a built-in paginator, use the NextMarker to
 paginate.
            bool hasMoreResults = true;

            var things = new List<ThingAttribute>();
            while (hasMoreResults)
```

```
            {
                things.AddRange(response.Things);

                // If NextMarker is not null, there are more results. Get the
next page of results.
                if (!String.IsNullOrEmpty(response.NextMarker))
                {
                    request.Marker = response.NextMarker;
                    response = await iotClient.ListThingsAsync(request);
                }
                else
                    hasMoreResults = false;
            }

            if (things is { Count: > 0 })
            {
                Console.WriteLine($"Found {things.Count} IoT Things:");
                foreach (var thing in things)
                {
                    Console.WriteLine($"- Thing Name: {thing.ThingName}");
                    Console.WriteLine($"  Thing ARN: {thing.ThingArn}");
                    Console.WriteLine($"  Thing Type: {thing.ThingTypeName ?? "No
type specified"}");
                    Console.WriteLine($"  Version: {thing.Version}");

                    if (thing.Attributes?.Count > 0)
                    {
                        Console.WriteLine("  Attributes:");
                        foreach (var attr in thing.Attributes)
                        {
                            Console.WriteLine($"    {attr.Key}: {attr.Value}");
                        }
                    }
                    Console.WriteLine();
                }
            }
            else
            {
                Console.WriteLine("No IoT Things found in your account.");
                Console.WriteLine("You can create IoT Things using the IoT Basics
scenario example.");
            }

            Console.WriteLine("Hello IoT completed successfully.");
```

```
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            Console.WriteLine($"Request throttled, please try again later:
 {ex.Message}");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't list Things. Here's why: {ex.Message}");
        }
    }
}
```

- Para obtener información sobre la API, consulte listThings en la Referencia de la API de AWS SDK for .NET .

C++

SDK para C++

Código para el CMake archivo CMake Lists.txt.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS iot)

# Set this project's name.
project("hello_iot")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
```

```
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

    # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
    # and set the proper subdirectory to the executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_iot.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Código del archivo de origen hello_iot.cpp.

```
#include <aws/core/Aws.h>
#include <aws/iot/IoTClient.h>
#include <aws/iot/model/ListThingsRequest.h>
#include <iostream>

/*
 *  A "Hello IoT" starter application which initializes an AWS IoT client and
 *  lists the AWS IoT topics in the current account.
 *
 *  main function
 *
 *  Usage: 'hello_iot'
 *
 */

int main(int argc, char **argv) {
```

```
    Aws::SDKOptions options;
    //  Optional: change the log level for debugging.
    //  options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::IoT::IoTClient iotClient(clientConfig);
        // List the things in the current account.
        Aws::IoT::Model::ListThingsRequest listThingsRequest;

        Aws::String nextToken; // Used for pagination.
        Aws::Vector<Aws::IoT::Model::ThingAttribute> allThings;

        do {
            if (!nextToken.empty()) {
                listThingsRequest.SetNextToken(nextToken);
            }

            Aws::IoT::Model::ListThingsOutcome listThingsOutcome =
iotClient.ListThings(
                    listThingsRequest);
            if (listThingsOutcome.IsSuccess()) {
                const Aws::Vector<Aws::IoT::Model::ThingAttribute> &things =
listThingsOutcome.GetResult().GetThings();
                allThings.insert(allThings.end(), things.begin(), things.end());
                nextToken = listThingsOutcome.GetResult().GetNextToken();
            }
            else {
                std::cerr << "List things failed"
                          << listThingsOutcome.GetError().GetMessage() <<
std::endl;
                break;
            }
        } while (!nextToken.empty());

        std::cout << allThings.size() << " thing(s) found." << std::endl;
        for (auto const &thing: allThings) {
            std::cout << thing.GetThingName() << std::endl;
        }
    }
```

```
    Aws::ShutdownAPI(options); // Should only be called once.
    return 0;
}
```

- Para obtener información sobre la API, consulte listThings en la Referencia de la API de AWS SDK para C++ .

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iot.IotClient;
import software.amazon.awssdk.services.iot.model.ListThingsRequest;
import software.amazon.awssdk.services.iot.model.ListThingsResponse;
import software.amazon.awssdk.services.iot.model.ThingAttribute;
import software.amazon.awssdk.services.iot.paginators.ListThingsIterable;

import java.util.List;

public class HelloIoT {
    public static void main(String[] args) {
        System.out.println("Hello AWS IoT. Here is a listing of your AWS IoT
 Things:");
        IotClient iotClient = IotClient.builder()
            .region(Region.US_EAST_1)
            .build();
```

```
        listAllThings(iotClient);
    }

    public static void listAllThings(IotClient iotClient) {
        iotClient.listThingsPaginator(ListThingsRequest.builder()
                .maxResults(10)
                .build())
            .stream()
            .flatMap(response -> response.things().stream())
            .forEach(attribute -> {
                System.out.println("Thing name: " + attribute.thingName());
                System.out.println("Thing ARN: " + attribute.thingArn());
            });
    }
}
```

- Para obtener información sobre la API, consulte listThings en la Referencia de la API de AWS SDK for Java 2.x .

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import aws.sdk.kotlin.services.iot.IotClient
import aws.sdk.kotlin.services.iot.model.ListThingsRequest

suspend fun main() {
    println("A listing of your AWS IoT Things:")
    listAllThings()
}

suspend fun listAllThings() {
    val thingsRequest =
        ListThingsRequest {
```

```
            maxResults = 10
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.listThings(thingsRequest)
        val thingList = response.things
        if (thingList != null) {
            for (attribute in thingList) {
                println("Thing name ${attribute.thingName}")
                println("Thing ARN: ${attribute.thingArn}")
            }
        }
    }
}
```

- Para obtener más información sobre la API, consulte listThings en la Referencia de la API de AWS SDK para Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
def hello_iot():
    """
    Use the AWS SDK for Python (Boto3) to create an AWS IoT client and list
    up to 10 things in your AWS IoT account.
    This example uses the default settings specified in your shared credentials
    and config files.
    """
    try:
        iot_client = boto3.client("iot")
        response = iot_client.list_things(maxResults=10)
        things = response.get("things", [])
```

```
        print("Hello, AWS IoT! Here are your things:")
        if things:
            for i, thing in enumerate(things, 1):
                print(f"{i}. {thing['thingName']}")
        else:
            print("No things found in your AWS IoT account.")
    except ClientError as e:
        if e.response["Error"]["Code"] == "UnauthorizedException":
            print("You don't have permission to access AWS IoT.")
        else:
            print(f"Couldn't access AWS IoT. Error: {e}")
    except NoCredentialsError:
        print("No AWS credentials found. Please configure your credentials.")
    except Exception as e:
        print(f"An unexpected error occurred: {e}")
```

- Para obtener información sobre la API, consulte listThings en la Referencia de la API de AWS SDK para Python (Boto3).

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteSe utiliza AWS IoT con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Aprenda los conceptos básicos del AWS IoT uso de un AWS SDK

En el siguiente ejemplo de código, se muestra cómo:

- Crea una AWS IoT cosa.
- Generar un certificado de dispositivo.
- Actualiza AWS IoT cualquier cosa con atributos.
- Devolver un punto de conexión único.
- Enumere sus AWS IoT certificados.
- Actualiza y AWS IoT oculta.
- Escribir la información de estado.
- Crear una regla.
- Mostrar las reglas.

- Buscar objetos con el nombre de objeto.
- Elimina cualquier AWS IoT cosa.

.NET

SDK for .NET (v4)

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Ejecute un escenario interactivo que demuestre AWS IoT las funciones.

```
/// <summary>
/// Scenario class for AWS IoT basics.
/// </summary>
public class IoTBasics
{
    public static bool IsInteractive = true;
    public static IoTWrapper? Wrapper = null;
    public static IAmazonCloudFormation? CloudFormationClient = null;
    public static ILogger<IoTBasics> logger = null!;
    private static IoTWrapper _iotWrapper = null!;
    private static IAmazonCloudFormation _amazonCloudFormation = null!;
    private static ILogger<IoTBasics> _logger = null!;

    private static string _stackName = "IoTBasicsStack";
    private static string _stackResourcePath = "../../../../../../scenarios/
basics/iot/iot_usecase/resources/cfn_template.yaml";

    /// <summary>
    /// Main method for the IoT Basics scenario.
    /// </summary>
    /// <param name="args">Command line arguments.</param>
    /// <returns>A Task object.</returns>
    public static async Task Main(string[] args)
    {
        // Set up dependency injection for the Amazon service.
        using var host = Host.CreateDefaultBuilder(args)
```

```
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonIoT>(new AWSOptions() { Region =
 RegionEndpoint.USEast1 })
                    .AddAWSService<IAmazonCloudFormation>()
                        .AddTransient<IoTWrapper>()
                        .AddLogging(builder => builder.AddConsole())
                        .AddSingleton<IAmazonIotData>(sp =>
                        {
                            var iotService = sp.GetRequiredService<IAmazonIoT>();
                            var request = new DescribeEndpointRequest
                            {
                                EndpointType = "iot:Data-ATS"
                            };
                            var response =
 iotService.DescribeEndpointAsync(request).Result;
                            return new AmazonIotDataClient($"https://
{response.EndpointAddress}/");
                        })
            )
            .Build();

        logger = LoggerFactory.Create(builder => builder.AddConsole())
            .CreateLogger<IoTBasics>();

        Wrapper = host.Services.GetRequiredService<IoTWrapper>();
        CloudFormationClient =
 host.Services.GetRequiredService<IAmazonCloudFormation>();

        // Set the private fields for backwards compatibility
        _logger = logger;
        _iotWrapper = Wrapper;
        _amazonCloudFormation = CloudFormationClient;

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Welcome to the AWS IoT example scenario.");
        Console.WriteLine("This example program demonstrates various interactions
 with the AWS Internet of Things (IoT) Core service.");
        Console.WriteLine();
        if (IsInteractive)
        {
            Console.WriteLine("Press Enter to continue...");
            Console.ReadLine();
        }
        Console.WriteLine(new string('-', 80));
```

```
        try
        {
            await RunScenarioAsync();
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "There was a problem running the scenario.");
            Console.WriteLine($"\nAn error occurred: {ex.Message}");
        }

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("The AWS IoT scenario has successfully completed.");
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Run the IoT Basics scenario.
    /// </summary>
    /// <returns>A Task object.</returns>
    public static async Task RunScenarioAsync()
    {
        // Use static properties if available, otherwise use private fields
        var iotWrapper = Wrapper ?? _iotWrapper;
        var cloudFormationClient = CloudFormationClient ?? _amazonCloudFormation;
        var scenarioLogger = logger ?? _logger;

        await RunScenarioInternalAsync(iotWrapper, cloudFormationClient,
 scenarioLogger);
    }

    /// <summary>
    /// Internal method to run the IoT Basics scenario with injected
 dependencies.
    /// </summary>
    /// <param name="iotWrapper">The IoT wrapper instance.</param>
    /// <param name="cloudFormationClient">The CloudFormation client instance.</
param>
    /// <param name="scenarioLogger">The logger instance.</param>
    /// <returns>A Task object.</returns>
    private static async Task RunScenarioInternalAsync(IoTWrapper iotWrapper,
 IAmazonCloudFormation cloudFormationClient, ILogger<IoTBasics> scenarioLogger)
    {
        string thingName = $"iot-thing-{Guid.NewGuid():N}";
```

```
        string certificateArn = "";
        string certificateId = "";
        string ruleName = $"iotruledefault";
        string snsTopicArn = "";

        try
        {
            // Step 1: Create an AWS IoT Thing
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("1. Create an AWS IoT Thing.");
            Console.WriteLine("An AWS IoT Thing represents a virtual entity in
the AWS IoT service that can be associated with a physical device.");
            Console.WriteLine();

            if (IsInteractive)
            {
                Console.Write("Enter Thing name: ");
                var userInput = Console.ReadLine();
                if (!string.IsNullOrEmpty(userInput))
                    thingName = userInput;
            }
            else
            {
                Console.WriteLine($"Using default Thing name: {thingName}");
            }

            var thingArn = await iotWrapper.CreateThingAsync(thingName);
            Console.WriteLine($"{thingName} was successfully created. The ARN
value is {thingArn}");
            Console.WriteLine(new string('-', 80));

            // Step 1.1: List AWS IoT Things
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("2. List AWS IoT Things.");
            Console.WriteLine("Now let's list the IoT Things to see the Thing we
just created.");
            Console.WriteLine();
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }

            var things = await iotWrapper.ListThingsAsync();
```

```
            Console.WriteLine($"Found {things.Count} IoT Things:");
            foreach (var thing in things.Take(10)) // Show first 10 things
            {
                Console.WriteLine($"Thing Name: {thing.ThingName}");
                Console.WriteLine($"Thing ARN: {thing.ThingArn}");
                if (thing.Attributes != null && thing.Attributes.Any())
                {
                    Console.WriteLine("Attributes:");
                    foreach (var attr in thing.Attributes)
                    {
                        Console.WriteLine($"  {attr.Key}: {attr.Value}");
                    }
                }
                Console.WriteLine("--------------");
            }
            Console.WriteLine();
            Console.WriteLine(new string('-', 80));

            // Step 2: Generate a Device Certificate
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("3. Generate a device certificate.");
            Console.WriteLine("A device certificate performs a role in securing
the communication between devices (Things) and the AWS IoT platform.");
            Console.WriteLine();

            var createCert = "y";
            if (IsInteractive)
            {
                Console.Write($"Do you want to create a certificate for
{thingName}? (y/n)");
                createCert = Console.ReadLine();
            }
            else
            {
                Console.WriteLine($"Creating certificate for {thingName}...");
            }

            if (createCert?.ToLower() == "y")
            {
                var certificateResult = await
iotWrapper.CreateKeysAndCertificateAsync();
                if (certificateResult.HasValue)
                {
                    var (certArn, certPem, certId) = certificateResult.Value;
```

```
                    certificateArn = certArn;
                    certificateId = certId;

                    Console.WriteLine($"\nCertificate:");
                    // Show only first few lines of certificate for brevity
                    var lines = certPem.Split('\n');
                    for (int i = 0; i < Math.Min(lines.Length, 5); i++)
                    {
                        Console.WriteLine(lines[i]);
                    }
                    if (lines.Length > 5)
                    {
                        Console.WriteLine("...");
                    }

                    Console.WriteLine($"\nCertificate ARN:");
                    Console.WriteLine(certificateArn);

                    // Step 3: Attach the Certificate to the AWS IoT Thing
                    Console.WriteLine("Attach the certificate to the AWS IoT
Thing.");
                    var attachResult = await
iotWrapper.AttachThingPrincipalAsync(thingName, certificateArn);
                    if (attachResult)
                    {
                        Console.WriteLine("Certificate attached to Thing
successfully.");
                    }
                    else
                    {
                        Console.WriteLine("Failed to attach certificate to
Thing.");
                    }

                    Console.WriteLine("Thing Details:");
                    Console.WriteLine($"Thing Name: {thingName}");
                    Console.WriteLine($"Thing ARN: {thingArn}");
                }
                else
                {
                    Console.WriteLine("Failed to create certificate.");
                }
            }
            Console.WriteLine(new string('-', 80));
```

```
            // Step 4: Update an AWS IoT Thing with Attributes
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("4. Update an AWS IoT Thing with Attributes.");
            Console.WriteLine("IoT Thing attributes, represented as key-value
pairs, offer a pivotal advantage in facilitating efficient data");
            Console.WriteLine("management and retrieval within the AWS IoT
ecosystem.");
            Console.WriteLine();
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }

            var attributes = new Dictionary<string, string>
            {
                { "Location", "Seattle" },
                { "DeviceType", "Sensor" },
                { "Firmware", "1.2.3" }
            };

            await iotWrapper.UpdateThingAsync(thingName, attributes);
            Console.WriteLine("Thing attributes updated successfully.");
            Console.WriteLine(new string('-', 80));

            // Step 5: Return a unique endpoint specific to the Amazon Web
Services account
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("5. Return a unique endpoint specific to the Amazon
Web Services account.");
            Console.WriteLine();
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }

            var endpoint = await iotWrapper.DescribeEndpointAsync();
            if (endpoint != null)
            {
                var subdomain = endpoint.Split('.')[0];
                Console.WriteLine($"Extracted subdomain: {subdomain}");
                Console.WriteLine($"Full Endpoint URL: https://{endpoint}");
```

```
            }
            else
            {
                Console.WriteLine("Failed to retrieve endpoint.");
            }
            Console.WriteLine(new string('-', 80));

            // Step 6: List your AWS IoT certificates
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("6. List your AWS IoT certificates");
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }

            var certificates = await iotWrapper.ListCertificatesAsync();
            foreach (var cert in certificates.Take(5)) // Show first 5
certificates
            {
                Console.WriteLine($"Cert id: {cert.CertificateId}");
                Console.WriteLine($"Cert Arn: {cert.CertificateArn}");
            }
            Console.WriteLine();
            Console.WriteLine(new string('-', 80));

            // Step 7: Create an IoT shadow
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("7. Update an IoT shadow that refers to a digital
representation or virtual twin of a physical IoT device");
            Console.WriteLine();
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }

            var shadowPayload = JsonSerializer.Serialize(new
            {
                state = new
                {
                    desired = new
                    {
                        temperature = 25,
```

```
                    humidity = 50
                }
            }
        });

        await iotWrapper.UpdateThingShadowAsync(thingName, shadowPayload);
        Console.WriteLine("Thing Shadow updated successfully.");
        Console.WriteLine(new string('-', 80));

        // Step 8: Write out the state information, in JSON format
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("8. Write out the state information, in JSON
format.");
        if (IsInteractive)
        {
            Console.WriteLine("Press Enter to continue...");
            Console.ReadLine();
        }

        var shadowData = await iotWrapper.GetThingShadowAsync(thingName);
        Console.WriteLine($"Received Shadow Data: {shadowData}");
        Console.WriteLine(new string('-', 80));

        // Step 9: Set up resources (SNS topic and IAM role) and create a
rule
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("9. Set up resources and create a rule");
        Console.WriteLine();

        // Deploy CloudFormation stack to create SNS topic and IAM role
        Console.WriteLine("Deploying CloudFormation stack to create SNS topic
and IAM role...");

        var deployStack = !IsInteractive || GetYesNoResponse("Would you like
to deploy the CloudFormation stack? (y/n) ");
        if (deployStack)
        {
            if (IsInteractive)
            {
                Console.Write(
                    $"Enter stack resource file path (or press Enter for
default '{_stackResourcePath}'): ");
                var userResourcePath = Console.ReadLine();
                if (!string.IsNullOrEmpty(userResourcePath))
```

```
                    _stackResourcePath = userResourcePath;
                }

                _stackName = PromptUserForStackName();

                var deploySuccess = await DeployCloudFormationStack(_stackName,
cloudFormationClient, scenarioLogger);

                if (deploySuccess)
                {
                    // Get stack outputs
                    var stackOutputs = await GetStackOutputs(_stackName,
cloudFormationClient, scenarioLogger);
                    if (stackOutputs != null)
                    {
                        snsTopicArn = stackOutputs["SNSTopicArn"];
                        string roleArn = stackOutputs["RoleArn"];

                        Console.WriteLine($"Successfully deployed stack. SNS
topic: {snsTopicArn}");
                        Console.WriteLine($"Successfully deployed stack. IAM
role: {roleArn}");

                        if (IsInteractive)
                        {
                            Console.Write($"Enter Rule name (press Enter for
default '{ruleName}'): ");
                            var userRuleName = Console.ReadLine();
                            if (!string.IsNullOrEmpty(userRuleName))
                                ruleName = userRuleName;
                        }
                        else
                        {
                            Console.WriteLine($"Using default rule name:
{ruleName}");
                        }

                        // Now create the IoT rule with the CloudFormation
outputs
                        var ruleResult = await
iotWrapper.CreateTopicRuleAsync(ruleName, snsTopicArn, roleArn);
                        if (ruleResult)
                        {
                            Console.WriteLine("IoT Rule created successfully.");
```

```
                        }
                        else
                        {
                            Console.WriteLine("Failed to create IoT rule.");
                        }
                    }
                    else
                    {
                        Console.WriteLine("Failed to get stack outputs. Skipping
rule creation.");
                    }
                }
                else
                {
                    Console.WriteLine("Failed to deploy CloudFormation stack.
Skipping rule creation.");
                }
            }
            else
            {
                Console.WriteLine("Skipping CloudFormation stack deployment and
rule creation.");
            }
            Console.WriteLine(new string('-', 80));

            // Step 10: List your rules
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("10. List your rules.");
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }

            var rules = await iotWrapper.ListTopicRulesAsync();
            Console.WriteLine("List of IoT Rules:");
            foreach (var rule in rules.Take(5)) // Show first 5 rules
            {
                Console.WriteLine($"Rule Name: {rule.RuleName}");
                Console.WriteLine($"Rule ARN: {rule.RuleArn}");
                Console.WriteLine("--------------");
            }
            Console.WriteLine();
            Console.WriteLine(new string('-', 80));
```

```
            // Step 11: Search things using the Thing name
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("11. Search things using the Thing name.");
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }

            var searchResults = await iotWrapper.SearchIndexAsync($"thingName:
{thingName}");
            if (searchResults.Any())
            {
                Console.WriteLine($"Thing id found using search is
 {searchResults.First().ThingId}");
            }
            else
            {
                Console.WriteLine($"No search results found for Thing:
 {thingName}");
            }
            Console.WriteLine(new string('-', 80));

            // Step 12: Cleanup - Detach and delete certificate
            if (!string.IsNullOrEmpty(certificateArn))
            {
                Console.WriteLine(new string('-', 80));
                var deleteCert = "y";
                if (IsInteractive)
                {
                    Console.Write($"Do you want to detach and delete the
 certificate for {thingName}? (y/n)");
                    deleteCert = Console.ReadLine();
                }
                else
                {
                    Console.WriteLine($"Detaching and deleting certificate for
 {thingName}...");
                }

                if (deleteCert?.ToLower() == "y")
                {
```

```
                    Console.WriteLine("12. You selected to detach and delete the
certificate.");
                    if (IsInteractive)
                    {
                        Console.WriteLine("Press Enter to continue...");
                        Console.ReadLine();
                    }

                    await iotWrapper.DetachThingPrincipalAsync(thingName,
certificateArn);
                    Console.WriteLine($"{certificateArn} was successfully removed
from {thingName}");

                    await iotWrapper.DeleteCertificateAsync(certificateId);
                    Console.WriteLine($"{certificateArn} was successfully
deleted.");
                }
                Console.WriteLine(new string('-', 80));
            }

            // Step 13: Delete the AWS IoT Thing
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("13. Delete the AWS IoT Thing.");
            var deleteThing = "y";
            if (IsInteractive)
            {
                Console.Write($"Do you want to delete the IoT Thing? (y/n)");
                deleteThing = Console.ReadLine();
            }
            else
            {
                Console.WriteLine($"Deleting IoT Thing {thingName}...");
            }

            if (deleteThing?.ToLower() == "y")
            {
                await iotWrapper.DeleteThingAsync(thingName);
                Console.WriteLine($"Deleted Thing {thingName}");
            }
            Console.WriteLine(new string('-', 80));

            // Step 14: Clean up CloudFormation stack
            if (!string.IsNullOrEmpty(snsTopicArn))
            {
```

```
                Console.WriteLine(new string('-', 80));
                Console.WriteLine("14. Clean up CloudFormation stack.");
                Console.WriteLine("Deleting the CloudFormation stack and all
 resources...");

                var cleanup = !IsInteractive || GetYesNoResponse("Do you want to
 delete the CloudFormation stack and all resources? (y/n) ");
                if (cleanup)
                {
                    var ruleCleanupSuccess = await
 iotWrapper.DeleteTopicRuleAsync(ruleName);

                    var stackCleanupSuccess = await
 DeleteCloudFormationStack(_stackName, cloudFormationClient, scenarioLogger);
                    if (ruleCleanupSuccess && stackCleanupSuccess)
                    {
                        Console.WriteLine("Successfully cleaned up CloudFormation
 stack and all resources.");
                    }
                    else
                    {
                        Console.WriteLine("Some cleanup operations failed. Check
 the logs for details.");
                    }
                }
                else
                {
                    Console.WriteLine($"Resources will remain. Stack name:
 {_stackName}");
                }
                Console.WriteLine(new string('-', 80));
            }
        }
        catch (Exception ex)
        {
            scenarioLogger.LogError(ex, "Error occurred during scenario
 execution.");

            // Cleanup on error
            if (!string.IsNullOrEmpty(certificateArn) && !
string.IsNullOrEmpty(thingName))
            {
                try
                {
```

```
                    await iotWrapper.DetachThingPrincipalAsync(thingName,
certificateArn);
                    await iotWrapper.DeleteCertificateAsync(certificateId);
                }
                catch (Exception cleanupEx)
                {
                    scenarioLogger.LogError(cleanupEx, "Error during cleanup.");
                }
            }

            if (!string.IsNullOrEmpty(thingName))
            {
                try
                {
                    await iotWrapper.DeleteThingAsync(thingName);
                }
                catch (Exception cleanupEx)
                {
                    scenarioLogger.LogError(cleanupEx, "Error during Thing
cleanup.");
                }
            }

            // Clean up CloudFormation stack on error
            if (!string.IsNullOrEmpty(snsTopicArn))
            {
                try
                {
                    await _iotWrapper.DeleteTopicRuleAsync(ruleName);
                    await DeleteCloudFormationStack(_stackName,
cloudFormationClient, scenarioLogger);
                }
                catch (Exception cleanupEx)
                {
                    scenarioLogger.LogError(cleanupEx, "Error during
CloudFormation stack cleanup.");
                }
            }

            throw;
        }
    }

    /// <summary>
```

```
    /// Deploys the CloudFormation stack with the necessary resources.
    /// </summary>
    /// <param name="stackName">The name of the CloudFormation stack.</param>
    /// <param name="cloudFormationClient">The CloudFormation client.</param>
    /// <param name="scenarioLogger">The logger.</param>
    /// <returns>True if the stack was deployed successfully.</returns>
    private static async Task<bool> DeployCloudFormationStack(string stackName,
IAmazonCloudFormation cloudFormationClient, ILogger<IoTBasics> scenarioLogger)
    {
        Console.WriteLine($"\nDeploying CloudFormation stack: {stackName}");

        try
        {
            var request = new CreateStackRequest
            {
                StackName = stackName,
                TemplateBody = await File.ReadAllTextAsync(_stackResourcePath),
                Capabilities = new List<string>
{ Capability.CAPABILITY_NAMED_IAM }
            };

            var response = await cloudFormationClient.CreateStackAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"CloudFormation stack creation started:
{stackName}");

                bool stackCreated = await
WaitForStackCompletion(response.StackId, cloudFormationClient, scenarioLogger);

                if (stackCreated)
                {
                    Console.WriteLine("CloudFormation stack created
successfully.");
                    return true;
                }
                else
                {
                    scenarioLogger.LogError($"CloudFormation stack creation
failed: {stackName}");
                    return false;
                }
            }
```

```
            else
            {
                scenarioLogger.LogError($"Failed to create CloudFormation stack:
{stackName}");
                return false;
            }
        }
        catch (AlreadyExistsException)
        {
            scenarioLogger.LogWarning($"CloudFormation stack '{stackName}'
already exists. Please provide a unique name.");
            var newStackName = PromptUserForStackName();
            return await DeployCloudFormationStack(newStackName,
cloudFormationClient, scenarioLogger);
        }
        catch (Exception ex)
        {
            scenarioLogger.LogError(ex, $"An error occurred while deploying the
CloudFormation stack: {stackName}");
            return false;
        }
    }

    /// <summary>
    /// Waits for the CloudFormation stack to be in the CREATE_COMPLETE state.
    /// </summary>
    /// <param name="stackId">The ID of the CloudFormation stack.</param>
    /// <param name="cloudFormationClient">The CloudFormation client.</param>
    /// <param name="scenarioLogger">The logger.</param>
    /// <returns>True if the stack was created successfully.</returns>
    private static async Task<bool> WaitForStackCompletion(string stackId,
IAmazonCloudFormation cloudFormationClient, ILogger<IoTBasics> scenarioLogger)
    {
        int retryCount = 0;
        const int maxRetries = 30;
        const int retryDelay = 10000;

        while (retryCount < maxRetries)
        {
            var describeStacksRequest = new DescribeStacksRequest
            {
                StackName = stackId
            };
```

```
            var describeStacksResponse = await
cloudFormationClient.DescribeStacksAsync(describeStacksRequest);

            if (describeStacksResponse.Stacks.Count > 0)
            {
                if (describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.CREATE_COMPLETE)
                {
                    return true;
                }
                if (describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.CREATE_FAILED ||
                    describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.ROLLBACK_COMPLETE)
                {
                    return false;
                }
            }

            Console.WriteLine("Waiting for CloudFormation stack creation to
complete...");
            await Task.Delay(retryDelay);
            retryCount++;
        }

        scenarioLogger.LogError("Timed out waiting for CloudFormation stack
creation to complete.");
        return false;
    }

    /// <summary>
    /// Gets the outputs from the CloudFormation stack.
    /// </summary>
    /// <param name="stackName">The name of the CloudFormation stack.</param>
    /// <param name="cloudFormationClient">The CloudFormation client.</param>
    /// <param name="scenarioLogger">The logger.</param>
    /// <returns>A dictionary of stack outputs.</returns>
    private static async Task<Dictionary<string, string>?> GetStackOutputs(string
stackName, IAmazonCloudFormation cloudFormationClient, ILogger<IoTBasics>
scenarioLogger)
    {
        try
        {
            var describeStacksRequest = new DescribeStacksRequest
```

```
            {
                StackName = stackName
            };

            var response = await
cloudFormationClient.DescribeStacksAsync(describeStacksRequest);

            if (response.Stacks.Count > 0)
            {
                var outputs = new Dictionary<string, string>();
                foreach (var output in response.Stacks[0].Outputs)
                {
                    outputs[output.OutputKey] = output.OutputValue;
                }
                return outputs;
            }

            return null;
        }
        catch (Exception ex)
        {
            scenarioLogger.LogError(ex, $"Failed to get stack outputs for
{stackName}");
            return null;
        }
    }

    /// <summary>
    /// Deletes the CloudFormation stack and waits for confirmation.
    /// </summary>
    /// <param name="stackName">The name of the CloudFormation stack.</param>
    /// <param name="cloudFormationClient">The CloudFormation client.</param>
    /// <param name="scenarioLogger">The logger.</param>
    /// <returns>True if the stack was deleted successfully.</returns>
    private static async Task<bool> DeleteCloudFormationStack(string stackName,
IAmazonCloudFormation cloudFormationClient, ILogger<IoTBasics> scenarioLogger)
    {
        try
        {
            var request = new DeleteStackRequest
            {
                StackName = stackName
            };
```

```
            await cloudFormationClient.DeleteStackAsync(request);
            Console.WriteLine($"CloudFormation stack '{stackName}' is being
deleted. This may take a few minutes.");

            bool stackDeleted = await WaitForStackDeletion(stackName,
cloudFormationClient, scenarioLogger);

            if (stackDeleted)
            {
                Console.WriteLine($"CloudFormation stack '{stackName}' has been
deleted.");
                return true;
            }
            else
            {
                scenarioLogger.LogError($"Failed to delete CloudFormation stack
'{stackName}'.");
                return false;
            }
        }
        catch (Exception ex)
        {
            scenarioLogger.LogError(ex, $"An error occurred while deleting the
CloudFormation stack: {stackName}");
            return false;
        }
    }

    /// <summary>
    /// Waits for the stack to be deleted.
    /// </summary>
    /// <param name="stackName">The name of the CloudFormation stack.</param>
    /// <param name="cloudFormationClient">The CloudFormation client.</param>
    /// <param name="scenarioLogger">The logger.</param>
    /// <returns>True if the stack was deleted successfully.</returns>
    private static async Task<bool> WaitForStackDeletion(string stackName,
IAmazonCloudFormation cloudFormationClient, ILogger<IoTBasics> scenarioLogger)
    {
        int retryCount = 0;
        const int maxRetries = 30;
        const int retryDelay = 10000;

        while (retryCount < maxRetries)
        {
```

```
            var describeStacksRequest = new DescribeStacksRequest
            {
                StackName = stackName
            };

            try
            {
                var describeStacksResponse = await
cloudFormationClient.DescribeStacksAsync(describeStacksRequest);

                if (describeStacksResponse.Stacks.Count == 0 ||
                    describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.DELETE_COMPLETE)
                {
                    return true;
                }
            }
            catch (AmazonCloudFormationException ex) when (ex.ErrorCode ==
"ValidationError")
            {
                return true;
            }

            Console.WriteLine($"Waiting for CloudFormation stack '{stackName}' to
be deleted...");
            await Task.Delay(retryDelay);
            retryCount++;
        }

        scenarioLogger.LogError($"Timed out waiting for CloudFormation stack
'{stackName}' to be deleted.");
        return false;
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
    private static bool GetYesNoResponse(string question)
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null && ynResponse.Equals("y",
StringComparison.InvariantCultureIgnoreCase);
        return response;
```

```
    }

    /// <summary>
    /// Prompts the user for a stack name.
    /// </summary>
    private static string PromptUserForStackName()
    {
        if (IsInteractive)
        {
            Console.Write($"Enter a name for the CloudFormation stack (press
 Enter for default '{_stackName}'): ");
            string? input = Console.ReadLine();
            if (!string.IsNullOrWhiteSpace(input))
            {
                var regex = new System.Text.RegularExpressions.Regex("[a-zA-Z][-
a-zA-Z0-9]*");
                if (!regex.IsMatch(input))
                {
                    Console.WriteLine($"Invalid stack name. Using default:
 {_stackName}");
                    return _stackName;
                }
                return input;
            }
        }
        return _stackName;
    }
}
```

Una clase contenedora para los métodos AWS IoT del SDK.

```
/// <summary>
/// Wrapper methods to use Amazon IoT Core with .NET.
/// </summary>
public class IoTWrapper
{
    private readonly IAmazonIoT _amazonIoT;
    private readonly IAmazonIotData _amazonIotData;
    private readonly ILogger<IoTWrapper> _logger;

    /// <summary>
    /// Constructor for the IoT wrapper.
```

```
    /// </summary>
    /// <param name="amazonIoT">The injected IoT client.</param>
    /// <param name="amazonIotData">The injected IoT Data client.</param>
    /// <param name="logger">The injected logger.</param>
    public IoTWrapper(IAmazonIoT amazonIoT, IAmazonIotData amazonIotData,
 ILogger<IoTWrapper> logger)
    {
        _amazonIoT = amazonIoT;
        _amazonIotData = amazonIotData;
        _logger = logger;
    }

    /// <summary>
    /// Creates an AWS IoT Thing.
    /// </summary>
    /// <param name="thingName">The name of the Thing to create.</param>
    /// <returns>The ARN of the Thing created, or null if creation failed.</
returns>
    public async Task<string?> CreateThingAsync(string thingName)
    {
        try
        {
            var request = new CreateThingRequest
            {
                ThingName = thingName
            };

            var response = await _amazonIoT.CreateThingAsync(request);
            _logger.LogInformation($"Created Thing {thingName} with ARN
 {response.ThingArn}");
            return response.ThingArn;
        }
        catch (Amazon.IoT.Model.ResourceAlreadyExistsException ex)
        {
            _logger.LogWarning($"Thing {thingName} already exists:
 {ex.Message}");
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't create Thing {thingName}. Here's why:
 {ex.Message}");
            return null;
        }
```

```
    }

    /// <summary>
    /// Creates a device certificate for AWS IoT.
    /// </summary>
    /// <returns>The certificate details including ARN and certificate PEM, or
 null if creation failed.</returns>
    public async Task<(string CertificateArn, string CertificatePem, string
 CertificateId)?> CreateKeysAndCertificateAsync()
    {
        try
        {
            var request = new CreateKeysAndCertificateRequest
            {
                SetAsActive = true
            };

            var response = await
 _amazonIoT.CreateKeysAndCertificateAsync(request);
            _logger.LogInformation($"Created certificate with ARN
 {response.CertificateArn}");
            return (response.CertificateArn, response.CertificatePem,
 response.CertificateId);
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
 {ex.Message}");
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't create certificate. Here's why:
 {ex.Message}");
            return null;
        }
    }

    /// <summary>
    /// Attaches a certificate to an IoT Thing.
    /// </summary>
    /// <param name="thingName">The name of the Thing.</param>
    /// <param name="certificateArn">The ARN of the certificate to attach.</
param>
```

```
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> AttachThingPrincipalAsync(string thingName, string
certificateArn)
    {
        try
        {
            var request = new AttachThingPrincipalRequest
            {
                ThingName = thingName,
                Principal = certificateArn
            };

            await _amazonIoT.AttachThingPrincipalAsync(request);
            _logger.LogInformation($"Attached certificate {certificateArn} to
Thing {thingName}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot attach certificate - resource not found:
{ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't attach certificate to Thing. Here's why:
{ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Updates an IoT Thing with attributes.
    /// </summary>
    /// <param name="thingName">The name of the Thing to update.</param>
    /// <param name="attributes">Dictionary of attributes to add.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> UpdateThingAsync(string thingName, Dictionary<string,
string> attributes)
    {
        try
        {
            var request = new UpdateThingRequest
            {
```

```
                ThingName = thingName,
                AttributePayload = new AttributePayload
                {
                    Attributes = attributes,
                    Merge = true
                }
            };

            await _amazonIoT.UpdateThingAsync(request);
            _logger.LogInformation($"Updated Thing {thingName} with attributes");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot update Thing - resource not found:
{ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't update Thing attributes. Here's why:
{ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Gets the AWS IoT endpoint URL.
    /// </summary>
    /// <returns>The endpoint URL, or null if retrieval failed.</returns>
    public async Task<string?> DescribeEndpointAsync()
    {
        try
        {
            var request = new DescribeEndpointRequest
            {
                EndpointType = "iot:Data-ATS"
            };

            var response = await _amazonIoT.DescribeEndpointAsync(request);
            _logger.LogInformation($"Retrieved endpoint:
{response.EndpointAddress}");
            return response.EndpointAddress;
        }
```

```
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't describe endpoint. Here's why:
{ex.Message}");
            return null;
        }
    }

    /// <summary>
    /// Lists all certificates associated with the account.
    /// </summary>
    /// <returns>List of certificate information, or empty list if listing
failed.</returns>
    public async Task<List<Certificate>> ListCertificatesAsync()
    {
        try
        {
            var request = new ListCertificatesRequest();
            var response = await _amazonIoT.ListCertificatesAsync(request);

            _logger.LogInformation($"Retrieved {response.Certificates.Count}
certificates");
            return response.Certificates;
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
            return new List<Certificate>();
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't list certificates. Here's why:
{ex.Message}");
            return new List<Certificate>();
        }
    }
```

```
    /// <summary>
    /// Updates the Thing's shadow with new state information.
    /// </summary>
    /// <param name="thingName">The name of the Thing.</param>
    /// <param name="shadowPayload">The shadow payload in JSON format.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> UpdateThingShadowAsync(string thingName, string
 shadowPayload)
    {
        try
        {
            var request = new UpdateThingShadowRequest
            {
                ThingName = thingName,
                Payload = new
 MemoryStream(System.Text.Encoding.UTF8.GetBytes(shadowPayload))
            };

            await _amazonIotData.UpdateThingShadowAsync(request);
            _logger.LogInformation($"Updated shadow for Thing {thingName}");
            return true;
        }
        catch (Amazon.IotData.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot update Thing shadow - resource not found:
 {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't update Thing shadow. Here's why:
 {ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Gets the Thing's shadow information.
    /// </summary>
    /// <param name="thingName">The name of the Thing.</param>
    /// <returns>The shadow data as a string, or null if retrieval failed.</
returns>
    public async Task<string?> GetThingShadowAsync(string thingName)
    {
```

```
        try
        {
            var request = new GetThingShadowRequest
            {
                ThingName = thingName
            };

            var response = await _amazonIotData.GetThingShadowAsync(request);
            using var reader = new StreamReader(response.Payload);
            var shadowData = await reader.ReadToEndAsync();

            _logger.LogInformation($"Retrieved shadow for Thing {thingName}");
            return shadowData;
        }
        catch (Amazon.IotData.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot get Thing shadow - resource not found:
 {ex.Message}");
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't get Thing shadow. Here's why:
 {ex.Message}");
            return null;
        }
    }

    /// <summary>
    /// Creates an IoT topic rule.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <param name="snsTopicArn">The ARN of the SNS topic for the action.</
param>
    /// <param name="roleArn">The ARN of the IAM role.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> CreateTopicRuleAsync(string ruleName, string
 snsTopicArn, string roleArn)
    {
        try
        {
            var request = new CreateTopicRuleRequest
            {
                RuleName = ruleName,
```

```
                TopicRulePayload = new TopicRulePayload
                {
                    Sql = "SELECT * FROM 'topic/subtopic'",
                    Description = $"Rule created by .NET example: {ruleName}",
                    Actions = new List<Amazon.IoT.Model.Action>
                    {
                        new Amazon.IoT.Model.Action
                        {
                            Sns = new SnsAction
                            {
                                TargetArn = snsTopicArn,
                                RoleArn = roleArn
                            }
                        }
                    },
                    RuleDisabled = false
                }
            };

            await _amazonIoT.CreateTopicRuleAsync(request);
            _logger.LogInformation($"Created IoT rule {ruleName}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceAlreadyExistsException ex)
        {
            _logger.LogWarning($"Rule {ruleName} already exists: {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't create topic rule. Here's why:
{ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Deletes an IoT topic rule.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> DeleteTopicRuleAsync(string ruleName)
    {
        try
```

```
        {
            var request = new DeleteTopicRuleRequest
            {
                RuleName = ruleName,
            };

            await _amazonIoT.DeleteTopicRuleAsync(request);
            _logger.LogInformation($"Deleted IoT rule {ruleName}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogWarning($"Rule {ruleName} not found: {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't delete topic rule. Here's why:
{ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Lists all IoT topic rules.
    /// </summary>
    /// <returns>List of topic rules, or empty list if listing failed.</returns>
    public async Task<List<TopicRuleListItem>> ListTopicRulesAsync()
    {
        try
        {
            var request = new ListTopicRulesRequest();
            var response = await _amazonIoT.ListTopicRulesAsync(request);

            _logger.LogInformation($"Retrieved {response.Rules.Count} IoT
rules");
            return response.Rules;
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
            return new List<TopicRuleListItem>();
        }
```

```
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't list topic rules. Here's why:
{ex.Message}");
            return new List<TopicRuleListItem>();
        }
    }

    /// <summary>
    /// Searches for IoT Things using the search index.
    /// </summary>
    /// <param name="queryString">The search query string.</param>
    /// <returns>List of Things that match the search criteria, or empty list if
search failed.</returns>
    public async Task<List<ThingDocument>> SearchIndexAsync(string queryString)
    {
        try
        {
            // First, try to perform the search
            var request = new SearchIndexRequest
            {
                QueryString = queryString
            };

            var response = await _amazonIoT.SearchIndexAsync(request);
            _logger.LogInformation($"Search found {response.Things.Count}
Things");
            return response.Things;
        }
        catch (Amazon.IoT.Model.IndexNotReadyException ex)
        {
            _logger.LogWarning($"Search index not ready, setting up indexing
configuration: {ex.Message}");
            return await SetupIndexAndRetrySearchAsync(queryString);
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex) when
(ex.Message.Contains("index") || ex.Message.Contains("Index"))
        {
            _logger.LogWarning($"Search index not configured, setting up indexing
configuration: {ex.Message}");
            return await SetupIndexAndRetrySearchAsync(queryString);
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
```

```
            _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
            return new List<ThingDocument>();
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't search index. Here's why: {ex.Message}");
            return new List<ThingDocument>();
        }
    }

    /// <summary>
    /// Sets up the indexing configuration and retries the search after waiting
for the index to be ready.
    /// </summary>
    /// <param name="queryString">The search query string.</param>
    /// <returns>List of Things that match the search criteria, or empty list if
setup/search failed.</returns>
    private async Task<List<ThingDocument>> SetupIndexAndRetrySearchAsync(string
queryString)
    {
        try
        {
            // Update indexing configuration to REGISTRY mode
            _logger.LogInformation("Setting up IoT search indexing
configuration...");
            await _amazonIoT.UpdateIndexingConfigurationAsync(
                new UpdateIndexingConfigurationRequest()
                {
                    ThingIndexingConfiguration = new ThingIndexingConfiguration()
                    {
                        ThingIndexingMode = ThingIndexingMode.REGISTRY
                    }
                });

            _logger.LogInformation("Indexing configuration updated. Waiting for
index to be ready...");

            // Wait for the index to be set up - this can take some time
            const int maxRetries = 10;
            const int retryDelaySeconds = 10;

            for (int attempt = 1; attempt <= maxRetries; attempt++)
            {
```

```
                try
                {
                    _logger.LogInformation($"Waiting for index to be ready
(attempt {attempt}/{maxRetries})...");
                    await Task.Delay(TimeSpan.FromSeconds(retryDelaySeconds));

                    // Try to get the current indexing configuration to see if
it's ready
                    var configResponse = await
_amazonIoT.GetIndexingConfigurationAsync(new GetIndexingConfigurationRequest());
                    if
(configResponse.ThingIndexingConfiguration?.ThingIndexingMode ==
ThingIndexingMode.REGISTRY)
                    {
                        // Try the search again
                        var request = new SearchIndexRequest
                        {
                            QueryString = queryString
                        };

                        var response = await
_amazonIoT.SearchIndexAsync(request);
                        _logger.LogInformation($"Search found
{response.Things.Count} Things after index setup");
                        return response.Things;
                    }
                }
                catch (Amazon.IoT.Model.IndexNotReadyException)
                {
                    // Index still not ready, continue waiting
                    _logger.LogInformation("Index still not ready, continuing to
wait...");
                    continue;
                }
                catch (Amazon.IoT.Model.InvalidRequestException ex) when
(ex.Message.Contains("index") || ex.Message.Contains("Index"))
                {
                    // Index still not ready, continue waiting
                    _logger.LogInformation("Index still not ready, continuing to
wait...");
                    continue;
                }
            }
```

```
            _logger.LogWarning("Timeout waiting for search index to be ready
 after configuration update");
            return new List<ThingDocument>();
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't set up search index configuration. Here's
 why: {ex.Message}");
            return new List<ThingDocument>();
        }
    }

    /// <summary>
    /// Detaches a certificate from an IoT Thing.
    /// </summary>
    /// <param name="thingName">The name of the Thing.</param>
    /// <param name="certificateArn">The ARN of the certificate to detach.</
param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> DetachThingPrincipalAsync(string thingName, string
 certificateArn)
    {
        try
        {
            var request = new DetachThingPrincipalRequest
            {
                ThingName = thingName,
                Principal = certificateArn
            };

            await _amazonIoT.DetachThingPrincipalAsync(request);
            _logger.LogInformation($"Detached certificate {certificateArn} from
 Thing {thingName}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot detach certificate - resource not found:
 {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
```

```
            _logger.LogError($"Couldn't detach certificate from Thing. Here's
why: {ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Deletes an IoT certificate.
    /// </summary>
    /// <param name="certificateId">The ID of the certificate to delete.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> DeleteCertificateAsync(string certificateId)
    {
        try
        {
            // First, update the certificate to inactive state
            var updateRequest = new UpdateCertificateRequest
            {
                CertificateId = certificateId,
                NewStatus = CertificateStatus.INACTIVE
            };
            await _amazonIoT.UpdateCertificateAsync(updateRequest);

            // Then delete the certificate
            var deleteRequest = new DeleteCertificateRequest
            {
                CertificateId = certificateId
            };

            await _amazonIoT.DeleteCertificateAsync(deleteRequest);
            _logger.LogInformation($"Deleted certificate {certificateId}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot delete certificate - resource not found:
{ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't delete certificate. Here's why:
{ex.Message}");
            return false;
```

```
        }
    }

    /// <summary>
    /// Deletes an IoT Thing.
    /// </summary>
    /// <param name="thingName">The name of the Thing to delete.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> DeleteThingAsync(string thingName)
    {
        try
        {
            var request = new DeleteThingRequest
            {
                ThingName = thingName
            };

            await _amazonIoT.DeleteThingAsync(request);
            _logger.LogInformation($"Deleted Thing {thingName}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot delete Thing - resource not found:
{ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't delete Thing. Here's why: {ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Lists IoT Things with pagination support.
    /// </summary>
    /// <returns>List of Things, or empty list if listing failed.</returns>
    public async Task<List<ThingAttribute>> ListThingsAsync()
    {
        try
        {
            // Use pages of 10.
            var request = new ListThingsRequest()
```

```
            {
                MaxResults = 10
            };
            var response = await _amazonIoT.ListThingsAsync(request);

            // Since there is not a built-in paginator, use the NextMarker to
 paginate.
            bool hasMoreResults = true;

            var things = new List<ThingAttribute>();
            while (hasMoreResults)
            {
                things.AddRange(response.Things);

                // If NextMarker is not null, there are more results. Get the
 next page of results.
                if (!String.IsNullOrEmpty(response.NextMarker))
                {
                    request.Marker = response.NextMarker;
                    response = await _amazonIoT.ListThingsAsync(request);
                }
                else
                    hasMoreResults = false;
            }

            _logger.LogInformation($"Retrieved {things.Count} Things");
            return things;
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
 {ex.Message}");
            return new List<ThingAttribute>();
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't list Things. Here's why: {ex.Message}");
            return new List<ThingAttribute>();
        }
    }

}
```

C++

SDK para C++

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Crea cualquier AWS IoT cosa.

```
    Aws::String thingName = askQuestion("Enter a thing name: ");

    if (!createThing(thingName, clientConfiguration)) {
        std::cerr << "Exiting because createThing failed." << std::endl;
        cleanup("", "", "", "", "", false, clientConfiguration);
        return false;
    }
```

```
//! Create an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::createThing(const Aws::String &thingName,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::CreateThingRequest createThingRequest;
    createThingRequest.SetThingName(thingName);

    Aws::IoT::Model::CreateThingOutcome outcome = iotClient.CreateThing(
            createThingRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to create thing " << thingName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
```

```
    }

    return outcome.IsSuccess();
}
```

Genere y asocie un certificado de dispositivo.

```
    Aws::String certificateARN;
    Aws::String certificateID;
    if (askYesNoQuestion("Would you like to create a certificate for your thing?
(y/n) ")) {
        Aws::String outputFolder;
        if (askYesNoQuestion(
                "Would you like to save the certificate and keys to file? (y/n)
")) {
            outputFolder = std::filesystem::current_path();
            outputFolder += "/device_keys_and_certificates";

            std::filesystem::create_directories(outputFolder);

            std::cout << "The certificate and keys will be saved to the folder: "
                      << outputFolder << std::endl;
        }

        if (!createKeysAndCertificate(outputFolder, certificateARN,
certificateID,
                                      clientConfiguration)) {
            std::cerr << "Exiting because createKeysAndCertificate failed."
                      << std::endl;
            cleanup(thingName, "", "", "", "", false, clientConfiguration);
            return false;
        }

        std::cout << "\nNext, the certificate will be attached to the thing.\n"
                  << std::endl;
        if (!attachThingPrincipal(certificateARN, thingName,
clientConfiguration)) {
            std::cerr << "Exiting because attachThingPrincipal failed." <<
std::endl;
            cleanup(thingName, certificateARN, certificateID, "", "",
                    false,
                    clientConfiguration);
```

```
            return false;
        }
    }
```

```
//! Create keys and certificate for an Aws IoT device.
//! This routine will save certificates and keys to an output folder, if
 provided.
/*!
  \param outputFolder: Location for storing output in files, ignored when string
 is empty.
  \param certificateARNResult: A string to receive the ARN of the created
 certificate.
  \param certificateID: A string to receive the ID of the created certificate.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::createKeysAndCertificate(const Aws::String &outputFolder,
                                           Aws::String &certificateARNResult,
                                           Aws::String &certificateID,
                                           const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient client(clientConfiguration);
    Aws::IoT::Model::CreateKeysAndCertificateRequest
 createKeysAndCertificateRequest;

    Aws::IoT::Model::CreateKeysAndCertificateOutcome outcome =
            client.CreateKeysAndCertificate(createKeysAndCertificateRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created a certificate and keys" << std::endl;
        certificateARNResult = outcome.GetResult().GetCertificateArn();
        certificateID = outcome.GetResult().GetCertificateId();
        std::cout << "Certificate ARN: " << certificateARNResult << ",
 certificate ID: "
                  << certificateID << std::endl;

        if (!outputFolder.empty()) {
            std::cout << "Writing certificate and keys to the folder '" <<
 outputFolder
                      << "'." << std::endl;
            std::cout << "Be sure these files are stored securely." << std::endl;
```

```
            Aws::String certificateFilePath = outputFolder + "/
certificate.pem.crt";
            std::ofstream certificateFile(certificateFilePath);
            if (!certificateFile.is_open()) {
                std::cerr << "Error opening certificate file, '" <<
 certificateFilePath
                          << "'."
                          << std::endl;
                return false;
            }
            certificateFile << outcome.GetResult().GetCertificatePem();
            certificateFile.close();

            const Aws::IoT::Model::KeyPair &keyPair =
 outcome.GetResult().GetKeyPair();

            Aws::String privateKeyFilePath = outputFolder + "/private.pem.key";
            std::ofstream privateKeyFile(privateKeyFilePath);
            if (!privateKeyFile.is_open()) {
                std::cerr << "Error opening private key file, '" <<
 privateKeyFilePath
                          << "'."
                          << std::endl;
                return false;
            }
            privateKeyFile << keyPair.GetPrivateKey();
            privateKeyFile.close();

            Aws::String publicKeyFilePath = outputFolder + "/public.pem.key";
            std::ofstream publicKeyFile(publicKeyFilePath);
            if (!publicKeyFile.is_open()) {
                std::cerr << "Error opening public key file, '" <<
 publicKeyFilePath
                          << "'."
                          << std::endl;
                return false;
            }
            publicKeyFile << keyPair.GetPublicKey();
        }
    }
    else {
        std::cerr << "Error creating keys and certificate: "
                  << outcome.GetError().GetMessage() << std::endl;
    }
```

```
    return outcome.IsSuccess();
}

//! Attach a principal to an AWS IoT thing.
/*!
  \param principal: A principal to attach.
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::attachThingPrincipal(const Aws::String &principal,
                                       const Aws::String &thingName,
                                       const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient client(clientConfiguration);
    Aws::IoT::Model::AttachThingPrincipalRequest request;
    request.SetPrincipal(principal);
    request.SetThingName(thingName);
    Aws::IoT::Model::AttachThingPrincipalOutcome outcome =
 client.AttachThingPrincipal(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully attached principal to thing." << std::endl;
    }
    else {
        std::cerr << "Failed to attach principal to thing." <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

Realiza varias operaciones en la AWS IoT cosa.

```
    if (!updateThing(thingName, { {"location", "Office"}, {"firmwareVersion",
 "v2.0"} }, clientConfiguration)) {
        std::cerr << "Exiting because updateThing failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, "", "", false,
                clientConfiguration);
        return false;
    }
```

```
   printAsterisksLine();

   std::cout << "Now an endpoint will be retrieved for your account.\n" <<
std::endl;
   std::cout << "An IoT Endpoint refers to a specific URL or Uniform Resource
Locator that serves as the entry point\n"
   << "for communication between IoT devices and the AWS IoT service." <<
std::endl;

   askQuestion("Press Enter to continue:", alwaysTrueTest);

   Aws::String endpoint;
   if (!describeEndpoint(endpoint, clientConfiguration)) {
       std::cerr << "Exiting because getEndpoint failed." << std::endl;
       cleanup(thingName, certificateARN, certificateID, "", "", false,
               clientConfiguration);
       return false;
   }
   std::cout <<"Your endpoint is " << endpoint << "." << std::endl;
   printAsterisksLine();

   std::cout << "Now the certificates in your account will be listed." <<
std::endl;
   askQuestion("Press Enter to continue:", alwaysTrueTest);

   if (!listCertificates(clientConfiguration)) {
       std::cerr << "Exiting because listCertificates failed." << std::endl;
       cleanup(thingName, certificateARN, certificateID, "", "", false,
               clientConfiguration);
       return false;
   }

   printAsterisksLine();

   std::cout << "Now the shadow for the thing will be updated.\n" << std::endl;
   std::cout << "A thing shadow refers to a feature that enables you to create a
virtual representation, or \"shadow,\"\n"
   << "of a physical device or thing. The thing shadow allows you to synchronize
and control the state of a device between\n"
   << "the cloud and the device itself. and the AWS IoT service. For example,
you can write and retrieve JSON data from a thing shadow." << std::endl;
   askQuestion("Press Enter to continue:", alwaysTrueTest);
```

```
    if (!updateThingShadow(thingName, R"({"state":{"reported":
{"temperature":25,"humidity":50}}})", clientConfiguration)) {
        std::cerr << "Exiting because updateThingShadow failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, "", "", false,
                clientConfiguration);
        return false;
    }

    printAsterisksLine();

    std::cout << "Now, the state information for the shadow will be retrieved.\n"
 << std::endl;
    askQuestion("Press Enter to continue:", alwaysTrueTest);

    Aws::String shadowState;
    if (!getThingShadow(thingName, shadowState, clientConfiguration)) {
        std::cerr << "Exiting because getThingShadow failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, "", "", false,
                clientConfiguration);
        return false;
    }
    std::cout << "The retrieved shadow state is: " << shadowState << std::endl;

    printAsterisksLine();

    std::cout << "A rule with now be added to to the thing.\n" << std::endl;
    std::cout << "Any user who has permission to create rules will be able to
 access data processed by the rule." << std::endl;
    std::cout << "In this case, the rule will use an Simple Notification Service
(SNS) topic and an IAM rule." << std::endl;
    std::cout << "These resources will be created using a CloudFormation
 template." << std::endl;
    std::cout << "Stack creation may take a few minutes." << std::endl;

    askQuestion("Press Enter to continue: ", alwaysTrueTest);
    Aws::Map<Aws::String, Aws::String> outputs
 =createCloudFormationStack(STACK_NAME,clientConfiguration);
    if (outputs.empty()) {
        std::cerr << "Exiting because createCloudFormationStack failed." <<
 std::endl;
        cleanup(thingName, certificateARN, certificateID, "", "", false,
                clientConfiguration);
        return false;
    }
```

```
    // Retrieve the topic ARN and role ARN from the CloudFormation stack outputs.
    auto topicArnIter = outputs.find(SNS_TOPIC_ARN_OUTPUT);
    auto roleArnIter = outputs.find(ROLE_ARN_OUTPUT);
    if ((topicArnIter == outputs.end()) || (roleArnIter == outputs.end())) {
        std::cerr << "Exiting because output '" << SNS_TOPIC_ARN_OUTPUT <<
        "' or '" << ROLE_ARN_OUTPUT << "'not found in the CloudFormation stack."
 << std::endl;
        cleanup(thingName, certificateARN, certificateID, STACK_NAME, "",
                false,
                clientConfiguration);
        return false;
    }

    Aws::String topicArn = topicArnIter->second;
    Aws::String roleArn = roleArnIter->second;
    Aws::String sqlStatement = "SELECT * FROM '";
    sqlStatement += MQTT_MESSAGE_TOPIC_FILTER;
    sqlStatement += "'";

    printAsterisksLine();

    std::cout << "Now a rule will be created.\n" << std::endl;
    std::cout << "Rules are an administrator-level action. Any user who has
 permission\n"
                 << "to create rules will be able to access data processed by the
 rule." << std::endl;
    std::cout << "In this case, the rule will use an SNS topic" << std::endl;
    std::cout << "and the following SQL statement '" << sqlStatement << "'." <<
 std::endl;
    std::cout << "For more information on IoT SQL, see https://
docs.aws.amazon.com/iot/latest/developerguide/iot-sql-reference.html" <<
 std::endl;
    Aws::String ruleName = askQuestion("Enter a rule name: ");
    if (!createTopicRule(ruleName, topicArn, sqlStatement, roleArn,
 clientConfiguration)) {
        std::cerr << "Exiting because createRule failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, STACK_NAME, "",
                false,
                clientConfiguration);
        return false;
    }

    printAsterisksLine();
```

```
    std::cout << "Now your rules will be listed.\n" << std::endl;
    askQuestion("Press Enter to continue: ", alwaysTrueTest);
    if (!listTopicRules(clientConfiguration)) {
        std::cerr << "Exiting because listRules failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, STACK_NAME, ruleName,
                false,
                clientConfiguration);
        return false;
    }

    printAsterisksLine();
    Aws::String queryString = "thingName:" + thingName;
    std::cout << "Now the AWS IoT fleet index will be queried with the query\n'"
    << queryString << "'.\n" << std::endl;
    std::cout << "For query information, see https://docs.aws.amazon.com/iot/
latest/developerguide/query-syntax.html" << std::endl;

    std::cout << "For this query to work, thing indexing must be enabled in your
 account.\n"
    << "This can be done with the awscli command line by calling 'aws iot update-
indexing-configuration'\n"
       << "or it can be done programmatically." << std::endl;
    std::cout << "For more information, see https://docs.aws.amazon.com/iot/
latest/developerguide/managing-index.html" << std::endl;
    if (askYesNoQuestion("Do you want to enable thing indexing in your account?
 (y/n) "))
    {
        Aws::IoT::Model::ThingIndexingConfiguration thingIndexingConfiguration;

 thingIndexingConfiguration.SetThingIndexingMode(Aws::IoT::Model::ThingIndexingMode::REGI

 thingIndexingConfiguration.SetThingConnectivityIndexingMode(Aws::IoT::Model::ThingConnec
        // The ThingGroupIndexingConfiguration object is ignored if not set.
        Aws::IoT::Model::ThingGroupIndexingConfiguration
 thingGroupIndexingConfiguration;
        if (!updateIndexingConfiguration(thingIndexingConfiguration,
 thingGroupIndexingConfiguration, clientConfiguration)) {
            std::cerr << "Exiting because updateIndexingConfiguration failed." <<
 std::endl;
            cleanup(thingName, certificateARN, certificateID, STACK_NAME,
                    ruleName, false,
                    clientConfiguration);
            return false;
```

```
        }
    }

    if (!searchIndex(queryString, clientConfiguration)) {

        std::cerr << "Exiting because searchIndex failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, STACK_NAME, ruleName,
                false,
                clientConfiguration);
        return false;
    }
```

```
//! Update an AWS IoT thing with attributes.
/*!
  \param thingName: The name for the thing.
  \param attributeMap: A map of key/value attributes/
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::updateThing(const Aws::String &thingName,
                              const std::map<Aws::String, Aws::String>
 &attributeMap,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::UpdateThingRequest request;
    request.SetThingName(thingName);
    Aws::IoT::Model::AttributePayload attributePayload;
    for (const auto &attribute: attributeMap) {
        attributePayload.AddAttributes(attribute.first, attribute.second);
    }
    request.SetAttributePayload(attributePayload);

    Aws::IoT::Model::UpdateThingOutcome outcome = iotClient.UpdateThing(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to update thing " << thingName << ":" <<
                  outcome.GetError().GetMessage() << std::endl;
    }
```

```
    return outcome.IsSuccess();
}

//! Describe the endpoint specific to the AWS account making the call.
/*!
  \param endpointResult: String to receive the endpoint result.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::describeEndpoint(Aws::String &endpointResult,
                                   const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::String endpoint;
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DescribeEndpointRequest describeEndpointRequest;
    describeEndpointRequest.SetEndpointType(
            "iot:Data-ATS"); // Recommended endpoint type.

    Aws::IoT::Model::DescribeEndpointOutcome outcome =
 iotClient.DescribeEndpoint(
            describeEndpointRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully described endpoint." << std::endl;
        endpointResult = outcome.GetResult().GetEndpointAddress();
    }
    else {
        std::cerr << "Error describing endpoint" <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}

//! List certificates registered in the AWS account making the call.
/*!
   \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::listCertificates(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::ListCertificatesRequest request;
```

```
    Aws::Vector<Aws::IoT::Model::Certificate> allCertificates;
    Aws::String marker; // Used to paginate results.
    do {
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::IoT::Model::ListCertificatesOutcome outcome =
 iotClient.ListCertificates(
                request);

        if (outcome.IsSuccess()) {
            const Aws::IoT::Model::ListCertificatesResult &result =
 outcome.GetResult();
            marker = result.GetNextMarker();
            allCertificates.insert(allCertificates.end(),
                                   result.GetCertificates().begin(),
                                   result.GetCertificates().end());
        }
        else {
            std::cerr << "Error: " << outcome.GetError().GetMessage() <<
 std::endl;
            return false;
        }
    } while (!marker.empty());

    std::cout << allCertificates.size() << " certificate(s) found." << std::endl;

    for (auto &certificate: allCertificates) {
        std::cout << "Certificate ID: " << certificate.GetCertificateId() <<
 std::endl;
        std::cout << "Certificate ARN: " << certificate.GetCertificateArn()
                  << std::endl;
        std::cout << std::endl;
    }

    return true;
}

//! Update the shadow of an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param document: The state information, in JSON format.
```

```
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::updateThingShadow(const Aws::String &thingName,
                                    const Aws::String &document,
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoTDataPlane::IoTDataPlaneClient
 iotDataPlaneClient(clientConfiguration);
    Aws::IoTDataPlane::Model::UpdateThingShadowRequest updateThingShadowRequest;
    updateThingShadowRequest.SetThingName(thingName);
    std::shared_ptr<std::stringstream> streamBuf =
 std::make_shared<std::stringstream>(
            document);
    updateThingShadowRequest.SetBody(streamBuf);
    Aws::IoTDataPlane::Model::UpdateThingShadowOutcome outcome =
 iotDataPlaneClient.UpdateThingShadow(
            updateThingShadowRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated thing shadow." << std::endl;
    }
    else {
        std::cerr << "Error while updating thing shadow."
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Get the shadow of an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param documentResult: String to receive the state information, in JSON format.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::getThingShadow(const Aws::String &thingName,
                                 Aws::String &documentResult,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoTDataPlane::IoTDataPlaneClient iotClient(clientConfiguration);
    Aws::IoTDataPlane::Model::GetThingShadowRequest request;
    request.SetThingName(thingName);
    auto outcome = iotClient.GetThingShadow(request);
```

```
    if (outcome.IsSuccess()) {
        std::stringstream ss;
        ss << outcome.GetResult().GetPayload().rdbuf();
        documentResult = ss.str();
    }
    else {
        std::cerr << "Error getting thing shadow: " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Create an AWS IoT rule with an SNS topic as the target.
/*!
  \param ruleName: The name for the rule.
  \param snsTopic: The SNS topic ARN for the action.
  \param sql: The SQL statement used to query the topic.
  \param roleARN: The IAM role ARN for the action.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::IoT::createTopicRule(const Aws::String &ruleName,
                             const Aws::String &snsTopicARN, const Aws::String
 &sql,
                             const Aws::String &roleARN,
                             const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::CreateTopicRuleRequest request;
    request.SetRuleName(ruleName);

    Aws::IoT::Model::SnsAction snsAction;
    snsAction.SetTargetArn(snsTopicARN);
    snsAction.SetRoleArn(roleARN);

    Aws::IoT::Model::Action action;
    action.SetSns(snsAction);

    Aws::IoT::Model::TopicRulePayload topicRulePayload;
    topicRulePayload.SetSql(sql);
    topicRulePayload.SetActions({action});
```

```
    request.SetTopicRulePayload(topicRulePayload);
    auto outcome = iotClient.CreateTopicRule(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created topic rule " << ruleName << "." <<
 std::endl;
    }
    else {
        std::cerr << "Error creating topic rule " << ruleName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    return outcome.IsSuccess();
}

//! Lists the AWS IoT topic rules.
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::listTopicRules(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::ListTopicRulesRequest request;

    Aws::Vector<Aws::IoT::Model::TopicRuleListItem> allRules;
    Aws::String nextToken; // Used for pagination.
    do {
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        Aws::IoT::Model::ListTopicRulesOutcome outcome =
 iotClient.ListTopicRules(
                request);

        if (outcome.IsSuccess()) {
            const Aws::IoT::Model::ListTopicRulesResult &result =
 outcome.GetResult();
            allRules.insert(allRules.end(),
                            result.GetRules().cbegin(),
                            result.GetRules().cend());

            nextToken = result.GetNextToken();
        }
```

```
        else {
            std::cerr << "ListTopicRules error: " <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }

    } while (!nextToken.empty());

    std::cout << "ListTopicRules: " << allRules.size() << " rule(s) found."
              << std::endl;
    for (auto &rule: allRules) {
        std::cout << "  Rule name: " << rule.GetRuleName() << ", rule ARN: "
                  << rule.GetRuleArn() << "." << std::endl;
    }

    return true;
}

//! Query the AWS IoT fleet index.
//! For query information, see https://docs.aws.amazon.com/iot/latest/
developerguide/query-syntax.html
/*!
  \param: query: The query string.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::searchIndex(const Aws::String &query,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::SearchIndexRequest request;
    request.SetQueryString(query);

    Aws::Vector<Aws::IoT::Model::ThingDocument> allThingDocuments;
    Aws::String nextToken; // Used for pagination.
    do {
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        Aws::IoT::Model::SearchIndexOutcome outcome =
 iotClient.SearchIndex(request);
```

```
        if (outcome.IsSuccess()) {
            const Aws::IoT::Model::SearchIndexResult &result =
 outcome.GetResult();
            allThingDocuments.insert(allThingDocuments.end(),
                                     result.GetThings().cbegin(),
                                     result.GetThings().cend());
            nextToken = result.GetNextToken();

        }
        else {
            std::cerr << "Error in SearchIndex: " <<
 outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!nextToken.empty());

    std::cout << allThingDocuments.size() << " thing document(s) found." <<
 std::endl;
    for (const auto thingDocument: allThingDocuments) {
        std::cout << "  Thing name: " << thingDocument.GetThingName() << "."
                  << std::endl;
    }
    return true;
}
```

Eliminación de recursos.

```
bool
AwsDoc::IoT::cleanup(const Aws::String &thingName, const Aws::String
 &certificateARN,
                     const Aws::String &certificateID, const Aws::String
 &stackName,
                     const Aws::String &ruleName, bool askForConfirmation,
                     const Aws::Client::ClientConfiguration &clientConfiguration)
 {
    bool result = true;

    if (!ruleName.empty() && (!askForConfirmation ||
                              askYesNoQuestion("Delete the rule '" + ruleName +
                                               "'? (y/n) "))) {
        result &= deleteTopicRule(ruleName, clientConfiguration);
```

```
    }

    Aws::CloudFormation::CloudFormationClient
 cloudFormationClient(clientConfiguration);

    if (!stackName.empty() && (!askForConfirmation ||
                               askYesNoQuestion(
                                       "Delete the CloudFormation stack '" +
 stackName +
                                       "'? (y/n) "))) {
        result &= deleteStack(stackName, clientConfiguration);
    }

    if (!certificateARN.empty() && (!askForConfirmation ||
                                    askYesNoQuestion("Delete the certificate '" +
                                                     certificateARN + "'? (y/n)
 "))) {
        result &= detachThingPrincipal(certificateARN, thingName,
 clientConfiguration);
        result &= deleteCertificate(certificateID, clientConfiguration);
    }

    if (!thingName.empty() && (!askForConfirmation ||
                               askYesNoQuestion("Delete the thing '" + thingName
 +
                                                "'? (y/n) "))) {
        result &= deleteThing(thingName, clientConfiguration);
    }

    return result;
}
```

```
//! Detach a principal from an AWS IoT thing.
/*!
  \param principal: A principal to detach.
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::detachThingPrincipal(const Aws::String &principal,
                                       const Aws::String &thingName,
```

```
                                        const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::DetachThingPrincipalRequest detachThingPrincipalRequest;
    detachThingPrincipalRequest.SetThingName(thingName);
    detachThingPrincipalRequest.SetPrincipal(principal);

    Aws::IoT::Model::DetachThingPrincipalOutcome outcome =
 iotClient.DetachThingPrincipal(
           detachThingPrincipalRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully detached principal " << principal << " from
 thing "
                  << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to detach principal " << principal << " from thing "
                  << thingName << ": "
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Delete a certificate.
/*!
  \param certificateID: The ID of a certificate.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::deleteCertificate(const Aws::String &certificateID,
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::DeleteCertificateRequest request;
    request.SetCertificateId(certificateID);

    Aws::IoT::Model::DeleteCertificateOutcome outcome =
 iotClient.DeleteCertificate(
           request);
```

```
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted certificate " << certificateID <<
 std::endl;
    }
    else {
        std::cerr << "Error deleting certificate " << certificateID << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Delete an AWS IoT rule.
/*!
  \param ruleName: The name for the rule.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::deleteTopicRule(const Aws::String &ruleName,
                                  const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DeleteTopicRuleRequest request;
    request.SetRuleName(ruleName);

    Aws::IoT::Model::DeleteTopicRuleOutcome outcome = iotClient.DeleteTopicRule(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted rule " << ruleName << std::endl;
    }
    else {
        std::cerr << "Failed to delete rule " << ruleName <<
                  ": " << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Delete an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
```

```
bool AwsDoc::IoT::deleteThing(const Aws::String &thingName,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DeleteThingRequest request;
    request.SetThingName(thingName);
    const auto outcome = iotClient.DeleteThing(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Error deleting thing " << thingName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

Java

SDK para Java 2.x

> **Note**
>
> Hay más en marcha GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Ejecute un escenario interactivo que demuestre AWS IoT las funciones.

```
import java.util.Scanner;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
```

```
 * This Java example performs these tasks:
 *
 * 1. Creates an AWS IoT Thing.
 * 2. Generate and attach a device certificate.
 * 3. Update an AWS IoT Thing with Attributes.
 * 4. Get an AWS IoT Endpoint.
 * 5. List your certificates.
 * 6. Updates the shadow for the specified thing..
 * 7. Write out the state information, in JSON format
 * 8. Creates a rule
 * 9. List rules
 * 10. Search things
 * 11. Detach amd delete the certificate.
 * 12. Delete Thing.
 */
public class IotScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) {
        final String usage =
            """
                Usage:
                    <roleARN> <snsAction>

                Where:
                    roleARN - The ARN of an IAM role that has permission to work
 with AWS IOT.
                    snsAction  - An ARN of an SNS topic.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        IotActions iotActions = new IotActions();
        String thingName;
        String ruleName;
        String roleARN = args[0];
        String snsAction = args[1];
        Scanner scanner = new Scanner(System.in);

        System.out.println(DASHES);
```

```
        System.out.println("Welcome to the AWS IoT basics scenario.");
        System.out.println("""
            This example program demonstrates various interactions with the AWS
Internet of Things (IoT) Core service. The program guides you through a series
of steps,
            including creating an IoT Thing, generating a device certificate,
updating the Thing with attributes, and so on.
            It utilizes the AWS SDK for Java V2 and incorporates functionality
for creating and managing IoT Things, certificates, rules,
            shadows, and performing searches. The program aims to showcase AWS
IoT capabilities and provides a comprehensive example for
            developers working with AWS IoT in a Java environment.

            Let's get started...

            """);
        System.out.println(DASHES);

        System.out.println("1. Create an AWS IoT Thing.");
        System.out.println("""
            An AWS IoT Thing represents a virtual entity in the AWS IoT service
that can be associated with
            a physical device.
            """);
        // Prompt the user for input.
        System.out.print("Enter Thing name: ");
        thingName = scanner.nextLine();
        iotActions.createIoTThing(thingName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Generate a device certificate.");
        System.out.println("""
            A device certificate performs a role in securing the communication
between devices (Things)
            and the AWS IoT platform.
            """);

        System.out.print("Do you want to create a certificate for " +thingName
+"? (y/n)");
        String certAns = scanner.nextLine();
        String certificateArn="" ;
        if (certAns != null && certAns.trim().equalsIgnoreCase("y")) {
            certificateArn = iotActions.createCertificate();
```

```
            System.out.println("Attach the certificate to the AWS IoT Thing.");
            iotActions.attachCertificateToThing(thingName, certificateArn);
        } else {
            System.out.println("A device certificate was not created.");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Update an AWS IoT Thing with Attributes.");
        System.out.println("""
             IoT Thing attributes, represented as key-value pairs, offer a
pivotal advantage in facilitating efficient data
             management and retrieval within the AWS IoT ecosystem.
            """);
        waitForInputToContinue(scanner);
        iotActions.updateShadowThing(thingName);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Return a unique endpoint specific to the Amazon
Web Services account.");
        System.out.println("""
            An IoT Endpoint refers to a specific URL or Uniform Resource Locator
that serves as the entry point for communication between IoT devices and the AWS
IoT service.
           """);
        waitForInputToContinue(scanner);
        String endpointUrl = iotActions.describeEndpoint();
        System.out.println("The endpoint is "+endpointUrl);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. List your AWS IoT certificates");
        waitForInputToContinue(scanner);
        if (certificateArn.length() > 0) {
            iotActions.listCertificates();
        } else {
            System.out.println("You did not create a certificates. Skipping this
step.");
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("6. Create an IoT shadow that refers to a digital
representation or virtual twin of a physical IoT device");
        System.out.println("""
            A Thing Shadow refers to a feature that enables you to create a
virtual representation, or "shadow,"
            of a physical device or thing. The Thing Shadow allows you to
synchronize and control the state of a device between
            the cloud and the device itself. and the AWS IoT service. For
example, you can write and retrieve JSON data from a Thing Shadow.
           """);
        waitForInputToContinue(scanner);
        iotActions.updateShadowThing(thingName);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Write out the state information, in JSON
format.");
        waitForInputToContinue(scanner);
        iotActions.getPayload(thingName);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Creates a rule");
        System.out.println("""
        Creates a rule that is an administrator-level action.
        Any user who has permission to create rules will be able to access data
processed by the rule.
        """);
        System.out.print("Enter Rule name: ");
        ruleName = scanner.nextLine();
        iotActions.createIoTRule(roleARN, ruleName, snsAction);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. List your rules.");
        waitForInputToContinue(scanner);
        iotActions.listIoTRules();
        waitForInputToContinue(scanner);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("10. Search things using the Thing name.");
        waitForInputToContinue(scanner);
        String queryString = "thingName:"+thingName ;
        iotActions.searchThings(queryString);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        if (certificateArn.length() > 0) {
            System.out.print("Do you want to detach and delete the certificate
for " +thingName +"? (y/n)");
            String delAns = scanner.nextLine();
            if (delAns != null && delAns.trim().equalsIgnoreCase("y")) {
                System.out.println("11. You selected to detach amd delete the
certificate.");
                waitForInputToContinue(scanner);
                iotActions.detachThingPrincipal(thingName, certificateArn);
                iotActions.deleteCertificate(certificateArn);
                waitForInputToContinue(scanner);
            } else {
                System.out.println("11. You selected not to delete the
certificate.");
            }
        } else {
            System.out.println("11. You did not create a certificate so there is
nothing to delete.");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Delete the AWS IoT Thing.");
        System.out.print("Do you want to delete the IoT Thing? (y/n)");
        String delAns = scanner.nextLine();
        if (delAns != null && delAns.trim().equalsIgnoreCase("y")) {
            iotActions.deleteIoTThing(thingName);
        } else {
            System.out.println("The IoT Thing was not deleted.");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The AWS IoT workflow has successfully completed.");
```

```
        System.out.println(DASHES);
    }


    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            System.out.println("");
            System.out.println("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                System.out.println("Continuing with the program...");
                System.out.println("");
                break;
            } else {
                // Handle invalid input.
                System.out.println("Invalid input. Please try again.");
            }
        }
    }
}
```

Una clase contenedora para los métodos AWS IoT del SDK.

```
import
 software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.core.retry.RetryPolicy;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iot.IotAsyncClient;
import software.amazon.awssdk.services.iot.model.Action;
import software.amazon.awssdk.services.iot.model.AttachThingPrincipalRequest;
import software.amazon.awssdk.services.iot.model.AttachThingPrincipalResponse;
import software.amazon.awssdk.services.iot.model.Certificate;
import
 software.amazon.awssdk.services.iot.model.CreateKeysAndCertificateResponse;
import software.amazon.awssdk.services.iot.model.CreateThingRequest;
import software.amazon.awssdk.services.iot.model.CreateThingResponse;
import software.amazon.awssdk.services.iot.model.CreateTopicRuleRequest;
```

```
import software.amazon.awssdk.services.iot.model.CreateTopicRuleResponse;
import software.amazon.awssdk.services.iot.model.DeleteCertificateRequest;
import software.amazon.awssdk.services.iot.model.DeleteCertificateResponse;
import software.amazon.awssdk.services.iot.model.DeleteThingRequest;
import software.amazon.awssdk.services.iot.model.DeleteThingResponse;
import software.amazon.awssdk.services.iot.model.DescribeEndpointRequest;
import software.amazon.awssdk.services.iot.model.DescribeEndpointResponse;
import software.amazon.awssdk.services.iot.model.DescribeThingRequest;
import software.amazon.awssdk.services.iot.model.DescribeThingResponse;
import software.amazon.awssdk.services.iot.model.DetachThingPrincipalRequest;
import software.amazon.awssdk.services.iot.model.DetachThingPrincipalResponse;
import software.amazon.awssdk.services.iot.model.IotException;
import software.amazon.awssdk.services.iot.model.ListCertificatesResponse;
import software.amazon.awssdk.services.iot.model.ListTopicRulesRequest;
import software.amazon.awssdk.services.iot.model.ListTopicRulesResponse;
import software.amazon.awssdk.services.iot.model.SearchIndexRequest;
import software.amazon.awssdk.services.iot.model.SearchIndexResponse;
import software.amazon.awssdk.services.iot.model.TopicRuleListItem;
import software.amazon.awssdk.services.iot.model.SnsAction;
import software.amazon.awssdk.services.iot.model.TopicRulePayload;
import software.amazon.awssdk.services.iotdataplane.IotDataPlaneAsyncClient;
import software.amazon.awssdk.services.iotdataplane.model.GetThingShadowRequest;
import software.amazon.awssdk.services.iotdataplane.model.GetThingShadowResponse;
import
 software.amazon.awssdk.services.iotdataplane.model.UpdateThingShadowRequest;
import
 software.amazon.awssdk.services.iotdataplane.model.UpdateThingShadowResponse;
import java.nio.charset.StandardCharsets;
import java.time.Duration;
import java.util.List;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class IotActions {

    private static IotAsyncClient iotAsyncClient;

    private static IotDataPlaneAsyncClient iotAsyncDataPlaneClient;

    private static final String TOPIC = "your-iot-topic";

    private static IotDataPlaneAsyncClient getAsyncDataPlaneClient() {
```

```
        SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
            .maxConcurrency(100)
            .connectionTimeout(Duration.ofSeconds(60))
            .readTimeout(Duration.ofSeconds(60))
            .writeTimeout(Duration.ofSeconds(60))
            .build();

        ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
            .apiCallTimeout(Duration.ofMinutes(2))
            .apiCallAttemptTimeout(Duration.ofSeconds(90))
            .retryPolicy(RetryPolicy.builder()
                .numRetries(3)
                .build())
            .build();

        if (iotAsyncDataPlaneClient == null) {
            iotAsyncDataPlaneClient = IotDataPlaneAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return iotAsyncDataPlaneClient;
    }


    private static IotAsyncClient getAsyncClient() {
        SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
            .maxConcurrency(100)
            .connectionTimeout(Duration.ofSeconds(60))
            .readTimeout(Duration.ofSeconds(60))
            .writeTimeout(Duration.ofSeconds(60))
            .build();

        ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
            .apiCallTimeout(Duration.ofMinutes(2))
            .apiCallAttemptTimeout(Duration.ofSeconds(90))
            .retryPolicy(RetryPolicy.builder()
                .numRetries(3)
                .build())
            .build();
```

```
        if (iotAsyncClient == null) {
            iotAsyncClient = IotAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return iotAsyncClient;
    }

    /**
     * Creates an IoT certificate asynchronously.
     *
     * @return The ARN of the created certificate.
     * <p>
     * This method initiates an asynchronous request to create an IoT
certificate.
     * If the request is successful, it prints the certificate details and
returns the certificate ARN.
     * If an exception occurs, it prints the error message.
     */
    public String createCertificate() {
        CompletableFuture<CreateKeysAndCertificateResponse> future =
getAsyncClient().createKeysAndCertificate();
        final String[] certificateArn = {null};
        future.whenComplete((response, ex) -> {
            if (response != null) {
                String certificatePem = response.certificatePem();
                certificateArn[0] = response.certificateArn();

                // Print the details.
                System.out.println("\nCertificate:");
                System.out.println(certificatePem);
                System.out.println("\nCertificate ARN:");
                System.out.println(certificateArn[0]);

            } else {
                Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
```

```
                    System.err.println("Unexpected error: " +
cause.getMessage());
                }
            }
        });

        future.join();
        return certificateArn[0];
    }

    /**
     * Creates an IoT Thing with the specified name asynchronously.
     *
     * @param thingName The name of the IoT Thing to create.
     *
     * This method initiates an asynchronous request to create an IoT Thing with
the specified name.
     * If the request is successful, it prints the name of the thing and its ARN
value.
     * If an exception occurs, it prints the error message.
     */
    public void createIoTThing(String thingName) {
        CreateThingRequest createThingRequest = CreateThingRequest.builder()
            .thingName(thingName)
            .build();

        CompletableFuture<CreateThingResponse> future =
getAsyncClient().createThing(createThingRequest);
        future.whenComplete((createThingResponse, ex) -> {
            if (createThingResponse != null) {
                System.out.println(thingName + " was successfully created. The
ARN value is " + createThingResponse.thingArn());
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                }
            }
        });
```

```
        future.join();
    }

    /**
     * Attaches a certificate to an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     * @param certificateArn The ARN of the certificate to attach.
     *
     * This method initiates an asynchronous request to attach a certificate to
an IoT Thing.
     * If the request is successful, it prints a confirmation message and
additional information about the Thing.
     * If an exception occurs, it prints the error message.
     */
    public void attachCertificateToThing(String thingName, String certificateArn)
{
        AttachThingPrincipalRequest principalRequest =
AttachThingPrincipalRequest.builder()
            .thingName(thingName)
            .principal(certificateArn)
            .build();

        CompletableFuture<AttachThingPrincipalResponse> future =
getAsyncClient().attachThingPrincipal(principalRequest);
        future.whenComplete((attachResponse, ex) -> {
            if (attachResponse != null &&
attachResponse.sdkHttpResponse().isSuccessful()) {
                System.out.println("Certificate attached to Thing
successfully.");

                // Print additional information about the Thing.
                describeThing(thingName);
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to attach certificate to Thing.
HTTP Status Code: " +
```

```
                        attachResponse.sdkHttpResponse().statusCode());
                }
            }
        });

        future.join();
    }

    /**
     * Describes an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     *
     * This method initiates an asynchronous request to describe an IoT Thing.
     * If the request is successful, it prints the Thing details.
     * If an exception occurs, it prints the error message.
     */
    private void describeThing(String thingName) {
        DescribeThingRequest thingRequest = DescribeThingRequest.builder()
            .thingName(thingName)
            .build();

        CompletableFuture<DescribeThingResponse> future =
getAsyncClient().describeThing(thingRequest);
        future.whenComplete((describeResponse, ex) -> {
            if (describeResponse != null) {
                System.out.println("Thing Details:");
                System.out.println("Thing Name: " +
describeResponse.thingName());
                System.out.println("Thing ARN: " + describeResponse.thingArn());
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to describe Thing.");
                }
            }
        });
```

```
        future.join();
    }

    /**
     * Updates the shadow of an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     *
     * This method initiates an asynchronous request to update the shadow of an
IoT Thing.
     * If the request is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void updateShadowThing(String thingName) {
        // Create Thing Shadow State Document.
        String stateDocument = "{\"state\":{\"reported\":{\"temperature\":25,
\"humidity\":50}}}";
        SdkBytes data = SdkBytes.fromString(stateDocument,
StandardCharsets.UTF_8);
        UpdateThingShadowRequest updateThingShadowRequest =
UpdateThingShadowRequest.builder()
            .thingName(thingName)
            .payload(data)
            .build();

        CompletableFuture<UpdateThingShadowResponse> future =
getAsyncDataPlaneClient().updateThingShadow(updateThingShadowRequest);
        future.whenComplete((updateResponse, ex) -> {
            if (updateResponse != null) {
                System.out.println("Thing Shadow updated successfully.");
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to update Thing Shadow.");
                }
            }
        });
```

```
        future.join();
    }

    /**
     * Describes the endpoint of the IoT service asynchronously.
     *
     * @return A CompletableFuture containing the full endpoint URL.
     *
     * This method initiates an asynchronous request to describe the endpoint of
 the IoT service.
     * If the request is successful, it prints and returns the full endpoint URL.
     * If an exception occurs, it prints the error message.
     */
    public String describeEndpoint() {
        CompletableFuture<DescribeEndpointResponse> future =
 getAsyncClient().describeEndpoint(DescribeEndpointRequest.builder().endpointType("iot:Da
ATS").build());
        final String[] result = {null};

        future.whenComplete((endpointResponse, ex) -> {
            if (endpointResponse != null) {
                String endpointUrl = endpointResponse.endpointAddress();
                String exString = getValue(endpointUrl);
                String fullEndpoint = "https://" + exString + "-ats.iot.us-
east-1.amazonaws.com";

                System.out.println("Full Endpoint URL: " + fullEndpoint);
                result[0] = fullEndpoint;
            } else {
                Throwable cause = (ex instanceof CompletionException) ?
 ex.getCause() : ex;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
 cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " +
 cause.getMessage());
                }
            }
        });

        future.join();
        return result[0];
    }
```

```
    /**
     * Extracts a specific value from the endpoint URL.
     *
     * @param input The endpoint URL to process.
     * @return The extracted value from the endpoint URL.
     */
    private static String getValue(String input) {
        // Define a regular expression pattern for extracting the subdomain.
        Pattern pattern = Pattern.compile("^(.*?)\\.iot\\.us-east-1\\.amazonaws\
\.com");

        // Match the pattern against the input string.
        Matcher matcher = pattern.matcher(input);

        // Check if a match is found.
        if (matcher.find()) {
            // Extract the subdomain from the first capturing group.
            String subdomain = matcher.group(1);
            System.out.println("Extracted subdomain: " + subdomain);
            return subdomain ;
        } else {
            System.out.println("No match found");
        }
        return "" ;
    }

    /**
     * Lists all certificates asynchronously.
     *
     * This method initiates an asynchronous request to list all certificates.
     * If the request is successful, it prints the certificate IDs and ARNs.
     * If an exception occurs, it prints the error message.
     */
    public void listCertificates() {
        CompletableFuture<ListCertificatesResponse> future =
 getAsyncClient().listCertificates();
        future.whenComplete((response, ex) -> {
            if (response != null) {
                List<Certificate> certList = response.certificates();
                for (Certificate cert : certList) {
                    System.out.println("Cert id: " + cert.certificateId());
                    System.out.println("Cert Arn: " + cert.certificateArn());
                }
```

```
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to list certificates.");
                }
            }
        });

        future.join();
    }

    /**
     * Retrieves the payload of a Thing's shadow asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     *
     * This method initiates an asynchronous request to get the payload of a
Thing's shadow.
     * If the request is successful, it prints the shadow data.
     * If an exception occurs, it prints the error message.
     */
    public void getPayload(String thingName) {
        GetThingShadowRequest getThingShadowRequest =
GetThingShadowRequest.builder()
            .thingName(thingName)
            .build();

        CompletableFuture<GetThingShadowResponse> future =
getAsyncDataPlaneClient().getThingShadow(getThingShadowRequest);
        future.whenComplete((getThingShadowResponse, ex) -> {
            if (getThingShadowResponse != null) {
                // Extracting payload from response.
                SdkBytes payload = getThingShadowResponse.payload();
                String payloadString = payload.asUtf8String();
                System.out.println("Received Shadow Data: " + payloadString);
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
```

```
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to get Thing Shadow payload.");
                }
            }
        });

        future.join();
    }

    /**
     * Creates an IoT rule asynchronously.
     *
     * @param roleARN The ARN of the IAM role that grants access to the rule's
actions.
     * @param ruleName The name of the IoT rule.
     * @param action The ARN of the action to perform when the rule is triggered.
     *
     * This method initiates an asynchronous request to create an IoT rule.
     * If the request is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void createIoTRule(String roleARN, String ruleName, String action) {
        String sql = "SELECT * FROM '" + TOPIC + "'";
        SnsAction action1 = SnsAction.builder()
            .targetArn(action)
            .roleArn(roleARN)
            .build();

        // Create the action.
        Action myAction = Action.builder()
            .sns(action1)
            .build();

        // Create the topic rule payload.
        TopicRulePayload topicRulePayload = TopicRulePayload.builder()
            .sql(sql)
            .actions(myAction)
            .build();
```

```
        // Create the topic rule request.
        CreateTopicRuleRequest topicRuleRequest =
CreateTopicRuleRequest.builder()
            .ruleName(ruleName)
            .topicRulePayload(topicRulePayload)
            .build();

        CompletableFuture<CreateTopicRuleResponse> future =
getAsyncClient().createTopicRule(topicRuleRequest);
        future.whenComplete((response, ex) -> {
            if (response != null) {
                System.out.println("IoT Rule created successfully.");
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to create IoT Rule.");
                }
            }
        });

        future.join();
    }

    /**
     * Lists IoT rules asynchronously.
     *
     * This method initiates an asynchronous request to list IoT rules.
     * If the request is successful, it prints the names and ARNs of the rules.
     * If an exception occurs, it prints the error message.
     */
    public void listIoTRules() {
        ListTopicRulesRequest listTopicRulesRequest =
ListTopicRulesRequest.builder().build();
        CompletableFuture<ListTopicRulesResponse> future =
getAsyncClient().listTopicRules(listTopicRulesRequest);
        future.whenComplete((listTopicRulesResponse, ex) -> {
            if (listTopicRulesResponse != null) {
                System.out.println("List of IoT Rules:");
```

```
                List<TopicRuleListItem> ruleList =
listTopicRulesResponse.rules();
                for (TopicRuleListItem rule : ruleList) {
                    System.out.println("Rule Name: " + rule.ruleName());
                    System.out.println("Rule ARN: " + rule.ruleArn());
                    System.out.println("--------------");
                }
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to list IoT Rules.");
                }
            }
        });

        future.join();
    }

    /**
     * Searches for IoT Things asynchronously based on a query string.
     *
     * @param queryString The query string to search for Things.
     *
     * This method initiates an asynchronous request to search for IoT Things.
     * If the request is successful and Things are found, it prints their IDs.
     * If no Things are found, it prints a message indicating so.
     * If an exception occurs, it prints the error message.
     */
    public void searchThings(String queryString) {
        SearchIndexRequest searchIndexRequest = SearchIndexRequest.builder()
            .queryString(queryString)
            .build();

        CompletableFuture<SearchIndexResponse> future =
getAsyncClient().searchIndex(searchIndexRequest);
        future.whenComplete((searchIndexResponse, ex) -> {
            if (searchIndexResponse != null) {
                // Process the result.
```

```
                if (searchIndexResponse.things().isEmpty()) {
                    System.out.println("No things found.");
                } else {
                    searchIndexResponse.things().forEach(thing ->
System.out.println("Thing id found using search is " + thing.thingId()));
                }
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to search for IoT Things.");
                }
            }
        });

        future.join();
    }

    /**
     * Detaches a principal (certificate) from an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     * @param certificateArn The ARN of the certificate to detach.
     *
     * This method initiates an asynchronous request to detach a certificate from
an IoT Thing.
     * If the detachment is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void detachThingPrincipal(String thingName, String certificateArn) {
        DetachThingPrincipalRequest thingPrincipalRequest =
DetachThingPrincipalRequest.builder()
            .principal(certificateArn)
            .thingName(thingName)
            .build();

        CompletableFuture<DetachThingPrincipalResponse> future =
getAsyncClient().detachThingPrincipal(thingPrincipalRequest);
        future.whenComplete((voidResult, ex) -> {
```

```
            if (ex == null) {
                System.out.println(certificateArn + " was successfully removed
from " + thingName);
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + ex.getMessage());
                }
            }
        });

        future.join();
    }

    /**
     * Deletes a certificate asynchronously.
     *
     * @param certificateArn The ARN of the certificate to delete.
     *
     * This method initiates an asynchronous request to delete a certificate.
     * If the deletion is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void deleteCertificate(String certificateArn) {
        DeleteCertificateRequest certificateProviderRequest =
DeleteCertificateRequest.builder()
            .certificateId(extractCertificateId(certificateArn))
            .build();

        CompletableFuture<DeleteCertificateResponse> future =
getAsyncClient().deleteCertificate(certificateProviderRequest);
        future.whenComplete((voidResult, ex) -> {
            if (ex == null) {
                System.out.println(certificateArn + " was successfully
deleted.");
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
```

```
                    System.err.println("Unexpected error: " + ex.getMessage());
                }
            }
        });

        future.join();
    }

    /**
     * Deletes an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing to delete.
     *
     * This method initiates an asynchronous request to delete an IoT Thing.
     * If the deletion is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void deleteIoTThing(String thingName) {
        DeleteThingRequest deleteThingRequest = DeleteThingRequest.builder()
            .thingName(thingName)
            .build();

        CompletableFuture<DeleteThingResponse> future =
getAsyncClient().deleteThing(deleteThingRequest);
        future.whenComplete((voidResult, ex) -> {
            if (ex == null) {
                System.out.println("Deleted Thing " + thingName);
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + ex.getMessage());
                }
            }
        });

        future.join();
    }

    // Get the cert Id  from the Cert ARN value.
    private String extractCertificateId(String certificateArn) {
        // Example ARN: arn:aws:iot:region:account-id:cert/certificate-id.
```

```
        String[] arnParts = certificateArn.split(":");
        String certificateIdPart = arnParts[arnParts.length - 1];
        return certificateIdPart.substring(certificateIdPart.lastIndexOf("/") +
 1);
    }
}
```

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import aws.sdk.kotlin.services.iot.IotClient
import aws.sdk.kotlin.services.iot.model.Action
import aws.sdk.kotlin.services.iot.model.AttachThingPrincipalRequest
import aws.sdk.kotlin.services.iot.model.AttributePayload
import aws.sdk.kotlin.services.iot.model.CreateThingRequest
import aws.sdk.kotlin.services.iot.model.CreateTopicRuleRequest
import aws.sdk.kotlin.services.iot.model.DeleteCertificateRequest
import aws.sdk.kotlin.services.iot.model.DeleteThingRequest
import aws.sdk.kotlin.services.iot.model.DescribeEndpointRequest
import aws.sdk.kotlin.services.iot.model.DescribeThingRequest
import aws.sdk.kotlin.services.iot.model.DetachThingPrincipalRequest
import aws.sdk.kotlin.services.iot.model.ListTopicRulesRequest
import aws.sdk.kotlin.services.iot.model.SearchIndexRequest
import aws.sdk.kotlin.services.iot.model.SnsAction
import aws.sdk.kotlin.services.iot.model.TopicRulePayload
import aws.sdk.kotlin.services.iot.model.UpdateThingRequest
import aws.sdk.kotlin.services.iotdataplane.IotDataPlaneClient
import aws.sdk.kotlin.services.iotdataplane.model.GetThingShadowRequest
import aws.sdk.kotlin.services.iotdataplane.model.UpdateThingShadowRequest
import aws.smithy.kotlin.runtime.content.ByteStream
import aws.smithy.kotlin.runtime.content.toByteArray
import java.util.Scanner
import java.util.regex.Pattern
```

```
import kotlin.system.exitProcess

/**
 * Before running this Kotlin code example, ensure that your development
 environment
 * is set up, including configuring your credentials.
 *
 * For detailed instructions, refer to the following documentation topic:
 * [Setting Up Your Development Environment](https://docs.aws.amazon.com/sdk-for-
kotlin/latest/developer-guide/setup.html)
 *
 * This code example requires an SNS topic and an IAM Role.
 * Follow the steps in the documentation to set up these resources:
 *
 * - [Creating an SNS Topic](https://docs.aws.amazon.com/sns/latest/dg/sns-
getting-started.html#step-create-topic)
 * - [Creating an IAM Role](https://docs.aws.amazon.com/IAM/latest/UserGuide/
id_roles_create.html)
 */

val DASHES = String(CharArray(80)).replace("\u0000", "-")
val TOPIC = "your-iot-topic"

suspend fun main(args: Array<String>) {
    val usage =
        """
        Usage:
            <roleARN> <snsAction>

        Where:
            roleARN - The ARN of an IAM role that has permission to work with AWS
 IOT.
            snsAction  - An ARN of an SNS topic.

        """.trimIndent()

    if (args.size != 2) {
        println(usage)
        exitProcess(1)
    }

    var thingName: String
    val roleARN = args[0]
    val snsAction = args[1]
```

```
    val scanner = Scanner(System.`in`)

    println(DASHES)
    println("Welcome to the AWS IoT example scenario.")
    println(
        """
        This example program demonstrates various interactions with the AWS
Internet of Things (IoT) Core service.
        The program guides you through a series of steps, including creating an
IoT thing, generating a device certificate,
        updating the thing with attributes, and so on.

        It utilizes the AWS SDK for Kotlin and incorporates functionality for
creating and managing IoT things, certificates, rules,
        shadows, and performing searches. The program aims to showcase AWS IoT
capabilities and provides a comprehensive example for
        developers working with AWS IoT in a Kotlin environment.
        """.trimIndent(),
    )

    print("Press Enter to continue...")
    scanner.nextLine()
    println(DASHES)

    println(DASHES)
    println("1. Create an AWS IoT thing.")
    println(
        """
        An AWS IoT thing represents a virtual entity in the AWS IoT service that
can be associated with a physical device.
        """.trimIndent(),
    )
    // Prompt the user for input.
    print("Enter thing name: ")
    thingName = scanner.nextLine()
    createIoTThing(thingName)
    describeThing(thingName)
    println(DASHES)

    println(DASHES)
    println("2. Generate a device certificate.")
    println(
        """
```

```
        A device certificate performs a role in securing the communication
between devices (things) and the AWS IoT platform.
        """.trimIndent(),
    )

    print("Do you want to create a certificate for $thingName? (y/n)")
    val certAns = scanner.nextLine()
    var certificateArn: String? = ""
    if (certAns != null && certAns.trim { it <= ' ' }.equals("y", ignoreCase =
true)) {
        certificateArn = createCertificate()
        println("Attach the certificate to the AWS IoT thing.")
        attachCertificateToThing(thingName, certificateArn)
    } else {
        println("A device certificate was not created.")
    }
    println(DASHES)

    println(DASHES)
    println("3. Update an AWS IoT thing with Attributes.")
    println(
        """
        IoT thing attributes, represented as key-value pairs, offer a pivotal
advantage in facilitating efficient data
        management and retrieval within the AWS IoT ecosystem.
        """.trimIndent(),
    )
    print("Press Enter to continue...")
    scanner.nextLine()
    updateThing(thingName)
    println(DASHES)

    println(DASHES)
    println("4. Return a unique endpoint specific to the Amazon Web Services
account.")
    println(
        """
        An IoT Endpoint refers to a specific URL or Uniform Resource Locator that
serves as the entry point for communication between IoT devices and the AWS IoT
service.
        """.trimIndent(),
    )
    print("Press Enter to continue...")
    scanner.nextLine()
```

```
    val endpointUrl = describeEndpoint()
    println(DASHES)

    println(DASHES)
    println("5. List your AWS IoT certificates")
    print("Press Enter to continue...")
    scanner.nextLine()
    if (certificateArn!!.isNotEmpty()) {
        listCertificates()
    } else {
        println("You did not create a certificates. Skipping this step.")
    }
    println(DASHES)

    println(DASHES)
    println("6. Create an IoT shadow that refers to a digital representation or
virtual twin of a physical IoT device")
    println(
        """
        A thing shadow refers to a feature that enables you to create a virtual
representation, or "shadow,"
        of a physical device or thing. The thing shadow allows you to synchronize
and control the state of a device between
        the cloud and the device itself. and the AWS IoT service. For example,
you can write and retrieve JSON data from a thing shadow.

        """.trimIndent(),
    )
    print("Press Enter to continue...")
    scanner.nextLine()
    updateShawdowThing(thingName)
    println(DASHES)

    println(DASHES)
    println("7. Write out the state information, in JSON format.")
    print("Press Enter to continue...")
    scanner.nextLine()
    getPayload(thingName)
    println(DASHES)

    println(DASHES)
    println("8. Creates a rule")
    println(
        """
```

```
        Creates a rule that is an administrator-level action.
        Any user who has permission to create rules will be able to access data
processed by the rule.
        """.trimIndent(),
    )
    print("Enter Rule name: ")
    val ruleName = scanner.nextLine()
    createIoTRule(roleARN, ruleName, snsAction)
    println(DASHES)

    println(DASHES)
    println("9. List your rules.")
    print("Press Enter to continue...")
    scanner.nextLine()
    listIoTRules()
    println(DASHES)

    println(DASHES)
    println("10. Search things using the name.")
    print("Press Enter to continue...")
    scanner.nextLine()
    val queryString = "thingName:$thingName"
    searchThings(queryString)
    println(DASHES)

    println(DASHES)
    if (certificateArn.length > 0) {
        print("Do you want to detach and delete the certificate for $thingName?
(y/n)")
        val delAns = scanner.nextLine()
        if (delAns != null && delAns.trim { it <= ' ' }.equals("y", ignoreCase =
true)) {
            println("11. You selected to detach amd delete the certificate.")
            print("Press Enter to continue...")
            scanner.nextLine()
            detachThingPrincipal(thingName, certificateArn)
            deleteCertificate(certificateArn)
        } else {
            println("11. You selected not to delete the certificate.")
        }
    } else {
        println("11. You did not create a certificate so there is nothing to
delete.")
    }
```

```
    println(DASHES)

    println(DASHES)
    println("12. Delete the AWS IoT thing.")
    print("Do you want to delete the IoT thing? (y/n)")
    val delAns = scanner.nextLine()
    if (delAns != null && delAns.trim { it <= ' ' }.equals("y", ignoreCase =
 true)) {
        deleteIoTThing(thingName)
    } else {
        println("The IoT thing was not deleted.")
    }
    println(DASHES)

    println(DASHES)
    println("The AWS IoT workflow has successfully completed.")
    println(DASHES)
}

suspend fun deleteIoTThing(thingNameVal: String) {
    val deleteThingRequest =
        DeleteThingRequest {
            thingName = thingNameVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteThing(deleteThingRequest)
        println("Deleted $thingNameVal")
    }
}

suspend fun deleteCertificate(certificateArn: String) {
    val certificateProviderRequest =
        DeleteCertificateRequest {
            certificateId = extractCertificateId(certificateArn)
        }
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteCertificate(certificateProviderRequest)
        println("$certificateArn was successfully deleted.")
    }
}

private fun extractCertificateId(certificateArn: String): String? {
    // Example ARN: arn:aws:iot:region:account-id:cert/certificate-id.
```

```
    val arnParts = certificateArn.split(":".toRegex()).dropLastWhile
 { it.isEmpty() }.toTypedArray()
    val certificateIdPart = arnParts[arnParts.size - 1]
    return certificateIdPart.substring(certificateIdPart.lastIndexOf("/") + 1)
}

suspend fun detachThingPrincipal(
    thingNameVal: String,
    certificateArn: String,
) {
    val thingPrincipalRequest =
        DetachThingPrincipalRequest {
            principal = certificateArn
            thingName = thingNameVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.detachThingPrincipal(thingPrincipalRequest)
        println("$certificateArn was successfully removed from $thingNameVal")
    }
}

suspend fun searchThings(queryStringVal: String?) {
    val searchIndexRequest =
        SearchIndexRequest {
            queryString = queryStringVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val searchIndexResponse = iotClient.searchIndex(searchIndexRequest)
        if (searchIndexResponse.things?.isEmpty() == true) {
            println("No things found.")
        } else {
            searchIndexResponse.things
                ?.forEach { thing -> println("Thing id found using search is
 ${thing.thingId}") }
        }
    }
}

suspend fun listIoTRules() {
    val listTopicRulesRequest = ListTopicRulesRequest {}

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
```

```
        val listTopicRulesResponse =
 iotClient.listTopicRules(listTopicRulesRequest)
        println("List of IoT rules:")
        val ruleList = listTopicRulesResponse.rules
        ruleList?.forEach { rule ->
            println("Rule name: ${rule.ruleName}")
            println("Rule ARN: ${rule.ruleArn}")
            println("--------------")
        }
    }
}

suspend fun createIoTRule(
    roleARNVal: String?,
    ruleNameVal: String?,
    action: String?,
) {
    val sqlVal = "SELECT * FROM '$TOPIC '"
    val action1 =
        SnsAction {
            targetArn = action
            roleArn = roleARNVal
        }

    val myAction =
        Action {
            sns = action1
        }

    val topicRulePayloadVal =
        TopicRulePayload {
            sql = sqlVal
            actions = listOf(myAction)
        }

    val topicRuleRequest =
        CreateTopicRuleRequest {
            ruleName = ruleNameVal
            topicRulePayload = topicRulePayloadVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.createTopicRule(topicRuleRequest)
        println("IoT rule created successfully.")
```

```
    }
}

suspend fun getPayload(thingNameVal: String?) {
    val getThingShadowRequest =
        GetThingShadowRequest {
            thingName = thingNameVal
        }

    IotDataPlaneClient.fromEnvironment { region = "us-east-1" }.use
 { iotPlaneClient ->
        val getThingShadowResponse =
 iotPlaneClient.getThingShadow(getThingShadowRequest)
        val payload = getThingShadowResponse.payload
        val payloadString = payload?.let { java.lang.String(it, Charsets.UTF_8) }
        println("Received shadow data: $payloadString")
    }
}

suspend fun listCertificates() {
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.listCertificates()
        val certList = response.certificates
        certList?.forEach { cert ->
            println("Cert id: ${cert.certificateId}")
            println("Cert Arn: ${cert.certificateArn}")
        }
    }
}

suspend fun describeEndpoint(): String? {
    val request = DescribeEndpointRequest {}
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val endpointResponse = iotClient.describeEndpoint(request)
        val endpointUrl: String? = endpointResponse.endpointAddress
        val exString: String = getValue(endpointUrl)
        val fullEndpoint = "https://$exString-ats.iot.us-east-1.amazonaws.com"
        println("Full endpoint URL: $fullEndpoint")
        return fullEndpoint
    }
}

private fun getValue(input: String?): String {
    // Define a regular expression pattern for extracting the subdomain.
```

```
    val pattern = Pattern.compile("^(.*?)\\.iot\\.us-east-1\\.amazonaws\\.com")

    // Match the pattern against the input string.
    val matcher = pattern.matcher(input)

    // Check if a match is found.
    if (matcher.find()) {
        val subdomain = matcher.group(1)
        println("Extracted subdomain: $subdomain")
        return subdomain
    } else {
        println("No match found")
    }
    return ""
}

suspend fun updateThing(thingNameVal: String?) {
    val newLocation = "Office"
    val newFirmwareVersion = "v2.0"
    val attMap: MutableMap<String, String> = HashMap()
    attMap["location"] = newLocation
    attMap["firmwareVersion"] = newFirmwareVersion

    val attributePayloadVal =
        AttributePayload {
            attributes = attMap
        }

    val updateThingRequest =
        UpdateThingRequest {
            thingName = thingNameVal
            attributePayload = attributePayloadVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        // Update the IoT thing attributes.
        iotClient.updateThing(updateThingRequest)
        println("$thingNameVal attributes updated successfully.")
    }
}

suspend fun updateShawdowThing(thingNameVal: String?) {
    // Create the thing shadow state document.
```

```
    val stateDocument = "{\"state\":{\"reported\":{\"temperature\":25, \"humidity
\":50}}}"
    val byteStream: ByteStream = ByteStream.fromString(stateDocument)
    val byteArray: ByteArray = byteStream.toByteArray()

    val updateThingShadowRequest =
        UpdateThingShadowRequest {
            thingName = thingNameVal
            payload = byteArray
        }

    IotDataPlaneClient.fromEnvironment { region = "us-east-1" }.use
 { iotPlaneClient ->
        iotPlaneClient.updateThingShadow(updateThingShadowRequest)
        println("The thing shadow was updated successfully.")
    }
}

suspend fun attachCertificateToThing(
    thingNameVal: String?,
    certificateArn: String?,
) {
    val principalRequest =
        AttachThingPrincipalRequest {
            thingName = thingNameVal
            principal = certificateArn
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.attachThingPrincipal(principalRequest)
        println("Certificate attached to $thingNameVal successfully.")
    }
}

suspend fun describeThing(thingNameVal: String) {
    val thingRequest =
        DescribeThingRequest {
            thingName = thingNameVal
        }

    // Print Thing details.
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val describeResponse = iotClient.describeThing(thingRequest)
        println("Thing details:")
```

```
        println("Thing name: ${describeResponse.thingName}")
        println("Thing ARN:  ${describeResponse.thingArn}")
    }
}

suspend fun createCertificate(): String? {
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.createKeysAndCertificate()
        val certificatePem = response.certificatePem
        val certificateArn = response.certificateArn

        // Print the details.
        println("\nCertificate:")
        println(certificatePem)
        println("\nCertificate ARN:")
        println(certificateArn)
        return certificateArn
    }
}

suspend fun createIoTThing(thingNameVal: String) {
    val createThingRequest =
        CreateThingRequest {
            thingName = thingNameVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.createThing(createThingRequest)
        println("Created $thingNameVal}")
    }
}
```

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Cree una clase contenedora de IoT para gestionar las operaciones.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)


    def create_thing(self, thing_name):
        """
        Creates an AWS IoT thing.

        :param thing_name: The name of the thing to create.
        :return: The name and ARN of the created thing.
        """
        try:
            response = self.iot_client.create_thing(thingName=thing_name)
            logger.info("Created thing %s.", thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
                logger.info("Thing %s already exists. Skipping creation.",
 thing_name)
                return None
            logger.error(
                "Couldn't create thing %s. Here's why: %s: %s",
                thing_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

```
    def list_things(self):
        """
        Lists AWS IoT things.

        :return: The list of things.
        """
        try:
            things = []
            paginator = self.iot_client.get_paginator("list_things")
            for page in paginator.paginate():
                things.extend(page["things"])
            logger.info("Retrieved %s things.", len(things))
            return things
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't list things. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise



    def create_keys_and_certificate(self):
        """
        Creates keys and a certificate for an AWS IoT thing.

        :return: The certificate ID, ARN, and PEM.
        """
        try:
            response =
self.iot_client.create_keys_and_certificate(setAsActive=True)
            logger.info("Created certificate %s.", response["certificateId"])
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't create keys and certificate. Here's why: %s: %s",
```

```
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
        else:
            return response


    def attach_thing_principal(self, thing_name, principal):
        """
        Attaches a certificate to an AWS IoT thing.

        :param thing_name: The name of the thing.
        :param principal: The ARN of the certificate.
        """
        try:
            self.iot_client.attach_thing_principal(
                thingName=thing_name, principal=principal
            )
            logger.info("Attached principal %s to thing %s.", principal,
thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot attach principal. Resource not found.")
                return
            logger.error(
                "Couldn't attach principal to thing. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def describe_endpoint(self, endpoint_type="iot:Data-ATS"):
        """
        Gets the AWS IoT endpoint.

        :param endpoint_type: The endpoint type.
        :return: The endpoint.
        """
        try:
            response =
self.iot_client.describe_endpoint(endpointType=endpoint_type)
            logger.info("Retrieved endpoint %s.", response["endpointAddress"])
```

```
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't describe endpoint. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
        else:
            return response["endpointAddress"]


    def list_certificates(self):
        """
        Lists AWS IoT certificates.

        :return: The list of certificates.
        """
        try:
            certificates = []
            paginator = self.iot_client.get_paginator("list_certificates")
            for page in paginator.paginate():
                certificates.extend(page["certificates"])
            logger.info("Retrieved %s certificates.", len(certificates))
            return certificates
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't list certificates. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise


    def detach_thing_principal(self, thing_name, principal):
        """
        Detaches a certificate from an AWS IoT thing.

        :param thing_name: The name of the thing.
```

```
        :param principal: The ARN of the certificate.
        """
        try:
            self.iot_client.detach_thing_principal(
                thingName=thing_name, principal=principal
            )
            logger.info("Detached principal %s from thing %s.", principal,
thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot detach principal. Resource not found.")
                return
            logger.error(
                "Couldn't detach principal from thing. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def delete_certificate(self, certificate_id):
        """
        Deletes an AWS IoT certificate.

        :param certificate_id: The ID of the certificate to delete.
        """
        try:
            self.iot_client.update_certificate(
                certificateId=certificate_id, newStatus="INACTIVE"
            )
            self.iot_client.delete_certificate(certificateId=certificate_id)
            logger.info("Deleted certificate %s.", certificate_id)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot delete certificate. Resource not found.")
                return
            logger.error(
                "Couldn't delete certificate. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

```
    def create_topic_rule(self, rule_name, topic, sns_action_arn, role_arn):
        """
        Creates an AWS IoT topic rule.

        :param rule_name: The name of the rule.
        :param topic: The MQTT topic to subscribe to.
        :param sns_action_arn: The ARN of the SNS topic to publish to.
        :param role_arn: The ARN of the IAM role.
        """
        try:
            self.iot_client.create_topic_rule(
                ruleName=rule_name,
                topicRulePayload={
                    "sql": f"SELECT * FROM '{topic}'",
                    "actions": [
                        {"sns": {"targetArn": sns_action_arn, "roleArn":
role_arn}}
                    ],
                },
            )
            logger.info("Created topic rule %s.", rule_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
                logger.info("Topic rule %s already exists. Skipping creation.",
rule_name)
                return
            logger.error(
                "Couldn't create topic rule. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def list_topic_rules(self):
        """
        Lists AWS IoT topic rules.

        :return: The list of topic rules.
        """
        try:
            rules = []
            paginator = self.iot_client.get_paginator("list_topic_rules")
            for page in paginator.paginate():
```

```
                rules.extend(page["rules"])
            logger.info("Retrieved %s topic rules.", len(rules))
            return rules
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't list topic rules. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise



    def search_index(self, query):
        """
        Searches the AWS IoT index.

        :param query: The search query.
        :return: The list of things found.
        """
        try:
            response = self.iot_client.search_index(queryString=query)
            logger.info("Found %s things.", len(response.get("things", [])))
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't search index. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
        else:
            return response.get("things", [])


    def update_indexing_configuration(self):
        """
        Updates the AWS IoT indexing configuration to enable thing indexing.
        """
```

```
        try:
            self.iot_client.update_indexing_configuration(
                thingIndexingConfiguration={"thingIndexingMode": "REGISTRY"}
            )
            logger.info("Updated indexing configuration.")
        except ClientError as err:
            logger.error(
                "Couldn't update indexing configuration. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def delete_thing(self, thing_name):
        """
        Deletes an AWS IoT thing.

        :param thing_name: The name of the thing to delete.
        """
        try:
            self.iot_client.delete_thing(thingName=thing_name)
            logger.info("Deleted thing %s.", thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot delete thing. Resource not found.")
                return
            logger.error(
                "Couldn't delete thing. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def delete_topic_rule(self, rule_name):
        """
        Deletes an AWS IoT topic rule.

        :param rule_name: The name of the rule to delete.
        """
        try:
            self.iot_client.delete_topic_rule(ruleName=rule_name)
            logger.info("Deleted topic rule %s.", rule_name)
```

```
        except ClientError as err:
            logger.error(
                "Couldn't delete topic rule. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def update_thing_shadow(self, thing_name, shadow_state):
        """
        Updates the shadow for an AWS IoT thing.

        :param thing_name: The name of the thing.
        :param shadow_state: The shadow state as a dictionary.
        """
        import json
        try:
            self.iot_data_client.update_thing_shadow(
                thingName=thing_name, payload=json.dumps(shadow_state)
            )
            logger.info("Updated shadow for thing %s.", thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot update thing shadow. Resource not found.")
                return
            logger.error(
                "Couldn't update thing shadow. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def get_thing_shadow(self, thing_name):
        """
        Gets the shadow for an AWS IoT thing.

        :param thing_name: The name of the thing.
        :return: The shadow state as a dictionary.
        """
        import json
        try:
```

```
            response =
 self.iot_data_client.get_thing_shadow(thingName=thing_name)
            shadow = json.loads(response["payload"].read())
            logger.info("Retrieved shadow for thing %s.", thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot get thing shadow. Resource not found.")
                return None
            logger.error(
                "Couldn't get thing shadow. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return shadow
```

Ejecute un escenario interactivo que demuestre los conceptos básicos de la IoT.

```
class IoTScenario:
    """Runs an interactive scenario that shows how to use AWS IoT."""

    is_interactive = True

    def __init__(self, iot_wrapper, iot_data_client, cfn_client,
 stack_name="IoTBasicsStack", template_path=None):
        """
        :param iot_wrapper: An instance of the IoTWrapper class.
        :param iot_data_client: A Boto3 IoT Data Plane client.
        :param cfn_client: A Boto3 CloudFormation client.
        :param stack_name: Name for the CloudFormation stack.
        :param template_path: Path to the CloudFormation template file.
        """
        self.iot_wrapper = iot_wrapper
        self.iot_data_client = iot_data_client
        self.cfn_client = cfn_client
        self.thing_name = None
        self.certificate_arn = None
        self.certificate_id = None
        self.rule_name = None
        self.stack_name = stack_name
```

```
        self.template_path = template_path or "../../../scenarios/basics/iot/
iot_usecase/resources/cfn_template.yaml"

    def _deploy_stack(self):
        """Deploy CloudFormation stack and return outputs."""
        with open(self.template_path, "r") as f:
            template_body = f.read()

        try:
            self.cfn_client.create_stack(
                StackName=self.stack_name,
                TemplateBody=template_body,
                Capabilities=["CAPABILITY_NAMED_IAM"]
            )

            waiter = self.cfn_client.get_waiter("stack_create_complete")
            waiter.wait(StackName=self.stack_name)

            response = self.cfn_client.describe_stacks(StackName=self.stack_name)
            outputs = {output["OutputKey"]: output["OutputValue"]
                       for output in response["Stacks"][0]["Outputs"]}
            return outputs["SNSTopicArn"], outputs["RoleArn"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "AlreadyExistsException":
                response =
 self.cfn_client.describe_stacks(StackName=self.stack_name)
                outputs = {output["OutputKey"]: output["OutputValue"]
                           for output in response["Stacks"][0]["Outputs"]}
                return outputs["SNSTopicArn"], outputs["RoleArn"]
            raise

    def _cleanup_stack(self):
        """Delete CloudFormation stack."""
        try:
            self.cfn_client.delete_stack(StackName=self.stack_name)
            waiter = self.cfn_client.get_waiter("stack_delete_complete")
            waiter.wait(StackName=self.stack_name)
            print("CloudFormation stack deleted successfully.")
        except ClientError as err:
            logger.error(f"Failed to delete stack: {err}")

    def run_scenario(self, thing_name, rule_name):
        """
        Runs the IoT basics scenario.
```

```
        :param thing_name: The name of the thing to create.
        :param rule_name: The name of the topic rule to create.
        """
        print("-" * 88)
        print("Welcome to the AWS IoT basics scenario!")
        print("-" * 88)
        print(
            "This scenario demonstrates how to interact with AWS IoT using the
AWS SDK for Python (Boto3).\n"
            "AWS IoT provides secure, bi-directional communication between
Internet-connected devices\n"
            "and the AWS cloud. You can manage device connections, process device
data, and build IoT applications.\n"
        )

        self.thing_name = thing_name
        self.rule_name = rule_name

        try:
            print("\nDeploying CloudFormation stack...")
            sns_topic_arn, role_arn = self._deploy_stack()
            print(f"Stack deployed. SNS Topic: {sns_topic_arn}")

            input("\nNext, we'll create an AWS IoT thing. Press Enter to
continue...") if self.is_interactive else None
            print("\n" + "-" * 88)
            print("1. Create an AWS IoT thing")
            print("-" * 88)
            response = self.iot_wrapper.create_thing(thing_name)
            print(f"Created thing: {response['thingName']}")
            print(f"Thing ARN: {response['thingArn']}")

            input("\nNext, we'll list things. Press Enter to continue...") if
self.is_interactive else None
            print("\n" + "-" * 88)
            print("2. List things")
            print("-" * 88)
            things = self.iot_wrapper.list_things()
            print(f"Found {len(things)} thing(s) in your account")
            for thing in things[:5]:  # Show first 5
                print(f"  Thing name: {thing['thingName']}")
```

```
            input("\nNext, we'll generate a device certificate. Press Enter to
continue...") if self.is_interactive else None
            print("\n" + "-" * 88)
            print("3. Generate a device certificate")
            print("-" * 88)
            cert_response = self.iot_wrapper.create_keys_and_certificate()
            self.certificate_arn = cert_response["certificateArn"]
            self.certificate_id = cert_response["certificateId"]
            print(f"Created certificate: {self.certificate_id}")

            input("\nNext, we'll attach the certificate to the thing. Press Enter
to continue...") if self.is_interactive else None
            print("\n" + "-" * 88)
            print("4. Attach the certificate to the thing")
            print("-" * 88)
            self.iot_wrapper.attach_thing_principal(thing_name,
self.certificate_arn)
            print(f"Attached certificate to thing: {thing_name}")

            input("\nNext, we'll update the thing shadow. Press Enter to
continue...") if self.is_interactive else None
            print("\n" + "-" * 88)
            print("5. Update the thing shadow")
            print("-" * 88)
            shadow_state = {"state": {"reported": {"temperature": 25, "humidity":
50}}}
            self.iot_wrapper.update_thing_shadow(thing_name, shadow_state)
            print(f"Updated shadow for thing: {thing_name}")

            input("\nNext, we'll get the thing shadow. Press Enter to
continue...") if self.is_interactive else None
            print("\n" + "-" * 88)
            print("6. Get the thing shadow")
            print("-" * 88)
            shadow = self.iot_wrapper.get_thing_shadow(thing_name)
            print(f"Shadow state: {json.dumps(shadow['state'], indent=2)}")

            input("\nNext, we'll get the AWS IoT endpoint. Press Enter to
continue...") if self.is_interactive else None
            print("\n" + "-" * 88)
            print("7. Get the AWS IoT endpoint")
            print("-" * 88)
            endpoint = self.iot_wrapper.describe_endpoint()
            print(f"IoT endpoint: {endpoint}")
```

```
            input("\nNext, we'll list certificates. Press Enter to continue...")
if self.is_interactive else None
            print("\n" + "-" * 88)
            print("8. List certificates")
            print("-" * 88)
            certificates = self.iot_wrapper.list_certificates()
            print(f"Found {len(certificates)} certificate(s)")
            for cert in certificates:
                print(f"  Certificate ID: {cert['certificateId']}")
                print(f"  Certificate ARN: {cert['certificateArn']}")
                print(f"  Status: {cert['status']}")

            input("\nNext, we'll create a topic rule. Press Enter to
continue...") if self.is_interactive else None
            print("\n" + "-" * 88)
            print("9. Create a topic rule")
            print("-" * 88)
            self.iot_wrapper.create_topic_rule(
                rule_name, f"device/{thing_name}/data", sns_topic_arn, role_arn
            )
            print(f"Created topic rule: {rule_name}")

            input("\nNext, we'll list topic rules. Press Enter to continue...")
if self.is_interactive else None
            print("\n" + "-" * 88)
            print("10. List topic rules")
            print("-" * 88)
            rules = self.iot_wrapper.list_topic_rules()
            print(f"Found {len(rules)} topic rule(s)")
            for rule in rules:
                print(f"  Rule name: {rule['ruleName']}")
                print(f"  Rule ARN: {rule['ruleArn']}")

            input("\nNext, we'll configure thing indexing. Press Enter to
continue...") if self.is_interactive else None
            print("\n" + "-" * 88)
            print("11. Configure thing indexing")
            print("-" * 88)
            self.iot_wrapper.update_indexing_configuration()
            print("Enabled thing indexing")
            print("Waiting for indexing to be ready...")
            time.sleep(10)
```

```
            input("\nNext, we'll search for things. Press Enter to continue...")
if self.is_interactive else None
            print("\n" + "-" * 88)
            print("12. Search for things")
            print("-" * 88)
            try:
                things = self.iot_wrapper.search_index(f"thingName:{thing_name}")
                if things:
                    print(f"Found {len(things)} thing(s) matching the query")
                    for thing in things:
                        print(f"  Thing name: {thing.get('thingName', 'N/A')}")
                        print(f"  Thing ID: {thing.get('thingId', 'N/A')}")
                else:
                    print("No things found. Indexing may take a few minutes.")
            except ClientError as err:
                if err.response["Error"]["Code"] in [
                    "IndexNotReadyException",
                    "InvalidRequestException",
                ]:
                    print("Search index not ready yet. This is expected.")
                else:
                    raise

        except ClientError as err:
            logger.error(
                "Scenario failed: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        finally:
            self._cleanup()

    def _cleanup(self):
        """Cleans up resources created during the scenario."""
        if not self.thing_name:
            return

        print("\n" + "-" * 88)
        print("Cleanup")
        print("-" * 88)

        if q.ask("Do you want to delete the resources? (y/n) ", q.is_yesno):
            try:
```

```
                if self.certificate_arn:
                    print(f"Detaching certificate from thing: {self.thing_name}")
                    self.iot_wrapper.detach_thing_principal(
                        self.thing_name, self.certificate_arn
                    )

                if self.certificate_id:
                    print(f"Deleting certificate: {self.certificate_id}")
                    self.iot_wrapper.delete_certificate(self.certificate_id)

                if self.thing_name:
                    print(f"Deleting thing: {self.thing_name}")
                    self.iot_wrapper.delete_thing(self.thing_name)

                if self.rule_name:
                    print(f"Deleting topic rule: {self.rule_name}")
                    self.iot_wrapper.delete_topic_rule(self.rule_name)

                self._cleanup_stack()
                print("Resources deleted successfully.")
            except ClientError as err:
                logger.error(
                    "Cleanup failed: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
        else:
            print("Resources will remain in your account.")

        print("\n" + "-" * 88)
        print("Thanks for using AWS IoT!")
        print("-" * 88)
```

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteSe utiliza AWS IoT con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Acciones de AWS IoT uso AWS SDKs

Los siguientes ejemplos de código muestran cómo realizar AWS IoT acciones individuales con AWS SDKs. Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones para configurar y ejecutar el código.

Los siguientes ejemplos incluyen solo las acciones que se utilizan con mayor frecuencia. Para ver una lista completa, consulte la Referencia de la API de AWS IoT.

Ejemplos

- Úselo AttachThingPrincipal con un AWS SDK o CLI
- Úselo CreateKeysAndCertificate con un AWS SDK o CLI
- Úselo CreateThing con un AWS SDK o CLI
- Úselo CreateTopicRule con un AWS SDK o CLI
- Úselo DeleteCertificate con un AWS SDK o CLI
- Úselo DeleteThing con un AWS SDK o CLI
- Úselo DeleteTopicRule con un AWS SDK o CLI
- Úselo DescribeEndpoint con un AWS SDK o CLI
- Úselo DescribeThing con un AWS SDK o CLI
- Úselo DetachThingPrincipal con un AWS SDK o CLI
- Úselo ListCertificates con un AWS SDK o CLI
- Úselo ListThings con un AWS SDK o CLI
- Úselo SearchIndex con un AWS SDK o CLI
- Úselo UpdateIndexingConfiguration con un AWS SDK o CLI
- Úselo UpdateThing con un AWS SDK o CLI

Úselo **`AttachThingPrincipal`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `AttachThingPrincipal`.

.NET

SDK for .NET (v4)

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    /// <summary>
    /// Attaches a certificate to an IoT Thing.
    /// </summary>
    /// <param name="thingName">The name of the Thing.</param>
    /// <param name="certificateArn">The ARN of the certificate to attach.</
param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> AttachThingPrincipalAsync(string thingName, string
 certificateArn)
    {
        try
        {
            var request = new AttachThingPrincipalRequest
            {
                ThingName = thingName,
                Principal = certificateArn
            };

            await _amazonIoT.AttachThingPrincipalAsync(request);
            _logger.LogInformation($"Attached certificate {certificateArn} to
 Thing {thingName}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot attach certificate - resource not found:
 {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
```

```
            _logger.LogError($"Couldn't attach certificate to Thing. Here's why:
 {ex.Message}");
            return false;
        }
    }
```

- Para obtener más información sobre la API, consulta AttachThingPrincipalla Referencia AWS SDK for .NET de la API.

C++

SDK para C++

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
//! Attach a principal to an AWS IoT thing.
/*!
  \param principal: A principal to attach.
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::attachThingPrincipal(const Aws::String &principal,
                                       const Aws::String &thingName,
                                       const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient client(clientConfiguration);
    Aws::IoT::Model::AttachThingPrincipalRequest request;
    request.SetPrincipal(principal);
    request.SetThingName(thingName);
    Aws::IoT::Model::AttachThingPrincipalOutcome outcome =
 client.AttachThingPrincipal(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully attached principal to thing." << std::endl;
    }
```

```
    else {
        std::cerr << "Failed to attach principal to thing." <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obtener más información sobre la API, consulta AttachThingPrincipalla Referencia AWS SDK para C++ de la API.

CLI

AWS CLI

Para asociar un certificado a un objeto

En el siguiente `attach-thing-principal` ejemplo, se adjunta un certificado a la MyTemperatureSensor cosa. El certificado se identifica con un ARN. Puedes encontrar el ARN de un certificado en la consola de AWS IoT.

```
aws iot attach-thing-principal \
    --thing-name MyTemperatureSensor \
    --principal arn:aws:iot:us-
west-2:123456789012:cert/2e1eb273792174ec2b9bf4e9b37e6c6c692345499506002a35159767055278e8
```

Este comando no genera ninguna salida.

Para obtener más información, consulte Administración de objetos con el registro en la Guía para desarrolladores de AWS IoT.

- Para obtener más información sobre la API, consulte AttachThingPrincipalla Referencia de AWS CLI comandos.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
   /**
    * Attaches a certificate to an IoT Thing asynchronously.
    *
    * @param thingName The name of the IoT Thing.
    * @param certificateArn The ARN of the certificate to attach.
    *
    * This method initiates an asynchronous request to attach a certificate to
an IoT Thing.
    * If the request is successful, it prints a confirmation message and
additional information about the Thing.
    * If an exception occurs, it prints the error message.
    */
   public void attachCertificateToThing(String thingName, String certificateArn)
{
       AttachThingPrincipalRequest principalRequest =
AttachThingPrincipalRequest.builder()
           .thingName(thingName)
           .principal(certificateArn)
           .build();

       CompletableFuture<AttachThingPrincipalResponse> future =
getAsyncClient().attachThingPrincipal(principalRequest);
       future.whenComplete((attachResponse, ex) -> {
           if (attachResponse != null &&
attachResponse.sdkHttpResponse().isSuccessful()) {
               System.out.println("Certificate attached to Thing
successfully.");

               // Print additional information about the Thing.
               describeThing(thingName);
           } else {
               Throwable cause = ex != null ? ex.getCause() : null;
```

```
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
 cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
 cause.getMessage());
                } else {
                    System.err.println("Failed to attach certificate to Thing.
 HTTP Status Code: " +
                        attachResponse.sdkHttpResponse().statusCode());
                }
            }
        });

        future.join();
    }
```

- Para obtener más información sobre la API, consulta AttachThingPrincipalla Referencia AWS SDK for Java 2.x de la API.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun attachCertificateToThing(
    thingNameVal: String?,
    certificateArn: String?,
) {
    val principalRequest =
        AttachThingPrincipalRequest {
            thingName = thingNameVal
            principal = certificateArn
        }
```

```
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.attachThingPrincipal(principalRequest)
        println("Certificate attached to $thingNameVal successfully.")
    }
}
```

- Para obtener más información sobre la API, consulta AttachThingPrincipalla referencia sobre el AWS SDK para la API de Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def attach_thing_principal(self, thing_name, principal):
        """
        Attaches a certificate to an AWS IoT thing.
```

```
        :param thing_name: The name of the thing.
        :param principal: The ARN of the certificate.
        """
        try:
            self.iot_client.attach_thing_principal(
                thingName=thing_name, principal=principal
            )
            logger.info("Attached principal %s to thing %s.", principal,
thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot attach principal. Resource not found.")
                return
            logger.error(
                "Couldn't attach principal to thing. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obtener más información sobre la API, consulta AttachThingPrincipalla AWS Referencia de API de SDK for Python (Boto3).

Para obtener una lista completa de las guías para desarrolladores del AWS SDK y ejemplos de código, consulte. Se utiliza AWS IoT con un SDK AWS En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`CreateKeysAndCertificate`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `CreateKeysAndCertificate`.

.NET

SDK for .NET (v4)

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
   /// <summary>
   /// Creates a device certificate for AWS IoT.
   /// </summary>
   /// <returns>The certificate details including ARN and certificate PEM, or
null if creation failed.</returns>
   public async Task<(string CertificateArn, string CertificatePem, string
CertificateId)?> CreateKeysAndCertificateAsync()
   {
       try
       {
           var request = new CreateKeysAndCertificateRequest
           {
               SetAsActive = true
           };

           var response = await
_amazonIoT.CreateKeysAndCertificateAsync(request);
           _logger.LogInformation($"Created certificate with ARN
{response.CertificateArn}");
           return (response.CertificateArn, response.CertificatePem,
response.CertificateId);
       }
       catch (Amazon.IoT.Model.ThrottlingException ex)
       {
           _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
           return null;
       }
       catch (Exception ex)
       {
           _logger.LogError($"Couldn't create certificate. Here's why:
{ex.Message}");
           return null;
       }
   }
```

- Para obtener más información sobre la API, consulta CreateKeysAndCertificatela Referencia AWS SDK for .NET de la API.

C++

SDK para C++

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
//! Create keys and certificate for an Aws IoT device.
//! This routine will save certificates and keys to an output folder, if
 provided.
/*!
  \param outputFolder: Location for storing output in files, ignored when string
 is empty.
  \param certificateARNResult: A string to receive the ARN of the created
 certificate.
  \param certificateID: A string to receive the ID of the created certificate.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::createKeysAndCertificate(const Aws::String &outputFolder,
                                           Aws::String &certificateARNResult,
                                           Aws::String &certificateID,
                                           const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient client(clientConfiguration);
    Aws::IoT::Model::CreateKeysAndCertificateRequest
 createKeysAndCertificateRequest;

    Aws::IoT::Model::CreateKeysAndCertificateOutcome outcome =
            client.CreateKeysAndCertificate(createKeysAndCertificateRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created a certificate and keys" << std::endl;
        certificateARNResult = outcome.GetResult().GetCertificateArn();
        certificateID = outcome.GetResult().GetCertificateId();
        std::cout << "Certificate ARN: " << certificateARNResult << ",
 certificate ID: "
                  << certificateID << std::endl;

        if (!outputFolder.empty()) {
```

```
            std::cout << "Writing certificate and keys to the folder '" <<
 outputFolder
                      << "'." << std::endl;
            std::cout << "Be sure these files are stored securely." << std::endl;

            Aws::String certificateFilePath = outputFolder + "/
certificate.pem.crt";
            std::ofstream certificateFile(certificateFilePath);
            if (!certificateFile.is_open()) {
                std::cerr << "Error opening certificate file, '" <<
 certificateFilePath
                          << "'."
                          << std::endl;
                return false;
            }
            certificateFile << outcome.GetResult().GetCertificatePem();
            certificateFile.close();

            const Aws::IoT::Model::KeyPair &keyPair =
 outcome.GetResult().GetKeyPair();

            Aws::String privateKeyFilePath = outputFolder + "/private.pem.key";
            std::ofstream privateKeyFile(privateKeyFilePath);
            if (!privateKeyFile.is_open()) {
                std::cerr << "Error opening private key file, '" <<
 privateKeyFilePath
                          << "'."
                          << std::endl;
                return false;
            }
            privateKeyFile << keyPair.GetPrivateKey();
            privateKeyFile.close();

            Aws::String publicKeyFilePath = outputFolder + "/public.pem.key";
            std::ofstream publicKeyFile(publicKeyFilePath);
            if (!publicKeyFile.is_open()) {
                std::cerr << "Error opening public key file, '" <<
 publicKeyFilePath
                          << "'."
                          << std::endl;
                return false;
            }
            publicKeyFile << keyPair.GetPublicKey();
        }
```

```
    }
    else {
        std::cerr << "Error creating keys and certificate: "
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obtener más información sobre la API, consulta CreateKeysAndCertificatela Referencia AWS SDK para C++ de la API.

CLI

AWS CLI

Para crear un par de claves RSA y emitir un certificado X.509

El siguiente `create-keys-and-certificate` crea un par de claves RSA de 2048 bits y emite un certificado X.509 mediante la clave pública emitida. Como esta es la única vez que el AWS IoT proporciona la clave privada para este certificado, asegúrese de guardarla en un lugar seguro.

```
aws iot create-keys-and-certificate \
    --certificate-pem-outfile "myTest.cert.pem" \
    --public-key-outfile "myTest.public.key" \
    --private-key-outfile "myTest.private.key"
```

Salida:

```
{
    "certificateArn": "arn:aws:iot:us-
west-2:123456789012:cert/9894ba17925e663f1d29c23af4582b8e3b7619c31f3fbd93adcb51ae54b83dc2
    "certificateId":
 "9894ba17925e663f1d29c23af4582b8e3b7619c31f3fbd93adcb51ae54b83dc2",
    "certificatePem": "
-----BEGIN CERTIFICATE-----
MIICiTCCEXAMPLE6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC
VVMxCzAJBgNVBAgEXAMPLEAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6
b24xFDASBgNVBAsTC0lBTSEXAMPLE2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd
BgkqhkiG9w0BCQEWEG5vb25lQGFtYEXAMPLEb20wHhcNMTEwNDI1MjA0NTIxWhcN
```

```
MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCEXAMPLEJBgNVBAgTAldBMRAwDgYD
VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDAEXAMPLEsTC0lBTSBDb25z
b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEXAMPLE25lQGFt
YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+aEXAMPLE
EXAMPLEfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9T
rDHudUZEXAMPLELG5M43q7Wgc/MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE
Ibb3OhjZnzcvQAEXAMPLEWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
nUhVVxYUntneD9+h8Mg9qEXAMPLEyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb
FFBjvSfpJIlJ00zbhNYS5f6GuoEDEXAMPLEBHjJnyp378OD8uTs7fLvjx79LjSTb
NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
-----END CERTIFICATE-----\n",
    "keyPair": {
        "PublicKey": "-----BEGIN PUBLIC KEY-----
\nMIIBIjANBgkqhkEXAMPLEQEFAAOCAQ8AMIIBCgKCAQEAEXAMPLE1nnyJwKSMHw4h\nMMEXAMPLEuuN/
dMAS3fyce8DW/4+EXAMPLEyjmoF/YVF/gHr99VEEXAMPLE5VF13\n59VK7cEXAMPLE67GK+y
+jikqXOgHh/xJTwo
+sGpWEXAMPLEDz18xOd2ka4tCzuWEXAMPLEahJbYkCPUBSU8opVkR7qkEXAMPLE1DR6sx2HocliOOLtu6Fkw91swQ
\GB3ZPrNh0PzQYvjUStZeccyNCx2EXAMPLEvp9mQOUXP6plfgxwKRX2fEXAMPLEDa
\nhJLXkX3rHU2xbxJSq7D+XEXAMPLEcw+LyFhI5mgFRl88eGdsAEXAMPLElnI9EesG\nFQIDAQAB
\n-----END PUBLIC KEY-----\n",
        "PrivateKey": "-----BEGIN RSA PRIVATE KEY-----\nkey omittted for security
 reasons\n-----END RSA PRIVATE KEY-----\n"
    }
}
```

Para obtener más información, consulte Crear y registrar un certificado de dispositivo de AWS IoT en la Guía para desarrolladores de AWS IoT.

- Para obtener más información sobre la API, consulte CreateKeysAndCertificatela Referencia de AWS CLI comandos.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    /**
```

```
    * Creates an IoT certificate asynchronously.
    *
    * @return The ARN of the created certificate.
    * <p>
    * This method initiates an asynchronous request to create an IoT
certificate.
    * If the request is successful, it prints the certificate details and
returns the certificate ARN.
    * If an exception occurs, it prints the error message.
    */
   public String createCertificate() {
       CompletableFuture<CreateKeysAndCertificateResponse> future =
getAsyncClient().createKeysAndCertificate();
       final String[] certificateArn = {null};
       future.whenComplete((response, ex) -> {
           if (response != null) {
               String certificatePem = response.certificatePem();
               certificateArn[0] = response.certificateArn();

               // Print the details.
               System.out.println("\nCertificate:");
               System.out.println(certificatePem);
               System.out.println("\nCertificate ARN:");
               System.out.println(certificateArn[0]);

           } else {
               Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
               if (cause instanceof IotException) {
                   System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
               } else {
                   System.err.println("Unexpected error: " +
cause.getMessage());
               }
           }
       });

       future.join();
       return certificateArn[0];
   }
```

- Para obtener más información sobre la API, consulta CreateKeysAndCertificatela Referencia AWS SDK for Java 2.x de la API.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun createCertificate(): String? {
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.createKeysAndCertificate()
        val certificatePem = response.certificatePem
        val certificateArn = response.certificateArn

        // Print the details.
        println("\nCertificate:")
        println(certificatePem)
        println("\nCertificate ARN:")
        println(certificateArn)
        return certificateArn
    }
}
```

- Para obtener más información sobre la API, consulta CreateKeysAndCertificatela referencia sobre el AWS SDK para la API de Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def create_keys_and_certificate(self):
        """
        Creates keys and a certificate for an AWS IoT thing.

        :return: The certificate ID, ARN, and PEM.
        """
        try:
            response =
 self.iot_client.create_keys_and_certificate(setAsActive=True)
            logger.info("Created certificate %s.", response["certificateId"])
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
```

```
                    "Couldn't create keys and certificate. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
        else:
            return response
```

- Para obtener más información sobre la API, consulta CreateKeysAndCertificatela AWS Referencia de API de SDK for Python (Boto3).

Para obtener una lista completa de las guías para desarrolladores del AWS SDK y ejemplos de código, consulte. Se utiliza AWS IoT con un SDK AWS En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`CreateThing`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `CreateThing`.

.NET

SDK for .NET (v4)

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    /// <summary>
    /// Creates an AWS IoT Thing.
    /// </summary>
    /// <param name="thingName">The name of the Thing to create.</param>
    /// <returns>The ARN of the Thing created, or null if creation failed.</
returns>
    public async Task<string?> CreateThingAsync(string thingName)
    {
        try
        {
```

```
            var request = new CreateThingRequest
            {
                ThingName = thingName
            };

            var response = await _amazonIoT.CreateThingAsync(request);
            _logger.LogInformation($"Created Thing {thingName} with ARN
{response.ThingArn}");
            return response.ThingArn;
        }
        catch (Amazon.IoT.Model.ResourceAlreadyExistsException ex)
        {
            _logger.LogWarning($"Thing {thingName} already exists:
{ex.Message}");
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't create Thing {thingName}. Here's why:
{ex.Message}");
            return null;
        }
    }
```

- Para obtener más información sobre la API, consulta CreateThingla Referencia AWS SDK for .NET de la API.

C++

SDK para C++

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
//! Create an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
```

```
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::createThing(const Aws::String &thingName,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::CreateThingRequest createThingRequest;
    createThingRequest.SetThingName(thingName);

    Aws::IoT::Model::CreateThingOutcome outcome = iotClient.CreateThing(
            createThingRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to create thing " << thingName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obtener más información sobre la API, consulta CreateThingla Referencia AWS SDK para C++ de la API.

CLI

AWS CLI

Ejemplo 1: creación de un registro de objetos en el registro

El siguiente `create-thing` ejemplo crea una entrada para un dispositivo en el registro de cosas de AWS IoT.

```
aws iot create-thing \
    --thing-name SampleIoTThing
```

Salida:

```
{
```

```
    "thingName": "SampleIoTThing",
    "thingArn": "arn:aws:iot:us-west-2: 123456789012:thing/SampleIoTThing",
    "thingId": " EXAMPLE1-90ab-cdef-fedc-ba987EXAMPLE "
}
```

Ejemplo 2: definición de un objeto que está asociado a un tipo de objeto

En el siguiente ejemplo de `create-thing` se crea un objeto que tiene el tipo de objeto especificado y sus atributos.

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

Salida:

```
{
    "thingName": "MyLightBulb",
    "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/MyLightBulb",
    "thingId": "40da2e73-c6af-406e-b415-15acae538797"
}
```

Para obtener más información, consulte How to Manage Things with the Registry y Thing Types en la Guía para desarrolladores de AWS IoT.

- Para obtener más información sobre la API, consulte CreateThingla Referencia de AWS CLI comandos.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    /**
```

```
    * Creates an IoT Thing with the specified name asynchronously.
    *
    * @param thingName The name of the IoT Thing to create.
    *
    * This method initiates an asynchronous request to create an IoT Thing with
the specified name.
    * If the request is successful, it prints the name of the thing and its ARN
value.
    * If an exception occurs, it prints the error message.
    */
   public void createIoTThing(String thingName) {
       CreateThingRequest createThingRequest = CreateThingRequest.builder()
           .thingName(thingName)
           .build();

       CompletableFuture<CreateThingResponse> future =
getAsyncClient().createThing(createThingRequest);
       future.whenComplete((createThingResponse, ex) -> {
           if (createThingResponse != null) {
               System.out.println(thingName + " was successfully created. The
ARN value is " + createThingResponse.thingArn());
           } else {
               Throwable cause = ex.getCause();
               if (cause instanceof IotException) {
                   System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
               } else {
                   System.err.println("Unexpected error: " +
cause.getMessage());
               }
           }
       });

       future.join();
   }
```

- Para obtener más información sobre la API, consulta CreateThingla Referencia AWS SDK for Java 2.x de la API.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun createIoTThing(thingNameVal: String) {
    val createThingRequest =
        CreateThingRequest {
            thingName = thingNameVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.createThing(createThingRequest)
        println("Created $thingNameVal}")
    }
}
```

- Para obtener más información sobre la API, consulta CreateThingla referencia sobre el AWS SDK para la API de Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""
```

```
    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def create_thing(self, thing_name):
        """
        Creates an AWS IoT thing.

        :param thing_name: The name of the thing to create.
        :return: The name and ARN of the created thing.
        """
        try:
            response = self.iot_client.create_thing(thingName=thing_name)
            logger.info("Created thing %s.", thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
                logger.info("Thing %s already exists. Skipping creation.",
thing_name)
                return None
            logger.error(
                "Couldn't create thing %s. Here's why: %s: %s",
                thing_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- Para obtener más información sobre la API, consulta CreateThingla AWS Referencia de API de SDK for Python (Boto3).

Para obtener una lista completa de las guías para desarrolladores del AWS SDK y ejemplos de código, consulte. Se utiliza AWS IoT con un SDK AWS En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`CreateTopicRule`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `CreateTopicRule`.

.NET

SDK for .NET (v4)

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    /// <summary>
    /// Creates an IoT topic rule.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <param name="snsTopicArn">The ARN of the SNS topic for the action.</
param>
    /// <param name="roleArn">The ARN of the IAM role.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> CreateTopicRuleAsync(string ruleName, string
 snsTopicArn, string roleArn)
    {
        try
        {
            var request = new CreateTopicRuleRequest
            {
                RuleName = ruleName,
                TopicRulePayload = new TopicRulePayload
                {
                    Sql = "SELECT * FROM 'topic/subtopic'",
                    Description = $"Rule created by .NET example: {ruleName}",
                    Actions = new List<Amazon.IoT.Model.Action>
                    {
                        new Amazon.IoT.Model.Action
                        {
```

```
                            Sns = new SnsAction
                            {
                                TargetArn = snsTopicArn,
                                RoleArn = roleArn
                            }
                        }
                    },
                    RuleDisabled = false
                }
            };

            await _amazonIoT.CreateTopicRuleAsync(request);
            _logger.LogInformation($"Created IoT rule {ruleName}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceAlreadyExistsException ex)
        {
            _logger.LogWarning($"Rule {ruleName} already exists: {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't create topic rule. Here's why:
{ex.Message}");
            return false;
        }
    }
```

- Para obtener más información sobre la API, consulta CreateTopicRulela Referencia AWS SDK for .NET de la API.

C++

SDK para C++

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
//! Create an AWS IoT rule with an SNS topic as the target.
/*!
  \param ruleName: The name for the rule.
  \param snsTopic: The SNS topic ARN for the action.
  \param sql: The SQL statement used to query the topic.
  \param roleARN: The IAM role ARN for the action.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::IoT::createTopicRule(const Aws::String &ruleName,
                             const Aws::String &snsTopicARN, const Aws::String
 &sql,
                             const Aws::String &roleARN,
                             const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::CreateTopicRuleRequest request;
    request.SetRuleName(ruleName);

    Aws::IoT::Model::SnsAction snsAction;
    snsAction.SetTargetArn(snsTopicARN);
    snsAction.SetRoleArn(roleARN);

    Aws::IoT::Model::Action action;
    action.SetSns(snsAction);

    Aws::IoT::Model::TopicRulePayload topicRulePayload;
    topicRulePayload.SetSql(sql);
    topicRulePayload.SetActions({action});

    request.SetTopicRulePayload(topicRulePayload);
    auto outcome = iotClient.CreateTopicRule(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created topic rule " << ruleName << "." <<
 std::endl;
    }
    else {
        std::cerr << "Error creating topic rule " << ruleName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    return outcome.IsSuccess();
```

```
}
```

- Para obtener más información sobre la API, consulta CreateTopicRulela Referencia AWS SDK para C++ de la API.

CLI

AWS CLI

Para crear una regla que envíe una alerta de Amazon SNS

En el siguiente ejemplo de `create-topic-rule` se crea una regla que envía un mensaje de Amazon SNS cuando las lecturas del nivel de humedad del suelo son bajas, tal como se encuentran en la sombra de dispositivo.

```
aws iot create-topic-rule \
    --rule-name "LowMoistureRule" \
    --topic-rule-payload file://plant-rule.json
```

El ejemplo requiere que se guarde el siguiente código JSON en un archivo denominado`plant-rule.json`:

```
{
    "sql": "SELECT * FROM '$aws/things/MyRPi/shadow/update/accepted' WHERE
 state.reported.moisture = 'low'\n",
    "description": "Sends an alert whenever soil moisture level readings are too
 low.",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [{
            "sns": {
                "targetArn": "arn:aws:sns:us-
west-2:123456789012:MyRPiLowMoistureTopic",
                "roleArn": "arn:aws:iam::123456789012:role/service-role/
MyRPiLowMoistureTopicRole",
                "messageFormat": "RAW"
            }
    }]
}
```

Este comando no genera ninguna salida.

Para obtener más información, consulte Creación de una regla de AWS IoT en la Guía para desarrolladores de AWS IoT.

- Para obtener más información sobre la API, consulte CreateTopicRulela Referencia de AWS CLI comandos.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    /**
     * Creates an IoT rule asynchronously.
     *
     * @param roleARN The ARN of the IAM role that grants access to the rule's
actions.
     * @param ruleName The name of the IoT rule.
     * @param action The ARN of the action to perform when the rule is triggered.
     *
     * This method initiates an asynchronous request to create an IoT rule.
     * If the request is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void createIoTRule(String roleARN, String ruleName, String action) {
        String sql = "SELECT * FROM '" + TOPIC + "'";
        SnsAction action1 = SnsAction.builder()
            .targetArn(action)
            .roleArn(roleARN)
            .build();

        // Create the action.
        Action myAction = Action.builder()
            .sns(action1)
            .build();
```

```
        // Create the topic rule payload.
        TopicRulePayload topicRulePayload = TopicRulePayload.builder()
            .sql(sql)
            .actions(myAction)
            .build();

        // Create the topic rule request.
        CreateTopicRuleRequest topicRuleRequest =
CreateTopicRuleRequest.builder()
            .ruleName(ruleName)
            .topicRulePayload(topicRulePayload)
            .build();

        CompletableFuture<CreateTopicRuleResponse> future =
getAsyncClient().createTopicRule(topicRuleRequest);
        future.whenComplete((response, ex) -> {
            if (response != null) {
                System.out.println("IoT Rule created successfully.");
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to create IoT Rule.");
                }
            }
        });

        future.join();
    }
```

- Para obtener más información sobre la API, consulta CreateTopicRulela Referencia AWS SDK for Java 2.x de la API.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun createIoTRule(
    roleARNVal: String?,
    ruleNameVal: String?,
    action: String?,
) {
    val sqlVal = "SELECT * FROM '$TOPIC '"
    val action1 =
        SnsAction {
            targetArn = action
            roleArn = roleARNVal
        }

    val myAction =
        Action {
            sns = action1
        }

    val topicRulePayloadVal =
        TopicRulePayload {
            sql = sqlVal
            actions = listOf(myAction)
        }

    val topicRuleRequest =
        CreateTopicRuleRequest {
            ruleName = ruleNameVal
            topicRulePayload = topicRulePayloadVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.createTopicRule(topicRuleRequest)
        println("IoT rule created successfully.")
```

```
    }
}
```

- Para obtener más información sobre la API, consulta CreateTopicRulela referencia sobre el AWS SDK para la API de Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def create_topic_rule(self, rule_name, topic, sns_action_arn, role_arn):
        """
        Creates an AWS IoT topic rule.

        :param rule_name: The name of the rule.
        :param topic: The MQTT topic to subscribe to.
        :param sns_action_arn: The ARN of the SNS topic to publish to.
```

```
        :param role_arn: The ARN of the IAM role.
        """
        try:
            self.iot_client.create_topic_rule(
                ruleName=rule_name,
                topicRulePayload={
                    "sql": f"SELECT * FROM '{topic}'",
                    "actions": [
                        {"sns": {"targetArn": sns_action_arn, "roleArn":
role_arn}}
                    ],
                },
            )
            logger.info("Created topic rule %s.", rule_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
                logger.info("Topic rule %s already exists. Skipping creation.",
rule_name)
                return
            logger.error(
                "Couldn't create topic rule. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obtener más información sobre la API, consulta CreateTopicRulela AWS Referencia de API de SDK for Python (Boto3).

Para obtener una lista completa de las guías para desarrolladores del AWS SDK y ejemplos de código, consulte. Se utiliza AWS IoT con un SDK AWS En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`DeleteCertificate`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `DeleteCertificate`.

.NET

SDK for .NET (v4)

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    /// <summary>
    /// Deletes an IoT certificate.
    /// </summary>
    /// <param name="certificateId">The ID of the certificate to delete.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> DeleteCertificateAsync(string certificateId)
    {
        try
        {
            // First, update the certificate to inactive state
            var updateRequest = new UpdateCertificateRequest
            {
                CertificateId = certificateId,
                NewStatus = CertificateStatus.INACTIVE
            };
            await _amazonIoT.UpdateCertificateAsync(updateRequest);

            // Then delete the certificate
            var deleteRequest = new DeleteCertificateRequest
            {
                CertificateId = certificateId
            };

            await _amazonIoT.DeleteCertificateAsync(deleteRequest);
            _logger.LogInformation($"Deleted certificate {certificateId}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot delete certificate - resource not found:
{ex.Message}");
            return false;
```

```
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't delete certificate. Here's why:
{ex.Message}");
            return false;
        }
    }
```

- Para obtener más información sobre la API, consulta DeleteCertificatela Referencia AWS SDK for .NET de la API.

C++

SDK para C++

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
//! Delete a certificate.
/*!
  \param certificateID: The ID of a certificate.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::deleteCertificate(const Aws::String &certificateID,
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::DeleteCertificateRequest request;
    request.SetCertificateId(certificateID);

    Aws::IoT::Model::DeleteCertificateOutcome outcome =
 iotClient.DeleteCertificate(
            request);
```

```
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted certificate " << certificateID <<
 std::endl;
    }
    else {
        std::cerr << "Error deleting certificate " << certificateID << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obtener más información sobre la API, consulta DeleteCertificatela Referencia AWS SDK para C++ de la API.

CLI

AWS CLI

Para eliminar un certificado de dispositivo

En el siguiente ejemplo de `delete-certificate` se elimina el certificado de dispositivo con el ID especificado.

```
aws iot delete-certificate \
    --certificate-
id c0c57bbc8baaf4631a9a0345c957657f5e710473e3ddbee1428d216d54d53ac9
```

Este comando no genera ninguna salida.

Para obtener más información, consulte DeleteCertificatela Referencia de la API de AWS IoT.

- Para obtener más información sobre la API, consulte DeleteCertificatela Referencia de AWS CLI comandos.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    /**
     * Deletes a certificate asynchronously.
     *
     * @param certificateArn The ARN of the certificate to delete.
     *
     * This method initiates an asynchronous request to delete a certificate.
     * If the deletion is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void deleteCertificate(String certificateArn) {
        DeleteCertificateRequest certificateProviderRequest =
DeleteCertificateRequest.builder()
                .certificateId(extractCertificateId(certificateArn))
                .build();

        CompletableFuture<DeleteCertificateResponse> future =
getAsyncClient().deleteCertificate(certificateProviderRequest);
        future.whenComplete((voidResult, ex) -> {
            if (ex == null) {
                System.out.println(certificateArn + " was successfully
deleted.");
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + ex.getMessage());
                }
            }
        });
```

```
        future.join();
    }
```

- Para obtener más información sobre la API, consulta DeleteCertificatela Referencia AWS SDK for Java 2.x de la API.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun deleteCertificate(certificateArn: String) {
    val certificateProviderRequest =
        DeleteCertificateRequest {
            certificateId = extractCertificateId(certificateArn)
        }
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteCertificate(certificateProviderRequest)
        println("$certificateArn was successfully deleted.")
    }
}
```

- Para obtener más información sobre la API, consulta DeleteCertificatela referencia sobre el AWS SDK para la API de Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def delete_certificate(self, certificate_id):
        """
        Deletes an AWS IoT certificate.

        :param certificate_id: The ID of the certificate to delete.
        """
        try:
            self.iot_client.update_certificate(
                certificateId=certificate_id, newStatus="INACTIVE"
            )
            self.iot_client.delete_certificate(certificateId=certificate_id)
            logger.info("Deleted certificate %s.", certificate_id)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot delete certificate. Resource not found.")
```

```
            return
        logger.error(
            "Couldn't delete certificate. Here's why: %s: %s",
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
        raise
```

- Para obtener más información sobre la API, consulta DeleteCertificatela AWS Referencia de API de SDK for Python (Boto3).

Para obtener una lista completa de las guías para desarrolladores del AWS SDK y ejemplos de código, consulte. Se utiliza AWS IoT con un SDK AWS En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`DeleteThing`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `DeleteThing`.

.NET

SDK for .NET (v4)

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
/// <summary>
/// Deletes an IoT Thing.
/// </summary>
/// <param name="thingName">The name of the Thing to delete.</param>
/// <returns>True if successful, false otherwise.</returns>
public async Task<bool> DeleteThingAsync(string thingName)
{
    try
    {
        var request = new DeleteThingRequest
```

```
            {
                ThingName = thingName
            };

            await _amazonIoT.DeleteThingAsync(request);
            _logger.LogInformation($"Deleted Thing {thingName}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot delete Thing - resource not found:
{ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't delete Thing. Here's why: {ex.Message}");
            return false;
        }
    }
```

- Para obtener más información sobre la API, consulta DeleteThingla Referencia AWS SDK for .NET de la API.

C++

SDK para C++

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
//! Delete an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
```

```
bool AwsDoc::IoT::deleteThing(const Aws::String &thingName,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DeleteThingRequest request;
    request.SetThingName(thingName);
    const auto outcome = iotClient.DeleteThing(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Error deleting thing " << thingName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obtener más información sobre la API, consulta DeleteThingla Referencia AWS SDK para C++ de la API.

CLI

AWS CLI

Para visualizar información detallada acerca de un objeto

En el siguiente `delete-thing` ejemplo, se elimina un elemento del registro de AWS IoT de tu AWS cuenta.

¿Fue para borrar algo --thing-name» FourthBulb

Este comando no genera ninguna salida.

Para obtener más información, consulte Administración de objetos con el registro en la Guía para desarrolladores de AWS IoT.

- Para obtener más información sobre la API, consulte la Referencia de comandos. DeleteThingAWS CLI

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    /**
     * Deletes an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing to delete.
     *
     * This method initiates an asynchronous request to delete an IoT Thing.
     * If the deletion is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void deleteIoTThing(String thingName) {
        DeleteThingRequest deleteThingRequest = DeleteThingRequest.builder()
            .thingName(thingName)
            .build();

        CompletableFuture<DeleteThingResponse> future =
getAsyncClient().deleteThing(deleteThingRequest);
        future.whenComplete((voidResult, ex) -> {
            if (ex == null) {
                System.out.println("Deleted Thing " + thingName);
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + ex.getMessage());
                }
            }
        });

        future.join();
    }
```

- Para obtener más información sobre la API, consulta DeleteThingla Referencia AWS SDK for Java 2.x de la API.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun deleteIoTThing(thingNameVal: String) {
    val deleteThingRequest =
        DeleteThingRequest {
            thingName = thingNameVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteThing(deleteThingRequest)
        println("Deleted $thingNameVal")
    }
}
```

- Para obtener más información sobre la API, consulta DeleteThingla referencia sobre el AWS SDK para la API de Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def delete_thing(self, thing_name):
        """
        Deletes an AWS IoT thing.

        :param thing_name: The name of the thing to delete.
        """
        try:
            self.iot_client.delete_thing(thingName=thing_name)
            logger.info("Deleted thing %s.", thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot delete thing. Resource not found.")
                return
            logger.error(
                "Couldn't delete thing. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obtener más información sobre la API, consulta DeleteThingla AWS Referencia de API de SDK for Python (Boto3).

Para obtener una lista completa de las guías para desarrolladores del AWS SDK y ejemplos de código, consulte. Se utiliza AWS IoT con un SDK AWS En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`DeleteTopicRule`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `DeleteTopicRule`.

C++

SDK para C++

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
//! Delete an AWS IoT rule.
/*!
  \param ruleName: The name for the rule.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::deleteTopicRule(const Aws::String &ruleName,
                                  const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DeleteTopicRuleRequest request;
    request.SetRuleName(ruleName);

    Aws::IoT::Model::DeleteTopicRuleOutcome outcome = iotClient.DeleteTopicRule(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted rule " << ruleName << std::endl;
    }
    else {
        std::cerr << "Failed to delete rule " << ruleName <<
                  ": " << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
```

```
}
```

- Para obtener más información sobre la API, consulta DeleteTopicRulela Referencia AWS SDK para C++ de la API.

CLI

AWS CLI

Para eliminar una regla

En el siguiente ejemplo de `delete-topic-rule` se elimina la regla especificada.

```
aws iot delete-topic-rule \
    --rule-name "LowMoistureRule"
```

Este comando no genera ninguna salida.

Para obtener más información, consulte Deleting a Rule en la Guía para desarrolladores de AWS IoT.

- Para obtener más información sobre la API, consulta DeleteTopicRulela Referencia de AWS CLI comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
```

```
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def delete_topic_rule(self, rule_name):
        """
        Deletes an AWS IoT topic rule.

        :param rule_name: The name of the rule to delete.
        """
        try:
            self.iot_client.delete_topic_rule(ruleName=rule_name)
            logger.info("Deleted topic rule %s.", rule_name)
        except ClientError as err:
            logger.error(
                "Couldn't delete topic rule. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obtener más información sobre la API, consulta DeleteTopicRulela AWS Referencia de API de SDK for Python (Boto3).

Para obtener una lista completa de las guías para desarrolladores del AWS SDK y ejemplos de código, consulte. Se utiliza AWS IoT con un SDK AWS En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`DescribeEndpoint`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `DescribeEndpoint`.

.NET

SDK for .NET (v4)

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    /// <summary>
    /// Gets the AWS IoT endpoint URL.
    /// </summary>
    /// <returns>The endpoint URL, or null if retrieval failed.</returns>
    public async Task<string?> DescribeEndpointAsync()
    {
        try
        {
            var request = new DescribeEndpointRequest
            {
                EndpointType = "iot:Data-ATS"
            };

            var response = await _amazonIoT.DescribeEndpointAsync(request);
            _logger.LogInformation($"Retrieved endpoint:
{response.EndpointAddress}");
            return response.EndpointAddress;
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't describe endpoint. Here's why:
{ex.Message}");
            return null;
        }
    }
```

- Para obtener más información sobre la API, consulta DescribeEndpointla Referencia AWS SDK for .NET de la API.

C++

SDK para C++

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
//! Describe the endpoint specific to the AWS account making the call.
/*!
  \param endpointResult: String to receive the endpoint result.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::describeEndpoint(Aws::String &endpointResult,
                                   const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::String endpoint;
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DescribeEndpointRequest describeEndpointRequest;
    describeEndpointRequest.SetEndpointType(
            "iot:Data-ATS"); // Recommended endpoint type.

    Aws::IoT::Model::DescribeEndpointOutcome outcome =
 iotClient.DescribeEndpoint(
            describeEndpointRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully described endpoint." << std::endl;
        endpointResult = outcome.GetResult().GetEndpointAddress();
    }
    else {
        std::cerr << "Error describing endpoint" <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }
```

```
    return outcome.IsSuccess();
}
```

- Para obtener más información sobre la API, consulta DescribeEndpointla Referencia AWS SDK para C++ de la API.

CLI

AWS CLI

Ejemplo 1: Para obtener su AWS punto final actual

El siguiente `describe-endpoint` ejemplo recupera el AWS punto final predeterminado al que se aplican todos los comandos.

```
aws iot describe-endpoint
```

Salida:

```
{
    "endpointAddress": "abc123defghijk.iot.us-west-2.amazonaws.com"
}
```

Para obtener más información, consulte DescribeEndpointla Guía para desarrolladores de AWS IoT.

Ejemplo 2: Obtención del punto de conexión ATS

En el siguiente ejemplo de `describe-endpoint` se recupera el punto de conexión de Amazon Trust Services (ATS).

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

Salida:

```
{
    "endpointAddress": "abc123defghijk-ats.iot.us-west-2.amazonaws.com"
```

```
}
```

Para obtener más información, consulte Certificados X.509 e AWS IoT en la Guía para desarrolladores de AWS IoT.

- Para obtener más información sobre la API, consulte DescribeEndpointla Referencia de AWS CLI comandos.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    /**
     * Describes the endpoint of the IoT service asynchronously.
     *
     * @return A CompletableFuture containing the full endpoint URL.
     *
     * This method initiates an asynchronous request to describe the endpoint of
 the IoT service.
     * If the request is successful, it prints and returns the full endpoint URL.
     * If an exception occurs, it prints the error message.
     */
    public String describeEndpoint() {
        CompletableFuture<DescribeEndpointResponse> future =
 getAsyncClient().describeEndpoint(DescribeEndpointRequest.builder().endpointType("iot:Da
ATS").build());
        final String[] result = {null};

        future.whenComplete((endpointResponse, ex) -> {
            if (endpointResponse != null) {
                String endpointUrl = endpointResponse.endpointAddress();
                String exString = getValue(endpointUrl);
                String fullEndpoint = "https://" + exString + "-ats.iot.us-
east-1.amazonaws.com";

                System.out.println("Full Endpoint URL: " + fullEndpoint);
```

```
                result[0] = fullEndpoint;
            } else {
                Throwable cause = (ex instanceof CompletionException) ?
 ex.getCause() : ex;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
 cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " +
 cause.getMessage());
                }
            }
        });

        future.join();
        return result[0];
    }
```

- Para obtener más información sobre la API, consulta DescribeEndpointla Referencia AWS SDK for Java 2.x de la API.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun describeEndpoint(): String? {
    val request = DescribeEndpointRequest {}
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val endpointResponse = iotClient.describeEndpoint(request)
        val endpointUrl: String? = endpointResponse.endpointAddress
        val exString: String = getValue(endpointUrl)
        val fullEndpoint = "https://$exString-ats.iot.us-east-1.amazonaws.com"
        println("Full endpoint URL: $fullEndpoint")
        return fullEndpoint
```

```
    }
}
```

- Para obtener más información sobre la API, consulta DescribeEndpointla referencia sobre el AWS SDK para la API de Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def describe_endpoint(self, endpoint_type="iot:Data-ATS"):
        """
        Gets the AWS IoT endpoint.

        :param endpoint_type: The endpoint type.
        :return: The endpoint.
        """
```

```
        try:
            response =
 self.iot_client.describe_endpoint(endpointType=endpoint_type)
            logger.info("Retrieved endpoint %s.", response["endpointAddress"])
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't describe endpoint. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
        else:
            return response["endpointAddress"]
```

- Para obtener más información sobre la API, consulta DescribeEndpointla AWS Referencia de API de SDK for Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
async fn show_address(client: &Client, endpoint_type: &str) -> Result<(), Error>
 {
    let resp = client
        .describe_endpoint()
        .endpoint_type(endpoint_type)
        .send()
        .await?;

    println!("Endpoint address: {}", resp.endpoint_address.unwrap());
```

```
    println!();

    Ok(())
}
```

- Para obtener más información sobre la API, consulta DescribeEndpointla referencia sobre la API de AWS SDK para Rust.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteSe utiliza AWS IoT con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`DescribeThing`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `DescribeThing`.

C++

SDK para C++

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
//! Describe an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::describeThing(const Aws::String &thingName,
                                const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::DescribeThingRequest request;
    request.SetThingName(thingName);
```

```
    Aws::IoT::Model::DescribeThingOutcome outcome =
iotClient.DescribeThing(request);

    if (outcome.IsSuccess()) {
        const Aws::IoT::Model::DescribeThingResult &result = outcome.GetResult();
        std::cout << "Retrieved thing '" << result.GetThingName() << "'" <<
std::endl;
        std::cout << "thingArn: " << result.GetThingArn() << std::endl;
        std::cout << result.GetAttributes().size() << " attribute(s) retrieved"
                  << std::endl;
        for (const auto &attribute: result.GetAttributes()) {
            std::cout << "  attribute: " << attribute.first << "=" <<
attribute.second
                      << std::endl;
        }
    }
    else {
        std::cerr << "Error describing thing " << thingName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obtener más información sobre la API, consulta DescribeThingla Referencia AWS SDK para C++ de la API.

CLI

AWS CLI

Para visualizar información detallada acerca de un objeto

El siguiente `describe-thing` ejemplo muestra información sobre una cosa (dispositivo) que está definida en el registro de AWS IoT de su AWS cuenta.

¿Cómo describir-thing --thing-name» MyLightBulb

Salida:

```
{
```

```
    "defaultClientId": "MyLightBulb",
    "thingName": "MyLightBulb",
    "thingId": "40da2e73-c6af-406e-b415-15acae538797",
    "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/MyLightBulb",
    "thingTypeName": "LightBulb",
    "attributes": {
        "model": "123",
        "wattage": "75"
    },
    "version": 1
}
```

Para obtener más información, consulte Administración de objetos con el registro en la Guía para desarrolladores de AWS IoT.

- Para obtener más información sobre la API, consulte la Referencia de comandos. DescribeThingAWS CLI

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    /**
     * Describes an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     *
     * This method initiates an asynchronous request to describe an IoT Thing.
     * If the request is successful, it prints the Thing details.
     * If an exception occurs, it prints the error message.
     */
    private void describeThing(String thingName) {
        DescribeThingRequest thingRequest = DescribeThingRequest.builder()
            .thingName(thingName)
            .build();
```

```
        CompletableFuture<DescribeThingResponse> future =
getAsyncClient().describeThing(thingRequest);
        future.whenComplete((describeResponse, ex) -> {
            if (describeResponse != null) {
                System.out.println("Thing Details:");
                System.out.println("Thing Name: " +
describeResponse.thingName());
                System.out.println("Thing ARN: " + describeResponse.thingArn());
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to describe Thing.");
                }
            }
        });

        future.join();
    }
```

- Para obtener más información sobre la API, consulta DescribeThingla Referencia AWS SDK for Java 2.x de la API.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun describeThing(thingNameVal: String) {
```

```
    val thingRequest =
        DescribeThingRequest {
            thingName = thingNameVal
        }

    // Print Thing details.
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val describeResponse = iotClient.describeThing(thingRequest)
        println("Thing details:")
        println("Thing name: ${describeResponse.thingName}")
        println("Thing ARN:  ${describeResponse.thingArn}")
    }
}
```

- Para obtener más información sobre la API, consulta DescribeThingla referencia sobre el AWS SDK para la API de Kotlin.

Para ver una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consultaSe utiliza AWS IoT con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`DetachThingPrincipal`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `DetachThingPrincipal`.

.NET

SDK for .NET (v4)

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    /// <summary>
    /// Detaches a certificate from an IoT Thing.
    /// </summary>
    /// <param name="thingName">The name of the Thing.</param>
```

```
    /// <param name="certificateArn">The ARN of the certificate to detach.</
param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> DetachThingPrincipalAsync(string thingName, string
 certificateArn)
    {
        try
        {
            var request = new DetachThingPrincipalRequest
            {
                ThingName = thingName,
                Principal = certificateArn
            };

            await _amazonIoT.DetachThingPrincipalAsync(request);
            _logger.LogInformation($"Detached certificate {certificateArn} from
 Thing {thingName}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot detach certificate - resource not found:
 {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't detach certificate from Thing. Here's
 why: {ex.Message}");
            return false;
        }
    }
```

- Para obtener más información sobre la API, consulta DetachThingPrincipalla Referencia AWS SDK for .NET de la API.

C++

SDK para C++

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
//! Detach a principal from an AWS IoT thing.
/*!
  \param principal: A principal to detach.
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::detachThingPrincipal(const Aws::String &principal,
                                       const Aws::String &thingName,
                                       const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::DetachThingPrincipalRequest detachThingPrincipalRequest;
    detachThingPrincipalRequest.SetThingName(thingName);
    detachThingPrincipalRequest.SetPrincipal(principal);

    Aws::IoT::Model::DetachThingPrincipalOutcome outcome =
 iotClient.DetachThingPrincipal(
            detachThingPrincipalRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully detached principal " << principal << " from
 thing "
                  << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to detach principal " << principal << " from thing "
                  << thingName << ": "
                  << outcome.GetError().GetMessage() << std::endl;
    }
```

```
    return outcome.IsSuccess();
}
```

- Para obtener más información sobre la API, consulta DetachThingPrincipalla Referencia AWS SDK para C++ de la API.

CLI

AWS CLI

Para separar una cosa certificate/principal de una cosa

En el siguiente ejemplo de `detach-thing-principal` se elimina un certificado que representa una entidad principal del objeto especificado.

```
aws iot detach-thing-principal \
    --thing-name "MyLightBulb" \
    --principal "arn:aws:iot:us-
west-2:123456789012:cert/604c48437a57b7d5fc5d137c5be75011c6ee67c9a6943683a1acb4b1626bac36
```

Este comando no genera ninguna salida.

Para obtener más información, consulte Administración de objetos con el registro en la Guía para desarrolladores de AWS IoT.

- Para obtener más información sobre la API, consulte DetachThingPrincipalla Referencia de AWS CLI comandos.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    /**
```

```
    * Detaches a principal (certificate) from an IoT Thing asynchronously.
    *
    * @param thingName The name of the IoT Thing.
    * @param certificateArn The ARN of the certificate to detach.
    *
    * This method initiates an asynchronous request to detach a certificate from
an IoT Thing.
    * If the detachment is successful, it prints a confirmation message.
    * If an exception occurs, it prints the error message.
    */
   public void detachThingPrincipal(String thingName, String certificateArn) {
       DetachThingPrincipalRequest thingPrincipalRequest =
DetachThingPrincipalRequest.builder()
           .principal(certificateArn)
           .thingName(thingName)
           .build();

       CompletableFuture<DetachThingPrincipalResponse> future =
getAsyncClient().detachThingPrincipal(thingPrincipalRequest);
       future.whenComplete((voidResult, ex) -> {
           if (ex == null) {
               System.out.println(certificateArn + " was successfully removed
from " + thingName);
           } else {
               Throwable cause = ex.getCause();
               if (cause instanceof IotException) {
                   System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
               } else {
                   System.err.println("Unexpected error: " + ex.getMessage());
               }
           }
       });

       future.join();
   }
```

- Para obtener más información sobre la API, consulta DetachThingPrincipalla Referencia AWS SDK for Java 2.x de la API.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun detachThingPrincipal(
    thingNameVal: String,
    certificateArn: String,
) {
    val thingPrincipalRequest =
        DetachThingPrincipalRequest {
            principal = certificateArn
            thingName = thingNameVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.detachThingPrincipal(thingPrincipalRequest)
        println("$certificateArn was successfully removed from $thingNameVal")
    }
}
```

- Para obtener más información sobre la API, consulta DetachThingPrincipalla referencia sobre el AWS SDK para la API de Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def detach_thing_principal(self, thing_name, principal):
        """
        Detaches a certificate from an AWS IoT thing.

        :param thing_name: The name of the thing.
        :param principal: The ARN of the certificate.
        """
        try:
            self.iot_client.detach_thing_principal(
                thingName=thing_name, principal=principal
            )
            logger.info("Detached principal %s from thing %s.", principal,
 thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot detach principal. Resource not found.")
                return
            logger.error(
                "Couldn't detach principal from thing. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obtener más información sobre la API, consulta DetachThingPrincipalla AWS Referencia de API de SDK for Python (Boto3).

Para obtener una lista completa de las guías para desarrolladores del AWS SDK y ejemplos de código, consulte. Se utiliza AWS IoT con un SDK AWS En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`ListCertificates`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `ListCertificates`.

.NET

SDK for .NET (v4)

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
   /// <summary>
   /// Lists all certificates associated with the account.
   /// </summary>
   /// <returns>List of certificate information, or empty list if listing
failed.</returns>
   public async Task<List<Certificate>> ListCertificatesAsync()
   {
       try
       {
           var request = new ListCertificatesRequest();
           var response = await _amazonIoT.ListCertificatesAsync(request);

           _logger.LogInformation($"Retrieved {response.Certificates.Count}
certificates");
           return response.Certificates;
       }
       catch (Amazon.IoT.Model.ThrottlingException ex)
       {
           _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
```

```
            return new List<Certificate>();
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't list certificates. Here's why:
 {ex.Message}");
            return new List<Certificate>();
        }
    }
```

- Para obtener más información sobre la API, consulta ListCertificatesla Referencia AWS SDK for .NET de la API.

C++

SDK para C++

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
//! List certificates registered in the AWS account making the call.
/*!
   \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::listCertificates(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::ListCertificatesRequest request;

    Aws::Vector<Aws::IoT::Model::Certificate> allCertificates;
    Aws::String marker; // Used to paginate results.
    do {
        if (!marker.empty()) {
            request.SetMarker(marker);
        }
```

```
        Aws::IoT::Model::ListCertificatesOutcome outcome =
 iotClient.ListCertificates(
                request);

        if (outcome.IsSuccess()) {
            const Aws::IoT::Model::ListCertificatesResult &result =
 outcome.GetResult();
            marker = result.GetNextMarker();
            allCertificates.insert(allCertificates.end(),
                                   result.GetCertificates().begin(),
                                   result.GetCertificates().end());
        }
        else {
            std::cerr << "Error: " << outcome.GetError().GetMessage() <<
 std::endl;
            return false;
        }
    } while (!marker.empty());

    std::cout << allCertificates.size() << " certificate(s) found." << std::endl;

    for (auto &certificate: allCertificates) {
        std::cout << "Certificate ID: " << certificate.GetCertificateId() <<
 std::endl;
        std::cout << "Certificate ARN: " << certificate.GetCertificateArn()
                  << std::endl;
        std::cout << std::endl;
    }

    return true;
}
```

- Para obtener más información sobre la API, consulta ListCertificatesla Referencia AWS SDK para C++ de la API.

CLI

AWS CLI

Ejemplo 1: Para enumerar los certificados registrados en su AWS cuenta

En el siguiente ejemplo de `list-certificates` se muestra una lista de todos los certificados registrados en la cuenta. Si supera el límite de paginación predeterminado de 25, puede utilizar el valor de respuesta `nextMarker` de este comando e introducirlo en el siguiente comando para obtener el siguiente lote de resultados. Repita el procedimiento hasta que `nextMarker` no devuelva ningún valor.

```
aws iot list-certificates
```

Salida:

```
{
    "certificates": [
        {
            "certificateArn": "arn:aws:iot:us-
west-2:123456789012:cert/604c48437a57b7d5fc5d137c5be75011c6ee67c9a6943683a1acb4b1626bac36
            "certificateId":
 "604c48437a57b7d5fc5d137c5be75011c6ee67c9a6943683a1acb4b1626bac36",
            "status": "ACTIVE",
            "creationDate": 1556810537.617
        },
        {
            "certificateArn": "arn:aws:iot:us-
west-2:123456789012:cert/262a1ac8a7d8aa72f6e96e365480f7313aa9db74b8339ec65d34dc3074e1c31e
            "certificateId":
 "262a1ac8a7d8aa72f6e96e365480f7313aa9db74b8339ec65d34dc3074e1c31e",
            "status": "ACTIVE",
            "creationDate": 1546447050.885
        },
        {
            "certificateArn": "arn:aws:iot:us-west-2:123456789012:cert/
b193ab7162c0fadca83246d24fa090300a1236fe58137e121b011804d8ac1d6b",
            "certificateId":
 "b193ab7162c0fadca83246d24fa090300a1236fe58137e121b011804d8ac1d6b",
            "status": "ACTIVE",
            "creationDate": 1546292258.322
        },
        {
            "certificateArn": "arn:aws:iot:us-
west-2:123456789012:cert/7aebeea3845d14a44ec80b06b8b78a89f3f8a706974b8b34d18f5adf0741db42
            "certificateId":
 "7aebeea3845d14a44ec80b06b8b78a89f3f8a706974b8b34d18f5adf0741db42",
            "status": "ACTIVE",
```

```
            "creationDate": 1541457693.453
        },
        {
            "certificateArn": "arn:aws:iot:us-
west-2:123456789012:cert/54458aa39ebb3eb39c91ffbbdcc3a6ca1c7c094d1644b889f735a6fc2cd9a7e3
            "certificateId":
 "54458aa39ebb3eb39c91ffbbdcc3a6ca1c7c094d1644b889f735a6fc2cd9a7e3",
            "status": "ACTIVE",
            "creationDate": 1541113568.611
        },
        {
            "certificateArn": "arn:aws:iot:us-
west-2:123456789012:cert/4f0ba725787aa94d67d2fca420eca022242532e8b3c58e7465c7778b443fd65e
            "certificateId":
 "4f0ba725787aa94d67d2fca420eca022242532e8b3c58e7465c7778b443fd65e",
            "status": "ACTIVE",
            "creationDate": 1541022751.983
        }
    ]
}
```

- Para obtener más información sobre la API, consulte ListCertificatesla Referencia de AWS CLI comandos.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    /**
     * Lists all certificates asynchronously.
     *
     * This method initiates an asynchronous request to list all certificates.
     * If the request is successful, it prints the certificate IDs and ARNs.
     * If an exception occurs, it prints the error message.
     */
```

```
    public void listCertificates() {
        CompletableFuture<ListCertificatesResponse> future =
getAsyncClient().listCertificates();
        future.whenComplete((response, ex) -> {
            if (response != null) {
                List<Certificate> certList = response.certificates();
                for (Certificate cert : certList) {
                    System.out.println("Cert id: " + cert.certificateId());
                    System.out.println("Cert Arn: " + cert.certificateArn());
                }
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to list certificates.");
                }
            }
        });

        future.join();
    }
```

- Para obtener más información sobre la API, consulta ListCertificatesla Referencia AWS SDK for Java 2.x de la API.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun listCertificates() {
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.listCertificates()
        val certList = response.certificates
        certList?.forEach { cert ->
            println("Cert id: ${cert.certificateId}")
            println("Cert Arn: ${cert.certificateArn}")
        }
    }
}
```

- Para obtener más información sobre la API, consulta ListCertificatesla referencia sobre el AWS SDK para la API de Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)
```

```
    def list_certificates(self):
        """
        Lists AWS IoT certificates.

        :return: The list of certificates.
        """
        try:
            certificates = []
            paginator = self.iot_client.get_paginator("list_certificates")
            for page in paginator.paginate():
                certificates.extend(page["certificates"])
            logger.info("Retrieved %s certificates.", len(certificates))
            return certificates
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't list certificates. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- Para obtener más información sobre la API, consulta ListCertificatesla AWS Referencia de API de SDK for Python (Boto3).

Para obtener una lista completa de las guías para desarrolladores del AWS SDK y ejemplos de código, consulte. Se utiliza AWS IoT con un SDK AWS En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`ListThings`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `ListThings`.

.NET

SDK for .NET (v4)

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    /// <summary>
    /// Lists IoT Things with pagination support.
    /// </summary>
    /// <returns>List of Things, or empty list if listing failed.</returns>
    public async Task<List<ThingAttribute>> ListThingsAsync()
    {
        try
        {
            // Use pages of 10.
            var request = new ListThingsRequest()
            {
                MaxResults = 10
            };
            var response = await _amazonIoT.ListThingsAsync(request);

            // Since there is not a built-in paginator, use the NextMarker to
paginate.
            bool hasMoreResults = true;

            var things = new List<ThingAttribute>();
            while (hasMoreResults)
            {
                things.AddRange(response.Things);

                // If NextMarker is not null, there are more results. Get the
next page of results.
                if (!String.IsNullOrEmpty(response.NextMarker))
                {
                    request.Marker = response.NextMarker;
                    response = await _amazonIoT.ListThingsAsync(request);
                }
                else
```

```
                    hasMoreResults = false;
            }

            _logger.LogInformation($"Retrieved {things.Count} Things");
            return things;
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
            return new List<ThingAttribute>();
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't list Things. Here's why: {ex.Message}");
            return new List<ThingAttribute>();
        }
    }
```

- Para obtener más información sobre la API, consulta ListThingsla Referencia AWS SDK for .NET de la API.

CLI

AWS CLI

Ejemplo 1: Creación de una lista de todos los objetos del registro

En el siguiente `list-things` ejemplo, se enumeran las cosas (dispositivos) que están definidas en el registro de AWS IoT de su AWS cuenta.

```
aws iot list-things
```

Salida:

```
{
    "things": [
        {
            "thingName": "ThirdBulb",
            "thingTypeName": "LightBulb",
```

```
            "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/ThirdBulb",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "version": 2
        },
        {
            "thingName": "MyOtherLightBulb",
            "thingTypeName": "LightBulb",
            "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/
MyOtherLightBulb",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "version": 3
        },
        {
            "thingName": "MyLightBulb",
            "thingTypeName": "LightBulb",
            "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/MyLightBulb",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "version": 1
        },
        {
        "thingName": "SampleIoTThing",
        "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/SampleIoTThing",
        "attributes": {},
        "version": 1
        }
    ]
}
```

Ejemplo 2: Creación de una lista de los objetos definidos que tienen un atributo específico

En el siguiente ejemplo de `list-things` se muestra una lista de objetos que tienen un atributo denominado `wattage`.

```
aws iot list-things \
```

```
    --attribute-name wattage
```

Salida:

```
{
    "things": [
        {
            "thingName": "MyLightBulb",
            "thingTypeName": "LightBulb",
            "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/MyLightBulb",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "version": 1
        },
        {
            "thingName": "MyOtherLightBulb",
            "thingTypeName": "LightBulb",
            "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/
MyOtherLightBulb",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "version": 3
        }
    ]
}
```

Para obtener más información, consulte Administración de objetos con el registro en la Guía para desarrolladores de AWS IoT.

- Para obtener más información sobre la API, consulte ListThingsla Referencia de AWS CLI comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def list_things(self):
        """
        Lists AWS IoT things.

        :return: The list of things.
        """
        try:
            things = []
            paginator = self.iot_client.get_paginator("list_things")
            for page in paginator.paginate():
                things.extend(page["things"])
            logger.info("Retrieved %s things.", len(things))
            return things
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
```

```
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't list things. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- Para obtener más información sobre la API, consulta ListThingsla AWS Referencia de API de SDK for Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
async fn show_things(client: &Client) -> Result<(), Error> {
    let resp = client.list_things().send().await?;

    println!("Things:");

    for thing in resp.things.unwrap() {
        println!(
            "  Name:  {}",
            thing.thing_name.as_deref().unwrap_or_default()
        );
        println!(
            "  Type:  {}",
            thing.thing_type_name.as_deref().unwrap_or_default()
        );
        println!(
            "  ARN:   {}",
```

```
            thing.thing_arn.as_deref().unwrap_or_default()
        );
        println!();
    }

    println!();

    Ok(())
}
```

- Para obtener más información sobre la API, consulta ListThingsla referencia sobre la API de AWS SDK para Rust.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteSe utiliza AWS IoT con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`SearchIndex`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `SearchIndex`.

.NET

SDK for .NET (v4)

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    /// <summary>
    /// Searches for IoT Things using the search index.
    /// </summary>
    /// <param name="queryString">The search query string.</param>
    /// <returns>List of Things that match the search criteria, or empty list if
search failed.</returns>
    public async Task<List<ThingDocument>> SearchIndexAsync(string queryString)
    {
        try
```

```
        {
            // First, try to perform the search
            var request = new SearchIndexRequest
            {
                QueryString = queryString
            };

            var response = await _amazonIoT.SearchIndexAsync(request);
            _logger.LogInformation($"Search found {response.Things.Count}
Things");
            return response.Things;
        }
        catch (Amazon.IoT.Model.IndexNotReadyException ex)
        {
            _logger.LogWarning($"Search index not ready, setting up indexing
configuration: {ex.Message}");
            return await SetupIndexAndRetrySearchAsync(queryString);
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex) when
(ex.Message.Contains("index") || ex.Message.Contains("Index"))
        {
            _logger.LogWarning($"Search index not configured, setting up indexing
configuration: {ex.Message}");
            return await SetupIndexAndRetrySearchAsync(queryString);
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
            return new List<ThingDocument>();
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't search index. Here's why: {ex.Message}");
            return new List<ThingDocument>();
        }
    }

    /// <summary>
    /// Sets up the indexing configuration and retries the search after waiting
for the index to be ready.
    /// </summary>
    /// <param name="queryString">The search query string.</param>
```

```
   /// <returns>List of Things that match the search criteria, or empty list if
setup/search failed.</returns>
   private async Task<List<ThingDocument>> SetupIndexAndRetrySearchAsync(string
queryString)
   {
       try
       {
           // Update indexing configuration to REGISTRY mode
           _logger.LogInformation("Setting up IoT search indexing
configuration...");
           await _amazonIoT.UpdateIndexingConfigurationAsync(
               new UpdateIndexingConfigurationRequest()
               {
                   ThingIndexingConfiguration = new ThingIndexingConfiguration()
                   {
                       ThingIndexingMode = ThingIndexingMode.REGISTRY
                   }
               });

           _logger.LogInformation("Indexing configuration updated. Waiting for
index to be ready...");

           // Wait for the index to be set up - this can take some time
           const int maxRetries = 10;
           const int retryDelaySeconds = 10;

           for (int attempt = 1; attempt <= maxRetries; attempt++)
           {
               try
               {
                   _logger.LogInformation($"Waiting for index to be ready
(attempt {attempt}/{maxRetries})...");
                   await Task.Delay(TimeSpan.FromSeconds(retryDelaySeconds));

                   // Try to get the current indexing configuration to see if
it's ready
                   var configResponse = await
_amazonIoT.GetIndexingConfigurationAsync(new GetIndexingConfigurationRequest());
                   if
(configResponse.ThingIndexingConfiguration?.ThingIndexingMode ==
ThingIndexingMode.REGISTRY)
                   {
                       // Try the search again
                       var request = new SearchIndexRequest
```

```
                        {
                            QueryString = queryString
                        };

                        var response = await
_amazonIoT.SearchIndexAsync(request);
                        _logger.LogInformation($"Search found
{response.Things.Count} Things after index setup");
                        return response.Things;
                    }
                }
                catch (Amazon.IoT.Model.IndexNotReadyException)
                {
                    // Index still not ready, continue waiting
                    _logger.LogInformation("Index still not ready, continuing to
wait...");
                    continue;
                }
                catch (Amazon.IoT.Model.InvalidRequestException ex) when
(ex.Message.Contains("index") || ex.Message.Contains("Index"))
                {
                    // Index still not ready, continue waiting
                    _logger.LogInformation("Index still not ready, continuing to
wait...");
                    continue;
                }
            }

            _logger.LogWarning("Timeout waiting for search index to be ready
after configuration update");
            return new List<ThingDocument>();
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't set up search index configuration. Here's
why: {ex.Message}");
            return new List<ThingDocument>();
        }
    }
```

- Para obtener más información sobre la API, consulta SearchIndexla Referencia AWS SDK for .NET de la API.

C++

SDK para C++

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
//! Query the AWS IoT fleet index.
//! For query information, see https://docs.aws.amazon.com/iot/latest/
developerguide/query-syntax.html
/*!
  \param: query: The query string.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::searchIndex(const Aws::String &query,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::SearchIndexRequest request;
    request.SetQueryString(query);

    Aws::Vector<Aws::IoT::Model::ThingDocument> allThingDocuments;
    Aws::String nextToken; // Used for pagination.
    do {
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        Aws::IoT::Model::SearchIndexOutcome outcome =
 iotClient.SearchIndex(request);

        if (outcome.IsSuccess()) {
            const Aws::IoT::Model::SearchIndexResult &result =
 outcome.GetResult();
            allThingDocuments.insert(allThingDocuments.end(),
                                     result.GetThings().cbegin(),
                                     result.GetThings().cend());
```

```
            nextToken = result.GetNextToken();

        }
        else {
            std::cerr << "Error in SearchIndex: " <<
 outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!nextToken.empty());

    std::cout << allThingDocuments.size() << " thing document(s) found." <<
 std::endl;
    for (const auto thingDocument: allThingDocuments) {
        std::cout << "  Thing name: " << thingDocument.GetThingName() << "."
                  << std::endl;
    }
    return true;
}
```

- Para obtener más información sobre la API, consulta SearchIndexla Referencia AWS SDK para C++ de la API.

CLI

AWS CLI

Para consultar el índice de objetos

En el siguiente ejemplo de `search-index` se consulta en el índice de `AWS_Things` los objetos que son del tipo `LightBulb`.

```
aws iot search-index \
    --index-name "AWS_Things" \
    --query-string "thingTypeName:LightBulb"
```

Salida:

```
{
    "things": [
        {
```

```
            "thingName": "MyLightBulb",
            "thingId": "40da2e73-c6af-406e-b415-15acae538797",
            "thingTypeName": "LightBulb",
            "thingGroupNames": [
                "LightBulbs",
                "DeadBulbs"
            ],
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "connectivity": {
                "connected": false
            }
        },
        {
            "thingName": "ThirdBulb",
            "thingId": "615c8455-33d5-40e8-95fd-3ee8b24490af",
            "thingTypeName": "LightBulb",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "connectivity": {
                "connected": false
            }
        },
        {
            "thingName": "MyOtherLightBulb",
            "thingId": "6dae0d3f-40c1-476a-80c4-1ed24ba6aa11",
            "thingTypeName": "LightBulb",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "connectivity": {
                "connected": false
            }
        }
    ]
}
```

Para obtener más información, consulte Managing Thing Indexing, en la Guía para desarrolladores de AWS IoT.

- Para obtener más información sobre la API, consulta SearchIndexla Referencia de AWS CLI comandos.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    /**
     * Searches for IoT Things asynchronously based on a query string.
     *
     * @param queryString The query string to search for Things.
     *
     * This method initiates an asynchronous request to search for IoT Things.
     * If the request is successful and Things are found, it prints their IDs.
     * If no Things are found, it prints a message indicating so.
     * If an exception occurs, it prints the error message.
     */
    public void searchThings(String queryString) {
        SearchIndexRequest searchIndexRequest = SearchIndexRequest.builder()
            .queryString(queryString)
            .build();

        CompletableFuture<SearchIndexResponse> future =
getAsyncClient().searchIndex(searchIndexRequest);
        future.whenComplete((searchIndexResponse, ex) -> {
            if (searchIndexResponse != null) {
                // Process the result.
                if (searchIndexResponse.things().isEmpty()) {
                    System.out.println("No things found.");
                } else {
                    searchIndexResponse.things().forEach(thing ->
System.out.println("Thing id found using search is " + thing.thingId()));
                }
```

```
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
 cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
 cause.getMessage());
                } else {
                    System.err.println("Failed to search for IoT Things.");
                }
            }
        });

        future.join();
    }
```

- Para obtener más información sobre la API, consulta SearchIndexla Referencia AWS SDK for Java 2.x de la API.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun searchThings(queryStringVal: String?) {
    val searchIndexRequest =
        SearchIndexRequest {
            queryString = queryStringVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val searchIndexResponse = iotClient.searchIndex(searchIndexRequest)
        if (searchIndexResponse.things?.isEmpty() == true) {
            println("No things found.")
```

```
        } else {
            searchIndexResponse.things
                ?.forEach { thing -> println("Thing id found using search is
 ${thing.thingId}") }
        }
    }
}
```

- Para obtener más información sobre la API, consulta SearchIndexla referencia sobre el AWS SDK para la API de Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def search_index(self, query):
        """
```

```
        Searches the AWS IoT index.

        :param query: The search query.
        :return: The list of things found.
        """
        try:
            response = self.iot_client.search_index(queryString=query)
            logger.info("Found %s things.", len(response.get("things", [])))
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't search index. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
        else:
            return response.get("things", [])
```

- Para obtener más información sobre la API, consulta SearchIndexla AWS Referencia de API de SDK for Python (Boto3).

Para obtener una lista completa de las guías para desarrolladores del AWS SDK y ejemplos de código, consulte. Se utiliza AWS IoT con un SDK AWS En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`UpdateIndexingConfiguration`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `UpdateIndexingConfiguration`.

C++

SDK para C++

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
//! Update the indexing configuration.
/*!
  \param thingIndexingConfiguration: A ThingIndexingConfiguration object which is
 ignored if not set.
  \param thingGroupIndexingConfiguration: A ThingGroupIndexingConfiguration
 object which is ignored if not set.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::updateIndexingConfiguration(
        const Aws::IoT::Model::ThingIndexingConfiguration
 &thingIndexingConfiguration,
        const Aws::IoT::Model::ThingGroupIndexingConfiguration
 &thingGroupIndexingConfiguration,
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::UpdateIndexingConfigurationRequest request;

    if (thingIndexingConfiguration.ThingIndexingModeHasBeenSet()) {
        request.SetThingIndexingConfiguration(thingIndexingConfiguration);
    }

    if (thingGroupIndexingConfiguration.ThingGroupIndexingModeHasBeenSet()) {

 request.SetThingGroupIndexingConfiguration(thingGroupIndexingConfiguration);
    }

    Aws::IoT::Model::UpdateIndexingConfigurationOutcome outcome =
 iotClient.UpdateIndexingConfiguration(
            request);
```

```
    if (outcome.IsSuccess()) {
        std::cout << "UpdateIndexingConfiguration succeeded." << std::endl;
    }
    else {
        std::cerr << "UpdateIndexingConfiguration failed."
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obtener más información sobre la API, consulta UpdateIndexingConfigurationla Referencia AWS SDK para C++ de la API.

CLI

AWS CLI

Para activar la indexación de objetos

El siguiente `update-indexing-configuration` ejemplo permite que la indexación de cosas permita buscar datos de registro, datos paralelos y estado de conectividad de cosas mediante el índice AWS_Things.

```
aws iot update-indexing-configuration
    --thing-indexing-
configuration thingIndexingMode=REGISTRY_AND_SHADOW,thingConnectivityIndexingMode=STATUS
```

Este comando no genera ninguna salida.

Para obtener más información, consulte Managing Thing Indexing en la Guía para desarrolladores de AWS IoT.

- Para obtener más información sobre la API, consulte la Referencia UpdateIndexingConfigurationde AWS CLI comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def update_indexing_configuration(self):
        """
        Updates the AWS IoT indexing configuration to enable thing indexing.
        """
        try:
            self.iot_client.update_indexing_configuration(
                thingIndexingConfiguration={"thingIndexingMode": "REGISTRY"}
            )
            logger.info("Updated indexing configuration.")
        except ClientError as err:
            logger.error(
                "Couldn't update indexing configuration. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
```

```
            raise
```

- Para obtener más información sobre la API, consulta UpdateIndexingConfigurationla AWS Referencia de API de SDK for Python (Boto3).

Para obtener una lista completa de las guías para desarrolladores del AWS SDK y ejemplos de código, consulte. Se utiliza AWS IoT con un SDK AWS En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`UpdateThing`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `UpdateThing`.

.NET

SDK for .NET (v4)

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
   /// <summary>
   /// Updates an IoT Thing with attributes.
   /// </summary>
   /// <param name="thingName">The name of the Thing to update.</param>
   /// <param name="attributes">Dictionary of attributes to add.</param>
   /// <returns>True if successful, false otherwise.</returns>
   public async Task<bool> UpdateThingAsync(string thingName, Dictionary<string,
string> attributes)
   {
       try
       {
           var request = new UpdateThingRequest
           {
               ThingName = thingName,
               AttributePayload = new AttributePayload
               {
```

```
                    Attributes = attributes,
                    Merge = true
                }
            };

            await _amazonIoT.UpdateThingAsync(request);
            _logger.LogInformation($"Updated Thing {thingName} with attributes");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot update Thing - resource not found:
{ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't update Thing attributes. Here's why:
{ex.Message}");
            return false;
        }
    }
```

- Para obtener más información sobre la API, consulta UpdateThingla Referencia AWS SDK for .NET de la API.

C++

SDK para C++

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
//! Update an AWS IoT thing with attributes.
/*!
  \param thingName: The name for the thing.
  \param attributeMap: A map of key/value attributes/
```

```
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::updateThing(const Aws::String &thingName,
                              const std::map<Aws::String, Aws::String>
 &attributeMap,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::UpdateThingRequest request;
    request.SetThingName(thingName);
    Aws::IoT::Model::AttributePayload attributePayload;
    for (const auto &attribute: attributeMap) {
        attributePayload.AddAttributes(attribute.first, attribute.second);
    }
    request.SetAttributePayload(attributePayload);

    Aws::IoT::Model::UpdateThingOutcome outcome = iotClient.UpdateThing(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to update thing " << thingName << ":" <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obtener más información sobre la API, consulta UpdateThingla Referencia AWS SDK para C++ de la API.

CLI

AWS CLI

Para asociar un objeto con un tipo de objeto

El siguiente `update-thing` ejemplo asocia una cosa del registro de AWS IoT a un tipo de cosa. Al realizar la asociación, se proporcionan valores para los atributos definidos por el tipo de objeto.

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

Este comando no proporciona ningún resultado. Para ver el resultado, use el comando `describe-thing`.

Para obtener más información, consulte Thing Types en la Guía para desarrolladores de AWS IoT.

- Para obtener más información sobre la API, consulte UpdateThingla Referencia de AWS CLI comandos.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun updateThing(thingNameVal: String?) {
    val newLocation = "Office"
    val newFirmwareVersion = "v2.0"
    val attMap: MutableMap<String, String> = HashMap()
    attMap["location"] = newLocation
    attMap["firmwareVersion"] = newFirmwareVersion

    val attributePayloadVal =
        AttributePayload {
            attributes = attMap
        }

    val updateThingRequest =
        UpdateThingRequest {
            thingName = thingNameVal
            attributePayload = attributePayloadVal
        }
```

```
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        // Update the IoT thing attributes.
        iotClient.updateThing(updateThingRequest)
        println("$thingNameVal attributes updated successfully.")
    }
}
```

- Para obtener más información sobre la API, consulta UpdateThingla referencia sobre el AWS SDK para la API de Kotlin.

Para ver una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consultaSe utiliza AWS IoT con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

AWS IoTCuotas de

Puede encontrar información sobre las cuotas de AWS IoT en la Referencia general de AWS.

- Para obtener más información sobre las cuotas de AWS IoT Core, consulte Puntos de enlace y cuotas de AWS IoT Core.
- Para obtener más información sobre las cuotas de AWS IoT Device Management, consulte Puntos de enlace y cuotas de AWS IoT Device Management.
- Para obtener más información sobre las cuotas de AWS IoT Device Defender, consulte Puntos de enlace y cuotas de AWS IoT Device Defender.

AWS IoT CorePrecios de

Puede encontrar información sobre los precios de AWS IoT Core en la página de marketing de AWS y en la Calculadora de precios de AWS.

- Para consultar la información sobre precios de AWS IoT Core, consulte Precios de AWS IoT Core.
- Para estimar el coste de su solución de arquitectura, consulte la Calculadora de precios de AWS.

Las traducciones son generadas a través de traducción automática. En caso de conflicto entre la traducción y la version original de inglés, prevalecerá la version en inglés.