



Panduan Developerr

AWS IoT Core



AWS IoT Core: Panduan Developer

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Merek dagang dan tampilan dagang Amazon tidak boleh digunakan sehubungan dengan produk atau layanan apa pun yang bukan milik Amazon, dengan cara apa pun yang dapat menyebabkan kebingungan antara para pelanggan, atau dengan cara apa pun yang menghina atau mendiskreditkan Amazon. Semua merek dagang lain yang tidak dimiliki oleh Amazon merupakan kekayaan masing-masing pemiliknya, yang mungkin atau mungkin tidak berafiliasi, terkait dengan, atau disponsori oleh Amazon.

Table of Contents

Apa itu AWS IoT?	1
Cara akses perangkat dan aplikasi AWS IoT	2
Apa yang AWS IoT bisa dilakukan	3
IoT dalam Industri	3
IoT dalam otomatisasi Rumah	3
Bagaimana cara AWS IoT kerja	4
Alam semesta IoT	4
AWS IoT ikhtisar layanan	7
AWS IoT Core layanan	12
Pelajari lebih lanjut tentang AWS IoT	17
Sumber daya pelatihan untuk AWS IoT	17
AWS IoT sumber daya dan panduan	17
AWS IoT di media sosial	18
AWS layanan yang digunakan oleh mesin AWS IoT Core aturan	18
Protokol komunikasi yang didukung oleh AWS IoT Core	20
Yang baru di AWS IoT konsol	20
Legenda	23
Bekerja dengan AWS SDK	24
Memulai dengan AWS IoT Core	26
Connect perangkat pertama Anda ke AWS IoT Core	26
Siapkan Anda Akun AWS	28
Mendaftar untuk Akun AWS	28
Buat pengguna dengan akses administratif	29
Buka AWS IoT konsol	30
Coba tutorial AWS IoT Core interaktif	31
Menghubungkan perangkat IoT	32
Menyimpan status perangkat offline	32
Merutekan data perangkat ke layanan	33
Coba koneksi AWS IoT cepat	34
Langkah 1. Mulai tutorialnya	35
Langkah 2. Buat objek benda	36
Langkah 3. Unduh file ke perangkat Anda	39
Langkah 4. Jalankan sampel	42
Langkah 5. Jelajahi lebih lanjut	45

Menguji konektivitas dengan titik akhir data perangkat	46
Jelajahi AWS IoT Core layanan dalam tutorial langsung	51
Opsi perangkat mana yang terbaik untuk Anda?	53
Buat AWS IoT sumber daya	54
Konfigurasi perangkat Anda	58
Lihat pesan MQTT dengan klien MQTT AWS IoT	98
Melihat pesan MQTT di klien MQTT	98
Menerbitkan pesan MQTT dari klien MQTT	101
Menguji Langganan Bersama di klien MQTT	103
Menghubungkan ke AWS IoT Core	105
AWS IoT Core- titik akhir bidang kontrol	105
AWS IoT titik akhir perangkat	106
AWS IoT Core untuk gateway dan perangkat LoRa WAN	108
Menghubungkan ke AWS IoT Core titik akhir layanan	109
AWS CLI untuk AWS IoT Core	109
AWS SDK	110
AWS SDK Seluler	116
REST API dari AWS IoT Core layanan	117
Menghubungkan perangkat ke AWS IoT	117
AWS IoT data perangkat dan titik akhir layanan	118
AWS IoT SDK perangkat	120
Protokol komunikasi perangkat	123
Topik MQTT	162
Titik akhir yang dapat dikonfigurasi	187
Menghubungkan ke titik AWS IoT akhir FIPS	208
AWS IoT Core- titik akhir bidang kontrol	208
AWS IoT Core- titik akhir bidang data	208
AWS IoT Device Management- titik akhir data pekerjaan	209
AWS IoT Device Management- Titik akhir Fleet Hub	209
AWS IoT Device Management- titik akhir terowongan aman	210
Tutorial AWS IoT	211
Membangun demo denganAWS IoTPerangkat	211
Prasyarat untuk membangun demo denganAWS IoTPerangkat	212
Mempersiapkan perangkat Anda untukAWS IoTklien perangkat	215
Menginstal dan mengkonfigurasiAWS IoTPerangkat klien	229
Menunjukkan komunikasi pesan MQTT denganAWS IoTklien perangkat	242

Menunjukkan tindakan jarak jauh (pekerjaan) denganAWS IoTPerangkat perangkat	
perangkat perangkat perangkat	262
Membersihkan	277
Membangun solusi denganAWS IoTPerangkat SDK	287
Mulai membangun solusi denganAWS IoTPerangkat SDK	287
Menyambungkan perangkatAWS IoT Core dengan menggunakanAWS IoT Device SDK	287
Membuat AWS IoT aturan untuk merutekan data perangkat ke layanan lain	311
Mempertahankan status perangkat saat perangkat offline dengan Device Shadows	355
Membuat otorisasi khusus untuk AWS IoT Core	386
Memantau kelembaban tanah dengan AWS IoT dan Raspberry Pi	405
Mengelola perangkat dengan AWS IoT	419
Cara mengelola sesuatu dengan registri	420
Buat sesuatu	420
Daftar hal-hal	420
Jelaskan hal-hal	423
Perbarui sesuatu	423
Hapus sesuatu	424
Lampirkan kepala sekolah pada suatu hal	424
Lepaskan kepala sekolah dari suatu hal	424
Jenis benda	425
Buat tipe benda	425
Daftar jenis hal	426
Jelaskan tipe benda	426
Kaitkan tipe benda dengan sesuatu	427
Menghentikan tipe benda	427
Hapus tipe benda	429
Kelompok benda statis	429
Buat grup benda statis	431
Jelaskan kelompok hal	432
Tambahkan sesuatu ke grup benda statis	433
Hapus sesuatu dari grup benda statis	434
Daftar hal-hal dalam kelompok hal	434
Daftar kelompok hal	435
Daftar grup untuk suatu hal	437
Perbarui grup benda statis	438
Hapus grup benda	438

Lampirkan kebijakan ke grup benda statis	439
Lepaskan kebijakan dari grup benda statis	440
Buat daftar kebijakan yang dilampirkan ke grup benda statis	440
Buat daftar grup untuk kebijakan	440
Dapatkan kebijakan yang efektif untuk suatu hal	441
Otorisasi uji untuk tindakan MQTT	442
Kelompok hal yang dinamis	444
Gunakan kasus kelompok benda dinamis	444
Buat grup hal yang dinamis	446
Jelaskan kelompok hal yang dinamis	447
Perbarui grup hal dinamis	448
Hapus grup benda dinamis	449
Batasan Grup Hal Dinamis dan Statis	449
Batasan Grup Hal Dinamis	449
Menandai sumber daya Anda AWS IoT	453
Dasar-dasar tanda	453
Pembatasan dan batasan tanda	454
Menggunakan tanda dengan kebijakan IAM	455
Grup penagihan	458
Melihat alokasi biaya dan data penggunaan	459
Keamanan	461
Keamanan di AWS IoT	462
Autentikasi	463
AWS pelatihan dan sertifikasi	463
Ikhtisar Sertifikat X.509	463
Otentikasi server	463
Autentikasi Klien	468
Otentikasi dan otorisasi khusus	503
Otorisasi	522
AWS pelatihan dan sertifikasi	525
AWS IoT Core kebijakan	525
Mengotorisasi panggilan langsung ke AWS layanan menggunakan penyedia AWS IoT Core kredensial	598
Akses lintas akun dengan IAM	604
Perlindungan data	606
Enkripsi data di AWS IoT	608

Keamanan transportasi di AWS IoT Core	608
Enkripsi data	614
Pengelolaan identitas dan akses	615
Audiens	615
Autentikasi dengan identitas IAM	616
Mengelola akses menggunakan kebijakan	619
Bagaimana AWS IoT bekerja dengan IAM	622
Contoh kebijakan berbasis identitas	655
AWS kebijakan terkelola	659
Pemecahan Masalah	674
Pembuatan Log dan Pemantauan	676
Alat Pemantauan	676
Validasi kepatuhan	678
Ketangguhan	680
Menggunakan AWS IoT Core dengan titik akhir VPC	680
Membuat titik akhir VPC untuk bidang data AWS IoT Core	681
Membuat titik akhir VPC untuk penyedia kredensi AWS IoT Core	682
Membuat titik akhir antarmuka VPC Amazon	683
Mengkonfigurasi zona host pribadi	684
Mengontrol Akses ke AWS IoT Core lebih dari titik akhir VPC	686
Batasan	688
Menskalakan titik akhir VPC dengan AWS IoT Core	688
Menggunakan domain khusus dengan titik akhir VPC	689
Ketersediaan titik akhir VPC untuk AWS IoT Core	689
Keamanan infrastruktur	689
Pemantauan keamanan	690
Praktik terbaik keamanan	690
Melindungi koneksi MQTT di AWS IoT	690
Jaga agar jam perangkat Anda tetap sinkron	693
Validasi sertifikat server	694
Gunakan satu identitas per perangkat	694
Gunakan detik Wilayah AWS sebagai cadangan	695
Gunakan hanya dalam penyediaan waktu	695
Izin untuk menjalankan pengujian AWS IoT Device Advisor	695
Pencegahan deputi kebingungan lintas layanan untuk Device Advisor	697
AWS pelatihan dan sertifikasi	698

Pemantauan AWS IoT	699
Konfigurasi AWS IoT logging	700
Konfigurasi peran dan kebijakan pencatatan	701
Konfigurasi logging default di AWS IoT (konsol)	703
Konfigurasi login default AWS IoT (CLI)	704
Konfigurasi login khusus sumber daya (CLI) AWS IoT	706
Tingkat Log	709
Pantau AWS IoT alarm dan metrik menggunakan Amazon CloudWatch	710
Menggunakan AWS IoT metrik	710
Membuat CloudWatch alarm di AWS IoT	711
AWS IoT metrik dan dimensi	715
Monitor AWS IoT menggunakan CloudWatch Log	737
Melihat AWS IoT log di CloudWatch konsol	738
CloudWatch Entri AWS IoT log log	739
Unggah log sisi perangkat ke Amazon CloudWatch	773
Cara kerjanya	773
Mengunggah log sisi perangkat dengan menggunakan aturan AWS IoT	774
Logging panggilan AWS IoT API menggunakan AWS CloudTrail	784
AWS IoT informasi di CloudTrail	784
Memahami entri file AWS IoT log	785
Aturan	788
Memberikan AWS IoT aturan akses yang dibutuhkannya	789
Melewati izin peran	792
Buat aturan	793
Buat aturan (Konsol)	794
Buat aturan (CLI)	795
Menandai aturan Anda	800
Melihat aturan Anda	801
Menghapus aturan	801
AWS IoT tindakan aturan	802
Apache Kafka	805
CloudWatch alarm	817
CloudWatch Log	819
CloudWatch metrik	821
DynamoDB	824
DynamoDBv2	827

Elasticsearch	829
HTTP	832
IoT Analytics	872
AWS IoT Events	875
AWS IoT SiteWise	877
Firehose	882
Kinesis Data Streams	885
Lambda	887
Lokasi	890
OpenSearch	894
Publikasikan ulang	897
S3	900
Salesforce IoT	902
SNS	903
SQS	906
Step Functions	908
Timestream	910
Memecahkan masalah aturan	918
Mengakses sumber daya lintas akun menggunakan aturan AWS IoT	918
Prasyarat	919
Penyiapan lintas akun untuk Amazon SQS	919
Penyiapan lintas akun untuk Amazon SNS	921
Penyiapan lintas akun untuk Amazon S3	923
Pengaturan lintas akun untuk AWS Lambda	925
Penanganan kesalahan (tindakan kesalahan)	927
Format pesan tindakan kesalahan	928
Contoh tindakan kesalahan	929
Mengurangi biaya pengiriman pesan dengan Basic Ingest	930
Menggunakan Basic Ingest	931
AWS IoT Referensi SQL	932
Klausula SELECT	933
Klausula FROM	935
Klausula WHERE	937
Jenis data	937
Operator	943
Fungsi	952

Literal	1022
Pernyataan kasus	1023
Ekstensi JSON	1024
Templat substitusi	1026
Kueri objek bersarang	1029
Muatan biner	1030
Versi SQL	1037
Layanan Bayangan Perangkat	1039
Menggunakan bayangan	1039
Memilih untuk menggunakan bayangan bernama atau tidak disebutkan namanya	1040
Mengakses bayangan	1041
Menggunakan bayangan di perangkat, aplikasi, dan layanan cloud lainnya	1041
Pesanan pesan	1042
Pangkas pesan bayangan	1044
Menggunakan bayangan di perangkat	1045
Menginisialisasi perangkat pada koneksi pertama ke AWS IoT	1046
Memproses pesan saat perangkat terhubung AWS IoT	1049
Memproses pesan saat perangkat tersambung kembali AWS IoT	1049
Menggunakan bayangan di aplikasi dan layanan	1050
Menginisialisasi aplikasi atau layanan pada koneksi ke AWS IoT	1051
Status pemrosesan berubah saat aplikasi atau layanan terhubung AWS IoT	1051
Mendeteksi perangkat terhubung	1051
Simulasi komunikasi layanan Device Shadow	1053
Menyiapkan simulasi	1054
Inisialisasi perangkat	1054
Kirim pembaruan dari aplikasi	1058
Menanggapi pembaruan di perangkat	1061
Amati pembaruan di aplikasi	1066
Melampaui simulasi	1067
Berinteraksi dengan bayangan	1068
Dukungan protokol	1068
Meminta dan melaporkan status	1069
Memperbarui bayangan	1069
Mengambil dokumen bayangan	1073
Menghapus data bayangan	1074
Device Shadow REST API	1077

GetThingShadow	1078
UpdateThingShadow	1080
DeleteThingShadow	1081
ListNamedShadowsForThing	1082
Topik Device Shadow MQTT	1083
/dapatkan	1085
/dapatkan/diterima	1085
/dapatkan/ditolak	1086
/perbarui	1087
/perbarui/delta	1088
/perbarui/diterima	1090
/pembaruan/dokumen	1090
/perbarui/ditolak	1091
/delete	1092
/hapus/diterima	1093
/hapus/ditolak	1094
Dokumen layanan Device Shadow	1095
Contoh dokumen bayangan	1095
Properti dokumen	1101
Negara bagian Delta	1102
Dokumen bayangan versi	1105
Token klien dalam dokumen bayangan	1105
Properti dokumen bayangan kosong	1105
Nilai array dalam dokumen bayangan	1106
Pesan kesalahan Device Shadow	1107
Tugas	1109
Mengakses pekerjaan AWS IoT	1109
AWS IoT Lowongan Kerja Wilayah dan titik akhir	1109
Apa itu operasi jarak jauh?	1109
Manfaat menggunakan AWS IoT Device Management Jobs untuk operasi jarak jauh	1110
Apa itu AWS IoT Jobs?	1112
Konsep kunci pekerjaan	1113
Pekerjaan dan status eksekusi pekerjaan	1117
Mengelola tugas	1123
Penandatanganan kode untuk pekerjaan	1123
Dokumen pekerjaan	1123

URLs yang ditandatangani sebelumnya	1124
Membuat dan mengelola pekerjaan menggunakan konsol	1126
Membuat dan mengelola pekerjaan menggunakan CLI	1129
Templat Job	1141
Template khusus dan AWS terkelola	1141
Gunakan templat AWS terkelola	1142
Buat templat pekerjaan khusus	1162
Konfigurasi Job	1170
Cara kerja konfigurasi pekerjaan	1171
Tentukan konfigurasi tambahan	1187
Perangkat dan pekerjaan	1197
Perangkat pemrograman untuk bekerja dengan pekerjaan	1200
Alur kerja perangkat	1200
Alur kerja pekerjaan	1202
Pemberitahuan pekerjaan	1207
AWS IoT pekerjaan API operations	1215
Manajemen pekerjaan dan kontrol API dan tipe data	1218
Pekerjaan perangkat MQTT dan HTTPS operasi API dan tipe data	1237
Mengamankan pengguna dan perangkat untuk Pekerjaan	1252
Jenis kebijakan yang diperlukan untuk AWS IoT Pekerjaan	1252
Mengotorisasi Jobs pengguna dan layanan cloud	1254
Mengotorisasi perangkat untuk menggunakan pekerjaan	1266
Batas Job	1270
Batas pekerjaan aktif dan bersamaan	1271
AWS IoT terowongan aman	1275
Apa itu terowongan aman?	1275
Konsep terowongan yang aman	1275
Cara kerja tunneling yang aman	1277
Siklus hidup terowongan yang aman	1278
AWS IoT tutorial tunneling aman	1279
Tutorial di bagian ini	1279
Buka terowongan dan mulai sesi SSH ke perangkat jarak jauh	1280
Buka terowongan untuk perangkat jarak jauh dan gunakan SSH berbasis browser	1298
Proksi lokal	1303
Cara menggunakan proxy lokal	1303
Konfigurasi proxy lokal untuk perangkat yang menggunakan proxy web	1309

Multiplexing dan koneksi TCP simultan	1318
Multiplexing beberapa aliran data	1318
Menggunakan koneksi TCP simultan	1322
Mengkonfigurasi perangkat jarak jauh dan menggunakan agen IoT	1325
Cuplikan agen IoT	1325
Mengontrol akses ke terowongan	1327
Prasyarat akses terowongan	1327
Kebijakan akses terowongan	1328
Menyelesaikan masalah konektivitas tunneling yang aman	1335
Kesalahan token akses klien tidak valid	1335
Kesalahan ketidakcocokan token klien	1336
Masalah konektivitas perangkat jarak jauh	1337
Penyediaan perangkat	1340
Penyediaan perangkat di AWS IoT	1341
API penyediaan armada	1342
Penyediaan perangkat yang tidak memiliki sertifikat perangkat menggunakan penyediaan armada	1343
Penyediaan dengan klaim	1344
Penyediaan oleh pengguna tepercaya	1346
Menggunakan kait pra-penyediaan dengan CLI AWS	1349
Penyediaan perangkat yang memiliki sertifikat perangkat	1352
Penyediaan satu hal	1353
ust-in-time Penyediaan J	1354
Registrasi massal	1360
Templat penyediaan	1361
Bagian parameter	1361
Bagian sumber daya	1362
Contoh template untuk pendaftaran massal	1367
Contoh template untuk just-in-time penyediaan (JITP)	1369
Penyediaan armada	1370
Kait pra-penyediaan	1374
Masukan kait pra-penyediaan	1375
Nilai pengembalian kait pra-penyediaan	1375
Contoh Lambda kait pra-penyediaan	1376
Penandatanganan sertifikat yang dikelola sendiri menggunakan penyedia AWS IoT Core sertifikat	1379

Cara kerja penandatanganan sertifikat yang dikelola sendiri dalam penyediaan armada	1380
Input fungsi Lambda penyedia sertifikat	1381
Nilai pengembalian fungsi Lambda penyedia sertifikat	1382
Contoh fungsi Lambda	1382
Penandatanganan sertifikat yang dikelola sendiri untuk penyediaan armada	1384
AWS CLI perintah untuk penyedia sertifikat	1386
Membuat kebijakan dan peran IAM untuk pengguna yang menginstal perangkat	1388
Membuat kebijakan IAM untuk pengguna yang akan menginstal perangkat	1389
Membuat peran IAM untuk pengguna yang akan menginstal perangkat	1390
Memperbarui kebijakan yang ada untuk mengotorisasi templat baru	1391
Penyediaan perangkat MQTT API	1392
CreateCertificateFromCsr	1393
CreateKeysAndCertificate	1395
RegisterThing	1397
Pengindeksan armada	1401
Mengelola pembaruan indeks	1401
Mencari di seluruh sumber data	1401
Kueri untuk data agregat	1401
Memantau data agregat dan membuat alarm dengan menggunakan metrik armada	1402
Mengelola pengindeksan armada	1402
Pengindeksan hal	1402
Pengindeksan kelompok hal	1404
Bidang yang dikelola	1404
Bidang kustom	1406
Kelola pengindeksan hal	1407
Kelola pengindeksan grup hal	1423
Kueri untuk data agregat	1425
GetStatistics	1425
GetCardinality	1428
GetPercentiles	1429
GetBucketsAgregasi	1431
Otorisasi	1433
Sintaks kueri	1433
Fitur yang didukung	1433
Fitur yang tidak didukung	1433
Catatan	1434

Contoh pertanyaan	1434
Contoh kueri grup hal	1439
Pengindeksan data lokasi	1440
Format data yang didukung	1441
Cara mengindeks data lokasi	1442
Perbarui konfigurasi pengindeksan hal	1442
Contoh geoqueries	1445
Tutorial memulai	1446
Metrik armada	1451
Tutorial memulai	1452
Mengelola metrik armada	1459
Pengiriman file berbasis MQTT	1466
Apa itu aliran?	1466
Mengelola aliran di AWS Cloud	1467
Berikan izin ke perangkat Anda	1468
Hubungkan perangkat Anda ke AWS IoT	1469
TagResource Pemakaian	1469
Menggunakan pengiriman AWS IoT file berbasis MQTT di perangkat	1470
Gunakan DescribeStream untuk mendapatkan data streaming	1471
Dapatkan blok data dari file streaming	1473
Menangani kesalahan dari pengiriman AWS IoT file berbasis MQTT	1479
Contoh kasus penggunaan di FreeRTOS OTA	1481
Penasihat Perangkat	1482
Menyiapkan	1484
Buat hal IoT	1484
Membuat peran IAM untuk digunakan sebagai peran perangkat	1484
Membuat kebijakan yang dikelola khusus untuk pengguna IAM untuk menggunakan Device Advisor	1487
Buat pengguna IAM untuk menggunakan Device Advisor	1488
Konfigurasi perangkat Anda	1490
Memulai dengan Device Advisor di konsol	1492
Alur kerja Device Advisor	1501
Prasyarat	1501
Buat definisi rangkaian pengujian	1501
Dapatkan definisi test suite	1504
Dapatkan titik akhir pengujian	1505

Memulai rangkaian uji coba	1505
Menjalankan test suite	1506
Hentikan uji coba yang dijalankan	1506
Dapatkan laporan kualifikasi untuk menjalankan rangkaian tes kualifikasi yang sukses	1507
Alur kerja konsol terperinci Device Advisor	1507
Prasyarat	1508
Buat definisi rangkaian pengujian	1508
Memulai rangkaian uji coba	1515
Hentikan uji coba yang dijalankan (opsional)	1517
Lihat detail dan log test suite run	1519
Unduh laporan AWS IoT kualifikasi	1521
Durasi panjang menguji alur kerja konsol	1521
Titik akhir VPC Penasihat Perangkat ()AWS PrivateLink	1530
Pertimbangan untuk titik akhir AWS IoT Core Device Advisor VPC	1530
Buat titik akhir VPC antarmuka untuk AWS IoT Core Device Advisor	1531
Mengontrol akses ke AWS IoT Core Device Advisor lebih dari titik akhir VPC	1532
Kasus uji Device Advisor	1533
Kasus uji Penasihat Perangkat agar memenuhi syarat untuk Program Kualifikasi AWS Perangkat.	1533
TLS	1534
MQTT	1541
Bayangan	1555
Eksekusi Job	1558
Izin dan kebijakan	1560
Tes durasi panjang	1561
Katalog Paket Perangkat Lunak	1579
Mempersiapkan untuk menggunakan Katalog Paket Perangkat Lunak	1579
.....	1579
Siklus hidup versi paket	1580
Konvensi penamaan versi Package	1582
Versi default	1582
Atribut versi	1582
Mengaktifkan pengindeksan AWS IoT armada	1583
Cadangan bernama bayangan	1583
Menghapus paket perangkat lunak	1585
Mempersiapkan keamanan	1585

Otentikasi berbasis sumber daya	1586
AWS IoT Hak pekerjaan untuk menyebarkan versi paket	1587
AWS IoT Hak pekerjaan untuk memperbarui bayangan bernama yang dilindungi undang-undang	1588
AWS IoT Izin pekerjaan untuk mengunduh dari Amazon S3	1591
Mempersiapkan pengindeksan armada	1591
Mengatur \$package bayangan sebagai sumber data	1591
Metrik ditampilkan di konsol	1592
Pola kueri	1593
Mengumpulkan distribusi versi paket melalui getBucketsAggregation	1595
Mempersiapkan AWS IoT Pekerjaan	1596
Parameter substitusi untuk pekerjaan AWS IoT	1596
Mempersiapkan dokumen pekerjaan dan versi paket untuk penyebaran	1598
Menamai paket dan versi saat menerapkan	1598
Menargetkan pekerjaan melalui kelompok hal yang AWS IoT dinamis	1599
Versi bayangan dan paket bernama yang dicadangkan	1599
Menghapus instalasi paket perangkat lunak	1600
Memulai	1600
Membuat paket dan versi	1601
Menerapkan versi paket	1603
Mengaitkan versi paket	1605
AWS IoT Core Lokasi Perangkat	1607
Jenis pengukuran dan pemecah	1607
Cara Kerja Lokasi AWS IoT Core Perangkat	1608
Cara menggunakan Lokasi AWS IoT Core Perangkat	1610
Menyelesaikan lokasi perangkat IoT	1611
Menyelesaikan lokasi perangkat (konsol)	1611
Menyelesaikan lokasi perangkat (API)	1615
Memecahkan masalah kesalahan saat menyelesaikan lokasi	1616
Menyelesaikan lokasi perangkat menggunakan topik MQTT	1618
Format topik MQTT lokasi perangkat	1618
Kebijakan untuk topik MQTT lokasi perangkat	1619
Topik dan muatan lokasi perangkat	1620
Pemecah lokasi dan muatan perangkat	1625
Pemecah berbasis Wi-Fi	1626
Pemecah berbasis seluler	1627

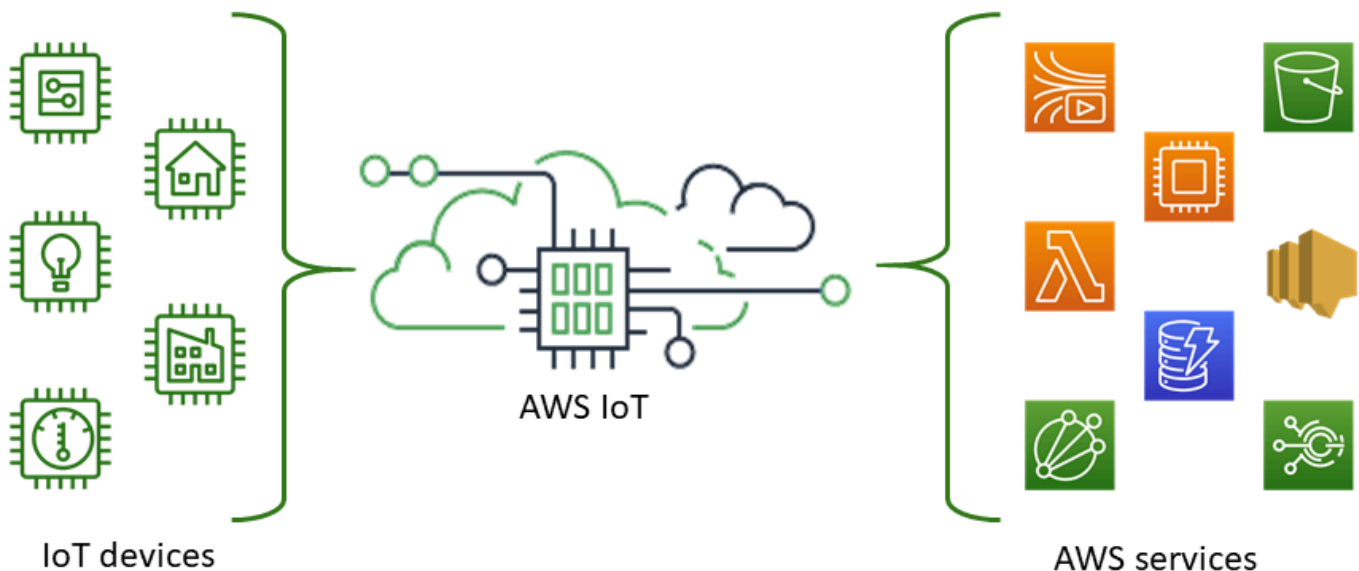
Pemecah pencarian terbalik IP	1632
Pemecah GNSS	1633
Pesan kejadian	1635
Bagaimana pesan acara dihasilkan	1635
Kebijakan untuk menerima pesan acara	1635
Aktifkan acara untuk AWS IoT	1636
Acara registri	1641
Peristiwa hal	1641
Acara jenis hal	1643
Acara kelompok hal	1646
Acara Lowongan	1652
Peristiwa siklus hidup	1656
Hubungkan/Putuskan acara	1657
Acara Berlangganan/Berhenti Berlangganan	1661
Memecahkan masalah	1663
AWS IoT Core panduan pemecahan masalah	1663
Mendiagnosis masalah konektivitas	1664
Mendiagnosis masalah aturan	1667
Mendiagnosis masalah dengan bayangan	1669
Mendiagnosis masalah tindakan Salesforce	1671
Mendiagnosis Batas Stream	1673
Pemecahan masalah armada perangkat terputus	1673
AWS IoT Panduan pemecahan masalah Device Advisor	1674
AWS IoT Device Management panduan pemecahan masalah	1677
AWS IoT Pemecahan Masalah Pekerjaan	1677
Panduan pemecahan masalah pengindeksan armada	1682
AWS IoT kesalahan	1685
AWS IoT SDK Perangkat, SDK Seluler, dan AWS IoT Klien Perangkat	1687
AWS IoT SDK perangkat	1687
AWS IoT Perangkat SDK untuk Embedded C	1689
Versi SDK AWS IoT Perangkat Sebelumnya	1690
AWS SDK Seluler	1690
AWS IoT Klien Perangkat	1691
Contoh kode	1693
Tindakan	1699
AttachThingPrincipal	1700

CreateKeysAndCertificate	1703
CreateThing	1709
CreateTopicRule	1712
DeleteCertificate	1717
DeleteThing	1720
DeleteTopicRule	1723
DescribeEndpoint	1724
DescribeThing	1728
DetachThingPrincipal	1732
ListCertificates	1735
ListThings	1740
SearchIndex	1743
UpdateIndexingConfiguration	1748
UpdateThing	1751
Skenario	1754
Bekerja dengan kasus penggunaan manajemen perangkat	1755
Kuota AWS IoT	1803
Harga AWS IoT Core	1804
.....	mdcccv

Apa itu AWS IoT?

AWS IoT menyediakan layanan cloud yang menghubungkan perangkat IoT Anda ke perangkat lain dan AWS layanan cloud. AWS IoT menyediakan perangkat lunak perangkat yang dapat membantu Anda mengintegrasikan perangkat IoT Anda ke dalam solusi AWS IoT berbasis. Jika perangkat Anda dapat terhubung AWS IoT, AWS IoT dapat menghubungkannya ke layanan cloud yang AWS menyediakan.

Untuk pengenalan langsung AWS IoT, kunjungi. [Memulai dengan AWS IoT Core](#)



AWS IoT memungkinkan Anda memilih yang paling tepat dan up-to-date teknologi untuk solusi Anda. Untuk membantu Anda mengelola dan mendukung perangkat IoT Anda di lapangan, AWS IoT Core mendukung protokol berikut:

- [MQTT \(Antrian Pesan dan Transportasi Telemetri\)](#)
- [MQTT melalui WSS \(Websockets Aman\)](#)
- [HTTPS \(Protokol Transfer Hypertext - Aman\)](#)
- [LoRaWAN \(Jaringan Area Luas Jarak Jauh\)](#)

Broker AWS IoT Core pesan mendukung perangkat dan klien yang menggunakan MQTT dan MQTT melalui protokol WSS untuk mempublikasikan dan berlangganan pesan. Ini juga mendukung perangkat dan klien yang menggunakan protokol HTTPS untuk mempublikasikan pesan.

AWS IoT Core untuk LoRa WAN membantu Anda menghubungkan dan mengelola perangkat LoRa WAN nirkabel (Jaringan Area Luas jarak jauh jarak jauh) nirkabel. AWS IoT Core untuk LoRa WAN menggantikan kebutuhan bagi Anda untuk mengembangkan dan mengoperasikan LoRa WAN Network Server (LNS).

Jika Anda tidak memerlukan AWS IoT fitur seperti komunikasi perangkat, [aturan](#), atau [pekerjaan](#), lihat [AWS Pesan](#) untuk informasi tentang layanan AWS IoT pesan lain yang mungkin lebih sesuai dengan kebutuhan Anda.

Cara akses perangkat dan aplikasi AWS IoT

AWS IoT menyediakan antarmuka berikut untuk [Tutorial AWS IoT](#):

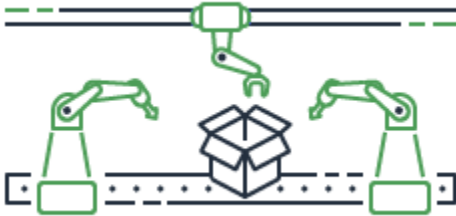
- AWS IoT SDK Perangkat —Membangun aplikasi di perangkat Anda yang mengirim pesan ke dan menerima pesan dari. AWS IoT Untuk informasi selengkapnya, lihat [AWS IoT SDK Perangkat, SDK Seluler, dan AWS IoT Klien Perangkat](#).
- AWS IoT Core untuk LoRa WAN —[Hubungkan dan kelola perangkat dan gateway WAN jarak jauh \(LoRaWAN\) Anda dengan menggunakan AWS IoT Core WAN. LoRa](#)
- AWS Command Line Interface (AWS CLI) —Jalankan perintah untuk AWS IoT Windows, macOS, dan Linux. Perintah ini memungkinkan Anda untuk membuat dan mengelola benda benda, sertifikat, aturan, pekerjaan, dan kebijakan. Untuk memulai, lihat [Panduan Pengguna AWS Command Line Interface](#). Untuk informasi selengkapnya tentang perintah untuk AWS IoT, lihat [iot](#) di AWS CLI Command Reference.
- AWS IoT API —Bangun aplikasi IoT Anda menggunakan permintaan HTTP atau HTTPS. Tindakan API ini memungkinkan Anda membuat dan mengelola objek, sertifikat, aturan, dan kebijakan secara terprogram. Untuk informasi selengkapnya tentang tindakan API AWS IoT, lihat [Tindakan](#) di Referensi AWS IoT API.
- AWS SDK —Bangun aplikasi IoT Anda menggunakan API khusus bahasa. SDK ini membungkus HTTP/HTTPS API dan memungkinkan Anda memprogram dalam bahasa apa pun yang didukung. Untuk informasi selengkapnya, lihat [AWS SDK dan Alat](#).

Anda juga dapat mengakses AWS IoT melalui [AWS IoT konsol](#), yang menyediakan antarmuka pengguna grafis (GUI) di mana Anda dapat mengonfigurasi dan mengelola objek benda, sertifikat, aturan, pekerjaan, kebijakan, dan elemen lain dari solusi IoT Anda.

Apa yang AWS IoT bisa dilakukan

Topik ini menjelaskan beberapa solusi yang mungkin Anda perlukan yang AWS IoT mendukung.

IoT dalam Industri



Ini adalah beberapa contoh AWS IoT solusi untuk [kasus penggunaan industri](#) yang menerapkan teknologi IoT untuk meningkatkan kinerja dan produktivitas proses industri.

Solusi untuk kasus penggunaan industri

- [Gunakan AWS IoT untuk membangun model kualitas prediktif dalam operasi industri](#)

Lihat AWS IoT bagaimana mengumpulkan dan menganalisis data dari operasi industri untuk membangun model kualitas prediktif. [Pelajari selengkapnya](#)

- [Gunakan AWS IoT untuk mendukung pemeliharaan prediktif dalam operasi industri](#)

Lihat bagaimana AWS IoT dapat membantu merencanakan pemeliharaan preventif untuk mengurangi waktu henti yang tidak direncanakan. [Pelajari selengkapnya](#)

IoT dalam otomatisasi Rumah



Ini adalah beberapa contoh AWS IoT solusi untuk [kasus penggunaan otomatisasi rumah](#) yang menerapkan teknologi IoT untuk membangun aplikasi IoT yang dapat diskalakan yang mengotomatiskan aktivitas rumah tangga menggunakan perangkat rumah yang terhubung.

Solusi untuk otomatisasi rumah

- [Gunakan AWS IoT di rumah Anda yang terhubung](#)

Lihat bagaimana AWS IoT dapat memberikan solusi otomatisasi rumah terintegrasi.

- [Gunakan AWS IoT untuk memberikan keamanan dan pemantauan rumah](#)

Lihat AWS IoT bagaimana menerapkan pembelajaran mesin dan komputasi tepi ke solusi otomatisasi rumah Anda.

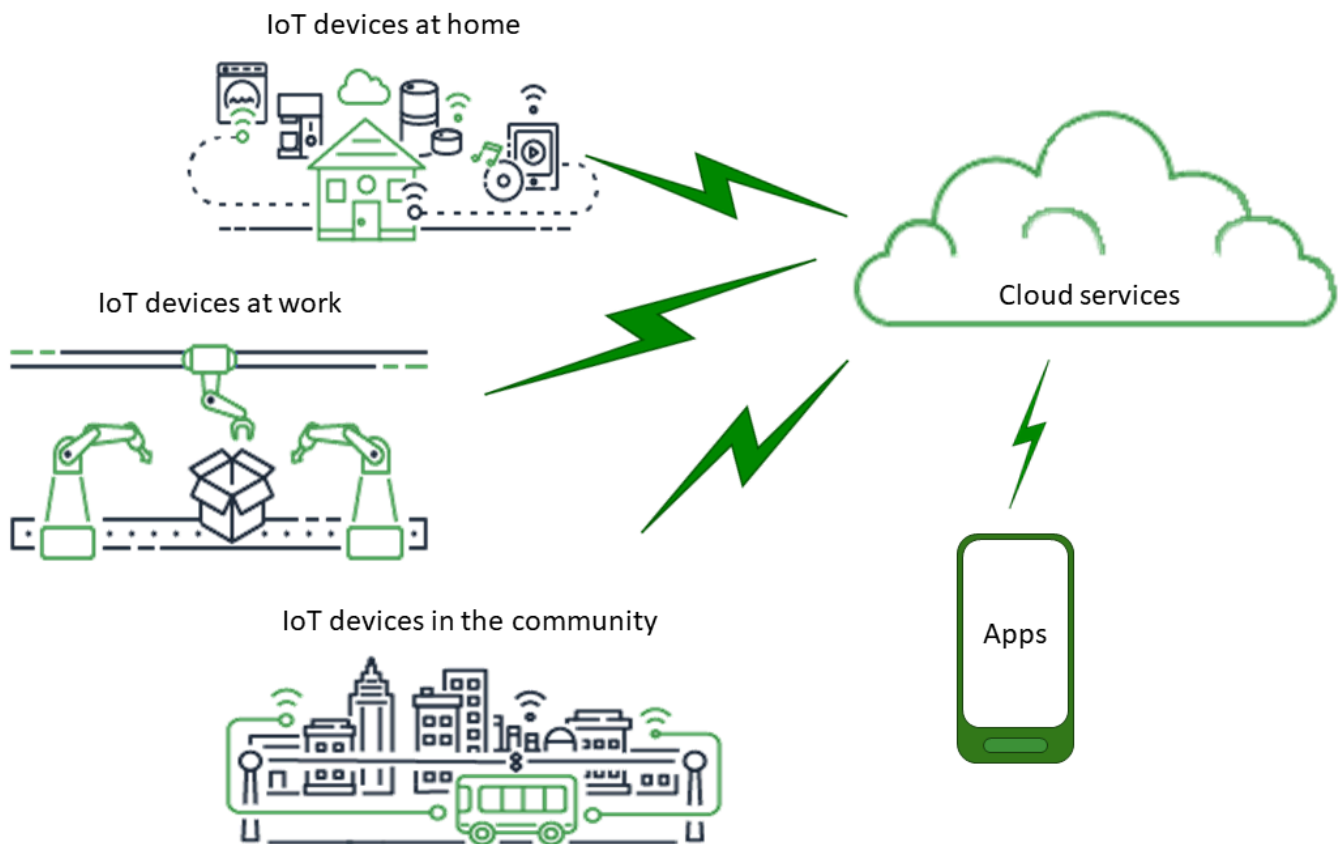
Untuk daftar solusi untuk kasus penggunaan industri, konsumen, dan komersial, lihat [Repositori AWS IoT Solusi](#).

Bagaimana cara AWS IoT kerja

AWS IoT menyediakan layanan cloud dan dukungan perangkat yang dapat Anda gunakan untuk mengimplementasikan solusi IoT. AWS menyediakan banyak layanan cloud untuk mendukung aplikasi berbasis IoT. Jadi untuk membantu Anda memahami dari mana harus memulai, bagian ini menyediakan diagram dan definisi konsep-konsep penting untuk memperkenalkan Anda ke alam semesta IoT.

Alam semesta IoT

Secara umum, Internet of Things (IoT) terdiri dari komponen-komponen kunci yang ditunjukkan dalam diagram ini.



Aplikasi

Aplikasi memberi pengguna akhir akses ke perangkat IoT dan fitur yang disediakan oleh layanan cloud yang terhubung dengan perangkat tersebut.

Layanan cloud

Layanan cloud didistribusikan, penyimpanan data skala besar dan layanan pemrosesan yang terhubung ke internet. Contohnya termasuk:

- Koneksi IoT dan layanan manajemen

AWS IoT adalah contoh koneksi IoT dan layanan manajemen.

- Layanan komputasi, seperti Amazon Elastic Compute Cloud dan AWS Lambda
- Layanan database, seperti Amazon DynamoDB

Komunikasi

Perangkat berkomunikasi dengan layanan cloud dengan menggunakan berbagai teknologi dan protokol. Contohnya termasuk:

- Wi-Fi/Internet Broadband
- Broadband data seluler
- Narrow-band data seluler
- Jaringan Area Luas Jarak Jauh (LoRaWAN)
- Komunikasi RF eksklusif

Perangkat

Perangkat adalah jenis perangkat keras yang mengelola antarmuka dan komunikasi. Perangkat biasanya terletak di dekat antarmuka dunia nyata yang mereka pantau dan kendalikan. Perangkat dapat mencakup sumber daya komputasi dan penyimpanan, seperti mikrokontroler, CPU, memori. Contohnya termasuk:

- Raspberry Pi
- Arduino
- Asisten antarmuka suara
- LoRaWAN dan perangkat
- Perangkat Amazon Sidewalk
- Perangkat IoT khusus

Antarmuka

Antarmuka adalah komponen yang menghubungkan perangkat ke dunia fisik.

- Antarmuka pengguna

Komponen yang memungkinkan perangkat dan pengguna untuk berkomunikasi satu sama lain.

- Antarmuka masukan

Memungkinkan pengguna untuk berkomunikasi dengan perangkat

Contoh: keypad, tombol

- Antarmuka keluaran

Mengaktifkan perangkat untuk berkomunikasi dengan pengguna

Contoh: Tampilan alfa-numerik, tampilan grafis, lampu indikator, bel alarm

- Sensor

Masukan komponen yang mengukur atau merasakan sesuatu di dunia luar dengan cara yang dipahami perangkat. Contohnya termasuk:

- Sensor suhu (mengubah suhu menjadi sinyal analog atau digital)
- Sensor kelembaban (mengubah kelembaban relatif menjadi sinyal analog atau digital)
- Konverter analog ke digital (mengubah tegangan analog ke nilai numerik)
- Unit pengukuran jarak ultrasonik (mengubah jarak ke nilai numerik)
- Sensor optik (mengubah tingkat cahaya ke nilai numerik)
- Kamera (mengubah data gambar menjadi data digital)

- Aktuator

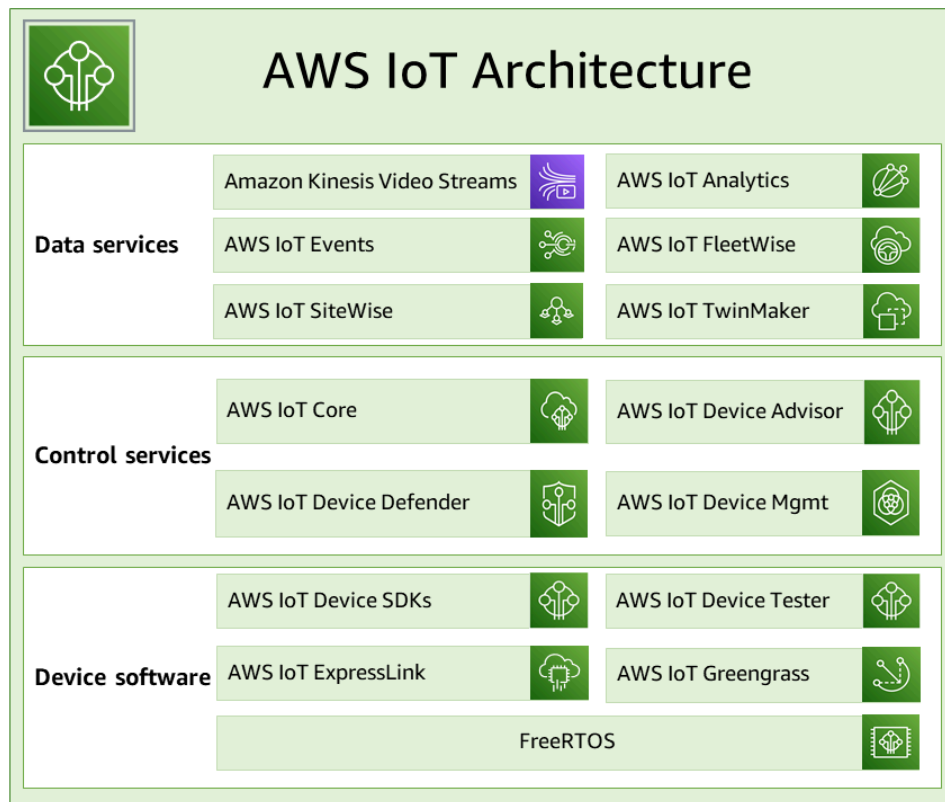
Komponen keluaran yang dapat digunakan perangkat untuk mengontrol sesuatu di dunia luar.

Contohnya termasuk:

- Motor stepper (ubah sinyal listrik menjadi gerakan)
- Relay (mengontrol tegangan dan arus listrik tinggi)

AWS IoT ikhtisar layanan

Di dunia IoT, AWS IoT menyediakan layanan yang mendukung perangkat yang berinteraksi dengan dunia dan data yang lewat di antara mereka dan. AWS IoT AWS IoT terdiri dari layanan yang ditampilkan dalam ilustrasi ini untuk mendukung solusi IoT Anda.



AWS IoT perangkat lunak

AWS IoT menyediakan perangkat lunak ini untuk mendukung perangkat IoT Anda.

AWS IoT SDK perangkat

[SDK AWS IoT Perangkat dan Seluler](#) membantu Anda menghubungkan perangkat secara efisien. AWS IoT SDK AWS IoT Perangkat dan Seluler mencakup pustaka sumber terbuka, panduan pengembang dengan sampel, dan panduan porting sehingga Anda dapat membangun produk atau solusi IoT inovatif pada platform perangkat keras pilihan Anda.

AWS IoT Device Tester

[AWS IoT Device Tester](#) untuk FreeRTOS AWS IoT Greengrass dan merupakan alat otomatisasi uji untuk mikrokontroler. AWS IoT Device Tester menguji perangkat Anda untuk menentukan apakah itu akan menjalankan FreeRTOS AWS IoT Greengrass atau dan beroperasi dengan layanan.

AWS IoT

AWS IoT ExpressLink

AWS IoT ExpressLink memberdayakan berbagai modul perangkat keras yang dikembangkan dan ditawarkan oleh [AWS Mitra](#). Modul konektivitas termasuk perangkat lunak AWS yang divalidasi,

membuatnya lebih cepat dan lebih mudah bagi Anda untuk menghubungkan perangkat dengan aman ke cloud dan terintegrasi dengan mulus dengan berbagai layanan. AWS Untuk informasi selengkapnya, kunjungi halaman [AWS IoT ExpressLink](#)ikhtisar atau lihat [Panduan AWS IoT ExpressLink Programmer](#).

AWS IoT Greengrass

[AWS IoT Greengrass](#)meluas AWS IoT ke perangkat edge sehingga mereka dapat bertindak secara lokal pada data yang mereka hasilkan, menjalankan prediksi berdasarkan model pembelajaran mesin, dan memfilter dan mengumpulkan data perangkat. AWS IoT Greengrass memungkinkan perangkat Anda mengumpulkan dan menganalisis data lebih dekat ke tempat data tersebut dihasilkan, bereaksi secara mandiri terhadap peristiwa lokal, dan berkomunikasi secara aman dengan perangkat lain di jaringan lokal. Anda dapat menggunakan AWS IoT Greengrass untuk membangun aplikasi edge menggunakan modul perangkat lunak pra-bangun, yang disebut komponen, yang dapat menghubungkan perangkat edge Anda ke AWS layanan atau layanan pihak ketiga.

FreeRTOS

[FreeRTOS](#) adalah sistem operasi open source dan real-time untuk mikrokontroler yang memungkinkan Anda memasukkan perangkat edge kecil berdaya rendah dalam solusi IoT Anda. FreeRTOS mencakup kernel dan serangkaian pustaka perangkat lunak yang terus berkembang yang mendukung banyak aplikasi. Sistem FreeRTOS dapat menghubungkan perangkat kecil dan berdaya rendah Anda dengan aman dan mendukung perangkat edge yang [AWS IoT](#)lebih kuat berjalan. [AWS IoT Greengrass](#)

AWS IoT layanan kontrol

Connect ke AWS IoT layanan berikut untuk mengelola perangkat dalam solusi IoT Anda.

AWS IoT Core

[AWS IoT Core](#)adalah layanan cloud terkelola yang memungkinkan perangkat yang terhubung berinteraksi secara aman dengan aplikasi cloud dan perangkat lain. AWS IoT Core dapat mendukung banyak perangkat dan pesan, dan dapat memproses dan merutekan pesan tersebut ke AWS IoT titik akhir dan perangkat lain. Dengan AWS IoT Core, aplikasi Anda dapat berinteraksi dengan semua perangkat Anda bahkan ketika mereka tidak terhubung.

AWS IoT Core Penasihat Perangkat

[AWS IoT Core Device Advisor](#) adalah kemampuan pengujian berbasis cloud yang dikelola sepenuhnya untuk memvalidasi perangkat IoT selama pengembangan perangkat lunak perangkat. Device Advisor menyediakan pengujian bawaan yang dapat Anda gunakan untuk memvalidasi perangkat IoT untuk konektivitas yang andal dan aman AWS IoT Core dengan, sebelum menerapkan perangkat ke produksi.

AWS IoT Pembela Perangkat

[AWS IoT Device Defender](#) membantu Anda mengamankan armada perangkat IoT Anda. AWS IoT Device Defender terus mengaudit konfigurasi IoT Anda untuk memastikan bahwa konfigurasi tersebut tidak menyimpang dari praktik terbaik keamanan. AWS IoT Device Defender mengirimkan peringatan ketika mendeteksi celah dalam konfigurasi IoT Anda yang mungkin menimbulkan risiko keamanan, seperti sertifikat identitas yang dibagikan di beberapa perangkat atau perangkat dengan sertifikat identitas yang dicabut yang mencoba terhubung. [AWS IoT Core](#)

AWS IoT Manajemen Perangkat

AWS IoT Layanan [Manajemen Perangkat](#) membantu Anda melacak, memantau, dan mengelola sejumlah besar perangkat yang terhubung yang membentuk armada perangkat Anda. AWS IoT Layanan Manajemen Perangkat membantu Anda memastikan bahwa perangkat IoT Anda berfungsi dengan baik dan aman setelah digunakan. Mereka juga menyediakan tunneling aman untuk mengakses perangkat Anda, memantau kesehatan mereka, mendeteksi dan memecahkan masalah dari jarak jauh, serta layanan untuk mengelola perangkat lunak perangkat dan pembaruan firmware.

AWS IoT layanan data

Analisis data dari perangkat dalam solusi IoT Anda dan lakukan tindakan yang tepat dengan menggunakan layanan berikut AWS IoT .

Amazon Kinesis Video Streams

[Amazon Kinesis Video](#) Streams memungkinkan Anda melakukan streaming video langsung dari perangkat AWS ke Cloud, yang disimpan, dienkripsi, dan diindeks dengan tahan lama, memungkinkan Anda mengakses data melalui API. easy-to-use Anda dapat menggunakan Amazon Kinesis Video Streams untuk menangkap sejumlah besar data video langsung dari jutaan sumber, termasuk ponsel cerdas, kamera keamanan, webcam, kamera yang disematkan di mobil, drone, dan sumber lainnya. Amazon Kinesis Video Streams memungkinkan Anda memutar video

untuk dilihat langsung dan sesuai permintaan, dan dengan cepat membangun aplikasi yang memanfaatkan visi komputer dan analitik video melalui integrasi dengan Amazon Rekognition Video, dan pustaka untuk kerangka kerja ML. Anda juga dapat mengirim data serial waktu non-video seperti data audio, citra termal, data kedalaman, data RADAR, dan banyak lagi.

Amazon Kinesis Video Streams dengan WebRTC

[Amazon Kinesis Video Streams](#) dengan WebRTC menyediakan implementasi WebRTC yang sesuai standar sebagai kemampuan yang dikelola sepenuhnya. Anda dapat menggunakan Amazon Kinesis Video Streams dengan WebRTC untuk media streaming langsung dengan aman atau melakukan interaksi audio atau video dua arah antara perangkat IoT kamera apa pun dan pemutar seluler atau web yang sesuai dengan WebRTC. Sebagai kemampuan yang dikelola sepenuhnya, Anda tidak perlu membangun, mengoperasikan, atau menskalakan infrastruktur cloud terkait WebRTC apa pun, seperti server pensinyalan atau relai media untuk mengalirkan media secara aman di seluruh aplikasi dan perangkat. Menggunakan Amazon Kinesis Video Streams dengan WebRTC, Anda dapat dengan mudah membangun peer-to-peer aplikasi untuk streaming media langsung, atau interaktivitas audio atau video real-time antara perangkat IoT kamera, browser web, dan perangkat seluler untuk berbagai kasus penggunaan.

AWS IoT Analitik

[AWS IoT Analytics](#) memungkinkan Anda menjalankan dan mengoperasikan analitik canggih secara efisien pada volume besar data IoT yang tidak terstruktur. AWS IoT Analytics mengotomatiskan setiap langkah sulit yang diperlukan untuk menganalisis data dari perangkat IoT. AWS IoT Analytics memfilter, mengubah, dan memperkaya data IoT sebelum menyimpannya di penyimpanan data deret waktu untuk dianalisis. Anda dapat menganalisis data Anda dengan menjalankan kueri satu kali atau terjadwal menggunakan mesin kueri SQL bawaan atau pembelajaran mesin.

AWS IoT Acara

[AWS IoT Peristiwa](#) mendeteksi dan merespons peristiwa dari sensor dan aplikasi IoT. Peristiwa adalah pola data yang mengidentifikasi keadaan yang lebih rumit dari yang diharapkan, seperti detektor gerakan yang menggunakan sinyal gerakan untuk mengaktifkan lampu dan kamera keamanan. AWS IoT Acara terus memantau data dari beberapa sensor dan aplikasi IoT, dan terintegrasi dengan layanan lain, seperti AWS IoT Core IoT, SiteWise DynamoDB, dan lainnya untuk memungkinkan deteksi dini dan wawasan unik.

AWS IoT FleetWise

[AWS IoT FleetWise](#) adalah layanan terkelola yang dapat Anda gunakan untuk mengumpulkan dan mentransfer data kendaraan ke cloud dalam waktu hampir nyata. Dengan AWS IoT

FleetWise, Anda dapat dengan mudah mengumpulkan dan mengatur data dari kendaraan yang menggunakan protokol dan format data yang berbeda. AWS IoT FleetWise membantu mengubah pesan tingkat rendah menjadi nilai yang dapat dibaca manusia dan menstandarisasi format data di cloud untuk analisis data. Anda juga dapat menentukan skema pengumpulan data untuk mengontrol data apa yang akan dikumpulkan di kendaraan dan kapan harus mentransfernya ke cloud.

AWS IoT SiteWise

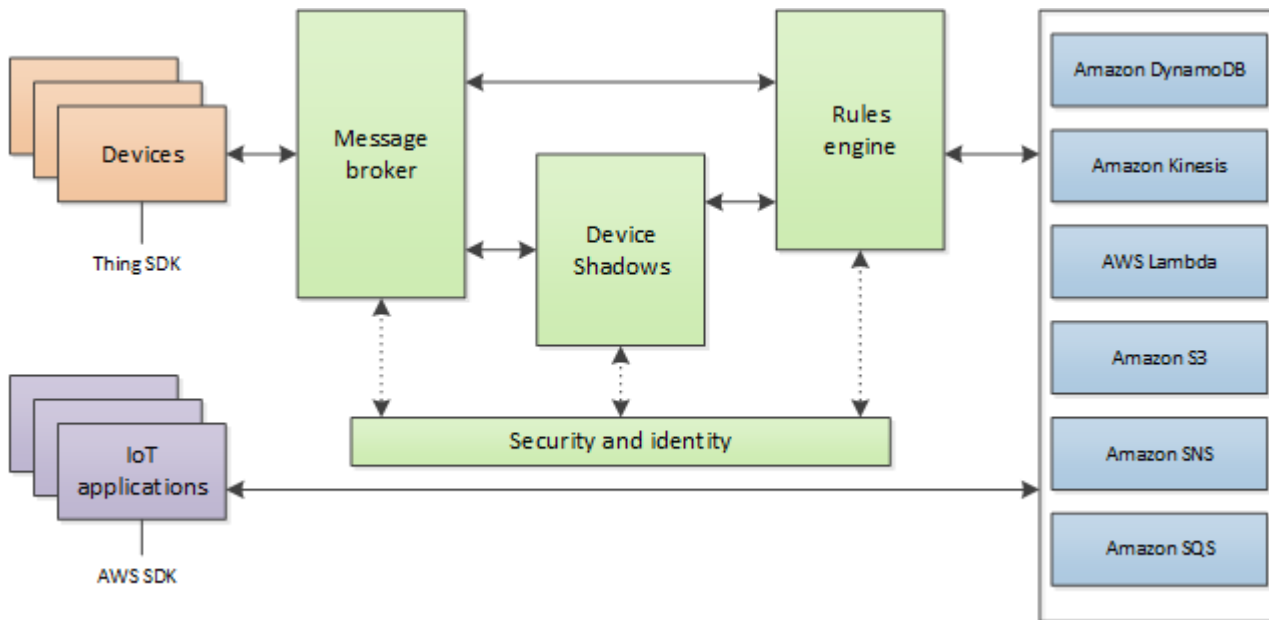
[AWS IoT SiteWise](#) mengumpulkan, menyimpan, mengatur, dan memantau data yang diteruskan dari peralatan industri oleh pesan MQTT atau API dalam skala besar dengan menyediakan perangkat lunak yang berjalan pada gateway di fasilitas Anda. Gateway terhubung dengan aman ke server data lokal Anda dan mengotomatiskan proses pengumpulan dan pengorganisasian data dan mengirimkannya ke Cloud. AWS

AWS IoT TwinMaker

[AWS IoT TwinMaker](#) membangun kembar digital operasional sistem fisik dan digital. AWS IoT TwinMaker menciptakan visualisasi digital menggunakan pengukuran dan analisis dari berbagai sensor dunia nyata, kamera, dan aplikasi perusahaan untuk membantu Anda melacak pabrik fisik, bangunan, atau pabrik industri Anda. Anda dapat menggunakan data dunia nyata untuk memantau operasi, mendiagnosis dan memperbaiki kesalahan, dan mengoptimalkan operasi.

AWS IoT Core layanan

AWS IoT Core menyediakan layanan yang menghubungkan perangkat IoT Anda ke AWS Cloud sehingga layanan dan aplikasi cloud lainnya dapat berinteraksi dengan perangkat Anda yang terhubung ke internet.



Bagian selanjutnya menjelaskan setiap AWS IoT Core layanan yang ditunjukkan dalam ilustrasi.

AWS IoT Core layanan pesan

Layanan AWS IoT Core konektivitas menyediakan komunikasi yang aman dengan perangkat IoT dan mengelola pesan yang lewat di antara mereka dan AWS IoT

Gateway perangkat

Memungkinkan perangkat untuk berkomunikasi dengan AWS IoT aman dan efisien. Komunikasi perangkat dijamin dengan protokol aman yang menggunakan sertifikat X.509.

Broker pesan

Menyediakan mekanisme yang aman untuk perangkat dan AWS IoT aplikasi untuk mempublikasikan dan menerima pesan dari satu sama lain. Anda dapat menggunakan protokol MQTT secara langsung atau MQTT untuk mempublikasikan dan berlangganan. WebSocket Untuk informasi selengkapnya tentang protokol yang AWS IoT mendukung, lihat [the section called “Protokol komunikasi perangkat”](#) Perangkat dan klien juga dapat menggunakan antarmuka HTTP REST untuk mempublikasikan data ke broker pesan.

Broker pesan mendistribusikan data perangkat ke perangkat yang telah berlangganan dan AWS IoT Core layanan lain, seperti layanan Device Shadow dan mesin aturan.

AWS IoT Core untuk LoRa WAN

AWS IoT Core untuk LoRa WAN memungkinkan untuk mengatur jaringan LoRa WAN pribadi dengan menghubungkan perangkat LoRa WAN dan gateway Anda AWS tanpa perlu mengembangkan dan mengoperasikan LoRa WAN Network Server (LNS). Pesan yang diterima dari perangkat LoRa WAN dikirim ke mesin aturan di mana mereka dapat diformat dan dikirim ke AWS IoT layanan lain.

Aturan mesin

Mesin Aturan menghubungkan data dari broker pesan ke AWS IoT layanan lain untuk penyimpanan dan pemrosesan tambahan. Misalnya, Anda dapat menyisipkan, memperbarui, atau menanyakan tabel DynamoDB atau menjalankan fungsi Lambda berdasarkan ekspresi yang Anda tentukan di mesin Aturan. Anda dapat menggunakan bahasa berbasis SQL untuk memilih data dari muatan pesan, lalu memproses dan mengirim data ke layanan lain, seperti Amazon Simple Storage Service (Amazon S3), Amazon DynamoDB, dan AWS Lambda. Anda juga dapat membuat aturan yang menerbitkan ulang pesan ke broker pesan dan ke pelanggan lain. Untuk informasi selengkapnya, lihat [Aturan untuk AWS IoT](#).

AWS IoT Core layanan kontrol

Layanan AWS IoT Core kontrol menyediakan fitur keamanan, manajemen, dan pendaftaran perangkat.

Layanan Otentikasi Kustom

Anda dapat menentukan otorisasi khusus yang memungkinkan Anda mengelola strategi otentikasi dan otorisasi Anda sendiri menggunakan layanan otentikasi khusus dan fungsi Lambda. Authorizer khusus memungkinkan AWS IoT untuk mengautentikasi perangkat Anda dan mengotorisasi operasi menggunakan otentikasi token pembawa dan strategi otorisasi.

Authorizer khusus dapat menerapkan berbagai strategi otentikasi; misalnya, verifikasi Token Web JSON atau callout penyedia OAuth. Mereka harus mengembalikan dokumen kebijakan yang digunakan oleh gateway perangkat untuk mengotorisasi operasi MQTT. Untuk informasi selengkapnya, lihat [Otentikasi dan otorisasi khusus](#).

Layanan Penyediaan Perangkat

Memungkinkan Anda menyediakan perangkat menggunakan templat yang menjelaskan sumber daya yang diperlukan untuk perangkat Anda: objek benda, sertifikat, dan satu atau beberapa kebijakan. Objek benda adalah entri dalam registri yang berisi atribut yang menggambarkan

perangkat. Perangkat menggunakan sertifikat untuk mengautentikasi dengan AWS IoT. Kebijakan menentukan operasi mana yang dapat dilakukan perangkat AWS IoT.

Template berisi variabel yang digantikan oleh nilai-nilai dalam kamus (peta). Anda dapat menggunakan template yang sama untuk menyediakan beberapa perangkat hanya dengan meneruskan nilai yang berbeda untuk variabel template dalam kamus. Untuk informasi selengkapnya, lihat [Penyediaan perangkat](#).

Registri grup

Grup memungkinkan Anda mengelola beberapa perangkat sekaligus dengan mengkategorikannya ke dalam grup. Grup juga dapat berisi grup — Anda dapat membangun hierarki grup. Tindakan apa pun yang Anda lakukan pada grup induk akan berlaku untuk grup turunannya. Tindakan yang sama juga berlaku untuk semua perangkat di grup induk dan semua perangkat di grup anak. Izin yang diberikan kepada grup akan berlaku untuk semua perangkat di grup dan di semua grup turunannya. Untuk informasi selengkapnya, lihat [Mengelola perangkat dengan AWS IoT](#).

Layanan Lowongan Kerja

Memungkinkan Anda menentukan satu set operasi jarak jauh yang dikirim ke dan dijalankan pada satu atau beberapa perangkat yang terhubung AWS IoT. Misalnya, Anda dapat menentukan pekerjaan yang menginstruksikan satu set perangkat untuk mengunduh dan menginstal pembaruan aplikasi atau firmware, reboot, memutar sertifikat, atau melakukan operasi pemecahan masalah jarak jauh.

Untuk membuat pekerjaan, Anda menentukan deskripsi operasi jarak jauh yang akan dilakukan dan daftar target yang harus melakukannya. Target dapat berupa perangkat individual, grup atau keduanya. Untuk informasi selengkapnya, lihat [Tugas](#).

Registri

Mengatur sumber daya yang terkait dengan setiap perangkat di AWS Cloud. Anda mendaftarkan perangkat Anda dan mengaitkan hingga tiga atribut khusus dengan masing-masing atribut. Anda juga dapat mengaitkan sertifikat dan ID klien MQTT dengan setiap perangkat untuk meningkatkan kemampuan Anda mengelola dan memecahkan masalah. Untuk informasi selengkapnya, lihat [Mengelola perangkat dengan AWS IoT](#).

Layanan Keamanan dan Identitas

Memberikan tanggung jawab bersama untuk keamanan di AWS Cloud. Perangkat Anda harus menjaga kredensialnya aman untuk mengirim data dengan aman ke broker pesan. Broker pesan

dan mesin aturan menggunakan fitur AWS keamanan untuk mengirim data dengan aman ke perangkat atau AWS layanan lain. Untuk informasi selengkapnya, lihat [Autentikasi](#).

AWS IoT Core layanan data

Layanan AWS IoT Core data membantu solusi IoT Anda memberikan pengalaman aplikasi yang andal bahkan dengan perangkat yang tidak selalu terhubung.

Bayangan perangkat

Dokumen JSON yang digunakan untuk menyimpan dan mengambil informasi status saat ini untuk perangkat.

Layanan Bayangan Perangkat

Layanan Device Shadow mempertahankan status perangkat sehingga aplikasi dapat berkomunikasi dengan perangkat apakah perangkat sedang online atau tidak. Saat perangkat offline, layanan Device Shadow mengelola datanya untuk aplikasi yang terhubung. Ketika perangkat terhubung kembali, itu menyinkronkan statusnya dengan bayangannya di layanan Device Shadow. Perangkat Anda juga dapat mempublikasikan statusnya saat ini ke bayangan untuk digunakan oleh aplikasi atau perangkat lain yang mungkin tidak terhubung sepanjang waktu. Untuk informasi selengkapnya, lihat [AWS IoT Layanan Device Shadow](#).

AWS IoT Core layanan dukungan

Integrasi Trotoar Amazon untuk AWS IoT Core

[Amazon Sidewalk](#) adalah jaringan bersama yang meningkatkan opsi konektivitas untuk membantu perangkat bekerja sama dengan lebih baik. Amazon Sidewalk mendukung berbagai perangkat pelanggan seperti yang menemukan hewan peliharaan atau barang berharga, yang menyediakan keamanan rumah pintar dan kontrol pencahayaan, dan yang menyediakan diagnostik jarak jauh untuk peralatan dan peralatan. Amazon Sidewalk Integration untuk AWS IoT Core memungkinkan produsen perangkat untuk menambahkan armada perangkat Trotoar mereka ke Cloud. AWS IoT

Untuk informasi selengkapnya, lihat [AWS IoT Core untuk Amazon Sidewalk](#).

Pelajari lebih lanjut tentang AWS IoT

Topik ini membantu Anda membiasakan diri dengan dunia AWS IoT. Anda bisa mendapatkan informasi umum tentang bagaimana solusi IoT diterapkan dalam berbagai kasus penggunaan, sumber daya pelatihan, tautan ke media sosial untuk AWS IoT dan semua AWS layanan lainnya, dan daftar layanan dan protokol komunikasi yang digunakan. AWS IoT

Sumber daya pelatihan untuk AWS IoT

Kami menyediakan kursus pelatihan ini untuk membantu Anda mempelajari AWS IoT dan bagaimana menerapkannya pada desain solusi Anda.

- [Pengantar AWS IoT](#)

Ikhtisar video AWS IoT dan layanan intinya.

- [Menyelami AWS IoT Otentikasi dan Otorisasi](#)

Kursus lanjutan yang mengeksplorasi konsep AWS IoT otentikasi dan otorisasi. Anda akan belajar cara mengautentikasi dan mengotorisasi klien untuk mengakses bidang AWS IoT kontrol dan API bidang data.

- [Seri Yayasan Internet of Things](#)

Jalur pembelajaran modul IoT eLearning pada berbagai teknologi dan fitur IoT.

AWS IoT sumber daya dan panduan

Ini adalah sumber daya teknis yang mendalam tentang aspek-aspek tertentu dari. AWS IoT

- [Lensa IoT — Kerangka Kerja yang Dirancang dengan Baik AWS IoT](#)

Dokumen yang menjelaskan praktik terbaik untuk merancang aplikasi IoT Anda. AWS

- [Merancang Topik MQTT untuk AWS IoT Core](#)

Whitepaper yang menjelaskan praktik terbaik untuk merancang topik MQTT AWS IoT Core dan memanfaatkan fitur dengan MQTT. AWS IoT Core

- [Abstrak dan pengantar](#)

Dokumen PDF yang menjelaskan berbagai cara yang AWS IoT menyediakan untuk menyediakan armada perangkat yang besar.

- [AWS IoT Core Penasihat Perangkat](#)

AWS IoT Core Device Advisor menyediakan pengujian bawaan yang dapat Anda gunakan untuk memvalidasi perangkat IoT untuk praktik terbaik konektivitas yang andal dan aman AWS IoT Core dengan, sebelum menerapkan perangkat ke produksi.

- [AWS IoT Sumber](#)

Sumber daya khusus IoT, seperti Panduan Teknis, Arsitektur Referensi, eBook, dan posting blog yang dikuratori, disajikan dalam indeks yang dapat dicari.

- [Atlas IoT](#)

Ikhtisar tentang cara mengatasi masalah desain IoT yang umum. IoT Atlas memberikan pandangan mendalam tentang tantangan desain yang mungkin Anda hadapi saat mengembangkan solusi IoT Anda.

- [AWS Whitepaper & Panduan](#)

Koleksi whitepaper dan panduan kami saat ini dan teknologi AWS IoT lainnya AWS .

AWS IoT di media sosial

Saluran media sosial ini memberikan informasi tentang AWS IoT dan topik AWS terkait.

- [Internet of Things on AWS IoT — Blog Resmi](#)
- [AWS IoT video di saluran Amazon Web Services di YouTube](#)

Akun media sosial ini mencakup semua AWS layanan, termasuk AWS IoT

- [Saluran Amazon Web Services di YouTube](#)
- [Amazon Web Services di Twitter](#)
- [Amazon Web Services di Facebook](#)
- [Amazon Web Services di Instagram](#)
- [Amazon Web Services di LinkedIn](#)

AWS layanan yang digunakan oleh mesin AWS IoT Core aturan

Mesin AWS IoT Core aturan dapat terhubung ke AWS layanan ini.

- [Amazon DynamoDB](#)

Amazon DynamoDB adalah layanan database NoSQL yang dapat diskalakan yang menyediakan kinerja database yang cepat dan dapat diprediksi.

- [Amazon Kinesis](#)

Amazon Kinesis memudahkan pengumpulan, proses, dan analisis data streaming real-time sehingga Anda bisa mendapatkan wawasan tepat waktu dan bereaksi cepat terhadap informasi baru. Amazon Kinesis dapat menyerap data real-time seperti video, audio, log aplikasi, clickstream situs web, dan data telemetri IoT untuk pembelajaran mesin, analitik, dan aplikasi lainnya.

- [AWS Lambda](#)

AWS Lambda memungkinkan Anda menjalankan kode tanpa menyediakan atau mengelola server. Anda dapat mengatur kode Anda untuk memicu secara otomatis dari AWS IoT data dan peristiwa atau memanggilnya langsung dari web atau aplikasi seluler.

- [Layanan Penyimpanan Sederhana Amazon](#)

Amazon Simple Storage Service (Amazon S3) dapat menyimpan dan mengambil sejumlah data kapan saja, dari mana saja di web. AWS IoT aturan dapat mengirim data ke Amazon S3 untuk penyimpanan.

- [Layanan Pemberitahuan Sederhana Amazon](#)

Amazon Simple Notification Service (Amazon SNS) adalah layanan web yang memungkinkan aplikasi, pengguna akhir, dan perangkat untuk mengirim dan menerima notifikasi dari cloud.

- [Layanan Antrian Sederhana Amazon](#)

Amazon Simple Queue Service (Amazon Simple Queue Service) adalah layanan antrian pesan yang memisahkan dan menskalakan layanan mikro, sistem terdistribusi, dan aplikasi tanpa server.

- [OpenSearch Layanan Amazon](#)

Amazon OpenSearch Service (OpenSearch Service) adalah layanan terkelola yang memudahkan penerapan, pengoperasian, dan skala OpenSearch, mesin pencari dan analitik sumber terbuka yang populer.

- [Amazon SageMaker](#)

Amazon SageMaker dapat membuat model pembelajaran mesin (ML) dengan menemukan pola dalam data IoT Anda. Layanan ini menggunakan model ini untuk memproses data baru dan menghasilkan prediksi untuk aplikasi Anda.

- [Amazon CloudWatch](#)

Amazon CloudWatch menyediakan solusi pemantauan yang andal, terukur, dan fleksibel untuk membantu menyiapkan, mengelola, dan menskalakan sistem dan infrastruktur pemantauan Anda sendiri.

Protokol komunikasi yang didukung oleh AWS IoT Core

Topik-topik ini memberikan informasi lebih lanjut tentang protokol komunikasi yang digunakan oleh AWS IoT. Untuk informasi selengkapnya tentang protokol yang digunakan oleh AWS IoT dan menghubungkan perangkat dan layanan AWS IoT, lihat. [Menghubungkan ke AWS IoT Core](#)

- [MQTT \(Transportasi Telemetri Antrian Pesan\)](#)

Halaman beranda situs MQTT.org tempat Anda dapat menemukan spesifikasi protokol MQTT. Untuk informasi selengkapnya tentang cara AWS IoT mendukung MQTT, lihat. [MQTT](#)

- [HTTPS \(Protokol Transfer Hypertext - Aman\)](#)

Perangkat dan aplikasi dapat mengakses AWS IoT layanan dengan menggunakan HTTPS.

- [LoRaWAN \(Jaringan Area Luas Jarak Jauh\)](#)

LoRaPerangkat WAN dan gateway dapat terhubung AWS IoT Core dengan menggunakan AWS IoT Core WAN. LoRa

- [TLS \(Keamanan Lapisan Transportasi\) v1.3](#)

Spesifikasi TLS v1.3 (RFC 5246). AWS IoT menggunakan TLS v1.3 untuk membangun koneksi aman antara perangkat dan perangkat. AWS IoT

Yang baru diAWS IoT konsol

Kami sedang dalam proses memperbarui antarmuka penggunaAWS IoT konsol ke pengalaman baru. Kami memperbarui antarmuka pengguna secara bertahap, sehingga beberapa halaman di konsol akan memiliki pengalaman baru, beberapa mungkin memiliki pengalaman asli dan baru, dan beberapa mungkin hanya memiliki pengalaman asli.

Tabel ini menampilkan keadaan masing-masing area antarmuka penggunaAWS IoT konsol pada 27 Januari 2022.

AWS IoT status antarmuka pengguna konsol

Halaman konsol	Pengalaman asli	Pengalaman baru	Comments
Memantau	Tidak tersedia	Tersedia	
Aktivitas	Tidak tersedia	Tersedia	
Onboard - Memulai	Tidak tersedia	Tersedia	Tidak tersedia di Wilayah CN
Onboard - Templat penyediaan armada	Tersedia	Tersedia	
Mengelola - Hal-hal	Tersedia	Tersedia	
Mengelola - Jenis	Tersedia	Tersedia	
Kelola - Kelompok hal	Tersedia	Tersedia	
Kelola - Grup penagihan	Tersedia	Tersedia	
Kelola - Lowongan	Tersedia	Tersedia	
Mengelola - Template Job	Tidak tersedia	Tersedia	
Kelola - Terowongan	Tidak tersedia	Tersedia	
Fleet Hub - Memulai	Tidak tersedia	Tersedia	Tidak tersedia di semua Wilayah AWS
Armada Hub - Aplikasi	Tidak tersedia	Tersedia	Tidak tersedia di semua Wilayah AWS
Greengrass - Memulai	Tidak tersedia	Tersedia	Tidak tersedia di semua Wilayah AWS
Greengrass - Perangkat inti	Tidak tersedia	Tersedia	Tidak tersedia di semua Wilayah AWS

Halaman konsol	Pengalaman asli	Pengalaman baru	Comments
Greengrass - Komponen	Tidak tersedia	Tersedia	Tidak tersedia di semuaWilayah AWS
Greengrass - Penyebaran	Tidak tersedia	Tersedia	Tidak tersedia di semuaWilayah AWS
Greengrass - Klasik (V1)	Tersedia	Tersedia	
Konektivitas nirkabel - Intro	Tidak tersedia	Tersedia	Tidak tersedia di semuaWilayah AWS
Konektivitas nirkabel - Gateway	Tidak tersedia	Tersedia	Tidak tersedia di semuaWilayah AWS
Konektivitas nirkabel - Perangkat	Tidak tersedia	Tersedia	Tidak tersedia di semuaWilayah AWS
Konektivitas nirkabel - Profil	Tidak tersedia	Tersedia	Tidak tersedia di semuaWilayah AWS
Konektivitas nirkabel - Tujuan	Tidak tersedia	Tersedia	Tidak tersedia di semuaWilayah AWS
Aman - Sertifikat	Tersedia	Tersedia	
Aman - Kebijakan	Tersedia	Tersedia	
Aman - CA	Tersedia	Tersedia	
Aman - Alias Peran	Tersedia	Tersedia	
Aman - Pengotorisasi	Tersedia	Tersedia	
Mempertahankan - Intro	Tidak tersedia	Tersedia	

Halaman konsol	Pengalaman asli	Pengalaman baru	Comments
Mempertahankan - Audit	Tidak tersedia	Tersedia	
Mempertahankan - Deteksi	Tidak tersedia	Tersedia	
Mempertahankan - Tindakan mitigasi	Tidak tersedia	Tersedia	
Defend - Pengaturan	Tidak tersedia	Tersedia	
Act - Aturan	Tersedia	Tersedia	
Act - Tujuan	Tersedia	Tersedia	
Uji - Device Advisor	Tersedia	Tersedia	Tidak tersedia di semuaWilayah AWS
Uji - klien uji MQTT	Tersedia	Tersedia	
Perangkat Lunak	Tersedia	Tersedia	
Pengaturan	Tidak tersedia	Tersedia	
Belajar	Tersedia	Belum tersedia	

Legenda

Nilai status

- Tersedia

Pengalaman antarmuka pengguna ini dapat digunakan.

- Tidak tersedia

Pengalaman antarmuka pengguna ini tidak dapat digunakan.

- Belum tersedia

Pengalaman antarmuka pengguna baru sedang dikerjakan, tetapi belum siap.

- Sedang berlangsung

Pengalaman antarmuka pengguna sedang dalam proses diperbarui. Namun, beberapa halaman mungkin masih memiliki pengalaman pengguna asli.

Menggunakan AWS IoT dengan AWS SDK

AWS kit pengembangan perangkat lunak (SDK) tersedia untuk banyak bahasa pemrograman populer. Setiap SDK menyediakan API, contoh kode, dan dokumentasi yang memudahkan developer untuk membangun aplikasi dalam bahasa pilihan mereka.

Dokumentasi SDK	Contoh kode
AWS SDK for C++	AWS SDK for C++ contoh kode
AWS CLI	AWS CLI contoh kode
AWS SDK for Go	AWS SDK for Go contoh kode
AWS SDK for Java	AWS SDK for Java contoh kode
AWS SDK for JavaScript	AWS SDK for JavaScript contoh kode
AWS SDK for Kotlin	AWS SDK for Kotlin contoh kode
AWS SDK for .NET	AWS SDK for .NET contoh kode
AWS SDK for PHP	AWS SDK for PHP contoh kode
AWS Tools for PowerShell	Alat untuk contoh PowerShell kode
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) contoh kode
AWS SDK for Ruby	AWS SDK for Ruby contoh kode
AWS SDK for Rust	AWS SDK for Rust contoh kode
AWS SDK untuk SAP ABAP	AWS SDK untuk SAP ABAP contoh kode

Dokumentasi SDK	Contoh kode
AWS SDK for Swift	AWS SDK for Swift contoh kode

 **Ketersediaan contoh**

Tidak dapat menemukan apa yang Anda butuhkan? Minta contoh kode menggunakan tautan Berikan umpan balik di bagian bawah halaman ini.

Memulai dengan AWS IoT Core

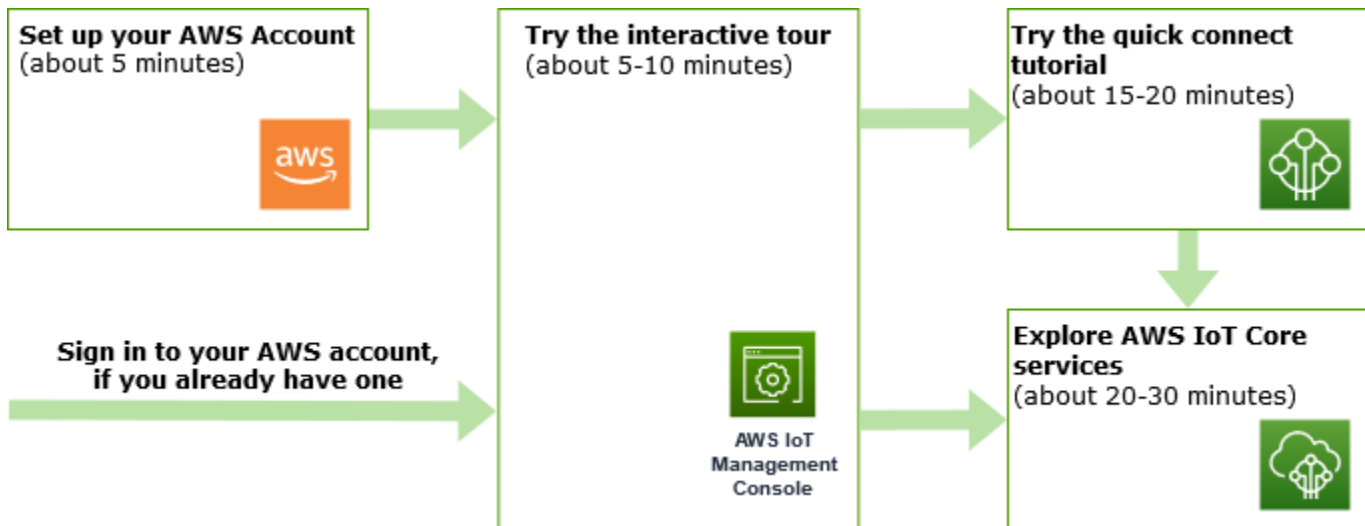
Apakah Anda baru mengenal IoT atau Anda memiliki pengalaman bertahun-tahun, sumber daya ini menyajikan AWS IoT konsep dan istilah yang akan membantu Anda mulai menggunakannya. AWS IoT

- Lihat ke dalam AWS IoT dan komponennya masuk [Bagaimana cara AWS IoT kerja](#).
- [Pelajari lebih lanjut AWS IoT](#) dari koleksi materi dan video pelatihan kami. Topik ini juga mencakup daftar layanan yang AWS IoT dapat terhubung, tautan media sosial, dan tautan ke spesifikasi protokol komunikasi.
- [the section called “Connect perangkat pertama Anda ke AWS IoT Core”](#).
- Kembangkan solusi IoT Anda dengan [Menghubungkan ke AWS IoT Core](#) dan jelajahi. [Tutorial AWS IoT](#)
- Uji dan validasi perangkat IoT Anda untuk komunikasi yang aman dan andal dengan menggunakan. [Penasihat Perangkat](#)
- Kelola solusi Anda dengan menggunakan layanan AWS IoT Core manajemen seperti [Pengeindeksan armada](#), [Tugas](#), dan [AWS IoT Device Defender](#).
- Analisis data dari perangkat Anda dengan menggunakan [AWS IoT layanan data](#).

Connect perangkat pertama Anda ke AWS IoT Core

AWS IoT Core layanan menghubungkan perangkat IoT ke AWS IoT layanan dan layanan lainnya AWS . AWS IoT Core termasuk gateway perangkat dan broker pesan, yang menghubungkan dan memproses pesan antara perangkat IoT Anda dan cloud.

Inilah cara Anda dapat memulai dengan AWS IoT Core dan AWS IoT.



Bagian ini menyajikan tur AWS IoT Core untuk memperkenalkan layanan utamanya dan memberikan beberapa contoh cara menghubungkan perangkat ke AWS IoT Core dan menyampaikan pesan di antara mereka. Meneruskan pesan antara perangkat dan cloud sangat penting untuk setiap solusi IoT dan bagaimana perangkat Anda dapat berinteraksi dengan layanan lain AWS .

- [Siapkan Anda Akun AWS](#)

Sebelum Anda dapat menggunakan AWS IoT layanan, Anda harus mengatur Akun AWS. Jika Anda sudah memiliki Akun AWS dan pengguna IAM untuk diri sendiri, Anda dapat menggunakannya dan melewati langkah ini.

- [Coba tutorial interaktif](#)

Demo ini paling baik jika Anda ingin melihat apa yang dapat dilakukan AWS IoT solusi dasar tanpa menghubungkan perangkat atau mengunduh perangkat lunak apa pun. Tutorial interaktif menyajikan solusi simulasi yang dibangun di atas AWS IoT Core layanan yang menggambarkan bagaimana mereka berinteraksi.

- [Coba tutorial koneksi cepat](#)

Tutorial ini paling baik jika Anda ingin memulai dengan cepat AWS IoT dan melihat cara kerjanya dalam skenario terbatas. Dalam tutorial ini, Anda akan memerlukan perangkat dan Anda akan menginstal beberapa AWS IoT perangkat lunak di dalamnya. Jika Anda tidak memiliki perangkat IoT, Anda dapat menggunakan komputer pribadi Windows, Linux, atau macOS Anda sebagai perangkat untuk tutorial ini. Jika Anda ingin mencoba AWS IoT, tetapi Anda tidak memiliki perangkat, coba opsi berikutnya.

- [Jelajahi AWS IoT Core layanan dengan tutorial langsung](#)

Tutorial ini terbaik untuk pengembang yang ingin memulai AWS IoT sehingga mereka dapat terus mengeksplorasi AWS IoT Core fitur lain seperti mesin aturan dan bayangan. Tutorial ini mengikuti proses yang mirip dengan tutorial koneksi cepat, tetapi memberikan rincian lebih lanjut pada setiap langkah untuk memungkinkan transisi yang lebih lancar ke tutorial yang lebih maju.

- [Lihat pesan MQTT dengan klien MQTT AWS IoT](#)

Pelajari cara menggunakan klien pengujian MQTT untuk menonton perangkat pertama Anda mempublikasikan pesan MQTT. AWS IoT Klien pengujian MQTT adalah alat yang berguna untuk memantau dan memecahkan masalah koneksi perangkat.

Note

Jika Anda ingin mencoba lebih dari satu tutorial memulai ini atau mengulangi tutorial yang sama, Anda harus menghapus objek benda yang Anda buat dari tutorial sebelumnya sebelum Anda memulai yang lain. Jika Anda tidak menghapus objek benda dari tutorial sebelumnya, Anda harus menggunakan nama hal yang berbeda untuk tutorial berikutnya. Ini karena nama benda harus unik di akun Anda dan Wilayah AWS.

Untuk informasi lebih lanjut tentang AWS IoT Core, lihat [Apa itu AWS IoT Core?](#)

Siapkan Anda Akun AWS

Sebelum Anda menggunakan AWS IoT Core untuk pertama kalinya, selesaikan tugas-tugas berikut:

Topik

- [Mendaftar untuk Akun AWS](#)
- [Buat pengguna dengan akses administratif](#)
- [Buka AWS IoT konsol](#)

Mendaftar untuk Akun AWS

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

Untuk mendaftar untuk Akun AWS

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

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

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

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

Buat pengguna dengan akses administratif

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

Amankan Anda Pengguna root akun AWS

1. Masuk ke [AWS Management Console](#) sebagai pemilik akun dengan memilih pengguna Root dan memasukkan alamat Akun AWS email Anda. Di laman berikutnya, masukkan kata sandi.

Untuk bantuan masuk dengan menggunakan pengguna root, lihat [Masuk sebagai pengguna root](#) di AWS Sign-In Panduan Pengguna.

2. Mengaktifkan autentikasi multi-faktor (MFA) untuk pengguna root Anda.

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

Buat pengguna dengan akses administratif

1. Aktifkan Pusat Identitas IAM.

Untuk mendapatkan petunjuk, silakan lihat [Mengaktifkan AWS IAM Identity Center](#) di Panduan Pengguna AWS IAM Identity Center .

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

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

Masuk sebagai pengguna dengan akses administratif

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

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

Tetapkan akses ke pengguna tambahan

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

Untuk petunjuknya, lihat [Membuat set izin](#) di Panduan AWS IAM Identity Center Pengguna.

2. Tetapkan pengguna ke grup, lalu tetapkan akses masuk tunggal ke grup.

Untuk petunjuk, lihat [Menambahkan grup](#) di Panduan AWS IAM Identity Center Pengguna.

- [Buka AWS IoT konsol](#)

Jika Anda sudah memiliki Akun AWS dan pengguna untuk diri sendiri, Anda dapat menggunakannya dan melompat ke depan [the section called “Buka AWS IoT konsol”](#).

Buka AWS IoT konsol

Sebagian besar topik berorientasi konsol di bagian ini dimulai dari konsol. AWS IoT Jika Anda belum masuk Akun AWS, masuk, lalu buka [AWS IoT konsol](#) dan lanjutkan ke bagian berikutnya untuk melanjutkan memulai AWS IoT.

Coba tutorial AWS IoT Core interaktif

Tutorial interaktif menunjukkan komponen solusi IoT sederhana yang dibangun di atasnya. AWS IoT Animasi tutorial menunjukkan bagaimana perangkat IoT berinteraksi dengan AWS IoT Core layanan. Topik ini memberikan pratinjau tutorial AWS IoT Core interaktif. Gambar di konsol menyertakan animasi yang tidak muncul dalam gambar dalam tutorial ini.

Untuk menjalankan demo, Anda harus terlebih dahulu [the section called “Siapkan Anda Akun AWS”](#). Tutorial, bagaimanapun, tidak memerlukan AWS IoT sumber daya, perangkat lunak tambahan, atau pengkodean apa pun.

Berharap untuk menghabiskan sekitar 5-10 menit pada demo ini. Memberi diri Anda 10 menit akan memungkinkan lebih banyak waktu untuk memahami setiap langkah.

Untuk menjalankan tutorial AWS IoT Core interaktif

1. Buka [AWS IoT halaman](#) beranda di AWS IoT konsol.

Di AWS IoT halaman beranda, di panel jendela Sumber pembelajaran, pilih Mulai tutorial.

The screenshot shows the AWS IoT console interface. The main heading is "AWS IoT Securely connect, test, and manage your IoT devices". Below this, there are sections for "How it works" (Connect, Test, Manage) and "Watch it work" (Interactive tutorial). On the right side, there are panels for "Get started with AWS IoT", "Pricing", "Learning resources", and "More resources". A red box highlights the "AWS IoT interactive tutorial" link in the Learning resources section, with a red arrow pointing to it.

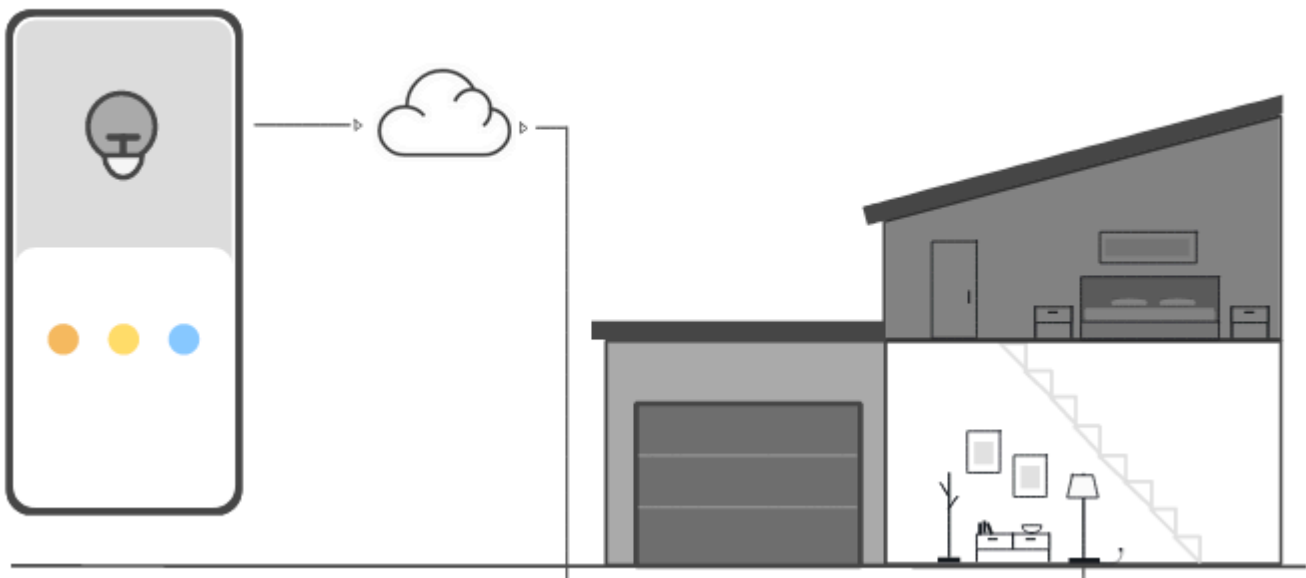
2. Di halaman Tutorial AWS IoT Konsol, tinjau bagian tutorial dan pilih bagian Mulai saat Anda siap untuk melanjutkan.

Bagian berikut menjelaskan bagaimana Tutorial AWS IoT Konsol menyajikan AWS IoT Core fitur-fitur ini:

- [Menghubungkan perangkat IoT](#)
- [Menyimpan status perangkat offline](#)
- [Merutekan data perangkat ke layanan](#)

Menghubungkan perangkat IoT

Pelajari cara perangkat IoT berkomunikasi. AWS IoT Core

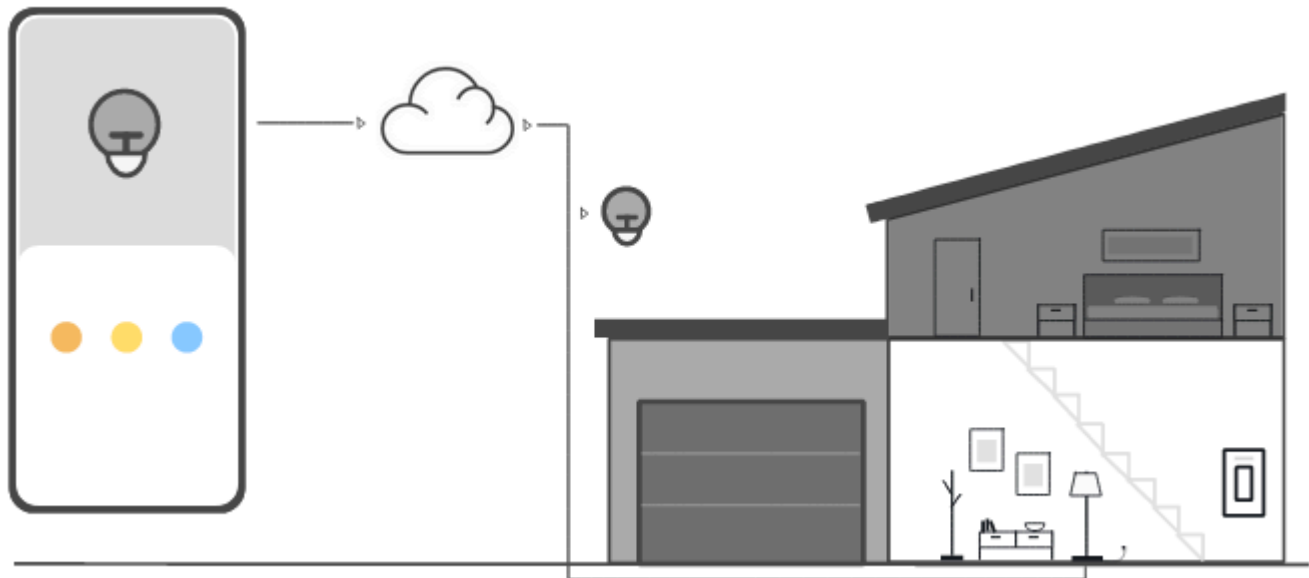


Animasi pada langkah ini menunjukkan bagaimana dua perangkat, perangkat kontrol di sebelah kiri dan lampu pintar di rumah di sebelah kanan, terhubung dan berkomunikasi dengan AWS IoT Core di awan. Animasi menunjukkan perangkat berkomunikasi dengan AWS IoT Core dan bereaksi terhadap pesan yang mereka terima.

Untuk informasi selengkapnya tentang menghubungkan perangkat AWS IoT Core, lihat [Menghubungkan ke AWS IoT Core](#).

Menyimpan status perangkat offline

Pelajari cara AWS IoT Core menyimpan status perangkat saat perangkat atau aplikasi sedang offline.



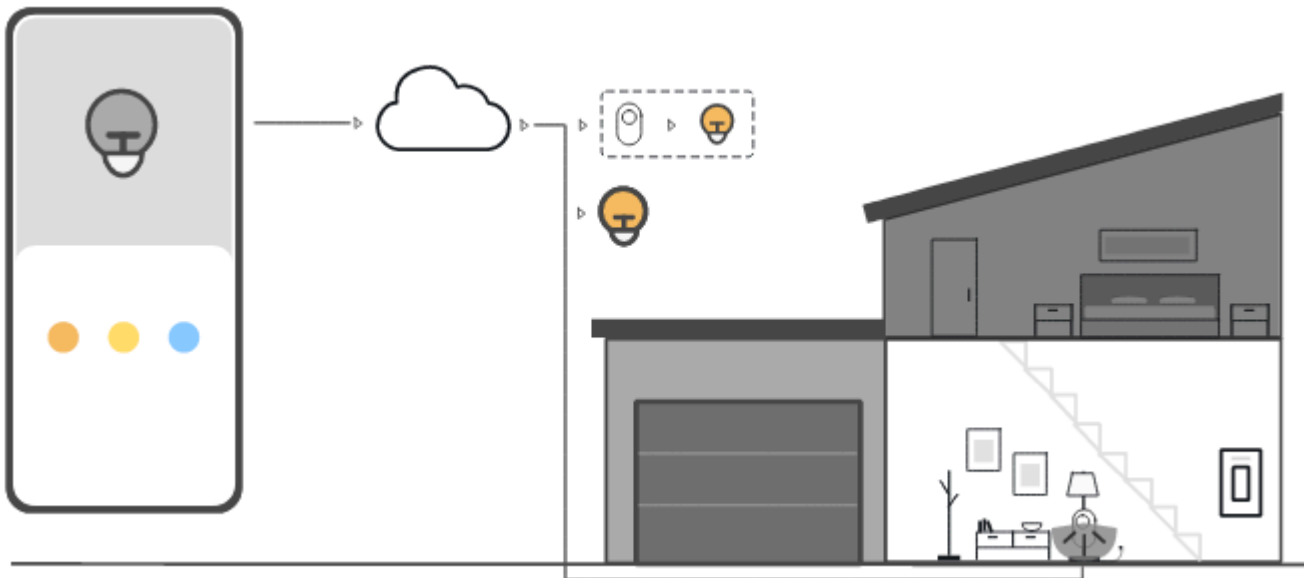
Animasi pada langkah ini menunjukkan bagaimana layanan Device Shadow AWS IoT Core menyimpan informasi status perangkat untuk perangkat kontrol dan lampu pintar. Saat lampu pintar sedang offline, Device Shadow menyimpan perintah dari perangkat kontrol.

Ketika lampu pintar terhubung kembali AWS IoT Core, ia mengambil perintah-perintah itu. Saat perangkat kontrol offline, Device Shadow menyimpan informasi status dari lampu pintar. Ketika perangkat kontrol terhubung kembali, ia mengambil keadaan lampu pintar saat ini untuk memperbarui tampilannya.

Untuk informasi selengkapnya tentang Device Shadows, lihat [AWS IoT Layanan Device Shadow](#).

Merutekan data perangkat ke layanan

Pelajari cara AWS IoT Core mengirim status perangkat ke AWS layanan lain.



Animasi pada langkah ini menunjukkan cara AWS IoT Core mengirim data dari perangkat ke AWS layanan lain dengan menggunakan AWS IoT aturan. AWS IoT aturan berlangganan pesan tertentu dari perangkat, menafsirkan data dalam pesan tersebut, dan merutekan data yang ditafsirkan ke layanan lain. Dalam contoh ini, AWS IoT aturan menafsirkan data dari sensor gerak dan mengirimkan perintah ke Device Shadow, yang kemudian mengirimkannya ke bohlam pintar. Seperti pada contoh sebelumnya, Device Shadow menyimpan info status perangkat untuk perangkat kontrol.

Untuk informasi selengkapnya tentang AWS IoT aturan, lihat [Aturan untuk AWS IoT](#).

Coba koneksi AWS IoT cepat

Dalam tutorial ini, Anda akan membuat objek hal pertama Anda, menghubungkan perangkat ke sana, dan melihatnya mengirim pesan MQTT.

Anda dapat mengharapkan untuk menghabiskan 15-20 menit pada tutorial ini.

Tutorial ini paling baik bagi orang-orang yang ingin memulai dengan cepat AWS IoT untuk melihat cara kerjanya dalam skenario terbatas. Jika Anda mencari contoh yang akan membantu Anda memulai sehingga Anda dapat menjelajahi lebih banyak fitur dan layanan, cobalah [Jelajahi AWS IoT Core layanan dalam tutorial langsung](#).

Dalam tutorial ini, Anda akan mengunduh dan menjalankan perangkat lunak pada perangkat yang terhubung ke sumber daya AWS IoT Core sebagai bagian dari solusi IoT yang sangat kecil. Perangkat ini dapat berupa perangkat IoT, seperti Raspberry Pi, atau bisa juga komputer yang

menjalankan Linux, OS dan OSX, atau Windows. Jika Anda ingin menghubungkan perangkat Long Range WAN (LoRaWAN) AWS IoT, lihat [tutorial> Menghubungkan perangkat dan gateway ke AWS IoT Core](#) WAN. LoRa

Jika perangkat Anda mendukung browser yang dapat menjalankan [AWS IoT konsol](#), kami sarankan Anda menyelesaikan tutorial ini di perangkat itu.

Note


Jika perangkat Anda tidak memiliki browser yang kompatibel, ikuti tutorial ini di komputer. Ketika prosedur meminta Anda untuk mengunduh file, unduh ke komputer Anda, dan kemudian transfer file yang diunduh ke perangkat Anda dengan menggunakan Secure Copy (SCP) atau proses serupa.

Tutorial ini mengharuskan perangkat IoT Anda untuk berkomunikasi dengan port 8443 pada titik akhir data perangkat Anda Akun AWS. Untuk menguji apakah ia dapat mengakses port itu, coba prosedurnya di [Menguji konektivitas dengan titik akhir data perangkat](#).

Langkah 1. Mulai tutorialnya

Jika memungkinkan, selesaikan prosedur ini di perangkat Anda; jika tidak, bersiaplah untuk mentransfer file ke perangkat Anda nanti dalam prosedur ini.

Untuk memulai tutorial, masuk ke [AWS IoT konsol](#). Di halaman beranda AWS IoT konsol, di sebelah kiri, pilih Connect dan kemudian pilih Connect one device.



Monitor

Connect

- Connect one device
- ▶ Connect many devices

Test


- ▶ Device Advisor
- MQTT test client
- Device Location [New](#)

Manage

- ▶ All devices
- ▶ Greengrass devices

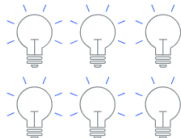
How it works

Connect devices to AWS IoT so they can send and receive data. **Bold** text refers to an entry in the **Connect** menu of the navigation pane.



Connect one device

The **Quick connect** wizard walks you through the steps to create the resources and download the software required to connect your IoT device to AWS IoT.



Connect many devices

Fleet provisioning templates define security policies and registry settings when a device connects to AWS IoT for the first time.

Langkah 2. Buat objek benda

1. Di bagian Siapkan perangkat Anda, ikuti petunjuk di layar untuk mempersiapkan perangkat Anda untuk AWS IoT terhubung.

AWS IoT ×

Monitor

Connect

Connect one device

Connect many devices

Test

Device Advisor

MQTT test client

Manage

All devices

Greengrass devices

LPWAN devices

Remote actions

Message Routing

Retained messages

Security

Fleet Hub

Device Software

Billing groups

Settings

Feature spotlight

Documentation

New console experience

Tell us what you think

AWS IoT > Connect > Connect one device

Step 1
Prepare your device

Step 2
Register and secure your device

Step 3
Choose platform and SDK

Step 4
Download connection kit

Step 5
Run connection kit

Prepare your device [Info](#)

How it works

In this wizard, we'll be creating a thing resource in AWS IoT. A thing resource is a digital representation of a physical device or logical entity.

A thing resource uses certificates to secure communication between your device and AWS IoT. AWS IoT policies control access to the AWS IoT resources. This wizard creates the certificate and policy for your device.

When a device connects to AWS IoT, policies enable it to subscribe and publish MQTT messages with AWS IoT message broker.

Prepare your device

1. Turn on your device and make sure it's connected to the internet.
2. Choose how you want to load files onto your device.
 1. If your device supports a browser, open the AWS IoT console on your device and run this wizard. You can download the files directly to your device from the browser.
 2. If your device doesn't support a browser, choose the best way to transfer files from the computer with the browser to your device. Some options to transfer files include using the file transfer protocol (FTP) and using a USB memory stick.
3. Make sure that you can access a command-line interface on your device.
 1. If you're running this wizard on your IoT device, open a terminal window on your device to access a command-line interface.
 2. If you're not running this on your IoT device, open an SSH terminal window on this device and connect it to your IoT device.
4. From the terminal window, enter this command:

After you complete these steps and get a successful ping response, you're ready to continue and connect your device to AWS IoT.

Cancel

2. Di bagian Daftar dan amankan perangkat Anda, pilih Buat hal baru atau Pilih hal yang sudah ada. Di bidang Thing name, masukkan nama untuk objek benda Anda. Nama benda yang digunakan dalam contoh ini adalah **TutorialTestThing**

Important

Periksa kembali nama benda Anda sebelum melanjutkan.

Nama benda tidak dapat diubah setelah objek benda dibuat. Jika Anda ingin mengubah nama benda, Anda harus membuat objek benda baru dengan nama benda yang benar dan kemudian menghapus yang dengan nama yang salah.

Di bagian Konfigurasi tambahan, sesuaikan sumber daya benda Anda lebih lanjut menggunakan konfigurasi opsional yang terdaftar.

Setelah Anda memberikan nama objek benda Anda dan memilih konfigurasi tambahan apa pun, pilih Berikutnya.

The screenshot shows the AWS IoT console interface for the 'Register and secure your device' wizard. The left sidebar contains navigation options like Monitor, Connect, Test, Manage, Device Software, and Billing groups. The main content area is titled 'Register and secure your device' and includes a progress indicator for five steps. Step 2 is active, showing the 'Register and secure your device' section. Below this, there are two radio buttons: 'Create a new thing' (selected) and 'Choose an existing thing'. A text input field for 'Thing name' is present with a placeholder 'Enter_name' and a note: 'Enter a unique name containing only: letters, numbers, hyphens, colons, or underscores. A thing name can't contain any spaces.' Underneath, there are four optional configuration sections: 'Thing type - optional', 'Searchable thing attributes - optional', 'Thing groups - optional', and 'Billing group - optional'. At the bottom, a blue information box states: 'Certificate and policy for your device: Your device requires a unique device certificate to securely authenticate its identity to AWS IoT, and an AWS IoT policy that authorizes it to send and receive messages. We'll create these resources for your device automatically. You can review and edit their properties later, if necessary.' At the bottom right, there are 'Cancel', 'Previous', and 'Next' buttons.

- Di bagian Pilih platform dan SDK, pilih platform dan bahasa SDK AWS IoT Perangkat yang ingin Anda gunakan. Contoh ini menggunakan platform Linux/OSX dan Python SDK. Pastikan bahwa

Anda memiliki python3 dan pip3 diinstal pada perangkat target Anda sebelum Anda melanjutkan ke langkah berikutnya.

Note

Pastikan untuk memeriksa daftar perangkat lunak prasyarat yang diperlukan oleh SDK pilihan Anda di bagian bawah halaman konsol.

Anda harus memiliki perangkat lunak yang diperlukan diinstal pada komputer target Anda sebelum Anda melanjutkan ke langkah berikutnya.

Setelah Anda memilih bahasa SDK platform dan perangkat, pilih Berikutnya.

The screenshot shows the AWS IoT console interface for the 'Choose platform and SDK' step. The breadcrumb navigation at the top reads 'AWS IoT > Connect > Connect one device'. On the left, a vertical sidebar lists five steps: 'Step 1 Prepare your device', 'Step 2 Register and secure your device', 'Step 3 Choose platform and SDK' (which is highlighted), 'Step 4 Download connection kit', and 'Step 5 Run connection kit'. The main content area is titled 'Choose platform and SDK' with an 'Info' link. Below the title is a section 'Choose the software for your device' featuring an illustration of a computer monitor and a lightbulb icon. A text box explains that the wizard helps download a software development kit (SDK) to the device, supporting various languages and OSes. The 'Platform and SDK' section contains two main choices: 'Device platform operating system' and 'AWS IoT Device SDK'. Under 'Device platform operating system', 'Linux / macOS' is selected with radio buttons, showing requirements for Linux (any version) and macOS (10.13+). 'Windows' (Version 10) is also listed. Under 'AWS IoT Device SDK', 'Python' is selected with radio buttons, showing requirements for Version 3.6+ and the need for Python and Git. 'Node.js' (Version 10+, requires Node.js and npm) and 'Java' (Version 8, requires Java JDK, Maven, and Git) are also listed. At the bottom right, there are three buttons: 'Cancel', 'Previous', and 'Next' (which is highlighted with a red border).

Langkah 3. Unduh file ke perangkat Anda

Halaman ini muncul AWS IoT setelah membuat kit koneksi, yang mencakup file dan sumber daya berikut yang dibutuhkan perangkat Anda:

- File sertifikat benda yang digunakan untuk mengautentikasi perangkat
 - Sumber daya kebijakan untuk mengotorisasi objek benda Anda untuk berinteraksi AWS IoT
 - Skrip untuk mengunduh SDK AWS Perangkat dan menjalankan program sampel di perangkat Anda
1. Saat Anda siap untuk melanjutkan, pilih Unduh kit koneksi untuk tombol untuk mengunduh kit koneksi untuk platform yang Anda pilih sebelumnya.

AWS IoT > Connect > Connect one device

Step 1
Prepare your device

Step 2
Register and secure your device



Step 3
Choose platform and SDK

Step 4
Download connection kit

Step 5
Run connection kit

Download connection kit Info

Install the software on your device

 →  We created the AWS IoT resources that your device needs to connect to AWS IoT. We also created a connection kit that includes the resources in a zipped file that you need to install on your device. The resources in the connection kit are listed below. In this step, you'll install them on your device.


Connection kit

Certificate TutorialTestThing.cert.pem	Private key TutorialTestThing.private.key	AWS IoT Device SDK Python
Script to send and receive messages start.sh	Policy TutorialTestThing-Policy View policy	



Download

If you are running this from a browser on the device, after you download the connection kit, it will be in the browser's download folder.


If you are not running this from a browser on your device, you'll need to transfer the connection kit from your browser's download folder to your device using the method you tested when you prepared your device in step 1.

 **Download connection kit**

Unzip connection kit on your device

  After the connection kit is on your device, unzip it using this command:

```
unzip connect_device_package.zip
```

 Copy

Cancel Previous **Next**

2. Jika Anda menjalankan prosedur ini di perangkat Anda, simpan file kit koneksi ke direktori tempat Anda dapat menjalankan perintah baris perintah.

Jika Anda tidak menjalankan prosedur ini di perangkat Anda, simpan file kit koneksi ke direktori lokal dan kemudian transfer file ke perangkat Anda.

3. Di kit koneksi Unzip di bagian perangkat Anda, masukkan `unzip connect_device_package.zip` di direktori tempat file kit koneksi berada.

Jika Anda menggunakan jendela PowerShell perintah Windows dan unzip perintah tidak berfungsi, ganti unzip dengan expand-archive, dan coba baris perintah lagi.

- Setelah Anda memiliki file kit koneksi pada perangkat, lanjutkan tutorial dengan memilih Berikutnya.

The screenshot shows the AWS IoT console interface for the 'Download connection kit' step. The breadcrumb navigation at the top reads 'AWS IoT > Connect > Connect one device'. On the left, a sidebar lists five steps: 'Step 1 Prepare your device', 'Step 2 Register and secure your device', 'Step 3 Choose platform and SDK', 'Step 4 Download connection kit' (which is highlighted in bold), and 'Step 5 Run connection kit'. The main content area is titled 'Download connection kit' with an 'Info' link. It is divided into three sections: 1. 'Install the software on your device' which explains that AWS IoT resources have been created and are packaged in a zip file. 2. 'Connection kit' which lists the contents: Certificate (TutorialTestThing.cert.pem), Private key (TutorialTestThing.private.key), AWS IoT Device SDK (Python), Script to send and receive messages (start.sh), and Policy (TutorialTestThing-Policy with a 'View policy' link). 3. 'Download' which provides instructions on how to get the zip file to the device, either via browser download or file transfer. A 'Download connection kit' button is present. Below this is the 'Unzip connection kit on your device' section, which shows a terminal command 'unzip connect_device_package.zip' in a text box and a 'Copy' button. At the bottom right, there are three navigation buttons: 'Cancel', 'Previous', and 'Next' (which is highlighted with a red border).

AWS IoT > Connect > Connect one device

Step 1
Prepare your device

Step 2
Register and secure your device

Step 3
Choose platform and SDK

Step 4
Download connection kit

Step 5
Run connection kit

Download connection kit [Info](#)

Install the software on your device

We created the AWS IoT resources that your device needs to connect to AWS IoT. We also created a connection kit that includes the resources in a zipped file that you need to install on your device. The resources in the connection kit are listed below. In this step, you'll install them on your device.

Connection kit

Certificate TutorialTestThing.cert.pem	Private key TutorialTestThing.private.key	AWS IoT Device SDK Python
Script to send and receive messages start.sh	Policy TutorialTestThing-Policy View policy	

Download

If you are running this from a browser on the device, after you download the connection kit, it will be in the browser's download folder.

If you are not running this from a browser on your device, you'll need to transfer the connection kit from your browser's download folder to your device using the method you tested when you prepared your device in step 1.

[Download connection kit](#)

Unzip connection kit on your device

After the connection kit is on your device, unzip it using this command:

```
unzip connect_device_package.zip
```

[Copy](#)

Cancel Previous **Next**

Langkah 4. Jalankan sampel

Anda melakukan prosedur ini di terminal atau jendela perintah pada perangkat Anda saat Anda mengikuti petunjuk yang ditampilkan di konsol. Perintah yang Anda lihat di konsol adalah untuk sistem operasi yang Anda pilih [the section called “Langkah 2. Buat objek benda”](#). Yang ditampilkan di sini adalah untuk sistem operasi Linux/OSX.

1. Di terminal atau jendela perintah di perangkat Anda, di direktori dengan file kit koneksi, lakukan langkah-langkah yang ditunjukkan di AWS IoT konsol.

AWS IoT > Connect > Connect one device

Step 1
Prepare your device

Step 2
Register and secure your device

Step 3
Choose platform and SDK

Step 4
Download connection kit

Step 5
Run connection kit

Run connection kit Info

How to display messages from your device

Step 1: Add execution permissions
On the device, launch a terminal window to copy and paste the command to add execution permissions.

```
chmod +x start.sh
```

Step 2: Run the start script
On the device, copy and paste the command to the terminal window and run the start script.

```
./start.sh
```

Step 3: Return to this screen to view your device's messages
After running the start script, return to this screen to see the messages between your device and AWS IoT. The messages from your device appear in the following list.

Subscriptions	sdk/test/Python	Pause	Clear
sdk/test/Python	Waiting for messages		

Cancel Previous Continue

- Setelah Anda memasukkan perintah dari Langkah 2 di konsol, Anda akan melihat output di terminal perangkat atau jendela perintah yang mirip dengan yang berikut ini. Output ini berasal dari pesan yang dikirim oleh program dan kemudian menerima kembali AWS IoT Core.

```
Running pub/sub sample application...
Connecting to a13hikvzkye6lx-ats.iot.us-east-1.amazonaws.com with client ID 'basicPubSub'...
Connected!
Subscribing to topic 'sdk/test/Python'...
Subscribed with QoS.AT_LEAST_ONCE
Sending messages until program killed
Publishing message to topic 'sdk/test/Python': Hello World! [1]
Received message from topic 'sdk/test/Python': b'"Hello World! [1]"'
Publishing message to topic 'sdk/test/Python': Hello World! [2]
Received message from topic 'sdk/test/Python': b'"Hello World! [2]"'
Publishing message to topic 'sdk/test/Python': Hello World! [3]
Received message from topic 'sdk/test/Python': b'"Hello World! [3]"'
```

Saat program sampel sedang berjalan, pesan pengujian Hello World! akan muncul juga. Pesan pengujian muncul di terminal atau jendela perintah di perangkat Anda.

Note

Untuk informasi selengkapnya tentang langganan topik dan publikasi, lihat contoh kode SDK yang Anda pilih.

- Untuk menjalankan program sampel lagi, Anda dapat mengulangi perintah dari Langkah 2 di konsol prosedur ini.
- (Opsional) Jika Anda ingin melihat pesan dari klien IoT Anda di [AWS IoT konsol](#), buka klien pengujian [MQTT di halaman Uji](#) konsol. AWS IoT Jika Anda memilih Python SDK, maka di klien pengujian MQTT, di filter Topik, masukkan topik, seperti berlangganan pesan dari **sdk/test/python** perangkat Anda. Filter topik peka huruf besar/kecil dan bergantung pada bahasa pemrograman SDK yang Anda pilih di Langkah 1. Untuk informasi selengkapnya tentang langganan topik dan publikasi, lihat contoh kode SDK yang Anda pilih.
- Setelah Anda berlangganan topik pengujian, jalankan `./start.sh` di perangkat Anda. Untuk informasi selengkapnya, lihat [the section called “Lihat pesan MQTT dengan klien MQTT AWS IoT”](#).

Setelah Anda menjalankan `./start.sh`, pesan muncul di klien MQTT, mirip dengan yang berikut ini:

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

sequenceNomor terbungkus secara [] bertahap oleh satu setiap kali Hello World! pesan baru diterima dan berhenti ketika Anda mengakhiri program.

- Untuk menyelesaikan tutorial dan melihat ringkasan, di AWS IoT konsol, pilih Lanjutkan.

AWS IoT > Connect > Connect one device

Step 1
Prepare your device

Step 2
Register and secure your device

Step 3
Choose platform and SDK

Step 4
Download connection kit

Step 5
Run connection kit

Run connection kit [Info](#)

How to display messages from your device

Step 1: Add execution permissions
On the device, launch a terminal window to copy and paste the command to add execution permissions.

```
chmod +x start.sh
```

[Copy](#)

Step 2: Run the start script
On the device, copy and paste the command to the terminal window and run the start script.

```
./start.sh
```

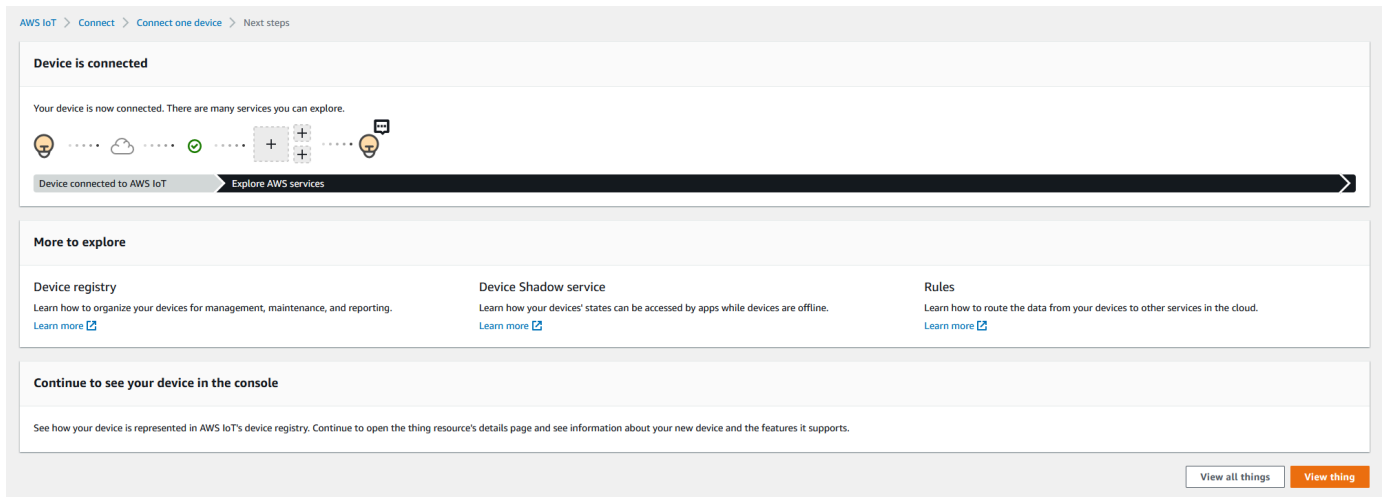
[Copy](#)

Step 3: Return to this screen to view your device's messages
After running the start script, return to this screen to see the messages between your device and AWS IoT. The messages from your device appear in the following list.

Subscriptions	sdk/test/Python	Resume	Clear
sdk/test/Python	<p>▼ sdk/test/Python September 14, 2022, 10:47:44 (UTC-0700)</p> <p>"Hello World! [3]"</p>		
	<p>▼ sdk/test/Python September 14, 2022, 10:47:43 (UTC-0700)</p> <p>"Hello World! [2]"</p>		
	<p>▼ sdk/test/Python September 14, 2022, 10:47:42 (UTC-0700)</p> <p>"Hello World! [1]"</p>		

[Cancel](#) [Previous](#) [Continue](#)

- Ringkasan tutorial koneksi AWS IoT cepat Anda sekarang akan muncul.



Langkah 5. Jelajahi lebih lanjut

Berikut adalah beberapa ide untuk dijelajahi AWS IoT lebih lanjut setelah Anda menyelesaikan awal yang cepat.

- [Lihat pesan MQTT di klien pengujian MQTT](#)

Dari [AWS IoT konsol](#), Anda dapat membuka [klien MQTT](#) di halaman Uji konsol. AWS IoT Di klien pengujian MQTT, berlangganan, dan kemudian#, di perangkat Anda, jalankan program `./start.sh` seperti yang dijelaskan pada langkah sebelumnya. Untuk informasi selengkapnya, lihat [the section called “Lihat pesan MQTT dengan klien MQTT AWS IoT”](#).

- Jalankan pengujian pada perangkat Anda dengan [Device Advisor](#)

Gunakan Device Advisor untuk menguji apakah perangkat Anda dapat terhubung dan berinteraksi dengan aman dan andal. AWS IoT

- [the section called “Coba tutorial AWS IoT Core interaktif”](#)

Untuk memulai tutorial interaktif, dari halaman Pelajari AWS IoT konsol, di ubin Lihat cara AWS IoT kerja, pilih Mulai tutorial.

- [Bersiaplah untuk menjelajahi lebih banyak tutorial](#)

Mulai cepat ini memberi Anda hanya contoh AWS IoT. Jika Anda ingin menjelajahi AWS IoT lebih jauh dan mempelajari fitur-fitur yang menjadikannya platform solusi IoT yang kuat, mulailah mempersiapkan platform pengembangan Anda dengan. [Jelajahi AWS IoT Core layanan dalam tutorial langsung](#)

Menguji konektivitas dengan titik akhir data perangkat

Topik ini menjelaskan cara menguji koneksi perangkat dengan titik akhir data perangkat akun Anda, titik akhir yang digunakan perangkat IoT Anda untuk terhubung. AWS IoT

Lakukan prosedur ini pada perangkat yang ingin Anda uji atau dengan menggunakan sesi terminal SSH yang terhubung ke perangkat yang ingin Anda uji.

Untuk menguji konektivitas perangkat dengan titik akhir data perangkat Anda.

- [Menemukan titik akhir data perangkat](#)
- [Uji koneksi dengan cepat](#)
- [Dapatkan aplikasi untuk menguji koneksi ke titik akhir dan port data perangkat Anda](#)
- [Uji koneksi ke titik akhir dan port data perangkat Anda](#)

Menemukan titik akhir data perangkat

Untuk menemukan titik akhir data perangkat

1. Di [AWS IoT konsol](#), di dekat bagian bawah panel navigasi, pilih Pengaturan.
2. Di halaman Pengaturan, di wadah titik akhir data perangkat, cari nilai Endpoint dan salin. Nilai endpoint Anda unik untuk Anda Akun AWS dan mirip dengan contoh ini: `a3qEXAMPLEsffp-ats.iot.eu-west-1.amazonaws.com`.
3. Simpan titik akhir data perangkat Anda untuk digunakan dalam prosedur berikut.

Uji koneksi dengan cepat

Prosedur ini menguji konektivitas umum dengan titik akhir data perangkat Anda, tetapi tidak menguji port spesifik yang akan digunakan perangkat Anda. Tes ini menggunakan program umum dan biasanya cukup untuk mengetahui apakah perangkat Anda dapat terhubung AWS IoT.

Jika Anda ingin menguji konektivitas dengan port spesifik yang akan digunakan perangkat Anda, lewati prosedur ini dan lanjutkan ke [Dapatkan aplikasi untuk menguji koneksi ke titik akhir dan port data perangkat Anda](#).

Untuk menguji titik akhir data perangkat dengan cepat

1. Di jendela terminal atau baris perintah pada perangkat Anda, ganti contoh titik akhir data perangkat (`a3qEXAMPLEsffp-ats.iot.eu-west-1.amazonaws.com`) dengan titik akhir data perangkat untuk akun Anda, lalu masukkan perintah ini.

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. Jika ping menampilkan output yang mirip dengan berikut ini, itu berhasil terhubung ke titik akhir data perangkat Anda. Meskipun tidak berkomunikasi AWS IoT secara langsung, ia menemukan server dan kemungkinan AWS IoT tersedia melalui titik akhir ini.

```
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
```

Jika Anda puas dengan hasil ini, Anda dapat berhenti menguji di sini.

Jika Anda ingin menguji konektivitas dengan port tertentu yang digunakan oleh AWS IoT, lanjutkan ke [Dapatkan aplikasi untuk menguji koneksi ke titik akhir dan port data perangkat Anda](#).

3. Jika ping tidak mengembalikan output yang berhasil, periksa nilai titik akhir untuk memastikan Anda memiliki titik akhir yang benar dan periksa koneksi perangkat dengan internet.

Dapatkan aplikasi untuk menguji koneksi ke titik akhir dan port data perangkat Anda

Tes konektivitas yang lebih menyeluruh dapat dilakukan dengan menggunakan nmap. Prosedur ini menguji untuk melihat nmap apakah diinstal pada perangkat Anda.

Untuk memeriksa **nmap** di perangkat

1. Di jendela terminal atau baris perintah pada perangkat yang ingin Anda uji, masukkan perintah ini untuk melihat nmap apakah diinstal.

```
nmap --version
```

2. Jika Anda melihat output yang mirip dengan berikut nmap ini, diinstal dan Anda dapat melanjutkan ke [the section called “Uji koneksi ke titik akhir dan port data perangkat Anda”](#).

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

3. Jika Anda tidak melihat respons yang mirip dengan yang ditunjukkan pada langkah sebelumnya, Anda harus menginstal nmap di perangkat. Pilih prosedur untuk sistem operasi perangkat Anda.

Linux

Prosedur ini mengharuskan Anda memiliki izin untuk menginstal perangkat lunak di komputer.

Untuk menginstal nmap di komputer Linux Anda

1. Di jendela terminal atau baris perintah pada perangkat Anda, masukkan perintah yang sesuai dengan versi Linux yang dijalankannya.
 - a. Debian atau Ubuntu:

```
sudo apt install nmap
```

- b. CentOS atau RHEL:

```
sudo yum install nmap
```

2. Uji instalasi dengan perintah ini:

```
nmap --version
```

3. Jika Anda melihat output yang mirip dengan berikut nmap ini, diinstal dan Anda dapat melanjutkan ke [the section called “Uji koneksi ke titik akhir dan port data perangkat Anda”](#).

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

macOS

Prosedur ini mengharuskan Anda memiliki izin untuk menginstal perangkat lunak di komputer.

Untuk menginstal nmap di komputer macOS Anda

1. Di browser, buka <https://nmap.org/download#macosx> dan unduh penginstal stabil terbaru.
Saat diminta, pilih Buka dengan DiskImageInstaller.
2. Di jendela instalasi, pindahkan paket ke folder Aplikasi.
3. Di Finder, cari nmap-xxxx-mpkg paket di folder Applications. Ctrl-click paket on dan pilih Buka untuk membuka paket.
4. Tinjau kotak dialog keamanan. Jika Anda siap untuk menginstal nmap, pilih Buka untuk menginstal nmap.
5. Di Terminal, uji instalasi dengan perintah ini.

```
nmap --version
```

6. Jika Anda melihat output yang mirip dengan berikut nmap ini, diinstal dan Anda dapat melanjutkan ke [the section called “Uji koneksi ke titik akhir dan port data perangkat Anda”](#).

```
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-libpcrc-7.6 nmap-libpcap-1.9.1 nmap-libdnet-1.12 ipv6  
Compiled without:
```



```
Available nsock engines: kqueue poll select
```

Windows

Prosedur ini mengharuskan Anda memiliki izin untuk menginstal perangkat lunak di komputer.

Untuk menginstal nmap di komputer Windows Anda

1. Di browser, buka <https://nmap.org/download#windows> dan unduh rilis stabil terbaru dari program pengaturan.

Jika diminta, pilih Simpan file. Setelah file diunduh, buka dari folder unduhan.

2. Setelah file pengaturan selesai diunduh, buka unduhan nmap-xxxx-setup.exe untuk menginstal aplikasi.
3. Terima pengaturan default saat program menginstal.

Anda tidak memerlukan aplikasi Npcap untuk tes ini. Anda dapat membatalkan pilihan itu jika Anda tidak ingin menginstalnya.

4. DiCommand, uji instalasi dengan perintah ini.

```
nmap --version
```

5. Jika Anda melihat output yang mirip dengan berikut nmap ini, diinstal dan Anda dapat melanjutkan ke [the section called “Uji koneksi ke titik akhir dan port data perangkat Anda”](#).

```
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-libpcrc-7.6 Npcap-1.50 nmap-libdnet-1.12 ipv6
Compiled without:
Available nsock engines: iocp poll select
```

Uji koneksi ke titik akhir dan port data perangkat Anda

Untuk menguji titik akhir dan port data perangkat

1. Di jendela terminal atau baris perintah pada perangkat Anda, ganti contoh titik akhir data perangkat (*a3qEXAMPLEsffp-ats.iot.eu-west-1.amazonaws.com*) dengan titik akhir data perangkat untuk akun Anda, lalu masukkan perintah ini.

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

2. Jika nmap menampilkan output yang mirip dengan berikut ini, berhasil nmap tersambung ke titik akhir data perangkat Anda di port yang dipilih.

```
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
```

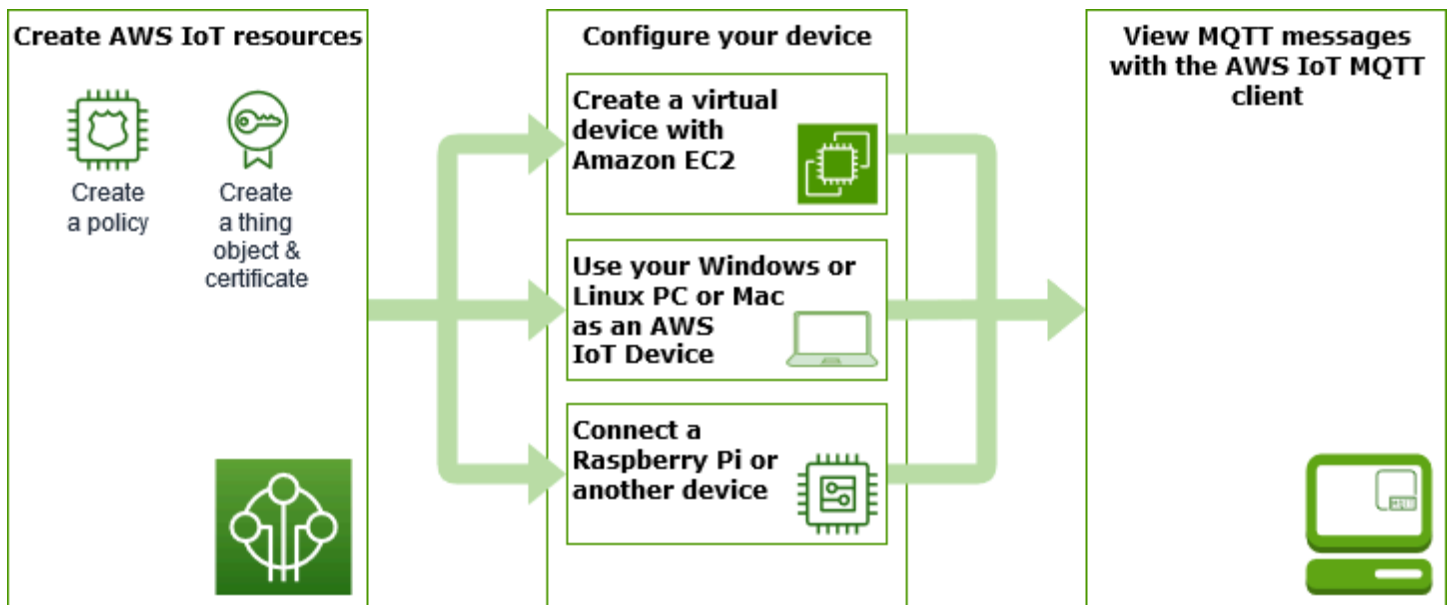
3. Jika nmap tidak menghasilkan output yang berhasil, periksa nilai titik akhir untuk memastikan Anda memiliki titik akhir yang benar dan periksa koneksi perangkat Anda dengan internet.

Anda dapat menguji port lain pada titik akhir data perangkat Anda, seperti port 443, port HTTPS utama, dengan mengganti port yang digunakan pada langkah **18443**, dengan port yang ingin Anda uji.

Jelajahi AWS IoT Core layanan dalam tutorial langsung

Dalam tutorial ini, Anda akan menginstal perangkat lunak dan membuat AWS IoT sumber daya yang diperlukan untuk menghubungkan perangkat AWS IoT Core sehingga dapat mengirim dan menerima pesan MQTT. AWS IoT Core Anda akan melihat pesan di klien MQTT di konsol. AWS IoT

Anda dapat menghabiskan 20-30 menit pada tutorial ini. Jika Anda menggunakan perangkat IoT atau Raspberry Pi, tutorial ini mungkin memakan waktu lebih lama jika, misalnya, Anda perlu menginstal sistem operasi dan mengkonfigurasi perangkat.



Tutorial ini paling baik untuk pengembang yang ingin memulai AWS IoT Core sehingga mereka dapat terus mengeksplorasi fitur-fitur yang lebih canggih, seperti [mesin aturan](#) dan [bayangan](#). Tutorial ini mempersiapkan Anda untuk terus belajar tentang AWS IoT Core dan bagaimana berinteraksi dengan AWS layanan lain dengan menjelaskan langkah-langkah secara lebih rinci daripada tutorial mulai [cepat](#). Jika Anda hanya mencari pengalaman Hello World yang cepat, cobalah [Coba koneksi AWS IoT cepat](#).

Setelah menyiapkan Akun AWS dan AWS IoT konsol, Anda akan mengikuti langkah-langkah ini untuk melihat cara menghubungkan perangkat dan mengirimkannya pesan AWS IoT Core.

Langkah selanjutnya

- [Pilih opsi perangkat mana yang terbaik untuk Anda](#)
- [the section called “Buat AWS IoT sumber daya”](#) jika Anda tidak akan membuat perangkat virtual dengan Amazon EC2
- [the section called “Konfigurasi perangkat Anda”](#)
- [the section called “Lihat pesan MQTT dengan klien MQTT AWS IoT”](#)

Untuk informasi lebih lanjut tentang AWS IoT Core, lihat [Apa itu AWS IoT Core?](#)

Opsi perangkat mana yang terbaik untuk Anda?

Jika Anda tidak yakin opsi mana yang harus dipilih, gunakan daftar keuntungan dan kerugian masing-masing opsi berikut untuk membantu Anda memutuskan mana yang terbaik untuk Anda.

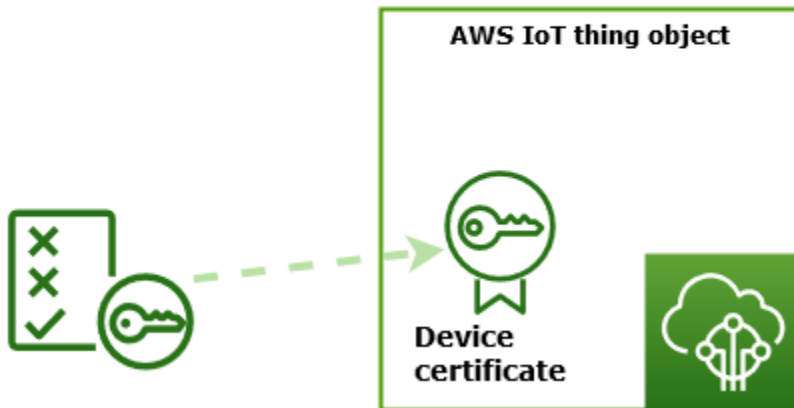
Opsi	Ini mungkin pilihan yang baik jika:	Ini mungkin bukan pilihan yang baik jika:
the section called “Buat perangkat virtual dengan Amazon EC2”	<ul style="list-style-type: none"> • Anda tidak memiliki perangkat sendiri untuk diuji. • Anda tidak ingin menginstal perangkat lunak apa pun di sistem Anda sendiri. • Anda ingin menguji pada OS Linux. 	<ul style="list-style-type: none"> • Anda tidak nyaman menggunakan perintah baris perintah. • Anda tidak ingin dikenakan AWS biaya tambahan. • Anda tidak ingin menguji pada OS Linux.
the section called “Gunakan Windows atau Linux PC atau Mac sebagai AWS IoT perangkat”	<ul style="list-style-type: none"> • Anda tidak ingin dikenakan AWS biaya tambahan. • Anda tidak ingin mengonfigurasi perangkat tambahan apa pun. 	<ul style="list-style-type: none"> • Anda tidak ingin menginstal perangkat lunak apa pun di komputer pribadi Anda. • Anda menginginkan platform pengujian yang lebih representatif.
the section called “Connect Raspberry Pi atau perangkat lain”	<ul style="list-style-type: none"> • Anda ingin menguji AWS IoT dengan perangkat yang sebenarnya. • Anda sudah memiliki perangkat untuk diuji. • Anda memiliki pengalaman mengintegrasikan perangkat keras ke dalam sistem. 	<ul style="list-style-type: none"> • Anda tidak ingin membeli atau mengkonfigurasi perangkat hanya untuk mencobanya. • Anda ingin menguji AWS IoT sesederhana mungkin, untuk saat ini.

Buat AWS IoT sumber daya

Dalam tutorial ini, Anda akan membuat AWS IoT sumber daya yang dibutuhkan perangkat untuk terhubung AWS IoT Core dan bertukar pesan.

Create an AWS IoT Core policy

Create a thing and its certificate



1. Buat dokumen AWS IoT kebijakan, yang akan mengizinkan perangkat Anda untuk berinteraksi dengan AWS IoT layanan.
2. Buat objek benda AWS IoT dan sertifikat perangkat X.509-nya, lalu lampirkan dokumen kebijakan. Objek benda adalah representasi virtual perangkat Anda di AWS IoT registri. Sertifikat mengautentikasi perangkat Anda AWS IoT Core, dan dokumen kebijakan mengizinkan perangkat Anda untuk berinteraksi. AWS IoT

Note

Jika Anda berencana untuk [the section called “Buat perangkat virtual dengan Amazon EC2”](#), Anda dapat melewati halaman ini dan melanjutkan ke [the section called “Konfigurasi perangkat Anda”](#). Anda akan membuat sumber daya ini ketika Anda membuat hal virtual Anda.

Tutorial ini menggunakan AWS IoT konsol untuk membuat AWS IoT sumber daya. Jika perangkat Anda mendukung browser web, mungkin lebih mudah untuk menjalankan prosedur ini di browser web perangkat karena Anda akan dapat mengunduh file sertifikat langsung ke perangkat Anda. Jika Anda menjalankan prosedur ini di komputer lain, Anda harus menyalin file sertifikat ke perangkat Anda sebelum dapat digunakan oleh aplikasi sampel.

Buat AWS IoT kebijakan

Perangkat menggunakan sertifikat X.509 untuk mengautentikasi dengan AWS IoT Core Sertifikat memiliki AWS IoT kebijakan yang dilampirkan padanya. Kebijakan ini menentukan AWS IoT operasi mana, seperti berlangganan atau menerbitkan topik MQTT, perangkat yang diizinkan untuk dilakukan. Perangkat Anda menampilkan sertifikatnya saat terhubung dan mengirim pesan ke AWS IoT Core.

Ikuti langkah-langkah untuk membuat kebijakan yang memungkinkan perangkat Anda melakukan AWS IoT operasi yang diperlukan untuk menjalankan program contoh. Anda harus membuat AWS IoT kebijakan sebelum dapat melampirkannya ke sertifikat perangkat, yang akan Anda buat nanti.

Untuk membuat AWS IoT kebijakan

1. Di [AWS IoT konsol](#), di menu sebelah kiri, pilih Keamanan dan kemudian pilih Kebijakan.
2. Pada halaman Anda belum memiliki kebijakan, pilih Buat kebijakan.

Jika akun Anda memiliki kebijakan yang ada, pilih Buat kebijakan.

3. Pada halaman Buat kebijakan:
 1. Di bagian Properti kebijakan, di bidang Nama kebijakan, masukkan nama untuk kebijakan (misalnya, **My_Iot_Policy**). Jangan gunakan informasi identitas pribadi dalam nama kebijakan Anda.
 2. Di bagian Dokumen kebijakan, buat pernyataan kebijakan yang memberikan atau menolak akses sumber daya ke AWS IoT Core operasi. Untuk membuat pernyataan kebijakan yang memberikan semua klien untuk melakukan **iot:Connect**, ikuti langkah-langkah berikut:
 - Di bidang Efek kebijakan, pilih Izinkan. Hal ini memungkinkan semua klien yang memiliki kebijakan ini dilampirkan pada sertifikat mereka untuk melakukan tindakan yang tercantum dalam bidang Tindakan Kebijakan.
 - Di bidang Tindakan kebijakan, pilih tindakan kebijakan seperti **iot:Connect**. Tindakan kebijakan adalah tindakan yang harus dilakukan oleh perangkat Anda saat menjalankan program contoh dari Device SDK.
 - Di bidang Sumber daya kebijakan, masukkan sumber daya Amazon Resource Name (ARN) atau `* A *` untuk memilih klien (perangkat) apa pun.

Untuk membuat pernyataan kebijakan untuk **iot:Receive**, dan **iot:Publishiot:Subscribe**, pilih Tambahkan pernyataan baru dan ulangi langkah-langkahnya.

Policy effect	Policy action	Policy resource	
Allow ▼	iot:Connect ▼	*	Remove
Allow ▼	iot:Receive ▼	*	Remove
Allow ▼	iot:Publish ▼	*	Remove
Allow ▼	iot:Subscribe ▼	*	Remove

Note

Dalam start cepat ini, karakter wildcard (*) digunakan untuk kesederhanaan. Untuk keamanan yang lebih tinggi, Anda harus membatasi klien (perangkat) mana yang dapat menghubungkan dan mempublikasikan pesan dengan menentukan ARN klien alih-alih karakter wildcard sebagai sumber daya. ARN klien mengikuti format ini: `arn:aws:iot:your-region:your-aws-account:client/my-client-id`. Namun, Anda harus terlebih dahulu membuat sumber daya (seperti perangkat klien atau bayangan benda) sebelum Anda dapat menetapkan ARN ke kebijakan. Untuk informasi selengkapnya, lihat [sumber AWS IoT Core tindakan](#).

4. Setelah memasukkan informasi untuk kebijakan Anda, pilih Buat.

Untuk informasi selengkapnya, lihat [Bagaimana AWS IoT bekerja dengan IAM](#).

Buat objek benda

Perangkat yang terhubung ke AWS IoT Core diwakili oleh benda benda dalam AWS IoT registri. Objek benda mewakili perangkat tertentu atau entitas logis. Ini bisa berupa perangkat fisik atau sensor (misalnya, bola lampu atau sakelar lampu di dinding). Ini juga bisa menjadi entitas logis, seperti contoh aplikasi atau entitas fisik yang tidak terhubung AWS IoT, tetapi terkait dengan perangkat lain yang melakukannya (misalnya, mobil yang memiliki sensor mesin atau panel kontrol).

Untuk membuat sesuatu di AWS IoT konsol

1. Di [AWS IoT konsol](#), di menu sebelah kiri, pilih Semua perangkat dan kemudian pilih Things.
2. Pada halaman Things, pilih Create things.

3. Pada halaman Create things, pilih Create a single, lalu pilih Next.
4. Pada halaman Tentukan properti benda, untuk nama Thing, masukkan nama untuk barang Anda, seperti **MyIotThing**.

Pilih nama benda dengan hati-hati, karena Anda tidak dapat mengubah nama apa pun nanti.

Untuk mengubah nama hal, Anda harus membuat hal baru, memberikan nama baru, dan kemudian menghapus hal lama.

Note

Jangan gunakan informasi yang dapat diidentifikasi secara pribadi dalam nama barang Anda. Nama benda dapat muncul dalam komunikasi dan laporan yang tidak terenkripsi.


5. Biarkan sisa bidang di halaman ini kosong. Pilih Selanjutnya.
6. Pada halaman Konfigurasi sertifikat perangkat - opsional, pilih Buat otomatis sertifikat baru (disarankan). Pilih Selanjutnya.
7. Pada halaman Lampirkan kebijakan ke sertifikat - opsional, pilih kebijakan yang Anda buat di bagian sebelumnya. Di bagian itu, kebijakan itu dinamai, **My_Iot_Policy**. Pilih Buat sesuatu.
8. Pada halaman Unduh sertifikat dan kunci:
 1. Unduh setiap sertifikat dan file kunci dan simpan untuk nanti. Anda harus menginstal file-file ini di perangkat Anda.

Saat Anda menyimpan file sertifikat, beri mereka nama di tabel berikut. Ini adalah nama file yang digunakan dalam contoh selanjutnya.

Nama berkas sertifikat

File	Jalur file
Kunci privat	<code>private.pem.key</code>
Kunci publik	(tidak digunakan dalam contoh ini)
Sertifikat perangkat	<code>device.pem.crt</code>
Sertifikat Root CA	<code>Amazon-root-CA-1.pem</code>

2. Untuk mengunduh file CA root untuk file-file ini, pilih tautan Unduh file sertifikat CA root yang sesuai dengan jenis titik akhir data dan rangkaian sandi yang Anda gunakan. Dalam tutorial ini, pilih Unduh di sebelah kanan kunci bit RSA 2048: Amazon Root CA 1 dan unduh kunci RSA 2048 bit: File sertifikat Amazon Root CA 1.

 Important

Anda harus menyimpan file sertifikat sebelum meninggalkan halaman ini. Setelah Anda meninggalkan halaman ini di konsol, Anda tidak akan lagi memiliki akses ke file sertifikat.

Jika Anda lupa mengunduh file sertifikat yang Anda buat pada langkah ini, Anda harus keluar dari layar konsol ini, buka daftar hal-hal di konsol, hapus objek benda yang Anda buat, lalu mulai ulang prosedur ini dari awal.

3. Pilih Selesai.

Setelah Anda menyelesaikan prosedur ini, Anda akan melihat objek hal baru dalam daftar hal-hal Anda.

Konfigurasi perangkat Anda

Bagian ini menjelaskan cara mengonfigurasi perangkat Anda untuk terhubung AWS IoT Core. Jika Anda ingin memulai AWS IoT Core tetapi belum memiliki perangkat, Anda dapat membuat perangkat virtual dengan menggunakan Amazon EC2 atau Anda dapat menggunakan PC Windows atau Mac sebagai perangkat IoT.

Pilih opsi perangkat terbaik untuk Anda coba AWS IoT Core. Tentu saja, Anda dapat mencoba semuanya, tetapi cobalah hanya satu per satu. Jika Anda tidak yakin opsi perangkat mana yang terbaik untuk Anda, baca tentang cara memilih [opsi perangkat mana yang terbaik](#), lalu kembali ke halaman ini.

Opsi perangkat

- [Buat perangkat virtual dengan Amazon EC2](#)
- [Gunakan Windows atau Linux PC atau Mac sebagai AWS IoT perangkat](#)
- [Connect Raspberry Pi atau perangkat lain](#)

Buat perangkat virtual dengan Amazon EC2

Dalam tutorial ini, Anda akan membuat instans Amazon EC2 untuk berfungsi sebagai perangkat virtual Anda di cloud.

Untuk menyelesaikan tutorial ini, Anda memerlukan file Akun AWS. Jika Anda tidak memilikinya, selesaikan langkah-langkah yang dijelaskan [Siapkan Anda Akun AWS](#) sebelum Anda melanjutkan.

Dalam tutorial ini, Anda akan:

- [Mengatur instans Amazon EC2](#)
- [Instal Git, Node.js dan konfigurasi AWS CLI](#)
- [Buat AWS IoT sumber daya untuk perangkat virtual Anda](#)
- [Instal SDK AWS IoT Perangkat untuk JavaScript](#)
- [Jalankan aplikasi sampel](#)
- [Melihat pesan dari aplikasi sampel di AWS IoT konsol](#)

Mengatur instans Amazon EC2


Langkah-langkah berikut menunjukkan cara membuat instans Amazon EC2 yang akan bertindak sebagai perangkat virtual Anda sebagai pengganti perangkat fisik.

Jika ini adalah pertama kalinya Anda membuat instans Amazon EC2, Anda mungkin menemukan petunjuknya di [Memulai instans Amazon EC2Linux](#) agar lebih bermanfaat.

Untuk meluncurkan sebuah instans

1. Buka konsol Amazon EC2 di <https://console.aws.amazon.com/ec2/>.
2. Dari menu konsol di sebelah kiri, perluas bagian Instans dan pilih Instans. Dari dasbor Instans, pilih Luncurkan instance di sebelah kanan untuk menampilkan daftar konfigurasi dasar.
3. Di bagian Nama dan tag, masukkan nama untuk instance dan tambahkan tag secara opsional.
4. Di bagian Application and OS Images (Amazon Machine Image), pilih template AMI untuk instans Anda, seperti Amazon Linux 2 AMI (HVM). Perhatikan bahwa AMI ini ditandai “Memenuhi syarat tingkat gratis”.
5. Di bagian Jenis instans, Anda dapat memilih konfigurasi perangkat keras instans Anda. Pilih jenis `t2.micro`, yang dipilih secara default. Perhatikan bahwa jenis instans ini memenuhi syarat untuk tingkat gratis.

6. Di bagian Key pair (login), pilih nama key pair dari daftar drop-down atau pilih Create a new key pair untuk membuat yang baru. Saat membuat key pair baru, pastikan Anda mengunduh file kunci pribadi dan menyimpannya di tempat yang aman, karena ini adalah satu-satunya kesempatan Anda untuk mengunduh dan menyimpannya. Anda harus menyediakan nama pasangan kunci saat meluncurkan sebuah instans dan kunci privat yang sesuai setiap kali Anda terhubung dengan instans tersebut.

 Warning

Jangan memilih opsi Proceed without a key pair. Jika Anda meluncurkan instans tanpa pasangan kunci, Anda tidak dapat terhubung dengan instans tersebut.

7. Di bagian Pengaturan jaringan dan bagian Konfigurasi penyimpanan, Anda dapat menyimpan pengaturan default. Saat Anda siap, pilih Launch instance.
8. Halaman konfirmasi memberi tahu Anda bahwa instans Anda sedang diluncurkan. Pilih Lihat Instans untuk menutup halaman konfirmasi dan kembali ke konsol.
9. Pada layar Instans, Anda dapat melihat status peluncuran. Hanya butuh waktu singkat untuk meluncurkan sebuah instans. Saat Anda meluncurkan sebuah instans, status awalnya adalah pending. Setelah instans dimulai, keadaannya berubah menjadi running dan instans menerima sebuah nama DNS publik. (Jika kolom DNS Publik (IPv4) tersembunyi, pilih Kolom Tampilkan/Sembunyikan (ikon berbentuk roda gigi) di sudut kanan atas halaman, lalu pilih DNS Publik (IPv4).)
10. Proses ini mungkin memerlukan waktu beberapa menit sampai instans siap, sehingga Anda dapat terhubung dengannya. Periksa apakah instans Anda telah lulus pemeriksaan statusnya; Anda dapat melihat informasi ini di kolom Pemeriksaan Status.

Setelah instans baru Anda lulus pemeriksaan statusnya, lanjutkan ke prosedur berikutnya dan sambungkan ke sana.

Untuk menyambung ke instans Anda

Anda dapat menyambung ke instans menggunakan klien berbasis browser dengan memilih instans dari konsol Amazon EC2 dan memilih untuk menyambung menggunakan Amazon EC2 Instance Connect. Instance Connect menangani izin dan menyediakan koneksi yang berhasil.

1. Buka konsol Amazon EC2 di <https://console.aws.amazon.com/ec2/>.
2. Di menu sebelah kiri, pilih Instans.

3. Pilih instans, lalu pilih Hubungkan.
4. Pilih Instans Amazon EC2 Connect, Connect.

Anda sekarang harus memiliki jendela Amazon EC2 Instance Connect yang masuk ke instans Amazon EC2 baru Anda.

Instal Git, Node.js dan konfigurasi AWS CLI

Di bagian ini, Anda akan menginstal Git dan Node.js pada instance Linux Anda.

Untuk menginstal Git

1. Di jendela Amazon EC2 Instance Connect, perbarui instans Anda dengan menggunakan perintah berikut.

```
sudo yum update -y
```

2. Di jendela Amazon EC2 Instance Connect, instal Git dengan menggunakan perintah berikut.

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

3. Untuk memeriksa apakah Git telah diinstal dan versi Git saat ini, jalankan perintah berikut:

```
git --version
```

Untuk menginstal Node.js

1. Di jendela Amazon EC2 Instance Connect, instal node version manager (nvm) dengan menggunakan perintah berikut.

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


Kami akan menggunakan nvm untuk menginstal Node.js karena nvm dapat menginstal beberapa versi Node.js dan memungkinkan Anda untuk beralih di antara mereka.

2. Di jendela Amazon EC2 Instance Connect, aktifkan nvm dengan menggunakan perintah ini.

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

3. Di jendela Amazon EC2 Instance Connect, gunakan nvm untuk menginstal versi terbaru Node.js dengan menggunakan perintah ini.

```
nvm install 16
```

 Note

Ini menginstal rilis LTS terbaru dari Node.js.

Menginstal Node.js juga menginstal Node Package Manager (npm) sehingga Anda dapat menginstal modul tambahan sesuai kebutuhan.

4. Di jendela Amazon EC2 Instance Connect, uji apakah Node.js diinstal dan berjalan dengan benar menggunakan perintah ini.

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

Tutorial ini membutuhkan Node v10.0 atau yang lebih baru. Untuk informasi selengkapnya, lihat [Tutorial: Menyiapkan Node.js di Instans Amazon EC2](#).

Untuk mengkonfigurasi AWS CLI

Instans Amazon EC2 Anda sudah dimuat sebelumnya dengan file. AWS CLI Namun, Anda harus melengkapi AWS CLI profil Anda. Untuk informasi selengkapnya tentang cara mengonfigurasi CLI Anda, lihat [Mengonfigurasi CLI. AWS CLI](#)

1. Contoh berikut menunjukkan nilai sampel. Gantilah dengan nilai-nilai Anda sendiri. Anda dapat menemukan nilai-nilai ini di [AWS konsol Anda di info akun Anda di bawah Kredensi keamanan](#).

Di jendela Amazon EC2 Instance Connect, masukkan perintah ini:

```
aws configure
```

Kemudian masukkan nilai dari akun Anda pada petunjuk yang ditampilkan.

```
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. Anda dapat menguji AWS CLI konfigurasi Anda dengan perintah ini:

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

Jika Anda AWS CLI dikonfigurasi dengan benar, perintah harus mengembalikan alamat titik akhir dari alamat Anda Akun AWS.

Buat AWS IoT sumber daya untuk perangkat virtual Anda

Bagian ini menjelaskan cara menggunakan AWS CLI untuk membuat objek benda dan file sertifikatnya langsung di perangkat virtual. Ini dilakukan langsung pada perangkat untuk menghindari potensi komplikasi yang mungkin timbul dari menyalinnya ke perangkat dari komputer lain. Di bagian ini, Anda akan membuat sumber daya berikut untuk perangkat virtual Anda:

- Objek benda untuk mewakili perangkat virtual Anda AWS IoT.
- Sertifikat untuk mengautentikasi perangkat virtual Anda.
- Dokumen kebijakan untuk mengotorisasi perangkat virtual Anda untuk terhubung ke AWS IoT, dan untuk mempublikasikan, menerima, dan berlangganan pesan.

Untuk membuat objek AWS IoT benda di instance Linux Anda

Perangkat yang terhubung ke AWS IoT diwakili oleh benda benda dalam AWS IoT registri. Objek benda mewakili perangkat tertentu atau entitas logis. Dalam hal ini, objek benda Anda akan mewakili perangkat virtual Anda, instans Amazon EC2 ini.

1. Di jendela Amazon EC2 Instance Connect, jalankan perintah berikut untuk membuat objek benda Anda.

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

2. Respons JSON akan terlihat seperti ini:

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

```
}

```

Untuk membuat dan melampirkan AWS IoT kunci dan sertifikat di instance Linux Anda

[create-keys-and-certificate](#) Perintah tersebut membuat sertifikat klien yang ditandatangani oleh otoritas sertifikat Amazon Root. Sertifikat ini digunakan untuk mengautentikasi identitas perangkat virtual Anda.

1. Di jendela Amazon EC2 Instance Connect, buat direktori untuk menyimpan sertifikat dan file kunci Anda.

```
mkdir ~/certs

```

2. Di jendela Amazon EC2 Instance Connect, unduh salinan sertifikat Amazon Certificate Authority (CA) dengan menggunakan perintah ini.

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

```

3. Di jendela Amazon EC2 Instance Connect, jalankan perintah berikut untuk membuat kunci pribadi, kunci publik, dan file sertifikat X.509. Perintah ini juga mendaftarkan dan mengaktifkan sertifikat dengan 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"

```

Respon tersebut terlihat seperti berikut. Simpan `certificateArn` sehingga Anda dapat menggunakannya dalam perintah berikutnya. Anda akan memerlukannya untuk melampirkan sertifikat Anda ke barang Anda dan melampirkan kebijakan ke sertifikat di langkah selanjutnya.

```
{
  "certificateArn": "arn:aws:iot:us-
west-2:123456789012:cert/9894ba17925e663f1d29c23af4582b8e3b7619c31f3fbd93adcb51ae54b83dc2",
  "certificateId":
  "9894ba17925e663f1d29c23af4582b8e3b7619c31f3fbd93adcb51ae54b83dc2",
  "certificatePem": "
-----BEGIN CERTIFICATE-----

```

```

MIICiTCCExAMPLE6m7oRw0uX0jANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC
VVMxCzAJBgNVBAGExAMPLEAwDgYDVQQHEwdTZWF0dGxLMQ8wDQYDVQQKEwZBbWF6
b24xFDASBgNVBA5TC0lBTSExAMPLE2xLMRIwEAYDVQQDEwLUZXN0Q2lsYWMxHzAd
BgkqhkiG9w0BCQEWEG5vb25lQGfYExAMPLEb20wHhcNMTEwNDI1MjA0NTIxWhcN
MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCExAMPLEJBgNVBAGTAldBMRAdG9YD
VQQHEwdTZWF0dGxLMQ8wDQYDVQQKEwZBbWF6b24xFDExAMPLEsTC0lBTSBDb25z
b2xLMRIwEAYDVQQDEwLUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQExAMPLE25lQGfY
YXpvi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+aEXAMPLE
EXAMPLEfEvySWtC2XADZ4nB+BLyGVIk60CpiwsZ3G93vUEI03IyNoH/f0wYK8m9T
rDHudUZExAMPLELg5M43q7Wgc/MbQITx0USQv7c7ugFFDzQGBzZswY6786m86gpE
Ibb30hjZnzcVQExAMPLEWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAAtCu4
nUhVVxYUntneD9+h8Mg9qExAMPLEyExzyLwaxLAoo7TJHidbtS4J5iNmZgXL0Fkb
FFBjvSfpJILJ0z0z0z0z0z0z0z0z0z0z0z0z0z0z0z0z0z0z0z0z0z0z0z0z0z0z0z
NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
-----END CERTIFICATE-----\n",
  "keyPair": {
    "PublicKey": "-----BEGIN PUBLIC
KEY-----\nMIIBIjANBgkqhkiG9w0BAQEAFAAAOCAG8AMIIBCGKAQEAEXAMPLE1nnyJwKSMHw4h
\nMMExAMPLEuuN/dMAS3fyce8DW/4+EXAMPLEyjmof/YVF/
gHr99VEEXAMPLE5VF13\n59VK7cEXAMPLE67GK+y+jikqX0gHh/xJTWO
+sGpWEXAMPLEDz18x0d2ka4tCzuWEXAMPLEeahJbYkCPUBSU8opVkr7qkEXAMPLE1DR6sx2Hocli00Ltu6Fkw91swQWE
\nGB3ZPrNh0PzQYvjUstZeccyNCx2EXAMPLEvp9mQ0UXP6plfgxwKRX2fEXAMPLEda
\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"
  }
}

```

- Di jendela Amazon EC2 Instance Connect, lampirkan objek benda Anda ke sertifikat yang baru saja Anda buat dengan menggunakan perintah berikut dan *CertificateArn* dalam respons dari perintah sebelumnya.

```

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

```

Jika berhasil, perintah ini tidak menampilkan output apapun.

Untuk membuat dan melampirkan kebijakan

1. Di jendela Amazon EC2 Instance Connect, buat file kebijakan dengan menyalin dan menempelkan dokumen kebijakan ini ke file bernama. **~/policy.json**

Jika Anda tidak memiliki editor Linux favorit, Anda dapat membukanya, dengan menggunakan perintah ini.

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

Tempelkan dokumen kebijakan `policy.json` ke dalamnya. Masukkan `ctrl-x` untuk keluar dari nano editor dan simpan file.

Isi dokumen kebijakan untuk `policy.json`.

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

2. Di jendela Amazon EC2 Instance Connect, buat kebijakan Anda menggunakan perintah berikut.

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

Output:

```
{
  "policyName": "MyIotThingPolicy",
  "policyArn": "arn:aws:iot:your-region: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"
}
```

- Di jendela Amazon EC2 Instance Connect, lampirkan kebijakan ke sertifikat perangkat virtual Anda dengan menggunakan perintah berikut.

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

Jika berhasil, perintah ini tidak menampilkan output apapun.

Instal SDK AWS IoT Perangkat untuk JavaScript

Di bagian ini, Anda akan menginstal AWS IoT Device SDK untuk JavaScript, yang berisi kode yang dapat digunakan aplikasi untuk berkomunikasi dengan AWS IoT dan program sampel. Untuk informasi selengkapnya, lihat [SDK AWS IoT Perangkat untuk JavaScript GitHub repositori](#).

Untuk menginstal AWS IoT Device SDK untuk JavaScript instans Linux Anda

1. Di jendela Amazon EC2 Instance Connect, kloning AWS IoT Device SDK untuk JavaScript repositori ke `aws-iot-device-sdk-js-v2` direktori direktori home Anda dengan menggunakan perintah ini.

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

2. Arahkan ke `aws-iot-device-sdk-js-v2` direktori yang Anda buat pada langkah sebelumnya.

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

3. Gunakan npm untuk menginstal SDK.

```
npm install
```

Jalankan aplikasi sampel

Perintah di bagian berikutnya mengasumsikan bahwa file kunci dan sertifikat Anda disimpan di perangkat virtual Anda seperti yang ditunjukkan dalam tabel ini.

Nama berkas sertifikat

File	Jalur file
Kunci privat	<code>~/certs/private.pem.key</code>
Sertifikat perangkat	<code>~/certs/device.pem.crt</code>
Sertifikat Root CA	<code>~/certs/Amazon-root-CA-1.pem</code>

Di bagian ini, Anda akan menginstal dan menjalankan aplikasi `pub-sub.js` sampel yang ditemukan di `aws-iot-device-sdk-js-v2/samples/node` direktori AWS IoT Device SDK untuk JavaScript. Aplikasi ini menunjukkan bagaimana perangkat, instans Amazon EC2 Anda, menggunakan pustaka MQTT untuk menerbitkan dan berlangganan pesan MQTT. Aplikasi `pub-sub.js` sampel berlangganan `topiktopic_1`, menerbitkan 10 pesan ke topik itu dan menampilkan pesan saat diterima dari broker pesan.

Untuk menginstal dan menjalankan aplikasi sampel

1. Di jendela Amazon EC2 Instance Connect, navigasikan ke `aws-iot-device-sdk-js-v2/samples/node/pub_sub` direktori yang dibuat SDK dan instal aplikasi sampel dengan menggunakan perintah ini.

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

2. Di jendela Amazon EC2 Instance Connect, dapatkan *your-iot-endpoint* dari AWS IoT dengan menggunakan perintah ini.

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

3. Di jendela Amazon EC2 Instance Connect, masukkan *your-iot-endpoint* seperti yang ditunjukkan dan jalankan perintah ini.

```
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
```

Aplikasi sampel:

1. Terhubung ke AWS IoT Core akun Anda.
2. Berlangganan topik pesan, `topic_1`, dan menampilkan pesan yang diterimanya pada topik itu.
3. Menerbitkan 10 pesan ke topik, `topic_1`.
4. Menampilkan output yang mirip dengan berikut ini:

```
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}
```

Jika Anda mengalami masalah dalam menjalankan aplikasi sampel, tinjau [the section called “Memecahkan masalah dengan aplikasi sampel”](#).

Anda juga dapat menambahkan `--verbosity debug` parameter ke baris perintah sehingga aplikasi sampel menampilkan pesan terperinci tentang apa yang dilakukannya. Informasi itu mungkin memberi Anda bantuan yang Anda butuhkan untuk memperbaiki masalah.

Melihat pesan dari aplikasi sampel di AWS IoT konsol

Anda dapat melihat pesan aplikasi sampel saat mereka melewati broker pesan dengan menggunakan klien pengujian MQTT di konsol.AWS IoT

Untuk melihat pesan MQTT yang diterbitkan oleh aplikasi sampel

1. Ulasan [Lihat pesan MQTT dengan klien MQTT AWS IoT](#). Ini membantu Anda mempelajari cara menggunakan klien pengujian MQTT di AWS IoT konsol untuk melihat pesan MQTT saat mereka melewati broker pesan.
2. Buka klien pengujian MQTT di konsol.AWS IoT
3. Di Berlangganan topik, Berlangganan topik, topic_1.
4. Di jendela Amazon EC2 Instance Connect, jalankan aplikasi sampel lagi dan tonton pesan di klien pengujian MQTT di konsol.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
```

[Untuk informasi selengkapnya tentang MQTT dan cara AWS IoT Core mendukung protokol, lihat MQTT.](#)

Gunakan Windows atau Linux PC atau Mac sebagai AWS IoT perangkat

Dalam tutorial ini, Anda akan mengkonfigurasi komputer pribadi untuk digunakan dengan AWS IoT. Instruksi ini mendukung PC dan Mac Windows dan Linux. Untuk mencapai ini, Anda perlu menginstal beberapa perangkat lunak di komputer Anda. Jika Anda tidak ingin menginstal perangkat lunak di komputer Anda, Anda dapat mencoba [Buat perangkat virtual dengan Amazon EC2](#), yang menginstal semua perangkat lunak pada mesin virtual.

Dalam tutorial ini, Anda akan:

- [Siapkan komputer pribadi Anda](#)
- [Instal Git, Python, dan AWS IoT Device SDK untuk Python](#)
- [Menyiapkan kebijakan dan menjalankan contoh aplikasi](#)
- [Melihat pesan dari aplikasi sampel di AWS IoT konsol](#)
- [Jalankan contoh Berlangganan Bersama dengan Python](#)

Siapkan komputer pribadi Anda

Untuk menyelesaikan tutorial ini, Anda memerlukan PC Windows atau Linux atau Mac dengan koneksi ke internet.

Sebelum Anda melanjutkan ke langkah berikutnya, pastikan Anda dapat membuka jendela baris perintah di komputer Anda. Gunakan cmd.exe pada PC Windows. Pada PC Linux atau Mac, gunakan Terminal.

Instal Git, Python, dan AWS IoT Device SDK untuk Python

Di bagian ini, Anda akan menginstal Python, dan AWS IoT Device SDK untuk Python di komputer Anda.

Instal versi terbaru Git dan Python

Untuk mengunduh dan menginstal Git dan Python di komputer Anda

1. Periksa untuk melihat apakah Anda telah menginstal Git di komputer Anda. Masukkan perintah ini di baris perintah.

```
git --version
```

Jika perintah menampilkan versi Git, Git diinstal dan Anda dapat melanjutkan ke langkah berikutnya.

Jika perintah menampilkan kesalahan, buka <https://git-scm.com/download> dan instal Git untuk komputer Anda.

2. Periksa untuk melihat apakah Anda telah menginstal Python. Masukkan perintah di baris perintah.

```
python -V
```

Note

Jika perintah ini memberikan kesalahan: Python was not found, mungkin karena sistem operasi Anda memanggil Python v3.x yang dapat dieksekusi sebagai Python3. Dalam hal ini, ganti semua contoh python dengan python3 dan lanjutkan sisa tutorial ini.

Jika perintah menampilkan versi Python, Python sudah diinstal. Tutorial ini membutuhkan Python v3.7 atau yang lebih baru.

3. Jika Python diinstal, Anda dapat melewati langkah-langkah lainnya di bagian ini. Jika tidak, lanjutkan.
4. Buka <https://www.python.org/downloads/> dan unduh penginstal untuk komputer Anda.
5. Jika unduhan tidak secara otomatis mulai diinstal, jalankan program yang diunduh untuk menginstal Python.
6. Verifikasi instalasi Python.

```
python -V
```

Konfirmasikan bahwa perintah menampilkan versi Python. Jika versi Python tidak ditampilkan, coba unduh dan instal Python lagi.

Instal AWS IoT Device SDK untuk Python

Untuk menginstal AWS IoT Device SDK untuk Python di komputer Anda

1. Instal v2 dari AWS IoT Device SDK untuk Python.

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

2. Kloning AWS IoT Device SDK untuk repositori Python ke direktori `aws-iot-device-sdk-python-v2` direktori home Anda. Prosedur ini mengacu pada direktori dasar untuk file yang Anda instal sebagai *rumah*.

Lokasi sebenarnya dari direktori *home* tergantung pada sistem operasi Anda.

Linux/macOS

Di macOS dan Linux, direktori *home* adalah. `~`

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

Windows

Di Windows, Anda dapat menemukan jalur direktori *home* dengan menjalankan perintah ini di cmd jendela.

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

Note

Jika Anda menggunakan Windows PowerShell sebagai lawan dari `cmd.exe`, maka gunakan perintah berikut.

```
echo $home
```


Untuk informasi selengkapnya, lihat [AWS IoT Device SDK for GitHub Python repositori](#).

Bersiaplah untuk menjalankan aplikasi sampel

Untuk mempersiapkan sistem Anda untuk menjalankan aplikasi sampel

- Buatlah direktori `certs`. Ke dalam `certs` direktori, salin kunci pribadi, sertifikat perangkat, dan file sertifikat CA root yang Anda simpan saat Anda membuat dan mendaftarkan objek benda tersebut [the section called “Buat AWS IoT sumber daya”](#). Nama file dari setiap file di direktori tujuan harus cocok dengan yang ada di tabel.

Perintah di bagian berikutnya mengasumsikan bahwa file kunci dan sertifikat Anda disimpan di perangkat Anda seperti yang ditunjukkan dalam tabel ini.

Linux/macOS

Jalankan perintah ini untuk membuat `certs` subdirektori yang akan Anda gunakan ketika Anda menjalankan aplikasi sampel.

```
mkdir ~/certs
```

Ke subdirektori baru, salin file ke jalur file tujuan yang ditunjukkan pada tabel berikut.

Nama berkas sertifikat

File	Jalur file
Kunci privat	<code>~/certs/private.pem.key</code>
Sertifikat perangkat	<code>~/certs/device.pem.crt</code>
Sertifikat Root CA	<code>~/certs/Amazon-root-CA-1.pem</code>

Jalankan perintah ini untuk membuat daftar file dalam `certs` direktori dan membandingkannya dengan yang tercantum dalam tabel.

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

Windows

Jalankan perintah ini untuk membuat `certs` subdirektori yang akan Anda gunakan ketika Anda menjalankan aplikasi sampel.

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

Ke subdirektori baru, salin file ke jalur file tujuan yang ditunjukkan pada tabel berikut.

Nama berkas sertifikat

File	Jalur file
Kunci privat	%USERPROFILE%\certs\private.pem.key
Sertifikat perangkat	%USERPROFILE%\certs\device.pem.crt
Sertifikat Root CA	%USERPROFILE%\certs\Amazon-root-CA-1.pem

Jalankan perintah ini untuk membuat daftar file dalam `certs` direktori dan membandingkannya dengan yang tercantum dalam tabel.

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

Menyiapkan kebijakan dan menjalankan contoh aplikasi

Di bagian ini, Anda akan menyiapkan kebijakan dan menjalankan skrip `pubsub.py` sampel yang ditemukan di `aws-iot-device-sdk-python-v2/samples` direktori AWS IoT Device SDK for Python. Skrip ini menunjukkan bagaimana perangkat Anda menggunakan pustaka MQTT untuk menerbitkan dan berlangganan pesan MQTT.

Aplikasi `pubsub.py` sampel berlangganan `topiktest/topic`, menerbitkan 10 pesan ke topik itu, dan menampilkan pesan saat diterima dari broker pesan.

Untuk menjalankan skrip `pubsub.py` sampel, Anda memerlukan informasi berikut:

Nilai parameter aplikasi

Parameter	Di mana menemukan nilainya
<i>your-iot-endpoint</i>	<ol style="list-style-type: none"> Di AWS IoT konsol, di menu kiri, pilih Pengaturan. Pada halaman Pengaturan, titik akhir Anda ditampilkan di bagian titik akhir data perangkat.

your-iot-endpoint Nilai memiliki format: *endpoint_id*-ats.iot.*region*.amazonaws.com, misalnya, a3qj468EXAMPLE-ats.iot.us-west-2.amazonaws.com.

Sebelum menjalankan skrip, pastikan kebijakan hal Anda memberikan izin untuk skrip sampel untuk terhubung, berlangganan, menerbitkan, dan menerima.

Untuk menemukan dan meninjau dokumen kebijakan untuk sumber daya sesuatu

- Di [AWS IoT konsol](#), di daftar Things, temukan sumber daya benda yang mewakili perangkat Anda.
- Pilih tautan Nama dari sumber daya benda yang mewakili perangkat Anda untuk membuka halaman Detail hal.
- Di halaman Detail hal, di tab Sertifikat, pilih sertifikat yang dilampirkan ke sumber daya benda. Seharusnya hanya ada satu sertifikat dalam daftar. Jika ada lebih dari satu, pilih sertifikat yang filenya diinstal pada perangkat Anda dan yang akan digunakan untuk terhubung AWS IoT Core.

Di halaman Detail sertifikat, di tab Kebijakan, pilih kebijakan yang dilampirkan pada sertifikat. Seharusnya hanya ada satu. Jika ada lebih dari satu, ulangi langkah berikutnya untuk masing-masing untuk memastikan bahwa setidaknya satu kebijakan memberikan akses yang diperlukan.

- Di halaman Ringkasan kebijakan, temukan editor JSON dan pilih Edit dokumen kebijakan untuk meninjau dan mengedit dokumen kebijakan sesuai kebutuhan.
- Kebijakan JSON ditampilkan dalam contoh berikut. Dalam "Resource" elemen, ganti *region:account* dengan Anda Wilayah AWS dan Akun AWS di setiap Resource nilai.

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

```
{
  "Effect": "Allow",
  "Action": [
    "iot:Publish",
    "iot:Receive"
  ],
  "Resource": [
    "arn:aws:iot:region:account:topic/test/topic"
  ]
},
{
  "Effect": "Allow",
  "Action": [
    "iot:Subscribe"
  ],
  "Resource": [
    "arn:aws:iot:region:account:topicfilter/test/topic"
  ]
},
{
  "Effect": "Allow",
  "Action": [
    "iot:Connect"
  ],
  "Resource": [
    "arn:aws:iot:region:account:client/test-*"
  ]
}
]
```

Linux/macOS

Untuk menjalankan skrip sampel di Linux/macOS

1. Di jendela baris perintah Anda, arahkan ke `~/aws-iot-device-sdk-python-v2/samples/node/pub_sub` direktori yang dibuat SDK dengan menggunakan perintah ini.

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

2. Di jendela baris perintah Anda, ganti *your-iot-endpoint* seperti yang ditunjukkan dan jalankan perintah ini.

```
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

Untuk menjalankan aplikasi sampel pada PC Windows

1. Di jendela baris perintah Anda, navigasikan ke %USERPROFILE%\aws-iot-device-sdk-python-v2\samples direktori yang dibuat SDK dan instal aplikasi sampel dengan menggunakan perintah ini.

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

2. Di jendela baris perintah Anda, ganti *your-iot-endpoint* seperti yang ditunjukkan dan jalankan perintah ini.

```
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
```

Contoh skrip:

1. Terhubung ke AWS IoT Core untuk akun Anda.
2. Berlangganan topik pesan, uji/topik, dan menampilkan pesan yang diterimanya pada topik itu.
3. Menerbitkan 10 pesan ke topik, uji/topik.
4. Menampilkan output yang mirip dengan berikut ini:

```
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!
```

Jika Anda mengalami masalah dalam menjalankan aplikasi sampel, tinjau [the section called “Memecahkan masalah dengan aplikasi sampel”](#).

Anda juga dapat menambahkan `--verbosity Debug` parameter ke baris perintah sehingga aplikasi sampel menampilkan pesan terperinci tentang apa yang dilakukannya. Informasi itu dapat membantu Anda memperbaiki masalah.

Melihat pesan dari aplikasi sampel di AWS IoT konsol

Anda dapat melihat pesan aplikasi sampel saat mereka melewati broker pesan dengan menggunakan klien pengujian MQTT di konsol AWS IoT

Untuk melihat pesan MQTT yang diterbitkan oleh aplikasi sampel

1. Ulasan [Lihat pesan MQTT dengan klien MQTT AWS IoT](#). Ini membantu Anda mempelajari cara menggunakan klien pengujian MQTT di AWS IoT konsol untuk melihat pesan MQTT saat mereka melewati broker pesan.
2. Buka klien pengujian MQTT di konsol AWS IoT
3. Di Berlangganan topik, berlangganan topik, uji/topik.
4. Di jendela baris perintah Anda, jalankan aplikasi sampel lagi dan tonton pesan di klien MQTT di konsol 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
```

[Untuk informasi selengkapnya tentang MQTT dan cara AWS IoT Core mendukung protokol, lihat MQTT.](#)

Jalankan contoh Berlangganan Bersama dengan Python

AWS IoT Core mendukung [Langganan Bersama](#) untuk MQTT 3 dan MQTT 5. Langganan Bersama memungkinkan beberapa klien untuk berbagi langganan ke suatu topik dan hanya satu klien yang akan menerima pesan yang dipublikasikan ke topik tersebut menggunakan distribusi acak. Untuk menggunakan Langganan Bersama, klien berlangganan [filter topik](#) Langganan Bersama:\$share/{ShareName}/{TopicFilter}.

Untuk menyiapkan kebijakan dan menjalankan contoh Langganan Bersama

1. Untuk menjalankan contoh Berlangganan Bersama, Anda harus menyiapkan kebijakan hal Anda seperti yang didokumentasikan dalam Langganan Bersama [MQTT 5](#).
2. Untuk menjalankan contoh Berlangganan Bersama, jalankan perintah berikut.

Linux/macOS

Untuk menjalankan skrip sampel di Linux/macOS

1. Di jendela baris perintah Anda, arahkan ke ~/aws-iot-device-sdk-python-v2/samples direktori yang dibuat SDK dengan menggunakan perintah ini.

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

2. Di jendela baris perintah Anda, ganti *your-iot-endpoint* seperti yang ditunjukkan dan jalankan perintah ini.

```
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

Untuk menjalankan aplikasi sampel pada PC Windows

1. Di jendela baris perintah Anda, navigasikan ke %USERPROFILE%\aws-iot-device-sdk-python-v2\samples direktori yang dibuat SDK dan instal aplikasi sampel dengan menggunakan perintah ini.

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

2. Di jendela baris perintah Anda, ganti *your-iot-endpoint* seperti yang ditunjukkan dan jalankan perintah ini.

```
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

Anda dapat secara opsional menentukan pengidentifikasi grup berdasarkan kebutuhan Anda saat menjalankan sampel (mis., `--group_identifier consumer`). Jika Anda tidak menentukan satu, `python-sample` adalah pengidentifikasi grup default.

3. Output di baris perintah Anda dapat terlihat seperti berikut:

```
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. Buka klien uji MQTT di konsol.AWS IoT Di Berlangganan topik, berlangganan topik Langganan Bersama seperti:\$share/consumer/test/topic. Anda dapat menentukan pengidentifikasi grup berdasarkan kebutuhan Anda saat menjalankan sampel (mis.,--group_identifier consumer). Jika Anda tidak menentukan pengidentifikasi grup, nilai defaultnya adalahpython-sample. Untuk informasi selengkapnya, lihat [contoh Python Berlangganan Bersama MQTT 5](#) dan Langganan [Bersama](#) dari Panduan Pengembang.AWS IoT Core

Di jendela baris perintah Anda, jalankan aplikasi sampel lagi dan saksikan distribusi pesan di klien pengujian MQTT AWS IoT konsol dan baris perintah.

Subscribe to a topic | **Publish to a topic**

Topic filter [Info](#)
The topic filter describes the topic(s) to which you want to subscribe. The topic filter can include MQTT wildcard characters.

\$share/consumer/test/topic

```
[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>
[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>
[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! [5]"
[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! [6]"
[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
[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]: 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'.
[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!
```

Subscriptions \$share/consumer/test/topic

\$share/consumer/test/topic ✕

▼ test/topic	April 21, 2023, 14:43:10 (UTC-0700)
"Hello World! [10]"	
▶ Properties	
▼ test/topic	April 21, 2023, 14:43:07 (UTC-0700)
"Hello World! [7]"	
▶ Properties	
▼ test/topic	April 21, 2023, 14:43:03 (UTC-0700)
"Hello World! [4]"	
▶ Properties	
▼ test/topic	April 21, 2023, 14:43:00 (UTC-0700)
"Hello World! [1]"	
▶ Properties	

Connect Raspberry Pi atau perangkat lain

Di bagian ini, kita akan mengkonfigurasi Raspberry Pi untuk digunakan dengan AWS IoT. Jika Anda memiliki perangkat lain yang ingin Anda sambungkan, petunjuk untuk Raspberry Pi menyertakan referensi yang dapat membantu Anda menyesuaikan instruksi ini ke perangkat Anda.

Ini biasanya memakan waktu sekitar 20 menit, tetapi bisa memakan waktu lebih lama jika Anda memiliki banyak peningkatan perangkat lunak sistem untuk diinstal.

Dalam tutorial ini, Anda akan:

- [Siapkan perangkat Anda](#)
- [Instal alat dan pustaka yang diperlukan untuk AWS IoT Device SDK](#)
- [Instal SDK AWS IoT Perangkat](#)
- [Instal dan jalankan aplikasi sampel](#)
- [Melihat pesan dari aplikasi sampel di AWS IoT konsol](#)

⚠ Important

Menyesuaikan instruksi ini ke perangkat dan sistem operasi lain dapat menjadi tantangan. Anda harus memahami perangkat Anda dengan cukup baik untuk dapat menafsirkan instruksi ini dan menerapkannya ke perangkat Anda.

Jika Anda mengalami kesulitan saat mengonfigurasi perangkat AWS IoT, Anda dapat mencoba salah satu opsi perangkat lain sebagai alternatif, seperti [Buat perangkat virtual dengan Amazon EC2](#) atau [Gunakan Windows atau Linux PC atau Mac sebagai AWS IoT perangkat](#).

Siapkan perangkat Anda

Tujuan dari langkah ini adalah untuk mengumpulkan apa yang Anda perlukan untuk mengkonfigurasi perangkat Anda sehingga dapat memulai sistem operasi (OS), terhubung ke internet, dan memungkinkan Anda untuk berinteraksi dengannya di antarmuka baris perintah.

Untuk menyelesaikan tutorial ini, Anda memerlukan hal berikut:

- Sebuah Akun AWS. Jika Anda tidak memilikinya, selesaikan langkah-langkah yang dijelaskan [Siapkan Anda Akun AWS](#) sebelum Anda melanjutkan.
- [Raspberry Pi 3 Model B](#) atau model yang lebih baru. Ini mungkin bekerja pada versi sebelumnya dari Raspberry Pi, tetapi mereka belum diuji.
- [Raspberry Pi OS \(32-bit\)](#) atau yang lebih baru. Kami merekomendasikan menggunakan versi terbaru dari Raspberry Pi OS. Versi OS sebelumnya mungkin berfungsi, tetapi belum diuji.

Untuk menjalankan contoh ini, Anda tidak perlu menginstal desktop dengan antarmuka pengguna grafis (GUI); Namun, jika Anda baru mengenal Raspberry Pi dan perangkat keras Raspberry Pi Anda mendukungnya, menggunakan desktop dengan GUI mungkin lebih mudah.

- Ethernet atau WiFi koneksi.
- Keyboard, mouse, monitor, kabel, catu daya, dan perangkat keras lain yang diperlukan oleh perangkat Anda.

⚠ Important

Sebelum Anda melanjutkan ke langkah berikutnya, perangkat Anda harus menginstal, mengkonfigurasi, dan menjalankan sistem operasinya. Perangkat harus terhubung ke

internet dan Anda harus dapat mengakses perangkat dengan menggunakan antarmuka baris perintahnya. Akses baris perintah dapat melalui keyboard, mouse, dan monitor yang terhubung langsung, atau dengan menggunakan antarmuka jarak jauh terminal SSH.

Jika Anda menjalankan sistem operasi pada Raspberry Pi Anda yang memiliki antarmuka pengguna grafis (GUI), buka jendela terminal pada perangkat dan lakukan instruksi berikut di jendela itu. Jika tidak, jika Anda terhubung ke perangkat Anda dengan menggunakan terminal jarak jauh, seperti PuTTY, buka terminal jarak jauh ke perangkat Anda dan gunakan itu.

Instal alat dan pustaka yang diperlukan untuk AWS IoT Device SDK

Sebelum Anda menginstal SDK AWS IoT Perangkat dan kode sampel, pastikan sistem Anda mutakhir dan memiliki alat dan pustaka yang diperlukan untuk menginstal SDK.

1. Perbarui sistem operasi dan instal perpustakaan yang diperlukan

Sebelum Anda menginstal AWS IoT Device SDK, jalankan perintah ini di jendela terminal pada perangkat Anda untuk memperbarui sistem operasi dan menginstal pustaka yang diperlukan.

```
sudo apt-get update
```

```
sudo apt-get upgrade
```

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

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

2. Instal Git

Jika sistem operasi perangkat Anda tidak disertakan dengan Git yang diinstal, Anda harus menginstalnya untuk menginstal AWS IoT Device SDK. JavaScript

a. Uji untuk melihat apakah Git sudah diinstal dengan menjalankan perintah ini.

```
git --version
```

b. Jika perintah sebelumnya mengembalikan versi Git, Git sudah diinstal dan Anda dapat melompat ke Langkah 3.

- c. Jika kesalahan ditampilkan saat Anda menjalankan git perintah, instal Git dengan menjalankan perintah ini.

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

- d. Uji lagi untuk melihat apakah Git diinstal dengan menjalankan perintah ini.

```
git --version
```

- e. Jika Git diinstal, lanjutkan ke bagian berikutnya. Jika tidak, pecahkan masalah dan perbaiki kesalahan sebelum melanjutkan. Anda memerlukan Git untuk menginstal AWS IoT Device SDK untuk JavaScript.

Instal SDK AWS IoT Perangkat

Instal SDK AWS IoT Perangkat.

Python

Di bagian ini, Anda akan menginstal Python, alat pengembangannya, dan AWS IoT Device SDK untuk Python di perangkat Anda. Instruksi ini untuk Raspberry Pi yang menjalankan OS Raspberry Pi terbaru. Jika Anda memiliki perangkat lain atau menggunakan sistem operasi lain, Anda mungkin perlu menyesuaikan petunjuk ini untuk perangkat Anda.

1. Instal Python dan alat pengembangannya

AWS IoT Perangkat SDK untuk Python memerlukan Python v3.5 atau yang lebih baru untuk diinstal pada Raspberry Pi Anda.

Di jendela terminal ke perangkat Anda, jalankan perintah ini.

1. Jalankan perintah ini untuk menentukan versi Python yang diinstal pada perangkat Anda.

```
python3 --version
```

Jika Python diinstal, itu akan menampilkan versinya.

2. Jika versi yang ditampilkan Python 3.5 atau lebih besar, Anda dapat melompat ke Langkah 2.

3. Jika versi yang ditampilkan kurang dari Python 3.5, Anda dapat menginstal versi yang benar dengan menjalankan perintah ini.

```
sudo apt install python3
```

4. Jalankan perintah ini untuk mengonfirmasi bahwa versi Python yang benar sekarang diinstal.

```
python3 --version
```

2. Uji untuk pip3

Di jendela terminal ke perangkat Anda, jalankan perintah ini.

1. Jalankan perintah ini untuk melihat pip3 apakah sudah diinstal.

```
pip3 --version
```

2. Jika perintah mengembalikan nomor versi, pip3 diinstal dan Anda dapat melompat ke Langkah 3.
3. Jika perintah sebelumnya mengembalikan kesalahan, jalankan perintah ini untuk menginstal pip3.

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

4. Jalankan perintah ini untuk melihat pip3 apakah sudah diinstal.

```
pip3 --version
```

3. Instal SDK AWS IoT Perangkat saat ini untuk Python

Instal AWS IoT Device SDK untuk Python dan unduh contoh aplikasi ke perangkat Anda.

Di perangkat Anda, jalankan perintah ini.

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

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

JavaScript

Di bagian ini, Anda akan menginstal Node.js, manajer paket npm, dan AWS IoT Device SDK untuk JavaScript di perangkat Anda. Instruksi ini untuk Raspberry Pi yang menjalankan Raspberry Pi OS. Jika Anda memiliki perangkat lain atau menggunakan sistem operasi lain, Anda mungkin perlu menyesuaikan petunjuk ini untuk perangkat Anda.

1. Instal versi terbaru dari Node.js

AWS IoT Perangkat SDK untuk JavaScript membutuhkan Node.js dan manajer paket npm untuk diinstal pada Raspberry Pi Anda.

- a. Unduh versi terbaru dari repositori Node dengan memasukkan perintah ini.

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

- b. Instal Node dan npm.

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

- c. Verifikasi instalasi Node.

```
node -v
```

Konfirmasikan bahwa perintah menampilkan versi Node. Tutorial ini membutuhkan Node v10.0 atau yang lebih baru. Jika versi Node tidak ditampilkan, coba unduh repositori Node lagi.

- d. Verifikasi instalasi npm.

```
npm -v
```

Konfirmasikan bahwa perintah menampilkan versi npm. Jika versi npm tidak ditampilkan, coba instal Node dan npm lagi.

- e. Mulai ulang perangkat.

```
sudo shutdown -r 0
```


2. Instal SDK AWS IoT Perangkat untuk JavaScript

Instal AWS IoT Device SDK untuk JavaScript Raspberry Pi Anda.

- a. *Kloning AWS IoT Device SDK untuk JavaScript repositori ke aws-iot-device-sdk-js-v2 direktori direktori home Anda.* Pada Raspberry Pi, direktori *home* adalah `~/`, yang digunakan sebagai direktori *home* dalam perintah berikut. Jika perangkat Anda menggunakan jalur yang berbeda untuk direktori *home*, Anda harus mengganti `~/` dengan jalur yang benar untuk perangkat Anda dalam perintah berikut.

Perintah ini membuat `~/aws-iot-device-sdk-js-v2` direktori dan menyalin kode SDK ke dalamnya.

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

- b. Ubah ke `aws-iot-device-sdk-js-v2` direktori yang Anda buat pada langkah sebelumnya dan jalankan `npm install` untuk menginstal SDK. Perintah `npm install` akan memanggil build `aws-crt` perpustakaan yang dapat memakan waktu beberapa menit untuk menyelesaikannya.

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

Instal dan jalankan aplikasi sampel

Di bagian ini, Anda akan menginstal dan menjalankan aplikasi pubsub sampel yang ditemukan di AWS IoT Device SDK. Aplikasi ini menunjukkan bagaimana perangkat Anda menggunakan pustaka MQTT untuk mempublikasikan dan berlangganan pesan MQTT. Aplikasi sampel berlangganan topik `topic_1`, menerbitkan 10 pesan ke topik itu, dan menampilkan pesan saat diterima dari broker pesan.

Instal file sertifikat

Aplikasi sampel memerlukan file sertifikat yang mengautentikasi perangkat yang akan diinstal pada perangkat.

Untuk menginstal file sertifikat perangkat untuk aplikasi sampel

1. Buat `certs` subdirektori di direktori *home* Anda dengan menjalankan perintah ini.

```
cd ~
mkdir certs
```

2. Ke `~/certs` direktori, salin kunci pribadi, sertifikat perangkat, dan sertifikat CA root yang Anda buat sebelumnya [the section called “Buat AWS IoT sumber daya”](#).

Cara Anda menyalin file sertifikat ke perangkat tergantung pada perangkat dan sistem operasi dan tidak dijelaskan di sini. Namun, jika perangkat Anda mendukung antarmuka pengguna grafis (GUI) dan memiliki browser web, Anda dapat melakukan prosedur yang dijelaskan [the section called “Buat AWS IoT sumber daya”](#) dari browser web perangkat Anda untuk mengunduh file yang dihasilkan langsung ke perangkat Anda.

Perintah di bagian berikutnya mengasumsikan bahwa file kunci dan sertifikat Anda disimpan di perangkat seperti yang ditunjukkan dalam tabel ini.

Nama berkas sertifikat

File	Jalur file
Sertifikat Root CA	<code>~/certs/Amazon-root-CA-1.pem</code>
Sertifikat perangkat	<code>~/certs/device.pem.crt</code>
Kunci privat	<code>~/certs/private.pem.key</code>

Untuk menjalankan aplikasi sampel, Anda memerlukan informasi berikut:

Nilai parameter aplikasi

Parameter	Di mana menemukan nilainya
<i><code>your-iot-endpoint</code></i>	Di AWS IoT konsol , pilih Semua perangkat, lalu pilih Things.

Parameter	Di mana menemukan nilainya
	Pada halaman Pengaturan di AWS IoT menu. Titik akhir Anda ditampilkan di bagian titik akhir data Perangkat.

your-iot-endpoint Nilai memiliki format: *endpoint_id*-ats.iot.*region*.amazonaws.com, misalnya, a3qj468EXAMPLE-ats.iot.us-west-2.amazonaws.com.

Python

Untuk menginstal dan menjalankan aplikasi sampel

1. Arahkan ke direktori aplikasi contoh.

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

2. Di jendela baris perintah, ganti *your-iot-endpoint* seperti yang ditunjukkan dan jalankan perintah ini.

```
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. Perhatikan bahwa aplikasi sampel:

1. Terhubung ke AWS IoT layanan untuk akun Anda.
2. Berlangganan topik pesan, topic_1, dan menampilkan pesan yang diterimanya pada topik itu.
3. Menerbitkan 10 pesan ke topik, topic_1.
4. Menampilkan output yang mirip dengan berikut ini:

```
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!
```

Jika Anda mengalami masalah dalam menjalankan aplikasi sampel, tinjau [the section called “Memecahkan masalah dengan aplikasi sampel”](#).

Anda juga dapat menambahkan `--verbosity Debug` parameter ke baris perintah sehingga aplikasi sampel menampilkan pesan terperinci tentang apa yang dilakukannya. Informasi itu mungkin memberi Anda bantuan yang Anda butuhkan untuk memperbaiki masalah.

JavaScript

Untuk menginstal dan menjalankan aplikasi sampel

1. Di jendela baris perintah Anda, navigasikan ke `~/aws-iot-device-sdk-js-v2/samples/node/pub_sub` direktori yang dibuat SDK dan instal aplikasi sampel dengan menggunakan perintah ini. Perintah `npm install` akan memanggil build `aws-crt` perpustakaan yang dapat memakan waktu beberapa menit untuk menyelesaikannya.

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

```
npm install
```

2. Di jendela baris perintah, ganti *your-iot-endpoint* seperti yang ditunjukkan dan jalankan perintah ini.

```
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. Perhatikan bahwa aplikasi sampel:
 1. Terhubung ke AWS IoT layanan untuk akun Anda.
 2. Berlangganan topik pesan, `topic_1`, dan menampilkan pesan yang diterimanya pada topik itu.
 3. Menerbitkan 10 pesan ke topik, `topic_1`.
 4. Menampilkan output yang mirip dengan berikut ini:

```
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}
```

Jika Anda mengalami masalah dalam menjalankan aplikasi sampel, tinjau [the section called “Memecahkan masalah dengan aplikasi sampel”](#).

Anda juga dapat menambahkan `--verbosity Debug` parameter ke baris perintah sehingga aplikasi sampel menampilkan pesan terperinci tentang apa yang dilakukannya. Informasi itu mungkin memberi Anda bantuan yang Anda butuhkan untuk memperbaiki masalah.

Melihat pesan dari aplikasi sampel di AWS IoT konsol

Anda dapat melihat pesan aplikasi sampel saat mereka melewati broker pesan dengan menggunakan klien pengujian MQTT di konsol.AWS IoT

Untuk melihat pesan MQTT yang diterbitkan oleh aplikasi sampel

1. Ulasan [Lihat pesan MQTT dengan klien MQTT AWS IoT](#). Ini membantu Anda mempelajari cara menggunakan klien pengujian MQTT di AWS IoT konsol untuk melihat pesan MQTT saat mereka melewati broker pesan.
2. Buka klien pengujian MQTT di konsol.AWS IoT
3. Berlangganan topik, `topic_1`.
4. Di jendela baris perintah Anda, jalankan aplikasi sampel lagi dan tonton pesan di klien MQTT di konsol.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
```

Memecahkan masalah dengan aplikasi sampel

Jika Anda mengalami kesalahan saat mencoba menjalankan aplikasi sampel, berikut adalah beberapa hal yang perlu diperiksa.

Periksa sertifikat

Jika sertifikat tidak aktif, tidak AWS IoT akan menerima upaya koneksi apa pun yang menggunakannya untuk otorisasi. Saat membuat sertifikat Anda, mudah untuk mengabaikan tombol Aktifkan. Untungnya, Anda dapat mengaktifkan sertifikat Anda dari [AWS IoT konsol](#).

Untuk memeriksa aktivasi sertifikat Anda

1. Di [AWS IoT konsol](#), di menu sebelah kiri, pilih Aman, lalu pilih Sertifikat.
2. Dalam daftar sertifikat, temukan sertifikat yang Anda buat untuk latihan dan periksa statusnya di kolom Status.

Jika Anda tidak ingat nama sertifikat, periksa apakah ada yang Tidak Aktif untuk melihat apakah mereka mungkin yang Anda gunakan.

Pilih sertifikat dalam daftar untuk membuka halaman detailnya. Di halaman detail, Anda dapat melihat tanggal Buat untuk membantu Anda mengidentifikasi sertifikat.

3. Untuk mengaktifkan sertifikat tidak aktif, dari halaman detail sertifikat, pilih Tindakan, lalu pilih Aktifkan.

Jika Anda menemukan sertifikat yang benar dan aktif, tetapi Anda masih mengalami masalah dalam menjalankan aplikasi sampel, periksa kebijakannya seperti yang dijelaskan langkah selanjutnya.

Anda juga dapat mencoba membuat hal baru dan sertifikat baru dengan mengikuti langkah-langkah di [the section called “Buat objek benda”](#). Jika Anda membuat hal baru, Anda harus memberinya nama hal baru dan mengunduh file sertifikat baru ke perangkat Anda.

Periksa kebijakan yang dilampirkan pada sertifikat

Kebijakan mengotorisasi tindakan di AWS IoT. Jika sertifikat yang digunakan untuk terhubung AWS IoT tidak memiliki kebijakan, atau tidak memiliki kebijakan yang memungkinkannya terhubung, koneksi akan ditolak, bahkan jika sertifikat aktif.

Untuk memeriksa kebijakan yang dilampirkan pada sertifikat

1. Temukan sertifikat seperti yang dijelaskan pada item sebelumnya dan buka halaman detailnya.
2. Di menu sebelah kiri halaman detail sertifikat, pilih Kebijakan untuk melihat kebijakan yang dilampirkan pada sertifikat.

3. Jika tidak ada kebijakan yang dilampirkan pada sertifikat, tambahkan satu dengan memilih menu Tindakan, lalu pilih Lampirkan kebijakan.

Pilih kebijakan yang Anda buat sebelumnya [the section called “Buat AWS IoT sumber daya”](#).

4. Jika ada kebijakan yang dilampirkan, pilih ubin kebijakan untuk membuka halaman detailnya.

Di halaman detail, tinjau dokumen Kebijakan untuk memastikan dokumen tersebut berisi informasi yang sama dengan yang Anda buat [the section called “Buat AWS IoT kebijakan”](#).

Periksa baris perintah

Pastikan Anda menggunakan baris perintah yang benar untuk sistem Anda. Perintah yang digunakan pada sistem Linux dan macOS seringkali berbeda dari yang digunakan pada sistem Windows.

Periksa alamat titik akhir

[Tinjau perintah yang Anda masukkan dan periksa kembali alamat titik akhir dalam perintah Anda ke yang ada di konsol Anda AWS IoT .](#)

Periksa nama file dari file sertifikat

Bandingkan nama file dalam perintah yang Anda masukkan ke nama file file sertifikat di `certs` direktori.

Beberapa sistem mungkin memerlukan nama file dalam tanda kutip agar berfungsi dengan benar.

Periksa instalasi SDK

Pastikan instalasi SDK Anda selesai dan benar.

Jika ragu, instal ulang SDK di perangkat Anda. Dalam kebanyakan kasus, itu masalah menemukan bagian tutorial berjudul Instal SDK AWS IoT Perangkat untuk **bahasa** SDK dan mengikuti prosedur lagi.

Jika Anda menggunakan AWS IoT Device SDK untuk JavaScript, ingatlah untuk menginstal contoh aplikasi sebelum Anda mencoba menjalankannya. Menginstal SDK tidak secara otomatis menginstal aplikasi sampel. Contoh aplikasi harus diinstal secara manual setelah SDK diinstal.

Lihat pesan MQTT dengan klien MQTT AWS IoT

Bagian ini menjelaskan cara menggunakan klien pengujian AWS IoT MQTT di [AWS IoT konsol](#) untuk menonton pesan MQTT yang dikirim dan diterima oleh. AWS IoT Contoh yang digunakan di bagian ini berkaitan dengan contoh yang digunakan [Memulai dengan AWS IoT Core](#); namun, Anda dapat mengganti `topicName` yang digunakan dalam contoh dengan nama topik [atau filter topik apa pun yang digunakan oleh solusi](#) IoT Anda.

Perangkat menerbitkan pesan MQTT yang diidentifikasi berdasarkan [topik](#) untuk mengkomunikasikan statusnya AWS IoT, dan AWS IoT menerbitkan pesan MQTT untuk menginformasikan perangkat dan aplikasi tentang perubahan dan peristiwa. Anda dapat menggunakan klien MQTT untuk berlangganan topik ini dan menonton pesan saat muncul. Anda juga dapat menggunakan klien pengujian MQTT untuk mempublikasikan pesan MQTT ke perangkat dan layanan berlangganan di perangkat dan layanan Anda. Akun AWS

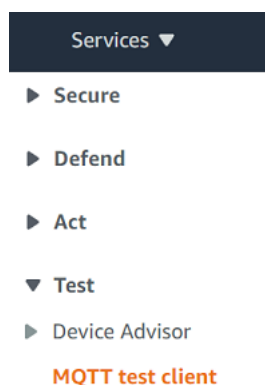
Daftar Isi

- [Melihat pesan MQTT di klien MQTT](#)
- [Menerbitkan pesan MQTT dari klien MQTT](#)
- [Menguji Langganan Bersama di klien MQTT](#)

Melihat pesan MQTT di klien MQTT

Untuk melihat pesan MQTT di klien pengujian MQTT

1. Di [AWS IoT konsol](#), di menu sebelah kiri, pilih Test dan kemudian pilih MQTT test client.



- Di tab Berlangganan topik, masukkan *TopicName* untuk berlangganan topik yang dipublikasikan perangkat Anda. Untuk aplikasi sampel yang memulai, berlangganan#, yang berlangganan semua topik pesan.

Melanjutkan contoh memulai, pada tab Berlangganan ke topik, di bidang Filter topik, masukkan#, lalu pilih Berlangganan.

Halaman log pesan topik, # terbuka dan # muncul di daftar Langganan. Jika perangkat yang Anda [the section called “Konfigurasi perangkat Anda”](#) konfigurasi menjalankan program contoh, Anda akan melihat pesan yang dikirimnya AWS IoT di log pesan #. Entri log pesan akan muncul di bawah bagian Publikasikan saat pesan dengan topik berlangganan diterima oleh AWS IoT

Subscriptions	#	Pause	Clear	Export	Edit
#	♡ X				

- Pada halaman log pesan #, Anda juga dapat mempublikasikan pesan ke topik, tetapi Anda harus menentukan nama topik. Anda tidak dapat mempublikasikan ke topik #.

Pesan yang dipublikasikan ke topik berlangganan muncul di log pesan saat diterima, dengan pesan terbaru terlebih dahulu.

Memecahkan masalah pesan MQTT

Gunakan filter topik wild card

Jika pesan Anda tidak muncul di log pesan seperti yang Anda harapkan, coba berlangganan filter topik wild card seperti yang dijelaskan dalam [Filter topik](#). Filter topik wild card multi-level MQTT adalah tanda hash atau pound (#) dan dapat digunakan sebagai filter topik di bidang topik Langganan.

Berlangganan filter # topik berlangganan setiap topik yang diterima oleh broker pesan. Anda dapat mempersempit filter dengan mengganti elemen jalur filter topik dengan karakter kartu liar # multi-level atau karakter kartu liar tingkat tunggal '+'.

Saat menggunakan kartu liar dalam filter topik

- Karakter kartu liar multi-level harus menjadi karakter terakhir dalam filter topik.
- Jalur filter topik hanya dapat memiliki satu karakter kartu liar tingkat tunggal per level topik.

Sebagai contoh:

Filter topik	Menampilkan pesan dengan
#	Nama topik apa pun
topic_1/#	Nama topik yang dimulai dengan topic_1/
topic_1/level_2/#	Nama topik yang dimulai dengan topic_1/level_2/
topic_1/+/level_3	Nama topik yang dimulai dengan topic_1/, diakhiri dengan /level_3, dan memiliki satu elemen dengan nilai apa pun di antaranya.

Untuk informasi selengkapnya tentang filter topik, lihat [Filter topik](#).

Periksa kesalahan nama topik

Nama topik dan filter topik MQTT peka huruf besar/kecil. Jika, misalnya, perangkat Anda menerbitkan pesan ke Topic_1 (dengan huruf besar T) alih-alih topic_1 topik yang Anda langgani, pesannya tidak akan muncul di klien pengujian MQTT. Namun, berlangganan filter topik wild card akan menunjukkan bahwa perangkat menerbitkan pesan dan Anda dapat melihat bahwa itu menggunakan nama topik yang bukan yang Anda harapkan.

Menerbitkan pesan MQTT dari klien MQTT

Untuk mempublikasikan pesan ke topik MQTT

1. Pada halaman klien pengujian MQTT, di tab **Publikasikan ke topik**, di bidang **Nama topik**, masukkan `topicName` pesan Anda. Dalam contoh ini, gunakan `my/topic`.

Note

Jangan gunakan informasi yang dapat diidentifikasi secara pribadi dalam nama topik, baik menggunakannya di klien pengujian MQTT atau dalam implementasi sistem Anda. Nama topik dapat muncul dalam komunikasi dan laporan yang tidak terenkripsi.

2. Di jendela payload pesan, masukkan JSON berikut:

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

3. Pilih **Publikasikan** untuk mempublikasikan pesan Anda AWS IoT.

Note

Pastikan Anda berlangganan topik `my/topic` sebelum mempublikasikan pesan Anda.

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 Service (QoS) of 0.

Q my/topic X

Message payload

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

► Additional configuration

Publish

4. Dalam daftar Langganan, pilih topik saya/untuk melihat pesan. Anda akan melihat pesan muncul di klien pengujian MQTT di bawah jendela payload pesan publikasi.

Subscriptions	#	Pause	Clear	Export	Edit
#	my/topic				

▼ my/topic November 02, 2021, 11:55:22 (UTC-0700)

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

Anda dapat mempublikasikan pesan MQTT ke topik lain dengan mengubah topicName di bidang Nama topik dan **memilih** tombol Publikasikan.

⚠ Important

Saat Anda membuat beberapa langganan dengan topik yang tumpang tindih (mis., Probe1/temperature dan probe1/#), ada kemungkinan bahwa satu pesan yang dipublikasikan ke topik yang cocok dengan kedua langganan akan dikirimkan beberapa kali, sekali untuk setiap langganan yang tumpang tindih.

Menguji Langganan Bersama di klien MQTT

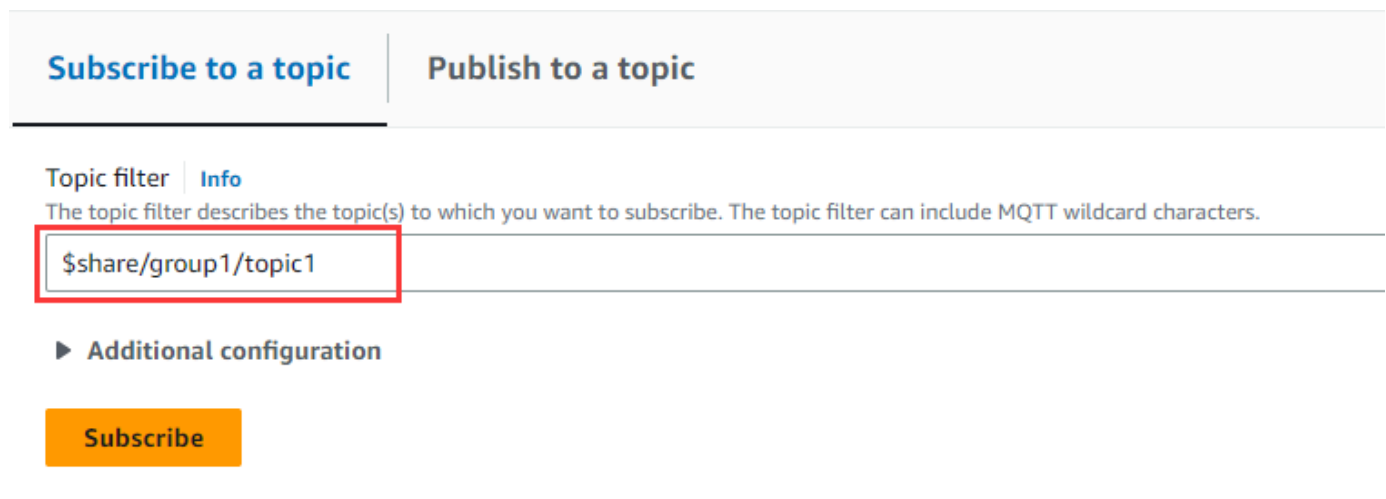
Bagian ini menjelaskan cara menggunakan klien AWS IoT MQTT di [AWS IoT konsol](#) untuk menonton pesan MQTT yang dikirim dan diterima menggunakan Langganan Bersama. AWS IoT [MQTT](#) memungkinkan beberapa klien untuk berbagi langganan ke topik dengan hanya satu klien menerima pesan yang dipublikasikan ke topik tersebut menggunakan distribusi acak. [Untuk mensimulasikan beberapa klien MQTT \(dalam contoh ini, dua klien MQTT\) berbagi langganan yang sama, Anda membuka klien AWS IoT MQTT di konsol dari beberapa browser web.](#) [AWS IoT](#) Contoh yang digunakan di bagian ini tidak berhubungan dengan contoh yang digunakan dalam [Memulai dengan AWS IoT Core](#). Untuk informasi selengkapnya, lihat [Langganan Bersama](#).

Untuk berbagi langganan ke topik MQTT

1. Di [AWS IoT konsol](#), di panel navigasi, pilih Uji dan kemudian pilih klien pengujian MQTT.
2. Di tab Berlangganan topik, masukkan *TopicName* untuk berlangganan topik yang dipublikasikan perangkat Anda. Untuk menggunakan Langganan Bersama, berlangganan filter topik Langganan Bersama sebagai berikut:

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

Contoh filter topik dapat berupa **\$share/group1/topic1**, yang berlangganan topik **topic1** pesan.



The screenshot shows the 'Subscribe to a topic' interface in the AWS IoT console. It features two tabs: 'Subscribe to a topic' (selected) and 'Publish to a topic'. Below the tabs, there is a 'Topic filter' field with an 'Info' icon. The field contains the text '\$share/group1/topic1', which is highlighted with a red rectangular box. Below the field, there is a section for 'Additional configuration' with a right-pointing triangle icon. At the bottom of the interface is an orange 'Subscribe' button.

3. Buka browser web lain dan ulangi langkah 1 dan langkah 2. Dengan cara ini, Anda mensimulasikan dua klien MQTT berbeda yang berbagi langganan yang sama. **\$share/group1/topic1**

4. **Pilih satu klien MQTT, di tab *Publikasikan ke topik*, di bidang *Nama topik*, masukkan *TopicName* pesan Anda.** Dalam contoh ini, gunakan **topic1**. Cobalah mempublikasikan pesan beberapa kali. Dari daftar Langganan kedua klien MQTT, Anda harus dapat melihat bahwa klien menerima pesan menggunakan distribusi acak. Dalam contoh ini, kami menerbitkan pesan yang sama “Halo dari AWS IoT konsol” tiga kali. Klien MQTT di sebelah kiri menerima pesan dua kali dan klien MQTT di sebelah kanan menerima pesan satu kali.

The image displays two side-by-side screenshots of the AWS IoT Core console, illustrating the process of publishing a message to a topic and the resulting subscriptions.

Left Screenshot (Subscription View):

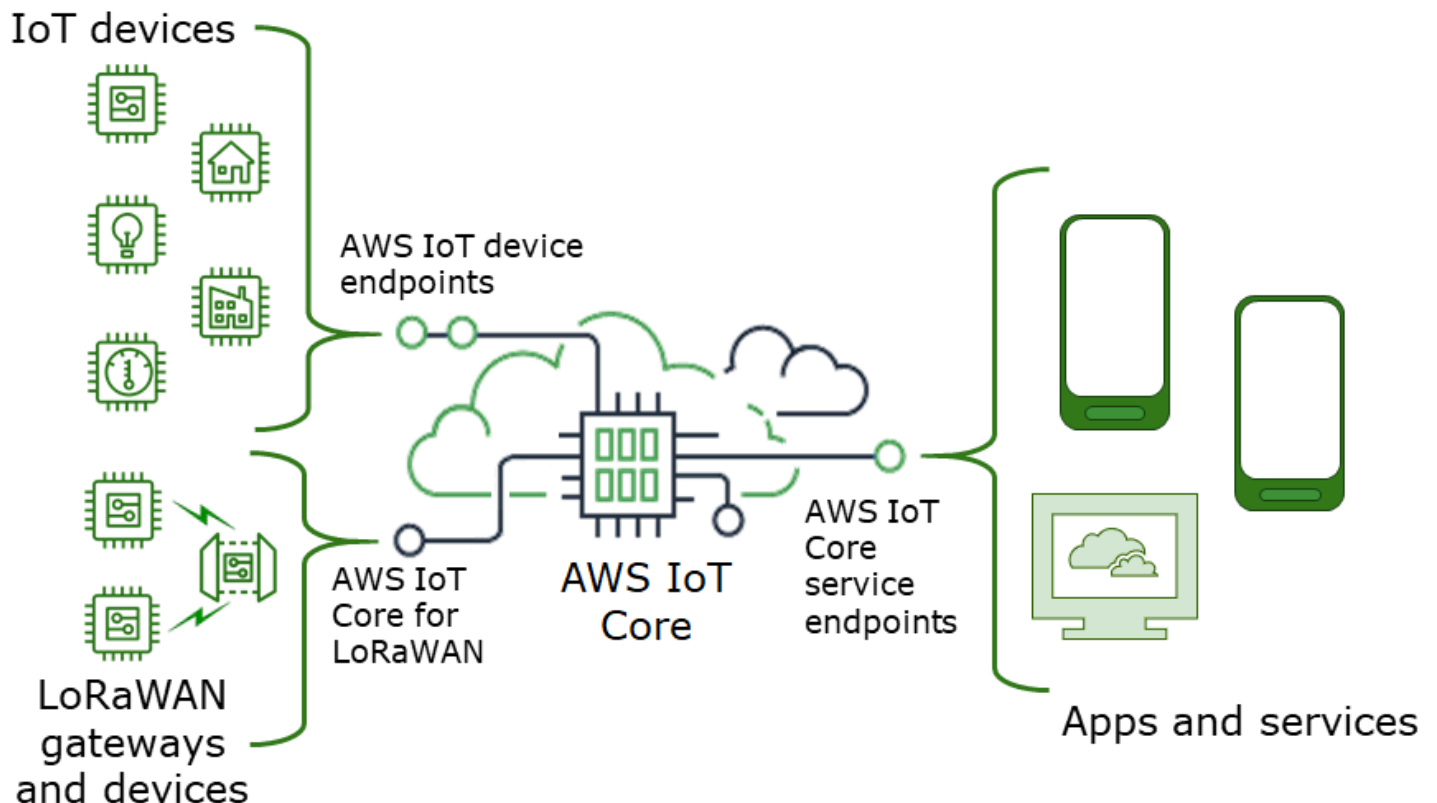
- Topic filter:** \$share/group1/topic1
- Additional configuration:** (Expanded)
- Subscribe:** (Button)
- Subscriptions:** \$share/group1/topic1
- Subscription List:** \$share/group1/topic1 (with heart and close icons)
- Message payload:** { "message": "Hello from AWS IoT console" }
- Additional configuration:** (Expanded)
- Publish:** (Button)
- Message Status:** No messages have been sent to this subscription yet. Please send a message to this subscription to see messages here.

Right Screenshot (Publish View):

- Topic filter:** \$share/group1/topic1
- Additional configuration:** (Expanded)
- Subscribe:** (Button)
- Subscriptions:** \$share/group1/topic1
- Subscription List:** \$share/group1/topic1 (with heart and close icons)
- Message payload:** { "message": "Hello from AWS IoT console" }
- Additional configuration:** (Expanded)
- Publish:** (Button)
- Message Status:** No messages have been sent to this subscription yet. Please send a message to this subscription to see messages here.

Menghubungkan ke AWS IoT Core

AWS IoT Core mendukung koneksi dengan perangkat IoT, gateway nirkabel, layanan, dan aplikasi. Perangkat terhubung AWS IoT Core sehingga mereka dapat mengirim data ke dan menerima data dari AWS IoT layanan dan perangkat lain. Aplikasi dan layanan lainnya juga terhubung AWS IoT Core untuk mengontrol dan mengelola perangkat IoT dan memproses data dari solusi IoT Anda. Bagian ini menjelaskan cara memilih cara terbaik untuk terhubung dan berkomunikasi AWS IoT Core untuk setiap aspek solusi IoT Anda.



Ada beberapa cara untuk berinteraksi AWS IoT. Aplikasi dan layanan dapat menggunakan [AWS IoT Core- titik akhir bidang kontrol](#) dan perangkat dapat terhubung AWS IoT Core dengan menggunakan [AWS IoT titik akhir perangkat](#) atau [AWS IoT Core untuk Wilayah LoRa WAN dan titik akhir](#).

AWS IoT Core- titik akhir bidang kontrol

Titik akhir bidang kontrol menyediakan akses ke fungsi yang mengontrol dan mengelola AWS IoT solusi Anda. AWS IoT Core

- Titik akhir

Bidang kontrol AWS IoT Core- dan titik akhir bidang kontrol AWS IoT Core Device Advisor adalah spesifik Wilayah dan tercantum dalam [AWS IoT Core Titik Akhir](#) dan Kuota. Format titik akhir adalah sebagai berikut.

Tujuan titik akhir	Format titik akhir	Melayani
AWS IoT Core- pesawat kontrol	<code>iot.<i>aws-region</i>.amazonaws.com</code>	AWS IoT Kontrol API Pesawat
AWS IoT Core Device Advisor - bidang kontrol	<code>api.iotdeviceadvisor.<i>aws-region</i>.amazonaws.com</code>	AWS IoT Core API Pesawat Kontrol Penasihat Perangkat

- SDK dan alat

[AWS SDK](#) menyediakan dukungan khusus bahasa untuk AWS IoT Core API, dan API layanan lainnya. AWS [AWS Mobile SDK](#) memberi pengembang aplikasi dukungan khusus platform untuk AWS IoT Core API, dan AWS layanan lainnya di perangkat seluler.

[AWS CLI](#) Menyediakan akses baris perintah ke fungsi yang disediakan oleh titik akhir AWS IoT layanan. [AWS Alat untuk PowerShell](#) menyediakan alat untuk mengelola AWS layanan dan sumber daya di lingkungan PowerShell scripting.

- Autentikasi

Endpoint layanan menggunakan pengguna IAM dan AWS kredensi untuk mengautentikasi pengguna.

- Pelajari selengkapnya

Untuk informasi selengkapnya dan tautan ke referensi SDK, lihat [the section called “Menghubungkan ke AWS IoT Core titik akhir layanan”](#).

AWS IoT titik akhir perangkat

Titik akhir AWS IoT perangkat mendukung komunikasi antara perangkat IoT Anda dan AWS IoT

- Titik akhir

Dukungan AWS IoT Core dan AWS IoT Device Management fungsi titik akhir perangkat. Mereka khusus untuk Anda Akun AWS dan Anda dapat melihat apa itu dengan menggunakan [describe-endpoint](#) perintah.

Tujuan titik akhir	Format titik akhir	Melayani
AWS IoT Core- pesawat data	Lihat ??? .	AWS IoT API Pesawat Data
AWS IoT Device Managemen t- data pekerjaan	Lihat ??? .	AWS IoT Lowongan Kerja Data Plane API
AWS IoT Device Advisor - bidang data	Lihat ??? .	Tidak berlaku
AWS IoT Device Managemen t- Hub Armada	Tidak berlaku	Tidak berlaku
AWS IoT Device Managemen t- terowongan aman	<code>api.tunneling.iot. <i>aws-region</i>.amazonaw s.com</code>	AWS IoT API Tunneling Aman

Untuk informasi lebih lanjut tentang titik akhir ini dan fungsi yang mereka dukung, lihat [the section called “AWS IoT data perangkat dan titik akhir layanan”](#).

- SDK

[SDK AWS IoT Perangkat](#) menyediakan dukungan khusus bahasa untuk protokol Message Queueing Telemetry Transport (MQTT) dan WebSocket Secure (WSS), yang digunakan perangkat untuk berkomunikasi. [AWS IoT AWS SDK Seluler](#) juga menyediakan dukungan untuk komunikasi perangkat MQTT, AWS IoT API, dan API AWS layanan lain di perangkat seluler.

- Autentikasi

Titik akhir perangkat menggunakan sertifikat X.509 atau pengguna AWS IAM dengan kredensial untuk mengautentikasi pengguna.

- Pelajari selengkapnya

Untuk informasi selengkapnya dan tautan ke referensi SDK, lihat [the section called “AWS IoT SDK perangkat”](#).

AWS IoT Core untuk gateway dan perangkat LoRa WAN

AWS IoT Core untuk LoRa WAN menghubungkan gateway nirkabel dan perangkat ke. AWS IoT Core

- Titik akhir

AWS IoT Core untuk LoRa WAN mengelola koneksi gateway ke akun dan titik akhir khusus Wilayah AWS IoT Core . Gateway dapat terhubung ke titik akhir Configuration and Update Server (CUPS) akun Anda yang AWS IoT Core disediakan LoRa WAN.

Tujuan titik akhir	Format titik akhir	Melayani
Konfigurasi dan Perbarui Server (CUPS)	<i>account-specific-prefix</i> .cups.lorawan. <i>aws-region</i> .amazonaws.com:443	Komunikasi gateway dengan Server Konfigurasi dan Pembaruan yang disediakan oleh AWS IoT Core untuk LoRa WAN
LoRaServer Jaringan WAN (LNS)	<i>account-specific-prefix</i> .gateway.lorawan. <i>aws-region</i> .amazonaws.com:443	Komunikasi gateway dengan Server Jaringan LoRa WAN yang disediakan oleh AWS IoT Core untuk LoRa WAN

- SDK

AWS IoT Wireless API yang AWS IoT Core untuk LoRa WAN dibangun didukung oleh AWS SDK. Untuk informasi selengkapnya, lihat [AWS SDK dan Kit Alat](#).

- Autentikasi

AWS IoT Core untuk komunikasi perangkat LoRa WAN gunakan sertifikat X.509 untuk mengamankan komunikasi dengan. AWS IoT

- Pelajari selengkapnya

Untuk informasi selengkapnya tentang mengonfigurasi dan menghubungkan perangkat nirkabel, lihat [AWS IoT Core Wilayah LoRa WAN dan titik akhir](#).

Menghubungkan ke AWS IoT Core titik akhir layanan

Anda dapat mengakses fitur AWS IoT Core- control plane dengan menggunakan AWS CLI, AWS SDK untuk bahasa pilihan Anda, atau dengan memanggil REST API secara langsung. Sebaiknya gunakan AWS CLI atau AWS SDK untuk berinteraksi AWS IoT Core karena mereka menggabungkan praktik terbaik untuk AWS layanan panggilan. Memanggil REST API secara langsung adalah pilihan, tetapi Anda harus memberikan [kredensyal keamanan yang diperlukan](#) yang memungkinkan akses ke API.

Note

Perangkat IoT harus digunakan. [AWS IoT SDK perangkat](#) SDK Perangkat dioptimalkan untuk digunakan pada perangkat, mendukung komunikasi MQTT dengan AWS IoT, dan mendukung AWS IoT API yang paling banyak digunakan oleh perangkat. Untuk informasi selengkapnya tentang SDK Perangkat dan fitur yang disediakan, lihat [AWS IoT SDK perangkat](#).

Perangkat seluler harus digunakan [AWS SDK Seluler](#). Mobile SDK menyediakan dukungan untuk AWS IoT API, komunikasi perangkat MQTT, dan API AWS layanan lain di perangkat seluler. Untuk informasi selengkapnya tentang SDK Seluler dan fitur yang mereka sediakan, lihat [AWS SDK Seluler](#).

Anda dapat menggunakan AWS Amplify alat dan sumber daya dalam aplikasi web dan seluler untuk terhubung dengan lebih mudah AWS IoT Core. Untuk informasi selengkapnya tentang menghubungkan AWS IoT Core dengan menggunakan Amplify, lihat [Sub Pub Memulai di dokumentasi](#) Amplify.

Bagian berikut menjelaskan alat dan SDK yang dapat Anda gunakan untuk mengembangkan dan berinteraksi dengan AWS IoT dan AWS layanan lainnya. Untuk daftar lengkap AWS alat dan kit pengembangan yang tersedia untuk membangun dan mengelola aplikasi AWS, lihat [Alat untuk Dibangun AWS](#).

AWS CLI untuk AWS IoT Core

AWS CLI Ini menyediakan akses baris perintah ke AWS API.

- Penginstalan

Untuk informasi tentang cara menginstal AWS CLI, lihat [Menginstal AWS CLI](#).

- Autentikasi

AWS CLI Menggunakan kredensi dari Anda. Akun AWS

- Referensi

Untuk informasi tentang AWS CLI perintah untuk AWS IoT Core layanan ini, lihat:

- [AWS CLI Referensi Perintah untuk IoT](#)
- [AWS CLI Referensi Perintah untuk data IoT](#)
- [AWS CLI Referensi Perintah untuk data pekerjaan IoT](#)
- [AWS CLI Referensi Perintah untuk tunneling aman IoT](#)

Untuk alat untuk mengelola AWS layanan dan sumber daya di lingkungan PowerShell skrip, lihat [AWS Alat untuk PowerShell](#).

AWS SDK

Dengan AWS SDK, aplikasi dan perangkat yang kompatibel dapat memanggil AWS IoT API dan API AWS layanan lainnya. Bagian ini menyediakan tautan ke AWS SDK dan dokumentasi referensi API untuk API AWS IoT Core layanan.

AWS SDK mendukung API ini AWS IoT Core

- [AWS IoT](#)
- [AWS IoT Pesawat Data](#)
- [AWS IoT Lowongan Kerja Data Plane](#)
- [AWS IoT Terowongan Aman](#)
- [AWS IoT Nirkabel](#)

C++

Untuk menginstal [AWS SDK for C++](#) dan menggunakannya untuk terhubung ke AWS IoT:

1. Ikuti petunjuk di [Memulai Menggunakan AWS SDK for C++](#)

Instruksi ini menjelaskan cara:

- Instal dan buat SDK dari file sumber

- Berikan kredensi untuk menggunakan SDK dengan Akun AWS
 - Menginisialisasi dan mematikan SDK di aplikasi atau layanan Anda
 - Membuat project CMake untuk membangun aplikasi atau layanan
2. Buat dan jalankan aplikasi sampel. Untuk contoh aplikasi yang menggunakan AWS SDK for C++ + [AWS SDK for C++](#) , lihat Contoh Kode.

Dokumentasi untuk AWS IoT Core layanan yang AWS SDK for C++ didukung

- [AWS: :IoTClient” dokumentasi referensi](#)
- [Aws: :IoTDataPlane: DataPlaneClient :Dokumentasi referensi IoT](#)
- [Aws: :IoTJobsDataPlane: JobsDataPlaneClient :Dokumentasi referensi IoT](#)
- [Aws: :IoTSecureTunneling: SecureTunnelingClient :Dokumentasi referensi IoT](#)

Go

Untuk menginstal [AWS SDK for Go](#) dan menggunakannya untuk terhubung ke AWS IoT:

1. Ikuti instruksi di [Memulai dengan AWS SDK for Go](#)

Instruksi ini menjelaskan cara:

- Instal AWS SDK for Go
 - Dapatkan kunci akses untuk SDK untuk mengakses Akun AWS
 - Impor paket ke kode sumber aplikasi atau layanan kami
2. Buat dan jalankan aplikasi sampel. Untuk contoh aplikasi yang menggunakan AWS SDK for Go, lihat [Contoh AWS SDK for Go Kode](#).

Dokumentasi untuk AWS IoT Core layanan yang AWS SDK for Go didukung

- [Dokumentasi referensi IoT](#)
- [Dokumentasi referensi IoT DataPlane](#)
- [Dokumentasi referensi IoT JobsDataPlane](#)
- [Dokumentasi referensi IoT SecureTunneling](#)

Java

Untuk menginstal [AWS SDK for Java](#) dan menggunakannya untuk terhubung ke AWS IoT:

1. Ikuti instruksi di [Memulai dengan AWS SDK for Java 2.x](#)

Instruksi ini menjelaskan cara:

- Mendaftar AWS dan Membuat Pengguna IAM
 - Unduh SDK
 - Menyiapkan AWS Kredensyal dan Wilayah
 - Gunakan SDK dengan Apache Maven
 - Menggunakan SDK dengan Gradle
2. Buat dan jalankan aplikasi contoh menggunakan salah satu [Contoh AWS SDK for Java 2.x Kode](#).
 3. Tinjau dokumentasi [referensi SDK API](#)

Dokumentasi untuk AWS IoT Core layanan yang AWS SDK for Java didukung

- [IoTClient dokumentasi referensi](#)
- [IoTDataPlaneClient dokumentasi referensi](#)
- [IoTJobsDataPlaneClient dokumentasi referensi](#)
- [Dokumentasi referensi IoT SecureTunnelingClient](#)

JavaScript

Untuk menginstal AWS SDK for JavaScript dan menggunakannya untuk terhubung ke AWS IoT:

1. Ikuti instruksi dalam [Menyiapkan AWS SDK for JavaScript](#). Instruksi ini berlaku untuk menggunakan AWS SDK for JavaScript di browser dan dengan Node.JS. Pastikan Anda mengikuti petunjuk yang berlaku untuk instalasi Anda.

Instruksi ini menjelaskan cara:

- Periksa prasyaratnya
- Instal SDK untuk JavaScript
- Muat SDK untuk JavaScript

2. Buat dan jalankan aplikasi contoh untuk memulai SDK seperti yang dijelaskan oleh opsi memulai untuk lingkungan Anda.
 - Memulai [AWS SDK untuk JavaScript di Browser](#), atau
 - Memulai dengan [AWS SDK untuk JavaScript di Node.js](#)

Dokumentasi untuk AWS IoT Core layanan yang AWS SDK for JavaScript didukung

- [AWS.Iot reference documentation](#)
- [AWS.IotData reference documentation](#)
- [AWS.IotJobsDataPlane reference documentation](#)
- [AWS.IotSecureTunneling reference documentation](#)

.NET

Untuk menginstal [AWS SDK for .NET](#) dan menggunakannya untuk terhubung ke AWS IoT:

1. Ikuti petunjuk dalam [Menyiapkan AWS SDK for .NET lingkungan Anda](#)
2. Ikuti petunjuk dalam [Menyiapkan AWS SDK for .NET proyek Anda](#)

Instruksi ini menjelaskan cara:

- Memulai proyek baru
 - Dapatkan dan konfigurasi AWS kredensial
 - Instal AWS paket SDK
3. Membuat dan menjalankan salah satu program sampel dalam [Bekerja dengan AWS layanan di AWS SDK for .NET](#)
 4. Tinjau dokumentasi [referensi SDK API](#)

Dokumentasi untuk AWS IoT Core layanan yang AWS SDK for .NET didukung

- [Dokumentasi referensi Amazon.iot.model](#)
- [Amazon. IotDataDokumentasi referensi. Model](#)
- [Dokumentasi referensi Amazon.IoT .Model JobsDataPlane](#)
- [Dokumentasi referensi Amazon.IoT .Model SecureTunneling](#)

PHP

Untuk menginstal [AWS SDK for PHP](#) dan menggunakannya untuk terhubung ke AWS IoT:

1. Ikuti petunjuk di [Memulai dengan AWS SDK for PHP Versi 3](#)

Instruksi ini menjelaskan cara:

- Periksa prasyaratnya
 - Instal SDK
 - Menerapkan SDK ke skrip PHP
2. Buat dan jalankan aplikasi sampel menggunakan salah satu [Contoh Kode AWS SDK for PHP Versi 3](#)

Dokumentasi untuk AWS IoT Core layanan yang AWS SDK for PHP didukung

- [Dokumentasi referensi IoTClient](#)
- [Dokumentasi referensi IoT DataPlaneClient](#)
- [Dokumentasi referensi IoT JobsDataPlaneClient](#)
- [Dokumentasi referensi IoT SecureTunnelingClient](#)

Python

Untuk menginstal [AWS SDK for Python \(Boto3\)](#) dan menggunakannya untuk terhubung ke AWS IoT:

1. Ikuti petunjuk di [AWS SDK for Python \(Boto3\) Quickstart](#)

Instruksi ini menjelaskan cara:

- Instal SDK
 - Konfigurasi SDK
 - Gunakan SDK dalam kode Anda
2. Membuat dan menjalankan program sampel yang menggunakan AWS SDK for Python (Boto3)

Program ini menampilkan opsi logging akun yang saat ini dikonfigurasi. Setelah Anda menginstal SDK dan mengkonfigurasinya untuk akun Anda, Anda harus dapat menjalankan program ini.

```
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))
```

Untuk informasi selengkapnya tentang fungsi yang digunakan dalam contoh ini, lihat [the section called “Konfigurasi AWS IoT logging”](#).

Dokumentasi untuk AWS IoT Core layanan yang AWS SDK for Python (Boto3) didukung

- [Dokumentasi referensi IoT](#)
- [Dokumentasi referensi IoT DataPlane](#)
- [Dokumentasi referensi IoT JobsDataPlane](#)
- [Dokumentasi referensi IoT SecureTunneling](#)

Ruby

Untuk menginstal [AWS SDK for Ruby](#) dan menggunakannya untuk terhubung ke AWS IoT:

- Ikuti instruksi di [Memulai dengan AWS SDK for Ruby](#)

Instruksi ini menjelaskan cara:

- Instal SDK
- Konfigurasi SDK
- Membuat dan menjalankan [Hello World Tutorial](#)

Dokumentasi untuk AWS IoT Core layanan yang didukung AWS SDK for Ruby

- [Aws: :IoT: :Dokumentasi referensi klien](#)
- [Aws: :IoTDataPlane: :Dokumentasi referensi klien](#)
- [Aws: :IoTJobsDataPlane: :Dokumentasi referensi klien](#)

- [Aws: :IoTSecureTunneling: :Dokumentasi referensi klien](#)

AWS SDK Seluler

AWS Mobile SDK menyediakan dukungan khusus platform pengembang aplikasi seluler untuk API layanan AWS IoT Core , komunikasi perangkat IoT menggunakan MQTT, dan API layanan lainnya. AWS

Android

AWS Mobile SDK for Android

AWS Mobile SDK for Android Ini berisi pustaka, sampel, dan dokumentasi bagi pengembang untuk membangun aplikasi seluler yang terhubung menggunakan AWS. SDK ini juga mencakup dukungan untuk komunikasi perangkat MQTT dan memanggil API layanan. AWS IoT Core Untuk informasi selengkapnya, lihat hal berikut:

- [AWS SDK seluler untuk Android aktif GitHub](#)
- [AWS SDK for Android Readme](#)
- [AWS Sampel SDK for Android Seluler](#)
- [AWS Referensi SDK for Android API](#)
- [AWSIoTClient Dokumentasi referensi kelas](#)

iOS

AWS Mobile SDK for iOS

AWS Mobile SDK for iOS Ini adalah kit pengembangan perangkat lunak open-source, didistribusikan di bawah lisensi Apache Open Source. SDK for iOS menyediakan pustaka, contoh kode, dan dokumentasi untuk membantu pengembang membangun aplikasi AWS seluler yang terhubung menggunakan. SDK ini juga mencakup dukungan untuk komunikasi perangkat MQTT dan memanggil API layanan. AWS IoT Core Untuk informasi selengkapnya, lihat hal berikut:

- [AWS Mobile SDK for iOS pada GitHub](#)
- [AWS SDK for iOS Readme](#)
- [AWS Sampel SDK for iOS](#)
- [AWS IoT Dokumen referensi kelas di AWS SDK for iOS](#)

REST API dari AWS IoT Core layanan

REST API dari AWS IoT Core layanan dapat dipanggil langsung dengan menggunakan permintaan HTTP.

- URL Titik Akhir

Titik akhir layanan yang mengekspos API REST AWS IoT Core layanan bervariasi menurut Wilayah dan tercantum dalam [AWS IoT Core Titik Akhir](#) dan Kuota. Anda harus menggunakan titik akhir untuk Wilayah yang memiliki AWS IoT sumber daya yang ingin Anda akses, karena AWS IoT sumber daya bersifat spesifik Wilayah.

- Autentikasi

REST API dari AWS IoT Core layanan menggunakan kredensi AWS IAM untuk otentikasi. Untuk informasi selengkapnya, lihat [Menandatangani permintaan AWS API](#) di Referensi AWS Umum.

- Referensi API

Untuk informasi tentang fungsi spesifik yang disediakan oleh REST API AWS IoT Core layanan, lihat:

- [Referensi API untuk IoT.](#)
- [Referensi API untuk data IoT.](#)
- [Referensi API untuk data pekerjaan IoT.](#)
- [Referensi API untuk tunneling aman IoT.](#)

Menghubungkan perangkat ke AWS IoT

Perangkat terhubung ke AWS IoT dan layanan lainnya melalui AWS IoT Core. Melalui AWS IoT Core, perangkat mengirim dan menerima pesan menggunakan titik akhir perangkat yang khusus untuk akun Anda. Komunikasi perangkat [the section called “AWS IoT SDK perangkat”](#) mendukung menggunakan protokol MQTT dan WSS. Untuk informasi selengkapnya tentang protokol yang dapat digunakan perangkat, lihat. [the section called “Protokol komunikasi perangkat”](#)

Broker pesan

AWS IoT mengelola komunikasi perangkat melalui broker pesan. Perangkat dan klien mempublikasikan pesan ke broker pesan dan juga berlangganan pesan yang diterbitkan oleh broker pesan. [Pesan diidentifikasi oleh topik yang ditentukan aplikasi.](#) Ketika broker pesan menerima pesan

yang diterbitkan oleh perangkat atau klien, ia menerbitkan kembali pesan itu ke perangkat dan klien yang telah berlangganan topik pesan. Broker pesan juga meneruskan pesan ke mesin AWS IoT [aturan](#), yang dapat bertindak berdasarkan konten pesan.

AWS IoT keamanan pesan

Koneksi perangkat untuk AWS IoT digunakan [the section called “Sertifikat klien X.509”](#) dan [AWS tanda tangan V4](#) untuk otentikasi. Komunikasi perangkat diamankan oleh TLS versi 1.3 dan AWS IoT mengharuskan perangkat untuk mengirim [ekstensi Server Name Indication \(SNI\)](#) ketika mereka terhubung. Untuk informasi selengkapnya, lihat [Keamanan Transportasi di AWS IoT](#).

AWS IoT data perangkat dan titik akhir layanan

Important

Anda dapat menyimpan atau menyimpan titik akhir di perangkat Anda. Ini berarti Anda tidak perlu menanyakan DescribeEndpoint API setiap kali perangkat baru terhubung. Titik akhir tidak akan berubah setelah AWS IoT Core membuatnya untuk akun Anda.

Setiap akun memiliki beberapa titik akhir perangkat yang unik untuk akun dan mendukung fungsi IoT tertentu. Titik akhir data AWS IoT perangkat mendukung protokol penerbitan/berlangganan yang dirancang untuk kebutuhan komunikasi perangkat IoT; namun, klien lain, seperti aplikasi dan layanan, juga dapat menggunakan antarmuka ini jika aplikasi mereka memerlukan fitur khusus yang disediakan titik akhir ini. Titik akhir layanan AWS IoT perangkat mendukung akses yang berpusat pada perangkat ke layanan keamanan dan manajemen.

Untuk mempelajari titik akhir data perangkat akun, Anda dapat menemukannya di halaman [Pengaturan](#) AWS IoT Core konsol.

Untuk mempelajari titik akhir perangkat akun Anda untuk tujuan tertentu, termasuk titik akhir data perangkat, gunakan perintah describe-endpoint CLI yang ditampilkan di sini, atau DescribeEndpoint REST API, dan berikan nilai *endpointType* parameter dari tabel berikut.

```
aws iot describe-endpoint --endpoint-type endpointType
```

Perintah ini mengembalikan *iot-endpoint dalam format* berikut: *account-specific-prefix.iot.aws-region.amazonaws.com*

Setiap pelanggan memiliki `iot:Data-ATS` dan `iot:Data` titik akhir. Setiap titik akhir menggunakan sertifikat X.509 untuk mengautentikasi klien. Kami sangat menyarankan agar pelanggan menggunakan tipe `iot:Data-ATS` endpoint yang lebih baru untuk menghindari masalah yang terkait dengan ketidakpercayaan yang meluas terhadap otoritas sertifikat Symantec. Kami menyediakan `iot:Data` titik akhir bagi perangkat untuk mengambil data dari titik akhir lama yang menggunakan VeriSign sertifikat untuk kompatibilitas mundur. Untuk informasi selengkapnya, lihat [Otentikasi Server](#).

AWS IoT titik akhir untuk perangkat

Tujuan titik akhir	Nilai <i>endpointType</i>	Deskripsi
AWS IoT Core- operasi pesawat data	<code>iot:Data-ATS</code>	Digunakan untuk mengirim dan menerima data ke dan dari broker pesan, Device Shadow , dan komponen Rules Engine dari AWS IoT. <code>iot:Data-ATS</code> mengembalikan titik akhir data yang ditandatangani ATS.
AWS IoT Core- operasi pesawat data (warisan)	<code>iot:Data</code>	<code>iot:Data</code> mengembalikan titik akhir data yang VeriSign ditandatangani yang disediakan untuk kompatibilitas mundur. MQTT 5 tidak didukung pada titik akhir Symantec (). <code>iot:Data</code>
AWS IoT Core akses kredensi	<code>iot:CredentialProvider</code>	Digunakan untuk menukar sertifikat X.509 bawaan perangkat dengan kredensial sementara untuk terhubung langsung dengan layanan lain. AWS Untuk informasi selengkapnya tentang menghubungkan ke AWS layanan lain, lihat Mengotori

Tujuan titik akhir	Nilai <i>endpointType</i>	Deskripsi
		sasi Panggilan Langsung ke AWS Layanan.
AWS IoT Device Management- operasi data pekerjaan	<code>iot:Jobs</code>	Digunakan untuk mengaktifkan perangkat berinteraksi dengan layanan AWS IoT Jobs menggunakan API HTTPS Perangkat Pekerjaan.
AWS IoT Operasi Device Advisor	<code>iot:DeviceAdvisor</code>	Jenis titik akhir pengujian yang digunakan untuk menguji perangkat dengan Device Advisor. Untuk informasi selengkapnya, lihat ??? .
AWS IoT Core data beta (pratinjau)	<code>iot:Data-Beta</code>	Tipe endpoint yang dicadangkan untuk rilis beta. Untuk informasi tentang penggunaannya saat ini, lihat ??? .

Anda juga dapat menggunakan nama domain yang sepenuhnya memenuhi syarat (FQDN) Anda sendiri, seperti *example.com*, dan sertifikat server terkait untuk menghubungkan perangkat dengan menggunakan AWS IoT [the section called “Titik akhir yang dapat dikonfigurasi”](#)

AWS IoT SDK perangkat

SDK AWS IoT Perangkat membantu Anda menghubungkan perangkat AWS IoT Core IoT dan mendukung MQTT dan MQTT melalui protokol WSS.

SDK AWS IoT Perangkat berbeda dari AWS SDK karena SDK AWS IoT Perangkat mendukung kebutuhan komunikasi khusus perangkat IoT, tetapi tidak mendukung semua layanan yang didukung oleh SDK. AWS SDK AWS IoT Perangkat kompatibel dengan AWS SDK yang mendukung semua AWS layanan; namun, mereka menggunakan metode otentikasi yang berbeda dan terhubung ke titik akhir yang berbeda, yang dapat membuat penggunaan AWS SDK tidak praktis pada perangkat IoT.

Perangkat seluler

[the section called “AWS SDK Seluler”](#) Mendukung komunikasi perangkat MQTT, beberapa API AWS IoT layanan, dan API layanan lainnya. AWS Jika Anda mengembangkan pada perangkat seluler yang didukung, tinjau SDK-nya untuk melihat apakah itu pilihan terbaik untuk mengembangkan solusi IoT Anda.

C++

AWS IoT SDK Perangkat C ++

AWS IoT C++ Device SDK memungkinkan pengembang untuk membangun aplikasi yang terhubung menggunakan AWS dan API layanan. AWS IoT Core Secara khusus, SDK ini dirancang untuk perangkat yang tidak dibatasi sumber daya dan memerlukan fitur-fitur canggih seperti antrian pesan, dukungan multi-threading, dan fitur bahasa terbaru. Untuk informasi selengkapnya, lihat hal berikut:

- [AWS IoT Perangkat SDK C++ v2 aktif GitHub](#)
- [AWS IoT Perangkat SDK C++ v2 Readme](#)
- [AWS IoT Sampel Perangkat SDK C++ v2](#)
- [AWS IoT Dokumentasi API SDK C++ v2 perangkat](#)

Python

AWS IoT Perangkat SDK untuk Python

AWS IoT Perangkat SDK untuk Python memungkinkan pengembang untuk menulis skrip Python untuk menggunakan perangkat mereka untuk mengakses AWS IoT platform melalui MQTT atau MQTT melalui protokol Secure (WSS). WebSocket Dengan menghubungkan perangkat mereka ke API AWS IoT Core layanan, pengguna dapat bekerja dengan aman dengan broker pesan, aturan, dan layanan Device Shadow yang AWS IoT Core menyediakan dan dengan AWS layanan lain seperti AWS Lambda, Amazon Kinesis, dan Amazon S3, dan banyak lagi.

- [AWS IoT Perangkat SDK untuk Python v2 aktif GitHub](#)
- [AWS IoT Perangkat SDK untuk Python v2 Readme](#)
- [AWS IoT SDK Perangkat untuk Sampel Python v2](#)
- [AWS IoT SDK perangkat untuk dokumentasi API Python v2](#)

JavaScript

AWS IoT Perangkat SDK untuk JavaScript

AWS IoT Perangkat SDK untuk JavaScript memungkinkan pengembang untuk menulis JavaScript aplikasi yang mengakses API AWS IoT Core menggunakan MQTT atau MQTT melalui protokol WebSocket. Ini dapat digunakan di lingkungan Node.js dan aplikasi browser. Untuk informasi selengkapnya, lihat hal berikut:

- [AWS IoT SDK perangkat untuk JavaScript v2 aktif GitHub](#)
- [AWS IoT Perangkat SDK untuk JavaScript v2 Readme](#)
- [AWS IoT SDK Perangkat untuk Sampel JavaScript v2](#)
- [AWS IoT SDK perangkat untuk dokumentasi API JavaScript v2](#)

Java

AWS IoT Perangkat SDK for Java

AWS IoT Device SDK for Java memungkinkan pengembang Java untuk mengakses API melalui MQTT atau MQTT melalui protokol. AWS IoT Core WebSocket SDK mendukung layanan Device Shadow. Anda dapat mengakses bayangan dengan menggunakan metode HTTP, termasuk GET, UPDATE, dan DELETE. SDK juga mendukung model akses bayangan yang disederhanakan, yang memungkinkan pengembang untuk bertukar data dengan bayangan dengan menggunakan metode pengambil dan penyetel, tanpa harus membuat serial atau deserialisasi dokumen JSON apa pun. Untuk informasi selengkapnya, lihat hal berikut:

- [AWS IoT Perangkat SDK for Java v2 aktif GitHub](#)
- [AWS IoT Perangkat SDK for Java v2 Readme](#)
- [AWS IoT Perangkat SDK for Java v2 Sampel](#)
- [AWS IoT Dokumentasi SDK for Java v2 API](#)

Embedded C

AWS IoT Perangkat SDK untuk Embedded C

⚠ Important

SDK ini dimaksudkan untuk digunakan oleh pengembang perangkat lunak tertanam yang berpengalaman.

C-SDK adalah kumpulan file sumber C di bawah lisensi sumber terbuka MIT yang dapat digunakan dalam aplikasi tertanam untuk menghubungkan perangkat IoT dengan aman ke IoT Core. AWS IoT Device SDK for Embedded C AWS Ini termasuk perpustakaan MQTT, JSON Parser, dan Device Shadow AWS IoT dan lainnya. Ini didistribusikan dalam bentuk sumber dan dimaksudkan untuk dibangun ke dalam firmware pelanggan bersama dengan kode aplikasi, perpustakaan lain dan, secara opsional, RTOS (Real Time Operating System).

Umumnya AWS IoT Device SDK for Embedded C ditargetkan pada perangkat terbatas sumber daya yang memerlukan runtime bahasa C yang dioptimalkan. Anda dapat menggunakan SDK pada sistem operasi apa pun dan menghostingnya pada semua jenis prosesor (misalnya, MCU dan MPU). Jika perangkat Anda memiliki memori yang cukup dan sumber daya pemrosesan yang tersedia, kami sarankan Anda menggunakan salah satu SDK AWS IoT Perangkat dan Seluler lainnya, seperti AWS IoT Device SDK for C++, Java, JavaScript atau Python.

Untuk informasi selengkapnya, lihat hal berikut:

- [AWS IoT Perangkat SDK untuk Embedded C aktif GitHub](#)
- [AWS IoT SDK Perangkat untuk Readme C Tertanam](#)
- [AWS IoT SDK Perangkat untuk Sampel C Tertanam](#)

Protokol komunikasi perangkat

AWS IoT Core mendukung perangkat dan klien yang menggunakan protokol MQTT dan MQTT over WebSocket Secure (WSS) untuk mempublikasikan dan berlangganan pesan, serta perangkat dan klien yang menggunakan protokol HTTPS untuk mempublikasikan pesan. Semua protokol mendukung IPv4 dan IPv6. Bagian ini menjelaskan opsi koneksi yang berbeda untuk perangkat dan klien.

TLS 1.2 dan TLS 1.3

AWS IoT Core menggunakan [TLS versi 1.2](#) dan [TLS versi 1.3](#) untuk mengenkripsi semua komunikasi. Saat menghubungkan perangkat ke AWS IoT Core, klien dapat mengirim [ekstensi](#)

[Server Name Indication \(SNI\)](#), yang tidak diperlukan tetapi sangat disarankan. Untuk menggunakan fitur seperti [pendaftaran multi-akun](#), [domain khusus](#), dan [titik akhir VPC](#), Anda harus menggunakan ekstensi SNI. Untuk informasi selengkapnya, lihat [Keamanan Transportasi di AWS IoT](#).

[AWS IoT SDK perangkat](#) Dukungan MQTT dan MQTT melalui WSS dan mendukung persyaratan keamanan koneksi klien. Kami merekomendasikan menggunakan [AWS IoT SDK perangkat](#) untuk menghubungkan klien ke AWS IoT.

Protokol, pemetaan port, dan otentikasi

Bagaimana perangkat atau klien terhubung ke broker pesan dengan menggunakan titik akhir perangkat tergantung pada protokol yang digunakannya. Tabel berikut mencantumkan protokol yang didukung endpoint AWS IoT perangkat dan metode otentikasi serta port yang mereka gunakan.

Protokol, otentikasi, dan pemetaan port

Protokol	Operasi yang didukung	Autentikasi	Port	Nama protokol ALPN
MQTT lebih WebSocket	Publikasikan, Berlangganan	Tanda Tangan Versi 4	443	N/A
MQTT lebih WebSocket	Publikasikan, Berlangganan	Otentikasi kustom	443	N/A
MQTT	Publikasikan, Berlangganan	Sertifikat klien X.509	443 †	x-amzn-mqtt-ca
MQTT	Publikasikan, Berlangganan	Sertifikat klien X.509	8883	N/A
MQTT	Publikasikan, Berlangganan	Otentikasi kustom	443 †	mqtt
HTTPS	Publikasikan saja	Tanda Tangan Versi 4	443	N/A
HTTPS	Publikasikan saja	Sertifikat klien X.509	443 †	x-amzn-https-ca

Protokol	Operasi yang didukung	Autentikasi	Port	Nama protokol ALPN
HTTPS	Publikasikan saja	Sertifikat klien X.509	8443	N/A
HTTPS	Publikasikan saja	Otentikasi kustom	443	N/A

Negosiasi Protokol Lapisan Aplikasi (ALPN)

† Klien yang terhubung pada port 443 dengan otentikasi sertifikat klien X.509 harus menerapkan ekstensi TLS [Application Layer Protocol Negotiation \(ALPN\)](#) dan menggunakan [nama protokol ALPN yang tercantum dalam ALPN](#) yang dikirim oleh klien sebagai bagian dari pesan `ProtocolNameList.ClientHello`

[Pada port 443, titik akhir IoT:Data-ATS mendukung ALPN x-amzn-http-ca HTTP, tetapi titik akhir IoT:Jobs tidak.](#)

[Pada port 8443 HTTPS dan port 443 MQTT dengan ALPN x-amzn-mqtt-ca, otentikasi khusus tidak dapat digunakan.](#)

Klien terhubung ke titik akhir perangkat mereka Akun AWS. Lihat [the section called “AWS IoT data perangkat dan titik akhir layanan”](#) untuk informasi tentang cara menemukan titik akhir perangkat akun Anda.

Note

AWS SDK tidak memerlukan seluruh URL. Mereka hanya memerlukan nama host endpoint seperti [pubsub.pycontoh untuk AWS IoT Device SDK for Python](#) on. GitHub Melewati seluruh URL seperti yang disediakan dalam tabel berikut dapat menghasilkan kesalahan seperti nama host yang tidak valid.

Menghubungkan ke AWS IoT Core

Protokol	Titik akhir atau URL
MQTT	<i>iot-endpoint</i>
MQTT melalui WSS	<code>wss://<i>iot-endpoint</i> /mqtt</code>
HTTPS	<code>https://<i>iot-endpoint</i> /topics</code>

Memilih protokol untuk komunikasi perangkat Anda

Untuk sebagian besar komunikasi perangkat IoT melalui titik akhir perangkat, Anda akan ingin menggunakan MQTT atau MQTT melalui protokol WSS; Namun, titik akhir perangkat juga mendukung HTTPS. Tabel berikut membandingkan bagaimana AWS IoT Core menggunakan dua protokol untuk komunikasi perangkat.

AWS IoT protokol perangkat side-by-side

Fitur	MQTT	HTTPS
Publikasi/Berlangganan dukungan	Publikasikan dan berlangganan	Publikasikan saja
Dukungan SDK	AWS SDK perangkat mendukung protokol MQTT dan WSS	Tidak ada dukungan SDK, tetapi Anda dapat menggunakan metode khusus bahasa untuk membuat permintaan HTTPS
Dukungan Kualitas Layanan	MQTT QoS level 0 dan 1	QoS didukung dengan melewati parameter string kueri <code>?qos=qos</code> di mana nilainya bisa 0 atau 1. Anda dapat menambahkan string kueri ini untuk mempublikasikan pesan dengan nilai QoS yang Anda inginkan.

Fitur	MQTT	HTTPS
Dapat menerima pesan yang terlewatkan saat perangkat sedang offline	Ya	Tidak
<code>clientId</code> dukungan lapangan	Ya	Tidak
Deteksi pemutusan perangkat	Ya	Tidak
Komunikasi yang aman	Ya. Lihat Protokol, pemetaan port, dan otentikasi	Ya. Lihat Protokol, pemetaan port, dan otentikasi
Definisi topik	Aplikasi didefinisikan	Aplikasi didefinisikan
Format data pesan	Aplikasi didefinisikan	Aplikasi didefinisikan
Protokol overhead	Lebih rendah	Lebih tinggi
Konsumsi daya	Lebih rendah	Lebih tinggi

Batas durasi koneksi

Koneksi HTTPS tidak dijamin bertahan lebih lama dari waktu yang diperlukan untuk menerima dan menanggapi permintaan.

Durasi koneksi MQTT tergantung pada fitur otentikasi yang Anda gunakan. Tabel berikut mencantumkan durasi koneksi maksimum dalam kondisi ideal untuk setiap fitur.

Durasi koneksi MQTT dengan fitur otentikasi

Fitur	Durasi ^{maksimum*}
Sertifikat klien X.509	1—2 minggu
Otentikasi kustom	1—2 minggu
Tanda Tangan Versi 4	Hingga 24 jam

* Tidak dijamin

Dengan sertifikat X.509 dan otentikasi khusus, durasi koneksi tidak memiliki batas keras, tetapi bisa sesingkat beberapa menit. Gangguan koneksi dapat terjadi karena berbagai alasan. Daftar berikut berisi beberapa alasan paling umum.

- Gangguan ketersediaan Wi-Fi
- Gangguan koneksi penyedia layanan Internet (ISP)
- Tambalan layanan
- Penyebaran layanan
- Penskalaan otomatis layanan
- Host layanan tidak tersedia
- Masalah dan pembaruan penyeimbang beban
- Kesalahan sisi klien

Perangkat Anda harus menerapkan strategi untuk mendeteksi pemutusan dan penyambungan kembali. Untuk informasi tentang peristiwa pemutusan sambungan dan panduan tentang cara menanganinya, lihat [??? di???](#).

MQTT

[MQTT](#) (Message Queuing Telemetry Transport) adalah protokol pesan ringan dan diadopsi secara luas yang dirancang untuk perangkat terbatas. AWS IoT Core dukungan untuk MQTT didasarkan pada spesifikasi MQTT [v3.1.1](#) dan [spesifikasi MQTT v5.0](#), dengan beberapa perbedaan, seperti yang [didokumentasikan](#) dalam [the section called “AWS IoT perbedaan dari spesifikasi MQTT”](#). Sebagai versi terbaru dari standar, MQTT 5 memperkenalkan beberapa fitur utama yang membuat sistem berbasis MQTT lebih kuat, termasuk peningkatan skalabilitas baru, pelaporan kesalahan yang ditingkatkan dengan respons kode alasan, pengatur waktu kedaluwarsa pesan dan sesi, dan header pesan pengguna khusus. Untuk informasi selengkapnya tentang fitur MQTT 5 yang AWS IoT Core mendukung, lihat fitur yang didukung [MQTT 5](#). AWS IoT Core juga mendukung komunikasi lintas versi MQTT (MQTT 3 dan MQTT 5). Penerbit MQTT 3 dapat mengirim pesan MQTT 3 ke pelanggan MQTT 5 yang akan menerima pesan publikasi MQTT 5, dan sebaliknya.

AWS IoT Core mendukung koneksi perangkat yang menggunakan protokol MQTT dan MQTT melalui protokol WSS dan yang diidentifikasi oleh ID klien. [AWS IoT SDK perangkat](#) Dukungan kedua protokol dan merupakan cara yang disarankan untuk menghubungkan perangkat ke. AWS IoT Core AWS IoT Perangkat SDK mendukung fungsi yang diperlukan untuk perangkat dan klien untuk terhubung dan mengakses AWS IoT layanan. Perangkat SDK mendukung protokol otentikasi yang diperlukan

AWS IoT layanan dan persyaratan ID koneksi yang diperlukan oleh protokol MQTT dan MQTT melalui protokol WSS. Untuk informasi tentang cara menyambung AWS IoT menggunakan SDK AWS Perangkat dan tautan ke contoh AWS IoT dalam bahasa yang didukung, lihat [the section called “Menghubungkan dengan MQTT menggunakan SDK Perangkat AWS IoT”](#). Untuk informasi selengkapnya tentang metode otentikasi dan pemetaan port untuk pesan MQTT, lihat [Protokol, pemetaan port, dan otentikasi](#)

Meskipun kami menyarankan untuk menggunakan SDK AWS IoT Perangkat untuk terhubung AWS IoT, mereka tidak diperlukan. Namun, jika Anda tidak menggunakan SDK AWS IoT Perangkat, Anda harus menyediakan keamanan koneksi dan komunikasi yang diperlukan. Klien harus mengirimkan [ekstensi TLS Server Name Indication \(SNI\)](#) dalam permintaan koneksi. Upaya koneksi yang tidak termasuk SNI ditolak. Untuk informasi selengkapnya, lihat [Keamanan Transportasi di AWS IoT](#). Klien yang menggunakan pengguna IAM dan AWS kredensialnya untuk mengautentikasi klien harus memberikan otentikasi [Signature](#) Version 4 yang benar.

Dalam topik ini:

- [Menghubungkan dengan MQTT menggunakan SDK Perangkat AWS IoT](#)
- [Opsi Kualitas Layanan \(QoS\) MQTT](#)
- [Sesi persisten MQTT](#)
- [Pesan yang disimpan MQTT](#)
- [Pesan MQTT Last Will and Testament \(LWT\)](#)
- [Menggunakan ConnectAttributes](#)
- [Fitur yang didukung MQTT 5](#)
- [MQTT 5 properti](#)
- [Kode alasan MQTT](#)
- [AWS IoT perbedaan dari spesifikasi MQTT](#)

Menghubungkan dengan MQTT menggunakan SDK Perangkat AWS IoT

Bagian ini berisi tautan ke SDK AWS IoT Perangkat dan kode sumber program contoh yang menggambarkan cara menghubungkan perangkat. AWS IoT Contoh aplikasi yang ditautkan di sini menunjukkan cara terhubung AWS IoT menggunakan protokol MQTT dan MQTT melalui WSS.

Note

AWS IoT Perangkat SDK telah merilis klien MQTT 5.

C++

Menggunakan SDK Perangkat AWS IoT C ++ untuk menghubungkan perangkat

- [Kode sumber aplikasi contoh yang menunjukkan contoh koneksi MQTT di C ++](#)
- [AWS IoT C ++ Perangkat SDK v2 aktif GitHub](#)

Python

Menggunakan AWS IoT Device SDK untuk Python untuk menghubungkan perangkat

- [Kode sumber aplikasi contoh yang menunjukkan contoh koneksi MQTT dengan Python](#)
- [AWS IoT Perangkat SDK untuk Python v2 aktif GitHub](#)

JavaScript

Menggunakan AWS IoT Device SDK JavaScript untuk menghubungkan perangkat

- [Kode sumber aplikasi contoh yang menunjukkan contoh koneksi MQTT di JavaScript](#)
- [AWS IoT SDK perangkat untuk JavaScript v2 aktif GitHub](#)

Java

Menggunakan AWS IoT Device SDK for Java untuk menghubungkan perangkat

Note

AWS IoT Device SDK for Java v2 sekarang mendukung pengembangan Android. Untuk informasi selengkapnya, lihat [SDK AWS IoT perangkat untuk Android](#).

- [Kode sumber aplikasi contoh yang menunjukkan contoh koneksi MQTT di Java](#)

- [AWS IoT Perangkat SDK for Java v2 aktif GitHub](#)

Embedded C

Menggunakan AWS IoT Device SDK untuk Embedded C untuk menghubungkan perangkat

Important

SDK ini dimaksudkan untuk digunakan oleh pengembang perangkat lunak tertanam yang berpengalaman.

- [Kode sumber aplikasi contoh yang menunjukkan contoh koneksi MQTT di Embedded C](#)
- [AWS IoT Perangkat SDK untuk Embedded C aktif GitHub](#)

Opsi Kualitas Layanan (QoS) MQTT

AWS IoT dan SDK AWS IoT Perangkat mendukung tingkat [MQTT Quality of Service \(QoS\)](#) dan. 0 1 Protokol MQTT mendefinisikan tingkat ketiga QoS, level2, tetapi tidak mendukungnya. AWS IoT Hanya protokol MQTT yang mendukung fitur QoS. HTTPS mendukung QoS dengan meneruskan parameter string kueri ?qos=qos di mana nilainya bisa 0 atau 1.

Tabel ini menjelaskan bagaimana setiap level QoS memengaruhi pesan yang dipublikasikan ke dan oleh broker pesan.

Dengan tingkat QoS...	Pesannya adalah...	Komentar
QoS tingkat 0	Dikirim nol atau lebih kali	Level ini harus digunakan untuk pesan yang dikirim melalui tautan komunikasi yang andal atau yang dapat dilewatkan tanpa masalah.
QoS tingkat 1	Dikirim setidaknya satu kali, dan kemudian berulang kali sampai PUBACK tanggapan diterima	Pesan tidak dianggap lengkap sampai pengirim menerima PUBACK tanggapan untuk

Dengan tingkat QoS...	Pesannya adalah...	Komentar
		menunjukkan pengiriman yang berhasil.

Sesi persisten MQTT

Sesi persisten menyimpan langganan dan pesan klien, dengan Quality of Service (QoS) 1, yang belum diakui oleh klien. Ketika perangkat terhubung kembali ke sesi persisten, sesi dilanjutkan, langganan dipulihkan, dan pesan berlangganan yang tidak diakui diterima dan disimpan sebelum koneksi ulang dikirim ke klien.

Pemrosesan pesan yang disimpan direkam dalam CloudWatch dan CloudWatch Log. Untuk informasi tentang entri yang ditulis ke CloudWatch dan CloudWatch Log, lihat [Metrik broker pesan](#) dan [Entri log antrian](#).

Membuat sesi persisten

Di MQTT 3, Anda membuat sesi persisten MQTT dengan mengirim CONNECT pesan dan menyetel flag ke `cleanSession 0`. Jika tidak ada sesi untuk klien yang mengirim CONNECT pesan, sesi persisten baru akan dibuat. Jika sesi sudah ada untuk klien, klien melanjutkan sesi yang ada. Untuk membuat sesi bersih, Anda mengirim CONNECT pesan dan mengatur `cleanSession` bendera ke 1, dan broker tidak akan menyimpan status sesi apa pun saat klien terputus.

Di MQTT 5, Anda menangani sesi persisten dengan menyetel bendera dan `Clean Start`. `Session Expiry Interval Clean Start` mengontrol awal sesi penghubung dan akhir sesi sebelumnya. Ketika Anda menetapkan `Clean Start = 1`, sesi baru dibuat dan sesi sebelumnya dihentikan jika ada. Saat Anda mengatur `Clean Start = 0`, sesi penghubung melanjutkan sesi sebelumnya jika ada. `Interval Kedaluwarsa Sesi` mengontrol akhir sesi penghubung. `Session Expiry Interval` menentukan waktu, dalam detik (4-byte integer), bahwa sesi akan bertahan setelah terputus. Pengaturan `Session Expiry interval = 0` menyebabkan sesi segera berakhir setelah terputus. Jika `Interval Kedaluwarsa Sesi` tidak ditentukan dalam pesan CONNECT, defaultnya adalah 0.

MQTT 5 Mulai Bersih dan Kedaluwarsa Sesi

Nilai properti	Deskripsi
<code>Clean Start= 1</code>	Membuat sesi baru dan mengakhiri sesi sebelumnya jika ada.

Nilai properti	Deskripsi
<code>Clean Start= 0</code>	Melanjutkan sesi jika sesi sebelumnya ada.
<code>Session Expiry Interval> 0</code>	Bertahan sesi.
<code>Session Expiry interval= 0</code>	Tidak bertahan sesi.

Dalam MQTT 5, jika Anda mengatur `Clean Start = 1` dan `Session Expiry Interval = 0`, ini setara dengan sesi bersih MQTT 3. Jika Anda mengatur `Clean Start = 0` dan `Session Expiry Interval > 0`, ini setara dengan sesi persisten MQTT 3.

Note

Versi Cross MQTT (MQTT 3 dan MQTT 5) sesi persisten tidak didukung. Sesi persisten MQTT 3 tidak dapat dilanjutkan sebagai sesi MQTT 5, dan sebaliknya.

Operasi selama sesi persisten

Klien menggunakan `sessionPresent` atribut dalam pesan koneksi yang diakui (CONNACK) untuk menentukan apakah ada sesi persisten. Jika `sessionPresent` ada1, sesi persisten hadir dan pesan yang disimpan untuk klien dikirim ke klien setelah klien menerima CONNACK, seperti yang dijelaskan dalam [lalu lintas Pesan setelah koneksi ulang ke sesi persisten](#). Jika `sessionPresent` ya1, klien tidak perlu berlangganan kembali. Namun, jika `sessionPresent` ada0, tidak ada sesi persisten yang hadir dan klien harus berlangganan kembali ke filter topiknya.

Setelah klien bergabung dengan sesi persisten, klien dapat mempublikasikan pesan dan berlangganan filter topik tanpa tanda tambahan pada setiap operasi.

Lalu lintas pesan setelah koneksi ulang ke sesi persisten

Sesi persisten mewakili hubungan yang sedang berlangsung antara klien dan broker pesan MQTT. Ketika klien terhubung ke broker pesan menggunakan sesi persisten, broker pesan menyimpan semua langganan yang dibuat klien selama koneksi. Ketika klien terputus, broker pesan menyimpan pesan QoS 1 yang tidak diakui dan pesan QoS 1 baru yang dipublikasikan ke topik yang menjadi

langganan klien. Pesan disimpan sesuai dengan batas akun. Pesan yang melebihi batas akan dihapus. Untuk informasi selengkapnya tentang batas pesan persisten, lihat [AWS IoT Core titik akhir dan kuota](#). Ketika klien terhubung kembali ke sesi persistennya, semua langganan dipulihkan dan semua pesan yang disimpan dikirim ke klien dengan kecepatan maksimum 10 pesan per detik. Di MQTT 5, jika QoS1 keluar dengan Interval Kedaluwarsa Pesan berakhir saat klien offline, setelah koneksi dilanjutkan, klien tidak akan menerima pesan kedaluwarsa.

Setelah koneksi ulang, pesan yang disimpan dikirim ke klien, dengan kecepatan yang dibatasi hingga 10 pesan tersimpan per detik, bersama dengan lalu lintas pesan saat ini hingga [Publish requests per second per connection](#) batas tercapai. Karena tingkat pengiriman pesan yang disimpan terbatas, akan memakan waktu beberapa detik untuk mengirimkan semua pesan yang disimpan jika sesi memiliki lebih dari 10 pesan tersimpan untuk dikirim setelah koneksi ulang.

Mengakhiri sesi persisten

Sesi persisten dapat diakhiri dengan cara-cara berikut:

- Waktu kedaluwarsa sesi persisten berlalu. Timer kedaluwarsa sesi persisten dimulai ketika broker pesan mendeteksi bahwa klien telah terputus, baik oleh klien memutuskan sambungan atau waktu koneksi habis.
- Klien mengirimkan CONNECT pesan yang menyetel `cleanSession` bendera ke 1.

Di MQTT 3, nilai default dari waktu kedaluwarsa sesi persisten adalah satu jam, dan ini berlaku untuk semua sesi di akun.

Di MQTT 5, Anda dapat mengatur Interval Kedaluwarsa Sesi untuk setiap sesi pada paket CONNECT dan DISCONNECT.

Untuk Interval Kedaluwarsa Sesi pada paket DISCONNECT:

- Jika sesi saat ini memiliki Interval Kedaluwarsa Sesi 0, Anda tidak dapat mengatur Interval Kedaluwarsa Sesi menjadi lebih besar dari 0 pada paket DISCONNECT.
- Jika sesi saat ini memiliki Interval Kedaluwarsa Sesi lebih besar dari 0, dan Anda mengatur Interval Kedaluwarsa Sesi ke 0 pada paket DISCONNECT, sesi akan berakhir pada DISCONNECT.
- Jika tidak, Interval Kedaluwarsa Sesi pada paket DISCONNECT akan memperbarui Interval Kedaluwarsa Sesi sesi saat ini.

Note

Pesan yang disimpan menunggu untuk dikirim ke klien ketika sesi berakhir dibuang; Namun, mereka masih ditagih pada tingkat pesan standar, meskipun mereka tidak dapat dikirim. Untuk informasi selengkapnya tentang harga pesan, lihat [AWS IoT Core Harga](#). Anda dapat mengonfigurasi interval waktu kedaluwarsa.

Koneksi ulang setelah sesi persisten telah kedaluwarsa

Jika klien tidak terhubung kembali ke sesi persistennya sebelum kedaluwarsa, sesi berakhir dan pesan yang disimpan akan dibuang. Ketika klien tersambung kembali setelah sesi kedaluwarsa dengan `cleanSession` tanda ke`0`, layanan akan membuat sesi persisten baru. Setiap langganan atau pesan dari sesi sebelumnya tidak tersedia untuk sesi ini karena mereka dibuang ketika sesi sebelumnya kedaluwarsa.

Biaya pesan sesi persisten

Pesan dibebankan kepada Anda Akun AWS ketika broker pesan mengirim pesan ke klien atau sesi persisten offline. Ketika perangkat offline dengan sesi persisten menghubungkan kembali dan melanjutkan sesi, pesan yang disimpan dikirim ke perangkat dan dibebankan ke akun Anda lagi. Untuk informasi selengkapnya tentang harga pesan, lihat [AWS IoT Core harga - Pesan](#).

Waktu kedaluwarsa sesi persisten default satu jam dapat ditingkatkan dengan menggunakan proses peningkatan batas standar. Perhatikan bahwa meningkatkan waktu kedaluwarsa sesi dapat meningkatkan biaya pesan Anda karena waktu tambahan dapat memungkinkan lebih banyak pesan disimpan untuk perangkat offline dan pesan tambahan tersebut akan dibebankan ke akun Anda dengan tarif pesan standar. Waktu kedaluwarsa sesi adalah perkiraan dan sesi dapat bertahan hingga 30 menit lebih lama dari batas akun; Namun, sesi tidak akan lebih pendek dari batas akun. Untuk informasi selengkapnya tentang batas sesi, lihat [AWS Service Quotas](#).

Pesan yang disimpan MQTT

AWS IoT Core mendukung flag `REACH` yang dijelaskan dalam protokol MQTT. Saat klien menyetel flag `RETAIN` pada pesan MQTT yang diterbitkannya, AWS IoT Core akan menyimpan pesan tersebut. Kemudian dapat dikirim ke pelanggan baru, diambil dengan memanggil [GetRetainedMessage](#) operasi, dan dilihat di [AWS IoT konsol](#).

Contoh menggunakan pesan yang dipertahankan MQTT

- Sebagai pesan konfigurasi awal

Pesan yang disimpan MQTT dikirim ke klien setelah klien berlangganan topik. Jika Anda ingin semua klien yang berlangganan topik menerima pesan yang disimpan MQTT tepat setelah langganan mereka, Anda dapat mempublikasikan pesan konfigurasi dengan set tanda RETAIN. Klien berlangganan juga menerima pembaruan konfigurasi itu setiap kali pesan konfigurasi baru diterbitkan.

- Sebagai pesan status terakhir yang diketahui

Perangkat dapat mengatur tanda RETAIN pada pesan status saat ini sehingga AWS IoT Core akan menyimpannya. Saat aplikasi terhubung atau terhubung kembali, mereka dapat berlangganan topik ini dan mendapatkan status terakhir yang dilaporkan segera setelah berlangganan topik pesan yang dipertahankan. Dengan cara ini mereka dapat menghindari keharusan menunggu hingga pesan berikutnya dari perangkat untuk melihat keadaan saat ini.

Di bagian ini:

- [Tugas umum dengan pesan yang disimpan MQTT di AWS IoT Core](#)
- [Penagihan dan pesan yang disimpan](#)
- [Membandingkan pesan yang dipertahankan MQTT dan sesi persisten MQTT](#)
- [MQTT menyimpan pesan dan Device Shadows AWS IoT](#)

Tugas umum dengan pesan yang disimpan MQTT di AWS IoT Core

AWS IoT Core menyimpan pesan MQTT dengan set bendera RESTAIN. Pesan yang disimpan ini dikirim ke semua klien yang telah berlangganan topik, sebagai pesan MQTT normal, dan mereka juga disimpan untuk dikirim ke pelanggan baru ke topik tersebut.

Pesan yang disimpan MQTT memerlukan tindakan kebijakan khusus untuk memberi wewenang kepada klien untuk mengaksesnya. Untuk contoh penggunaan kebijakan pesan yang dipertahankan, lihat [Contoh kebijakan pesan yang dipertahankan](#).

Bagian ini menjelaskan operasi umum yang melibatkan pesan yang disimpan.

- Membuat pesan yang disimpan

Klien menentukan apakah pesan dipertahankan saat memublikasikan pesan MQTT. Klien dapat menyetel flag REACH saat memublikasikan pesan menggunakan [Device SDK](#). Aplikasi dan layanan dapat mengatur tanda RETAIN saat mereka menggunakan [Publishtindakan](#) untuk memublikasikan pesan MQTT.

Hanya satu pesan per nama topik yang dipertahankan. Pesan baru dengan set tanda RETAIN yang dipublikasikan ke topik menggantikan pesan tertahan yang sudah ada yang dikirim ke topik sebelumnya.

CATATAN: Anda tidak dapat memublikasikan ke [topik yang dicadangkan](#) dengan set tanda RETAIN.

- Berlangganan topik pesan yang dipertahankan

Klien berlangganan topik pesan yang dipertahankan seperti halnya topik pesan MQTT lainnya. Pesan yang disimpan yang diterima dengan berlangganan topik pesan yang dipertahankan memiliki tanda RETAIN disetel.

Pesan yang disimpan akan dihapus dari AWS IoT Core saat klien menerbitkan pesan yang disimpan dengan muatan pesan 0-byte ke topik pesan yang dipertahankan. Klien yang telah berlangganan topik pesan yang dipertahankan juga akan menerima pesan 0-byte.

Berlangganan filter topik wild card yang menyertakan topik pesan yang dipertahankan memungkinkan klien menerima pesan berikutnya yang dipublikasikan ke topik pesan yang disimpan, tetapi tidak mengirimkan pesan yang disimpan saat berlangganan.

CATATAN: Untuk menerima pesan yang disimpan saat berlangganan, filter topik dalam permintaan langganan harus sama persis dengan topik pesan yang disimpan.

Pesan yang disimpan yang diterima saat berlangganan topik pesan yang dipertahankan memiliki tanda RETAIN disetel. Pesan yang disimpan yang diterima oleh klien berlangganan setelah berlangganan, jangan.

- Mengambil pesan yang disimpan

Pesan yang disimpan dikirim ke klien secara otomatis ketika mereka berlangganan topik dengan pesan yang disimpan. Agar klien menerima pesan yang disimpan saat berlangganan, klien harus berlangganan nama topik yang tepat dari pesan yang disimpan. Berlangganan filter topik wild card yang menyertakan topik pesan yang dipertahankan memungkinkan klien menerima pesan

berikutnya yang dipublikasikan ke topik pesan yang disimpan, tetapi tidak mengirimkan pesan yang disimpan saat berlangganan.

Layanan dan aplikasi dapat mencantumkan dan mengambil pesan yang disimpan dengan menelepon [ListRetainedMessages](#) dan [GetRetainedMessage](#)

Klien tidak dicegah untuk memublikasikan pesan ke topik pesan yang dipertahankan tanpa menyetel tanda RETAIN. Hal ini dapat menyebabkan hasil yang tidak terduga, seperti pesan yang disimpan tidak cocok dengan pesan yang diterima dengan berlangganan topik.

Dengan MQTT 5, jika pesan yang disimpan memiliki Interval Kedaluwarsa Pesan yang disetel dan pesan yang disimpan kedaluwarsa, pelanggan baru yang berlangganan topik tersebut tidak akan menerima pesan yang disimpan setelah berlangganan berhasil.

- Daftar topik pesan yang dipertahankan

[Anda dapat mencantumkan pesan yang disimpan dengan menelepon ListRetainedMessages dan pesan yang disimpan dapat dilihat di konsol.AWS IoT](#)

- Mendapatkan detail pesan yang disimpan

Anda bisa mendapatkan detail pesan yang disimpan dengan menelepon [GetRetainedMessage](#) dan mereka dapat dilihat di [AWS IoT konsol](#).

- Mempertahankan pesan Will

MQTT [Akan pesan](#) yang dibuat ketika perangkat terhubung dapat dipertahankan dengan mengatur Will Retain bendera di bidang. Connect Flag bits

- Menghapus pesan yang disimpan

Perangkat, aplikasi, dan layanan dapat menghapus pesan yang disimpan dengan menerbitkan pesan dengan set tanda RETAIN dan muatan pesan kosong (0-byte) ke nama topik pesan yang disimpan untuk dihapus. Pesan semacam itu menghapus pesan yang disimpan dari AWS IoT Core, dikirim ke klien dengan berlangganan topik, tetapi tidak dipertahankan oleh. AWS IoT Core

[Pesan yang disimpan juga dapat dihapus secara interaktif dengan mengakses pesan yang disimpan di konsol.AWS IoT](#) Pesan yang disimpan yang dihapus dengan menggunakan [AWS IoT konsol](#) juga mengirim pesan 0-byte ke klien yang telah berlangganan topik pesan yang dipertahankan.

Pesan yang disimpan tidak dapat dipulihkan setelah dihapus. Klien perlu memublikasikan pesan tertahan baru untuk menggantikan pesan yang dihapus.

- Debugging dan pemecahan masalah pesan yang disimpan

[AWS IoT Konsol](#) menyediakan beberapa alat untuk membantu Anda memecahkan masalah pesan yang disimpan:

- Halaman [pesan yang ditahan](#)

Halaman pesan yang disimpan di AWS IoT konsol menyediakan daftar paginasi pesan yang disimpan yang telah disimpan oleh Akun Anda di Wilayah saat ini. Dari halaman ini, Anda dapat:

- Lihat detail setiap pesan yang disimpan, seperti payload pesan, QoS, waktu diterima.
- Perbarui isi pesan yang disimpan.
- Hapus pesan yang disimpan.
- Klien [uji MQTT](#)

Halaman klien pengujian MQTT di AWS IoT konsol dapat berlangganan dan mempublikasikan ke topik MQTT. Opsi terbitkan memungkinkan Anda menyetel tanda RETAIN pada pesan yang Anda terbitkan untuk mensimulasikan perilaku perangkat Anda.

Beberapa hasil yang tidak terduga mungkin merupakan hasil dari aspek-aspek ini tentang bagaimana pesan yang disimpan diimplementasikan AWS IoT Core.

- Batas pesan yang dipertahankan

Ketika akun telah menyimpan jumlah maksimum pesan yang disimpan, AWS IoT Core mengembalikan respons terbatas ke pesan yang diterbitkan dengan set RETAIN dan muatan lebih besar dari 0 byte hingga beberapa pesan yang disimpan dihapus dan jumlah pesan yang dipertahankan turun di bawah batas.

- Pesanan pengiriman pesan yang dipertahankan

Urutan pesan yang disimpan dan pengiriman pesan berlangganan tidak dijamin.

Penagihan dan pesan yang disimpan

Menerbitkan pesan dengan tanda RETAIN yang disetel dari klien, menggunakan AWS IoT konsol, atau dengan menelepon [Publish](#) menimbulkan biaya pesan tambahan yang dijelaskan dalam [AWS IoT Core harga - Pesan](#).

Mengambil pesan yang disimpan oleh klien, dengan menggunakan AWS IoT konsol, atau dengan menelepon [GetRetainedMessage](#) menimbulkan biaya pengiriman pesan selain biaya penggunaan API normal. Biaya tambahan dijelaskan dalam [AWS IoT Core harga - Pesan](#).

[MQTT Akan pesan yang dipublikasikan ketika perangkat terputus secara tak terduga dikenakan biaya pesan yang dijelaskan dalam harga - Pesan.AWS IoT Core](#)

Untuk informasi selengkapnya tentang biaya pengiriman pesan, lihat [AWS IoT Core harga - Pesan](#).

Membandingkan pesan yang dipertahankan MQTT dan sesi persisten MQTT

Pesan yang disimpan dan sesi persisten adalah fitur standar MQTT yang memungkinkan perangkat menerima pesan yang diterbitkan saat sedang offline. Pesan yang disimpan dapat dipublikasikan dari sesi persisten. Bagian ini menjelaskan aspek-aspek kunci dari fitur-fitur ini dan bagaimana mereka bekerja sama.

	Pesan yang disimpan	Sesi persisten
Fitur utama	<p>Pesan yang disimpan dapat digunakan untuk mengonfigurasi atau memberi tahu kelompok besar perangkat setelah terhubung.</p> <p>Pesan yang disimpan juga dapat digunakan di mana Anda ingin perangkat hanya menerima pesan terakhir yang dipublikasikan ke topik setelah koneksi ulang.</p>	<p>Sesi persisten berguna untuk perangkat yang memiliki konektivitas intermiten dan dapat melewatkan beberapa pesan penting.</p> <p>Perangkat dapat terhubung dengan sesi persisten untuk menerima pesan yang dikirim saat sedang offline.</p>
Contoh	<p>Pesan yang disimpan dapat memberikan informasi konfigurasi perangkat tentang lingkungan mereka ketika mereka online. Konfigurasi awal dapat mencakup daftar topik pesan lain yang harus berlangganan atau informasi tentang bagaimana seharusnya mengkonfigurasi zona waktu lokalnya.</p>	<p>Perangkat yang terhubung melalui jaringan seluler dengan konektivitas intermiten dapat menggunakan sesi persisten untuk menghindari hilangnya pesan penting yang dikirim saat perangkat berada di luar jangkauan jaringan atau perlu mematikan radio selulernya.</p>

	Pesan yang disimpan	Sesi persisten
Pesan yang diterima saat berlangganan awal suatu topik	Setelah berlangganan topik dengan pesan yang disimpan, pesan tertahan terbaru diterima.	Setelah berlangganan topik tanpa pesan yang dipertahankan, tidak ada pesan yang diterima sampai seseorang dipublikasikan ke topik tersebut.
Topik berlangganan setelah koneksi ulang	Tanpa sesi persisten, klien harus berlangganan topik setelah koneksi ulang.	Topik berlangganan dipulihkan setelah koneksi ulang.
Pesan diterima setelah koneksi ulang	Setelah berlangganan topik dengan pesan yang disimpan, pesan tertahan terbaru diterima.	Semua pesan yang diterbitkan dengan QOS = 1 dan berlangganan dengan QOS = 1 saat perangkat terputus dikirim setelah perangkat terhubung kembali.
Kedaluwarsa data/sesi	Di MQTT 3, pesan yang disimpan tidak kedaluwarsa. Mereka disimpan sampai diganti atau dihapus. Di MQTT 5, pesan yang disimpan akan kedaluwarsa setelah interval kedaluwarsa pesan yang Anda tetapkan. Untuk informasi selengkapnya, lihat Kedaluwarsa Pesan .	Sesi persisten kedaluwarsa jika klien tidak terhubung kembali dalam periode batas waktu. Setelah sesi persisten berakhir, langganan klien dan pesan tersimpan yang diterbitkan dengan QOS = 1 dan berlangganan dengan QOS = 1 saat perangkat terputus akan dihapus. Pesan kedaluwarsa tidak akan terkirim. Untuk informasi selengkapnya tentang kedaluwarsa sesi dengan sesi persisten, lihat the section called “Sesi persisten MQTT”

Untuk informasi tentang sesi persisten, lihat [the section called “Sesi persisten MQTT”](#).

Dengan Pesan Tertahan, klien penerbitan menentukan apakah pesan harus disimpan dan dikirim ke perangkat setelah tersambung, apakah pesan tersebut memiliki sesi sebelumnya atau tidak. Pilihan untuk menyimpan pesan dibuat oleh penerbit dan pesan yang disimpan dikirimkan ke semua klien saat ini dan masa depan yang berlangganan dengan langganan QoS 0 atau QoS 1. Pesan yang disimpan hanya menyimpan satu pesan pada topik tertentu pada satu waktu.

Ketika akun telah menyimpan jumlah maksimum pesan yang disimpan, AWS IoT Core mengembalikan respons terbatas ke pesan yang diterbitkan dengan set RETAIN dan muatan lebih besar dari 0 byte hingga beberapa pesan yang disimpan dihapus dan jumlah pesan yang dipertahankan turun di bawah batas.

MQTT menyimpan pesan dan Device Shadows AWS IoT

Pesan yang disimpan dan Device Shadows menyimpan data dari perangkat, tetapi berperilaku berbeda dan melayani tujuan yang berbeda. Bagian ini menjelaskan persamaan dan perbedaan mereka.

	Pesan yang disimpan	Bayangan Perangkat
Payload pesan memiliki struktur atau skema yang telah ditentukan sebelumnya	Seperti yang didefinisikan oleh implementasi. MQTT tidak menentukan struktur atau skema untuk muatan pesannya.	AWS IoT mendukung struktur data tertentu.
Memperbarui payload pesan menghasilkan pesan acara	Menerbitkan pesan yang disimpan akan mengirimkan pesan ke klien berlangganan, tetapi tidak menghasilkan pesan pembaruan tambahan.	Memperbarui Device Shadow menghasilkan pesan pembaruan yang menjelaskan perubahan .
Pembaruan pesan diberi nomor	Pesan yang disimpan tidak diberi nomor secara otomatis.	Dokumen Device Shadow memiliki nomor versi otomatis dan stempel waktu.

	Pesan yang disimpan	Bayangan Perangkat
Muatan pesan dilampirkan ke sumber daya benda	Pesan yang disimpan tidak dilampirkan ke sumber daya sesuatu.	Device Shadows dilampirkan ke sumber daya benda.
Memperbarui elemen individual dari payload pesan	Elemen individual pesan tidak dapat diubah tanpa memperbarui seluruh payload pesan.	Elemen individual dari dokumen Device Shadow dapat diperbarui tanpa perlu memperbarui seluruh dokumen Device Shadow.
Klien menerima data pesan saat berlangganan	Klien secara otomatis menerima pesan yang disimpan setelah berlangganan topik dengan pesan yang disimpan.	Klien dapat berlangganan pembaruan Device Shadow, tetapi mereka harus meminta status saat ini dengan sengaja.
Pengindeksan dan kemampuan pencarian	Pesan yang disimpan tidak diindeks untuk pencarian.	Pengindeksan armada mengindeks data Device Shadow untuk pencarian dan agregasi.

Pesan MQTT Last Will and Testament (LWT)

Last Will and Testament (LWT) adalah fitur di MQTT. Dengan LWT, klien dapat menentukan pesan yang akan dipublikasikan broker ke topik yang ditentukan klien dan mengirimkannya ke semua klien yang berlangganan topik ketika pemutusan yang belum tahu terjadi. Pesan yang ditentukan klien disebut pesan LWT atau Pesan Will, dan topik yang ditentukan klien disebut sebagai Topik Will. Anda dapat menentukan pesan LWT saat perangkat terhubung ke broker. Pesan-pesan ini dapat dipertahankan dengan menyetel `Will Retain` bendera di `Connect Flag bits` bidang selama koneksi. Misalnya, jika `Will Retain` bendera disetel ke 1, Pesan Will akan disimpan di broker di Topik Will terkait. Untuk informasi selengkapnya, lihat [Akan Pesan](#).

Broker akan menyimpan Pesan Will sampai terjadi pemutusan yang belum diketahui. Ketika itu terjadi, broker akan mempublikasikan pesan ke semua klien yang berlangganan Topik Will untuk memberi tahu pemutusan. Jika klien terputus dari broker dengan pemutusan yang diprakarsai klien menggunakan pesan MQTT DISCONNECT, broker tidak akan mempublikasikan pesan LWT

yang disimpan. Dalam semua kasus lain, pesan LWT akan dikirim. [Untuk daftar lengkap skenario pemutusan saat broker akan mengirim pesan LWT, lihat peristiwa Hubungkan/Putuskan sambungan.](#)

Menggunakan ConnectAttributes

ConnectAttributes memungkinkan Anda untuk menentukan atribut apa yang ingin Anda gunakan dalam pesan connect Anda dalam kebijakan IAM Anda seperti PersistentConnect dan LastWill. Dengan ConnectAttributes, Anda dapat membuat kebijakan yang tidak memberikan akses perangkat ke fitur baru secara default, yang dapat membantu jika perangkat dikompromikan.

connectAttributes mendukung fitur-fitur berikut:

PersistentConnect

Gunakan PersistentConnect fitur ini untuk menyimpan semua langganan yang dibuat klien selama koneksi ketika koneksi antara klien dan broker terputus.

LastWill

Gunakan LastWill fitur ini untuk mempublikasikan pesan ke LastWillTopic saat klien tiba-tiba terputus.

Secara default, kebijakan Anda memiliki koneksi non-persisten dan tidak ada atribut yang diteruskan untuk koneksi ini. Anda harus menentukan koneksi persisten dalam kebijakan IAM Anda jika Anda ingin memilikinya.

Untuk ConnectAttributes contoh, lihat [Connect Policy Examples](#).

Fitur yang didukung MQTT 5

AWS IoT Core dukungan untuk MQTT 5 didasarkan pada [spesifikasi MQTT v5.0](#) dengan beberapa perbedaan seperti yang didokumentasikan dalam [the section called "AWS IoT perbedaan dari spesifikasi MQTT"](#)

AWS IoT Core mendukung fitur MQTT 5 berikut:

- [Langganan Bersama](#)
- [Mulai Bersih dan Kedaluwarsa Sesi](#)
- [Kode Alasan pada semua ACK](#)
- [Alias Topik](#)

- [Kedaluwarsa Pesan](#)
- [Fitur MQTT 5 lainnya](#)

Langganan Bersama

AWS IoT Core mendukung Langganan Bersama untuk MQTT 3 dan MQTT 5. Langganan Bersama memungkinkan beberapa klien untuk berbagi langganan ke suatu topik dan hanya satu klien yang akan menerima pesan yang dipublikasikan ke topik tersebut menggunakan distribusi acak. Langganan Bersama dapat secara efektif memuat pesan MQTT saldo di sejumlah pelanggan. Misalnya, Anda memiliki 1.000 perangkat yang menerbitkan topik yang sama, dan 10 aplikasi backend yang memproses pesan tersebut. Dalam hal ini, aplikasi backend dapat berlangganan topik yang sama dan masing-masing akan secara acak menerima pesan yang diterbitkan oleh perangkat ke topik bersama. Ini secara efektif “berbagi” beban pesan-pesan itu. Langganan Bersama juga memungkinkan ketahanan yang lebih baik. Ketika aplikasi backend terputus, broker mendistribusikan beban ke pelanggan yang tersisa dalam grup.

Untuk menggunakan Langganan Bersama, klien berlangganan [filter topik](#) Langganan Bersama sebagai berikut:

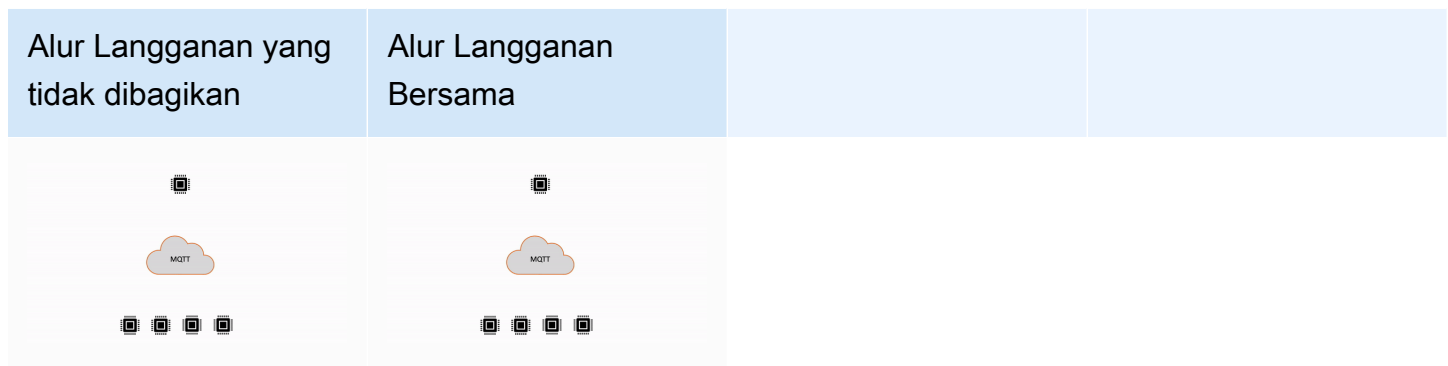
```
$share/{ShareName}/{TopicFilter}
```

- `$share` adalah string literal untuk menunjukkan filter topik Berlangganan Bersama, yang harus dimulai dengan `$share`.
- `{ShareName}` adalah string karakter untuk menentukan nama bersama yang digunakan oleh sekelompok pelanggan. Filter topik Langganan Bersama harus berisi `ShareName` dan diikuti oleh `/` karakter. Tidak `{ShareName}` boleh menyertakan karakter berikut: `/`, `+`, atau `#`. Ukuran maksimum untuk `{ShareName}` adalah 128 byte.
- `{TopicFilter}` mengikuti sintaks [filter topik](#) yang sama dengan Langganan Non-Shared. Ukuran maksimum untuk `{TopicFilter}` adalah 256 byte.
- Dua garis miring yang diperlukan (`/`) untuk `$share/{ShareName}/{TopicFilter}` tidak termasuk dalam [Jumlah garis miring maksimum dalam batas filter topik dan topik](#).

Langganan yang memiliki hal yang sama `{ShareName}/{TopicFilter}` termasuk dalam grup Berlangganan Bersama yang sama. Anda dapat membuat beberapa grup Langganan Bersama dan tidak melebihi [batas Langganan Bersama per grup](#). Untuk informasi selengkapnya, lihat [AWS IoT Core titik akhir dan kuota dari Referensi AWS Umum](#).

Tabel berikut membandingkan Langganan Non-Bersama dan Langganan Bersama:

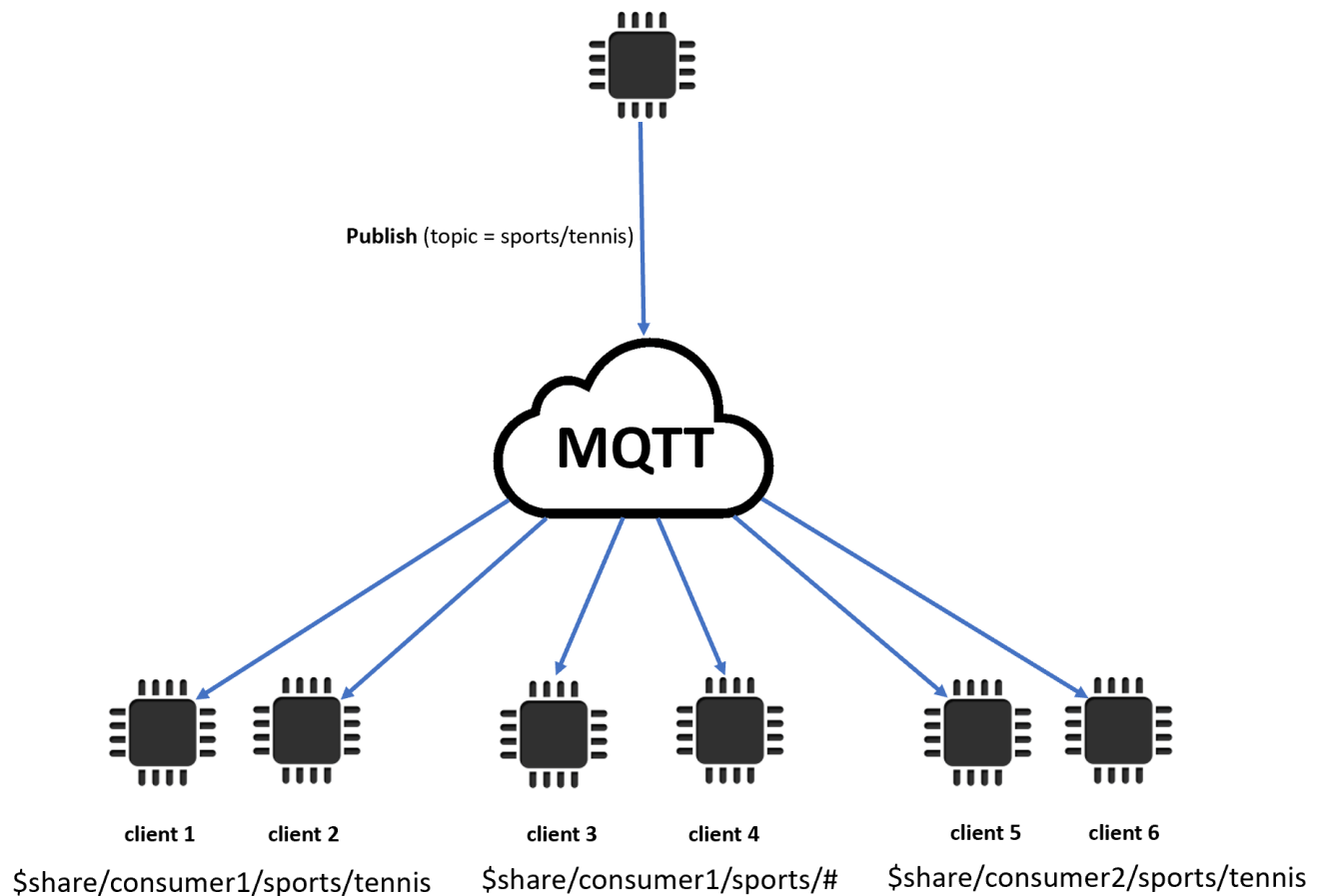
Langganan	Deskripsi	Contoh filter topik
Langganan yang tidak dibagikan	Setiap klien membuat langganan terpisah untuk menerima pesan yang dipublikasikan. Saat pesan dipublikasikan ke suatu topik, semua pelanggan topik tersebut menerima salinan pesan tersebut.	<code>sports/tennis</code> <code>sports/#</code>
Langganan Bersama	Beberapa klien dapat berbagi langganan ke topik dan hanya satu klien yang akan menerima pesan yang dipublikasikan ke topik itu pada distribusi acak.	<code>\$share/consumer/sports/tennis</code> <code>\$share/consumer/sports/#</code>



Catatan penting untuk menggunakan Langganan Bersama

- Ketika upaya publikasi ke pelanggan QoS0 gagal, tidak ada upaya coba lagi yang akan terjadi, dan pesan akan dihapus.
- Ketika upaya publikasi ke pelanggan QoS1 dengan sesi bersih gagal, pesan akan dikirim ke pelanggan lain dalam grup untuk beberapa upaya coba lagi. Pesan yang gagal dikirimkan setelah semua upaya coba lagi akan dibatalkan.
- Ketika upaya publikasi ke pelanggan QoS1 dengan [sesi persisten](#) gagal karena pelanggan sedang offline, pesan tidak akan antri dan akan dicoba ke pelanggan lain dalam grup. Pesan yang gagal dikirimkan setelah semua upaya coba lagi akan dibatalkan.
- Langganan Bersama tidak menerima pesan yang [disimpan](#).

- Jika Langganan Bersama berisi karakter wildcard (# atau +), mungkin ada beberapa Langganan Bersama yang cocok dengan topik. Jika itu terjadi, broker pesan menyalin pesan penerbitan dan mengirimkannya ke klien acak di setiap Langganan Bersama yang cocok. Perilaku wildcard Langganan Bersama dapat dijelaskan dalam diagram berikut.



Dalam contoh ini, ada tiga Langganan Bersama yang cocok dengan topik MQTT penerbitan. sports/tennis Broker pesan menyalin pesan yang diterbitkan dan mengirim pesan ke klien acak di setiap grup yang cocok.

Klien 1 dan klien 2 berbagi langganan: \$share/consumer1/sports/tennis

Klien 3 dan klien 4 berbagi langganan: \$share/consumer1/sports/#

Klien 5 dan klien 6 berbagi langganan: \$share/consumer2/sports/tennis

Untuk informasi selengkapnya tentang batas Langganan Bersama, lihat [AWS IoT Core titik akhir dan kuota](#) dari Referensi Umum.AWS [Untuk menguji Langganan Bersama menggunakan klien AWS IoT](#)

[MQTT di konsol, lihat.AWS IoT ???](#) Untuk informasi selengkapnya tentang Langganan Bersama, lihat [Langganan Bersama](#) dari spesifikasi MQTTv5.0.

Mulai Bersih dan Kedaluwarsa Sesi

Anda dapat menggunakan Clean Start dan Session Expiry untuk menangani sesi persisten Anda dengan lebih fleksibel. Bendera Mulai Bersih menunjukkan apakah sesi harus dimulai tanpa menggunakan sesi yang ada. Interval Kedaluwarsa Sesi menunjukkan berapa lama untuk mempertahankan sesi setelah pemutusan. Interval kedaluwarsa sesi dapat dimodifikasi pada saat pemutusan. Untuk informasi selengkapnya, lihat [the section called “Sesi persisten MQTT”](#).

Kode Alasan pada semua ACK

Anda dapat men-debug atau memproses pesan kesalahan dengan lebih mudah menggunakan kode alasan. Kode alasan dikembalikan oleh broker pesan berdasarkan jenis interaksi dengan broker (Berlangganan, Publikasikan, Akui). Untuk informasi lebih lanjut, lihat kode [alasan MQTT](#). Untuk daftar lengkap kode alasan MQTT, lihat Spesifikasi [MQTT](#) v5.

Alias Topik

Anda dapat mengganti nama topik dengan alias topik, yang merupakan bilangan bulat dua byte. Menggunakan alias topik dapat mengoptimalkan transmisi nama topik untuk berpotensi mengurangi biaya data pada layanan data terukur. AWS IoT Core memiliki batas default 8 alias topik. Untuk informasi selengkapnya, lihat [AWS IoT Core titik akhir dan kuota dari Referensi AWS](#) Umum.

Kedaluwarsa Pesan

Anda dapat menambahkan nilai kedaluwarsa pesan ke pesan yang dipublikasikan. Nilai-nilai ini mewakili interval kedaluwarsa pesan dalam hitungan detik. Jika pesan belum dikirim ke pelanggan dalam interval tersebut, pesan akan kedaluwarsa dan dihapus. Jika Anda tidak menyetel nilai kedaluwarsa pesan, pesan tidak akan kedaluwarsa.

Pada outbound, pelanggan akan menerima pesan dengan sisa waktu yang tersisa dalam interval kedaluwarsa. Misalnya, jika pesan publikasi masuk memiliki pesan kedaluwarsa 30 detik, dan dialihkan ke pelanggan setelah 20 detik, bidang kedaluwarsa pesan akan diperbarui menjadi 10. Dimungkinkan untuk pesan yang diterima oleh pelanggan untuk memiliki MEI yang diperbarui sebesar 0. Ini karena segera setelah waktu yang tersisa adalah 999 ms atau kurang, itu akan diperbarui ke 0.

Dalam AWS IoT Core, interval kedaluwarsa pesan minimum adalah 1. Jika interval diatur ke 0 dari sisi klien, itu akan disesuaikan ke 1. Interval kadaluwarsa pesan maksimum adalah 604800 (7 hari). Nilai apa pun yang lebih tinggi dari ini akan disesuaikan dengan nilai maksimum.

Dalam komunikasi lintas versi, perilaku kedaluwarsa pesan ditentukan oleh versi MQTT dari pesan publikasi masuk. Misalnya, pesan dengan kedaluwarsa pesan yang dikirim oleh sesi yang terhubung melalui MQTT5 dapat kedaluwarsa untuk perangkat yang berlangganan sesi MQTT3. Tabel di bawah ini mencantumkan cara kedaluwarsa pesan mendukung jenis pesan publikasi berikut:

Publikasikan Jenis Pesan	Interval Kedaluwarsa Pesan
Publikasikan Reguler	Jika server gagal mengirimkan pesan dalam waktu yang ditentukan, pesan yang kedaluwarsa akan dihapus dan pelanggan tidak akan menerimanya. Ini termasuk situasi seperti ketika perangkat tidak mempublikasikan pesan QoS 1 mereka.
Mempertahankan	Jika pesan yang disimpan kedaluwarsa dan klien baru berlangganan topik, klien tidak akan menerima pesan saat berlangganan.
Kehendak Terakhir	Interval untuk pesan terakhir akan dimulai setelah klien terputus dan server mencoba mengirimkan pesan wasiat terakhir kepada pelanggannya.
Pesan antrian	Jika QoS1 keluar dengan Interval Kedaluwarsa Pesan kedaluwarsa saat klien offline, setelah sesi persisten dilanjutkan, klien tidak akan menerima pesan kedaluwarsa.

Fitur MQTT 5 lainnya

Putuskan sambungan server

Ketika pemutusan terjadi, server dapat secara proaktif mengirim klien DISCONNECT untuk memberi tahu penutupan koneksi dengan kode alasan untuk pemutusan.

Permintaan/Tanggapan

Penayang dapat meminta tanggapan dikirim oleh penerima ke topik yang ditentukan penerbit pada saat penerimaan.

Ukuran Paket Maksimum

Klien dan Server dapat secara independen menentukan ukuran paket maksimum yang mereka dukung.

Format payload dan jenis konten

Anda dapat menentukan format payload (biner, teks) dan jenis konten saat pesan dipublikasikan. Ini diteruskan ke penerima pesan.

MQTT 5 properti

Properti MQTT 5 adalah tambahan penting untuk standar MQTT untuk mendukung fitur MQTT 5 baru seperti Session Expiry dan pola Permintaan/Respons. Di AWS IoT Core, Anda dapat membuat [aturan](#) yang dapat meneruskan properti dalam pesan keluar, atau menggunakan [HTTP Publish untuk mempublikasikan](#) pesan MQTT dengan beberapa properti baru.

Tabel berikut mencantumkan semua properti MQTT 5 yang mendukung. AWS IoT Core

Properti	Deskripsi	Jenis masuk	Paket
Indikator Format Muatan	Nilai boolean yang menunjukkan apakah payload diformat sebagai UTF-8.	Byte	TERBITKAN, SAMBUNGAN
Jenis Konten	String UTF-8 yang menggambarkan isi muatan.	Tali UTF-8	TERBITKAN, SAMBUNGAN
Topik Respon	String UTF-8 yang menjelaskan topik yang harus dipublikasikan oleh penerima sebagai bagian dari alur permintaan-respons. Topik tidak boleh memiliki karakter wildcard.	Tali UTF-8	TERBITKAN, SAMBUNGAN
Data Korelasi	Data biner yang digunakan oleh pengirim pesan permintaan untuk mengidentifikasi permintaan pesan respons.	Biner	TERBITKAN, SAMBUNGAN
Properti Pengguna	Pasangan string UTF-8. Properti ini dapat muncul beberapa kali dalam satu paket. Penerima akan menerima pasangan kunci-	Pasangan string UTF-8	HUBUNGAN, PUBLIKASIKAN, Akan Properti, BERLANGGANAN, PUTUSKAN

Properti	Deskripsi	Jenis masuk	Paket
	nilai dalam urutan yang sama dengan yang dikirim.		SAMBUNGAN, BERHENTI BERLANGGANAN
Interval Kedaluwarsa Pesan	Integer 4-byte yang mewakili interval kedaluwarsa pesan dalam hitungan detik. Jika tidak ada, pesan tidak kedaluwarsa.	Bilangan bulat 4-byte	TERBITKAN, SAMBUNGAN
Interval Kedaluwarsa Sesi	Integer 4-byte yang mewakili interval kedaluwarsa sesi dalam hitungan detik. AWS IoT Core mendukung maksimal 7 hari, dengan default maksimal satu jam. Jika nilai yang Anda tetapkan melebihi maksimum akun Anda, AWS IoT Core akan mengembalikan nilai yang disesuaikan di CONNACK.	Bilangan bulat 4-byte	SAMBUNGAN, SAMBUNGAN, PUTUSKAN SAMBUNGAN
Pengenal Klien yang Ditugaskan	ID klien acak yang dihasilkan AWS IoT Core ketika ID klien tidak ditentukan oleh perangkat. ID klien acak harus merupakan pengidentifikasi klien baru yang tidak digunakan oleh sesi lain yang saat ini dikelola oleh broker.	Tali UTF-8	CONNACK
Server Tetap Hidup	Sebuah integer 2-byte yang mewakili keep alive time yang ditetapkan oleh server. Server akan memutuskan koneksi klien jika klien tidak aktif selama lebih dari waktu tetap hidup.	Bilangan bulat 2 byte	CONNACK
Minta Informasi Masalah	Nilai boolean yang menunjukkan apakah Reason String atau User Properties dikirim dalam kasus kegagalan.	Byte	MENGHUBUNG

Properti	Deskripsi	Jenis masuk	Paket
Menerima Maksimum	Bilangan bulat 2-byte yang mewakili jumlah maksimum paket PUBLISH QOS > 0 yang dapat dikirim tanpa menerima PUBACK.	Bilangan bulat 2 byte	SAMBUNGGAN, CONNACK
Topik Alias Maksimum	Nilai ini menunjukkan nilai tertinggi yang akan diterima sebagai Alias Topik. Default-nya adalah 0.	Bilangan bulat 2 byte	SAMBUNGGAN, CONNACK
QoS Maksimum	Nilai maksimum QoS yang AWS IoT Core mendukung. Defaultnya adalah 1. AWS IoT Core tidak mendukung QoS2.	Byte	CONNACK
Pertahankan Tersedia	Nilai boolean yang menunjukkan apakah broker AWS IoT Core pesan mendukung pesan yang disimpan. Default-nya adalah 1.	Byte	CONNACK
Ukuran Paket Maksimum	Ukuran paket maksimum yang AWS IoT Core menerima dan mengirim. Tidak dapat melebihi 128KB.	Bilangan bulat 4-byte	SAMBUNGGAN, CONNACK
Berlangganan Wildcard Tersedia	Nilai boolean yang menunjukkan apakah broker AWS IoT Core pesan mendukung Langganan Wildcard Tersedia. Default-nya adalah 1.	Byte	CONNACK
Pengenal Berlangganan Tersedia	Nilai boolean yang menunjukkan apakah broker AWS IoT Core pesan mendukung Pengenal Langganan Tersedia. Default-nya adalah 0.	Byte	CONNACK

Kode alasan MQTT

MQTT 5 memperkenalkan pelaporan kesalahan yang ditingkatkan dengan respons kode alasan. AWS IoT Core dapat mengembalikan kode alasan termasuk tetapi tidak terbatas pada berikut dikelompokkan berdasarkan paket. Untuk daftar lengkap kode alasan yang didukung oleh MQTT 5, lihat spesifikasi [MQTT 5](#).

Kode Alasan CONNACK

Nilai	Hex	Nama Kode Alasan	Deskripsi
0	0x00	Berhasil	Koneksi diterima.
128	0x80	Kesalahan yang tidak ditentukan	Server tidak ingin mengungkapkan alasan kegagalan, atau tidak ada kode alasan lain yang berlaku.
133	0x85	Client Identifier tidak valid	Pengidentifikasi klien adalah string yang valid tetapi tidak diizinkan oleh server.
134	0x86	Nama Pengguna atau Kata Sandi Buruk	Server tidak menerima nama pengguna atau kata sandi yang ditentukan oleh klien.
135	0x87	Tidak diotorisasi	Klien tidak berwenang untuk terhubung.
144	0x90	Nama Topik tidak valid	Nama Topik Will dibentuk dengan benar tetapi tidak diterima oleh server.
151	0x97	Kuota terlampaui	Batas implementasi atau administrasi yang diberlakukan telah terlampaui.
155	0x9B	QoS tidak didukung	Server tidak mendukung QoS yang ditetapkan di Will QoS.

Kode Alasan PUBACK

Nilai	Hex	Nama Kode Alasan	Deskripsi
0	0x00	Berhasil	Pesan diterima. Publikasi pesan QoS 1 berlangsung.
128	0x80	Kesalahan yang tidak ditentukan	Penerima tidak menerima publikasi, tetapi tidak ingin mengungkapkan alasannya, atau tidak cocok dengan salah satu nilai lainnya.
135	0x87	Tidak diotorisasi	PUBLISH tidak diotorisasi.
144	0x90	Nama Topik tidak valid	Nama topik tidak cacat, tetapi tidak diterima oleh klien atau server.
145	0x91	Pengenal paket yang digunakan	Packet identifier sudah digunakan. Ini mungkin menunjukkan ketidakcocokan dalam keadaan sesi antara klien dan server.
151	0x97	Kuota terlampaui	Batas implementasi atau administrasi yang diberlakukan telah terlampaui.

PUTUSKAN Kode Alasan

Nilai	Hex	Nama Kode Alasan	Deskripsi
129	0x81	Paket cacat	Paket yang diterima tidak sesuai dengan spesifikasi ini.
130	0x82	Kesalahan Protokol	Paket yang tidak terduga atau rusak diterima.
135	0x87	Tidak diotorisasi	Permintaan tidak diizinkan.
139	0x8B	Server dimatikan	Server dimatikan.

Nilai	Hex	Nama Kode Alasan	Deskripsi
141	0x8D	Keep Alive timeout	Koneksi ditutup karena tidak ada paket yang diterima selama 1,5 kali waktu Keep Alive.
142	0x8E	Sesi diambil alih	Koneksi lain menggunakan clientID yang sama telah terhubung, menyebabkan koneksi ini ditutup.
143	0x8F	Filter Topik tidak valid	Filter topik dibentuk dengan benar tetapi tidak diterima oleh server.
144	0x90	Nama Topik tidak valid	Nama topik dibentuk dengan benar tetapi tidak diterima oleh klien atau server ini.
147	0x93	Menerima maksimum terlampaui	Klien atau server telah menerima lebih dari publikasi Terima Maksimum yang belum dikirim PUBACK atau PUBCOMP.
148	0x94	Topik Alias tidak valid	Klien atau server telah menerima paket PUBLISH yang berisi alias topik yang lebih besar dari Alias Topik Maksimum yang dikirim dalam paket CONNECT atau CONNACK.
151	0x97	Kuota terlampaui	Batas implementasi atau administrasi yang diberlakukan telah terlampaui.
152	0x98	Tindakan administratif	Koneksi ditutup karena tindakan administratif.
155	0x9B	QoS tidak didukung	Klien menentukan QoS lebih besar dari QoS yang ditentukan dalam QoS Maksimum di CONNACK.
161	0xA1	Pengidentifikasi Langganan tidak didukung	Server tidak mendukung pengidentifikasi langganan; langganan tidak diterima.

Kode Alasan SUBACK

Nilai	Hex	Nama Kode Alasan	Deskripsi
0	0x00	Diberikan QoS 0	Langganan diterima dan QoS maksimum yang dikirim adalah QoS 0. Ini mungkin QoS yang lebih rendah dari yang diminta.
1	0x01	Diberikan QoS 1	Langganan diterima dan QoS maksimum yang dikirim adalah QoS 1. Ini mungkin QoS yang lebih rendah dari yang diminta.
128	0x80	Kesalahan yang tidak ditentukan	Langganan tidak diterima dan Server tidak ingin mengungkapkan alasan atau tidak ada Kode Alasan lainnya yang berlaku.
135	0x87	Tidak diotorisasi	Klien tidak berwenang untuk membuat langganan ini.
143	0x8F	Filter Topik tidak valid	Filter Topik dibentuk dengan benar tetapi tidak diizinkan untuk Klien ini.
145	0x91	Packet Identifier sedang digunakan	Packet Identifier yang ditentukan sudah digunakan.
151	0x97	Kuota terlampaui	Batas implementasi atau administrasi yang diberlakukan telah terlampaui.

Kode Alasan UNSUBACK

Nilai	Hex	Nama Kode Alasan	Deskripsi
0	0x00	Berhasil	Langganan dihapus.
128	0x80	Kesalahan yang tidak ditentukan	Berhenti berlangganan tidak dapat diselesaikan dan Server tidak ingin mengungkapkan alasan atau tidak ada Kode Alasan lainnya yang berlaku.

Nilai	Hex	Nama Kode Alasan	Deskripsi
143	0x8F	Filter Topik tidak valid	Filter Topik dibentuk dengan benar tetapi tidak diizinkan untuk Klien ini.
145	0x91	Packet Identifier sedang digunakan	Packet Identifier yang ditentukan sudah digunakan.

AWS IoT perbedaan dari spesifikasi MQTT

Implementasi broker pesan didasarkan pada spesifikasi [MQTT v3.1.1 dan spesifikasi MQTT v5.0](#), [tetapi berbeda dari spesifikasi](#) dengan cara ini:

- AWS IoT tidak mendukung paket berikut untuk MQTT 3: PUBREC, PUBREL, dan PUBCOMP.
- AWS IoT tidak mendukung paket berikut untuk MQTT 5: PUBREC, PUBREL, PUBCOMP, dan AUTH.
- AWS IoT tidak mendukung pengalihan server MQTT 5.
- AWS IoT mendukung kualitas layanan MQTT (QoS) level 0 dan 1 saja. AWS IoT tidak mendukung penerbitan atau berlangganan dengan QoS level 2. Ketika QoS level 2 diminta, broker pesan tidak mengirim PUBACK atau SUBACK.
- Di AWS IoT, berlangganan topik dengan QoS level 0 berarti pesan dikirim nol kali atau lebih. Sebuah pesan dapat disampaikan lebih dari satu kali. Pesan yang dikirim lebih dari satu kali dapat dikirim dengan ID paket yang berbeda. Dalam kasus ini, bendera DUP tidak disetel.
- Saat menanggapi permintaan koneksi, broker pesan mengirim pesan CONNACK. Pesan ini berisi bendera untuk menunjukkan apakah koneksi melanjutkan sesi sebelumnya.
- Sebelum mengirim paket kontrol tambahan atau permintaan pemutusan sambungan, klien harus menunggu pesan CONNACK diterima di perangkat mereka dari broker pesan. AWS IoT
- Ketika klien berlangganan topik, mungkin ada penundaan antara waktu broker pesan mengirim SUBACK dan waktu klien mulai menerima pesan baru yang cocok.
- Saat klien menggunakan karakter wildcard # dalam filter topik untuk berlangganan topik, semua string di dan di bawah levelnya dalam hierarki topik akan dicocokkan. Namun, topik induk tidak cocok. Misalnya, langganan topik sensor/# menerima pesan yang dipublikasikan ke

topiksensor/, sensor/temperaturesensor/temperature/room1, tetapi bukan pesan yang dipublikasikan kesensor. Untuk informasi selengkapnya tentang wildcard, lihat [Filter topik](#).

- Broker pesan menggunakan ID klien untuk mengidentifikasi setiap klien. ID klien diteruskan dari klien ke broker pesan sebagai bagian dari muatan MQTT. Dua klien dengan ID klien yang sama tidak dapat dihubungkan secara bersamaan ke broker pesan. Ketika klien terhubung ke broker pesan menggunakan ID klien yang digunakan klien lain, koneksi klien baru diterima dan klien yang terhubung sebelumnya terputus.
- Pada kesempatan yang jarang terjadi, broker pesan mungkin mengirim ulang pesan PUBLISH logis yang sama dengan ID paket yang berbeda.
- Berlangganan filter topik yang berisi karakter wildcard tidak dapat menerima pesan yang disimpan. Untuk menerima pesan yang disimpan, permintaan berlangganan harus berisi filter topik yang sama persis dengan topik pesan yang disimpan.
- Broker pesan tidak menjamin urutan pesan dan ACK diterima.
- AWS IoT mungkin memiliki batasan yang berbeda dari spesifikasi. Untuk informasi lebih lanjut, lihat [broker AWS IoT Core pesan dan batas protokol dan kuota](#) dari Panduan AWS IoT Referensi.
- Bendera DUP MQTT tidak didukung.

HTTPS

Klien dapat mempublikasikan pesan dengan membuat permintaan ke REST API menggunakan protokol HTTP 1.0 atau 1.1. Untuk otentikasi dan pemetaan port yang digunakan oleh permintaan HTTP, lihat [Protokol, pemetaan port, dan otentikasi](#)

Note

HTTPS tidak mendukung `clientId` nilai seperti yang dilakukan MQTT. `clientId` tersedia saat menggunakan MQTT, tetapi tidak tersedia saat menggunakan HTTPS.

URL pesan HTTPS

Perangkat dan klien mempublikasikan pesan mereka dengan membuat permintaan POST ke titik akhir khusus klien dan URL khusus topik:

```
https://IoT_data_endpoint/topics/url_encoded_topic_name?qos=1
```

- *IOT_DATA_ENDPOINT* adalah titik akhir [data perangkat.AWS IoT](#) Anda dapat menemukan titik akhir di AWS IoT konsol di halaman detail benda atau di klien dengan menggunakan AWS CLI perintah:

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

Titik akhir akan terlihat seperti ini: a3qjEXAMPLEffp-ats.iot.us-west-2.amazonaws.com

- *url_encoded_topic_name* adalah nama [topik lengkap dari pesan yang dikirim](#).

Contoh kode pesan HTTPS

Ini adalah beberapa contoh cara mengirim pesan HTTPS ke 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

Anda dapat menggunakan [curl](#) dari klien atau perangkat untuk mengirim pesan ke AWS IoT.

Untuk menggunakan curl untuk mengirim pesan dari perangkat AWS IoT klien

1. Periksa curl versinya.

- a. Pada klien Anda, jalankan perintah ini pada command prompt.

```
curl --help
```

Dalam teks bantuan, cari opsi TLS. Anda harus melihat `--tlsv1.2` opsi.

- b. Jika Anda melihat `--tlsv1.2` opsi, lanjutkan.
- c. Jika Anda tidak melihat `--tlsv1.2` opsi atau Anda mendapatkan command `not found` kesalahan, Anda mungkin perlu memperbarui atau menginstal curl pada klien Anda atau menginstal `openssl` sebelum melanjutkan.

2. Instal sertifikat pada klien Anda.

Salin file sertifikat yang Anda buat saat Anda mendaftarkan klien Anda (benda) di AWS IoT konsol. Pastikan Anda memiliki tiga file sertifikat ini di klien Anda sebelum melanjutkan.

- File sertifikat CA (*Amazon-Root-CA-1.PEM* dalam contoh ini).
- File sertifikat klien (*device.pem.crt* dalam contoh ini).
- File kunci pribadi klien (*private.pem.key* dalam contoh ini).

3. Buat baris curl perintah, ganti nilai yang dapat diganti untuk akun dan sistem Anda.

```
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`

Gunakan TLS 1.2 (SSL).


```
--cacert Amazon-akar-CA-1.PEM
```

Nama file dan jalur, jika perlu, dari sertifikat CA untuk memverifikasi rekan.

```
--perangkat cert.pem.crt
```

Nama file sertifikat klien dan jalur, jika perlu.

```
--key private.pem.key
```

Nama file kunci pribadi klien dan jalur, jika perlu.

```
--permintaan POST
```

Jenis permintaan HTTP (dalam hal ini, POST).

```
--data "{\"message\": \"Halo, dunia\"}"
```

Data HTTP POST yang ingin Anda publikasikan. Dalam hal ini, ini adalah string JSON, dengan tanda kutip internal lolos dengan karakter garis miring terbalik (\).

```
"https://iot_data_endpoint:8443/topik/ topik? qos=1"
```

URL titik akhir data AWS IoT perangkat klien Anda, diikuti oleh port HTTPS:8443, yang kemudian diikuti oleh kata kunci, /topics/ dan nama topik *topik*, dalam hal ini. Tentukan Kualitas Layanan sebagai parameter kueri, ?qos=1.

4. Buka klien pengujian MQTT di konsol. AWS IoT

Ikuti petunjuk [Lihat pesan MQTT dengan klien MQTT AWS IoT](#) dan konfigurasi konsol untuk berlangganan pesan dengan nama topik *topik* yang digunakan dalam curl perintah Anda, atau gunakan filter topik wildcard. #

5. Uji perintahnya.

Saat memantau topik di klien pengujian AWS IoT konsol, buka klien Anda dan keluarkan baris perintah curl yang Anda buat di langkah 3. Anda akan melihat pesan klien Anda di konsol.

Topik MQTT

Topik MQTT mengidentifikasi pesan. AWS IoT AWS IoT klien mengidentifikasi pesan yang mereka terbitkan dengan memberikan nama topik pesan. Klien mengidentifikasi pesan yang ingin mereka

berlangganan (terima) dengan mendaftarkan filter topik AWS IoT Core. Broker pesan menggunakan nama topik dan filter topik untuk merutekan pesan dari klien penerbitan ke klien berlangganan.

Broker pesan menggunakan topik untuk mengidentifikasi pesan yang dikirim menggunakan MQTT dan dikirim menggunakan HTTP ke. [URL pesan HTTPS](#)

Sementara AWS IoT mendukung beberapa [topik sistem cadangan](#), sebagian besar topik MQTT dibuat dan dikelola oleh Anda, perancang sistem. AWS IoT menggunakan topik untuk mengidentifikasi pesan yang diterima dari klien penerbitan dan memilih pesan untuk dikirim ke klien berlangganan, seperti yang dijelaskan di bagian berikut. Sebelum Anda membuat namespace topik untuk sistem Anda, tinjau karakteristik topik MQTT untuk membuat hierarki nama topik yang paling sesuai untuk sistem IoT Anda.

Nama topik

Nama topik dan filter topik adalah string yang dikodekan UTF-8. Mereka dapat mewakili hierarki informasi dengan menggunakan karakter garis miring maju (/) untuk memisahkan tingkat hierarki. Misalnya, nama topik ini bisa merujuk ke sensor suhu di ruangan 1:

- `sensor/temperature/room1`

Dalam contoh ini, mungkin juga ada jenis sensor lain di ruangan lain dengan nama topik seperti:

- `sensor/temperature/room2`
- `sensor/humidity/room1`
- `sensor/humidity/room2`

Note

Saat Anda mempertimbangkan nama topik untuk pesan di sistem Anda, ingatlah:

- Nama topik dan filter topik peka huruf besar/kecil.
- Nama topik tidak boleh berisi informasi yang dapat diidentifikasi secara pribadi.
- Nama topik yang dimulai dengan \$ adalah [topik yang dicadangkan](#) untuk digunakan hanya oleh AWS IoT Core.
- AWS IoT Core tidak dapat mengirim atau menerima pesan antara Akun AWS s atau Wilayah.

Untuk informasi selengkapnya tentang mendesain nama topik dan namespace Anda, lihat whitepaper kami, [Merancang Topik MQTT](#) untuk AWS IoT Core

Untuk contoh bagaimana aplikasi dapat mempublikasikan dan berlangganan pesan, mulailah dengan [Memulai dengan AWS IoT Core](#) dan [AWS IoT SDK Perangkat, SDK Seluler, dan AWS IoT Klien Perangkat](#).

Important

Namespace topik terbatas pada Akun AWS dan Wilayah. Misalnya, `sensor/temp/room1` topik yang digunakan oleh Akun AWS dalam satu Wilayah berbeda dari `sensor/temp/room1` topik yang digunakan oleh AWS akun yang sama di Wilayah lain atau digunakan oleh orang lain Akun AWS di Wilayah mana pun.

Topik ARN

Semua topik ARN (Nama Sumber Daya Amazon) memiliki formulir berikut:

```
arn:aws:iot:aws-region:AWS-account-ID:topic/Topic
```

Misalnya, `arn:aws:iot:us-west-2:123EXAMPLE456:topic/application/topic/device/sensor` adalah ARN untuk topik tersebut. `application/topic/device/sensor`

Filter topik

Klien berlangganan mendaftarkan filter topik dengan broker pesan untuk menentukan topik pesan yang harus dikirim oleh broker pesan kepada mereka. Filter topik dapat berupa nama topik tunggal untuk berlangganan satu nama topik atau dapat menyertakan karakter wildcard untuk berlangganan beberapa nama topik secara bersamaan.

Klien penerbitan tidak dapat menggunakan karakter wildcard dalam nama topik yang mereka terbitkan.

Tabel berikut mencantumkan karakter wildcard yang dapat digunakan dalam filter topik.

Wildcard topik

Karakter wildcard	Pertandingan	Catatan
#	Semua string di dan di bawah levelnya dalam hierarki topik.	<p>Harus menjadi karakter terakhir dalam filter topik.</p> <p>Harus menjadi satu-satunya karakter dalam tingkat hierarki topiknya.</p> <p>Dapat digunakan dalam filter topik yang juga berisi karakter wildcard +.</p>
+	String apa pun di level yang berisi karakter.	<p>Harus menjadi satu-satunya karakter dalam tingkat hierarki topiknya.</p> <p>Dapat digunakan di berbagai tingkatan filter topik.</p>

Menggunakan wildcard dengan contoh nama topik sensor sebelumnya:

- Langganan untuk `sensor/#` menerima pesan yang dipublikasikan ke `sensor/`, `sensor/temperaturesensor/temperature/room1`, tetapi bukan pesan yang dipublikasikan ke `sensor`.
- Langganan untuk `sensor/+/room1` menerima pesan yang dipublikasikan ke `sensor/temperature/room1` dan `sensor/humidity/room1`, tetapi bukan pesan yang dikirim ke `sensor/temperature/room2` atau `sensor/humidity/room2`.

Filter topik ARN

Semua filter topik ARN (Nama Sumber Daya Amazon) memiliki formulir berikut:

```
arn:aws:iot:aws-region:AWS-account-ID:topicfilter/TopicFilter
```

Misalnya, `arn:aws:iot:us-west-2:123EXAMPLE456:topicfilter/application/topic/+ /sensor` adalah ARN untuk filter topik. `application/topic/+ /sensor`

Muatan pesan MQTT

Payload pesan yang dikirim dalam pesan MQTT Anda tidak ditentukan oleh AWS IoT, kecuali untuk salah satu pesan. [the section called “Topik yang dipesan” Untuk mengakomodasi kebutuhan aplikasi Anda, kami sarankan Anda menentukan payload pesan untuk topik Anda dalam batasan Service Quotas AWS IoT Core for Protocols.](#)

Menggunakan format JSON untuk payload pesan Anda memungkinkan mesin AWS IoT aturan untuk mengurai pesan Anda dan menerapkan kueri SQL ke dalamnya. Jika aplikasi Anda tidak memerlukan mesin aturan untuk menerapkan kueri SQL ke muatan pesan, Anda dapat menggunakan format data apa pun yang diperlukan aplikasi Anda. Untuk informasi tentang batasan dan karakter yang dicadangkan dalam dokumen JSON yang digunakan dalam kueri SQL, lihat. [Ekstensi JSON](#)

Untuk informasi selengkapnya tentang mendesain topik MQTT dan muatan pesan yang sesuai, lihat [Merancang](#) Topik MQTT untuk. AWS IoT Core

Jika batas ukuran pesan melebihi kuota layanan, itu akan menghasilkan alasan `CLIENT_ERROR` dengan `PAYLOAD_LIMIT_EXCEEDED` dan “Payload pesan melebihi batas ukuran untuk jenis pesan.” Untuk informasi selengkapnya tentang batas ukuran [AWS IoT Core pesan, lihat batas dan kuota broker pesan.](#)

Topik yang dipesan

Topik yang dimulai dengan tanda dolar (\$) dicadangkan untuk digunakan oleh AWS IoT. Anda dapat berlangganan dan mempublikasikan topik yang dipesan ini jika memungkinkan; Namun, Anda tidak dapat membuat topik baru yang dimulai dengan tanda dolar. Operasi publikasi atau berlangganan yang tidak didukung ke topik yang dicadangkan dapat mengakibatkan koneksi terputus.

Topik model aset

Topik	Operasi klien diizinkan	Deskripsi
<code><i>\$aws/sitewise/asset-models//asset/assetID /properti</i></code>	Langganan	AWS IoT SiteWise menerbitkan pemberitahuan properti aset untuk topik ini.

Topik	Operasi klien diizinkan	Deskripsi
<code>es/propertyId assetModelId</code>		Untuk informasi selengkapnya, lihat Berinteraksi dengan AWS layanan lain di Panduan AWS IoT SiteWise Pengguna.

AWS IoT Device Defender topik

Pesan-pesan ini mendukung buffer respons dalam format Concise Binary Object Representation (CBOR) dan JavaScript Object Notation (JSON), tergantung pada format payload topik. AWS IoT Device Defender topik hanya mendukung publikasi MQTT.

format muatan	Jenis data format respons
cbor	Representasi Objek Biner Ringkas (CBOR)
json	JavaScript Notasi Objek (JSON)

Untuk informasi selengkapnya, lihat [Mengirim metrik dari perangkat](#).

Topik	Operasi yang diizinkan	Deskripsi
<code>\$aws/things/thingName/defender/metrik/format muatan</code>	Publikasikan	AWS IoT Device Defender agen mempublikasikan metrik untuk topik ini. Untuk informasi selengkapnya, lihat Mengirim metrik dari perangkat .
<code>\$aws/things/thingName/defender/metrik/format payload/diterima</code>	Langganan	AWS IoT <i>menerbitkan ke topik ini setelah AWS IoT Device Defender agen menerbitkan pesan yang berhasil ke \$aws/things/ThingName/defender/metrics/payload-format.</i> Untuk

Topik	Operasi yang diizinkan	Deskripsi
		informasi selengkapnya, lihat Mengirim metrik dari perangkat .
<code>\$aws/things/ThingName/defender/metrik/formatpayload/ditolak</code>	Langganan	AWS IoT <i>menerbitkan ke topik ini setelah AWS IoT Device Defender agen menerbitkan pesan yang gagal ke \$aws/things/ThingName/defender/metrics/payload-format</i> . Untuk informasi selengkapnya, lihat Mengirim metrik dari perangkat .

AWS IoT Core Topik Lokasi Perangkat

AWS IoT Core Lokasi Perangkat dapat menyelesaikan data pengukuran dari perangkat Anda dan memberikan perkiraan lokasi perangkat IoT Anda. Data pengukuran dari perangkat dapat mencakup GNSS, Wi-Fi, seluler, dan alamat IP. AWS IoT Core Lokasi Perangkat kemudian memilih jenis pengukuran yang memberikan akurasi terbaik dan memecahkan informasi lokasi perangkat. Untuk informasi selengkapnya, lihat [AWS IoT Core Lokasi Perangkat](#) dan [Menyelesaikan lokasi perangkat menggunakan topik AWS IoT Core MQTT Lokasi Perangkat](#).

Topik	Operasi yang diizinkan	Deskripsi
<code>\$aws/device_location/customer_device_id/get_position_estimate</code>	Publikasikan	Perangkat memublikasikan topik ini agar data pengukuran mentah yang dipindai diselesaikan oleh Lokasi AWS IoT Core Perangkat.
<code>\$aws/device_location/customer_device_id/get_position_estimate/diterima</code>	Langganan	AWS IoT Core Lokasi Perangkat memublikasikan topik ini setelah berhasil menyelesaikan lokasi perangkat.

Topik	Operasi yang diizinkan	Deskripsi
<code>\$aws/device_location/customer_device_id/get_position_estimate/ditolak</code>	Langganan	AWS IoT Core Lokasi Perangkat memublikasikan topik ini ketika tidak dapat menyelesaikan lokasi perangkat dengan sukses karena kesalahan 4xx.

Topik acara

Note

Untuk informasi selengkapnya tentang topik MQTT yang dipesan untuk acara LoRa WAN, lihat Acara status [koneksi](#).

Topik	Operasi klien diizinkan	Deskripsi
<code>\$ aws/acara/sertifikat/terdaftar/caCertificateId</code>	Langganan	AWS IoT menerbitkan pesan ini ketika AWS IoT secara otomatis mendaftarkan sertifikat dan ketika klien menyajikan sertifikat dengan status. PENDING_ACTIVATION Untuk informasi selengkapnya, lihat the section called “Konfigurasi koneksi pertama oleh klien untuk pendaftaran otomatis” .
<code>\$ aws/acara/pekerjaan/jobBid/dibatalkan</code>	Langganan	AWS IoT memublikasikan pesan ini saat pekerjaan dibatalkan. Untuk informasi selengkapnya, lihat Acara Lowongan .
<code>\$aws/acara/pekerjaan/jobBid/cancellation_in_progress</code>	Langganan	AWS IoT memublikasikan pesan ini saat pembatalan pekerjaan sedang berlangsung

Topik	Operasi klien diizinkan	Deskripsi
		ng. Untuk informasi selengkapnya, lihat Acara Lowongan .
<code>\$ aws/acara/pekerjaan/jobId/ selesai</code>	Langganan	AWS IoT mempublikasikan pesan ini ketika pekerjaan telah selesai. Untuk informasi selengkapnya, lihat Acara Lowongan .
<code>\$ aws/acara/pekerjaan/jobId /dihapus</code>	Langganan	AWS IoT memublikasikan pesan ini saat pekerjaan dihapus. Untuk informasi selengkapnya, lihat Acara Lowongan .
<code>\$aws/acara/pekerjaan/jobId /deletion_in_progress</code>	Langganan	AWS IoT memublikasikan pesan ini saat penghapusan pekerjaan sedang berlangsung. Untuk informasi selengkapnya, lihat Acara Lowongan .
<code>\$AWS/acara/JobExecution/JobID/dibatalkan</code>	Langganan	AWS IoT memublikasikan pesan ini saat eksekusi pekerjaan dibatalkan. Untuk informasi selengkapnya, lihat Acara Lowongan .
<code>\$ AWS/acara/JobExecution/jobID/dihapus</code>	Langganan	AWS IoT memublikasikan pesan ini saat eksekusi pekerjaan dihapus. Untuk informasi selengkapnya, lihat Acara Lowongan .
<code>\$ AWS/acara/JobExecution/jobID/gagal</code>	Langganan	AWS IoT memublikasikan pesan ini ketika eksekusi pekerjaan gagal. Untuk informasi selengkapnya, lihat Acara Lowongan .
<code>\$AWS/acara/JobExecution/JobID/ditolak</code>	Langganan	AWS IoT menerbitkan pesan ini ketika eksekusi pekerjaan ditolak. Untuk informasi selengkapnya, lihat Acara Lowongan .

Topik	Operasi klien diizinkan	Deskripsi
<code>\$ AWS/acara/JobExecution/JobID/dihapus</code>	Langganan	AWS IoT memublikasikan pesan ini ketika eksekusi pekerjaan telah dihapus. Untuk informasi selengkapnya, lihat Acara Lowongan .
<code>\$AWS/acara/JobExecution/JobID/berhasil</code>	Langganan	AWS IoT menerbitkan pesan ini ketika eksekusi pekerjaan berhasil. Untuk informasi selengkapnya, lihat Acara Lowongan .
<code>\$AWS/acara/jobexecution/jobID/timed_out</code>	Langganan	AWS IoT memublikasikan pesan ini saat eksekusi pekerjaan habis. Untuk informasi selengkapnya, lihat Acara Lowongan .
<code>\$ aws/acara/kehadiran/terhubung/clientID</code>	Langganan	AWS IoT menerbitkan ke topik ini ketika klien MQTT dengan ID klien yang ditentukan terhubung ke. AWS IoT Untuk informasi selengkapnya, lihat Hubungkan/ Putuskan acara .
<code>\$ aws/acara/kehadiran/terputus/clientID</code>	Langganan	AWS IoT menerbitkan ke topik ini ketika klien MQTT dengan ID klien yang ditentukan terputus. AWS IoT Untuk informasi selengkapnya, lihat Hubungkan/ Putuskan acara .
<code>\$ aws/acara/langganan/berlangganan/clientID</code>	Langganan	AWS IoT menerbitkan topik ini ketika klien MQTT dengan ID klien yang ditentukan berlangganan topik MQTT. Untuk informasi selengkapnya, lihat Acara Berlangganan/Berhenti Berlangganan .

Topik	Operasi klien diizinkan	Deskripsi
<code>\$ aws/acara/langganan/berhenti berlangganan/clientID</code>	Langganan	AWS IoT menerbitkan topik ini ketika klien MQTT dengan ID klien yang ditentukan berhenti berlangganan topik MQTT. Untuk informasi selengkapnya, lihat Acara Berlangganan/Berhenti Berlangganan .
<code>\$aws/events/thing/ThingName /dibuat</code>	Langganan	AWS IoT menerbitkan ke topik ini ketika hal <i>ThingName</i> dibuat. Untuk informasi selengkapnya, lihat the section called “Acara registri” .
<code>\$ aws/acara/hal/ThingName /diperbarui</code>	Langganan	AWS IoT menerbitkan ke topik ini ketika hal <i>ThingName</i> diperbarui. Untuk informasi selengkapnya, lihat the section called “Acara registri” .
<code>\$ aws/acara/hal/ThingName /dihapus</code>	Langganan	AWS IoT menerbitkan ke topik ini ketika hal <i>ThingName</i> dihapus. Untuk informasi selengkapnya, lihat the section called “Acara registri” .
<code>thingGroupName \$AWS/acara/Thinggroup//dibuat</code>	Langganan	AWS IoT menerbitkan ke topik ini ketika grup sesuatu <i>thingGroupName</i> dibuat. Untuk informasi selengkapnya, lihat the section called “Acara registri” .
<code>\$ thingGroupName AWS/acara/Thinggroup//diperbarui</code>	Langganan	AWS IoT menerbitkan ke topik ini ketika grup hal <i>thingGroupName</i> diperbarui. Untuk informasi selengkapnya, lihat the section called “Acara registri” .

Topik	Operasi klien diizinkan	Deskripsi
<code>\$ <i>thingGroupName</i> AWS/acara/Thinggroup//dihapus</code>	Langganan	AWS IoT menerbitkan ke topik ini ketika grup sesuatu <i>thingGroupName</i> dihapus. Untuk informasi selengkapnya, lihat the section called “Acara registri” .
<code><i>thingTypeName</i> \$AWS/Event/ThingType//dibuat</code>	Langganan	AWS IoT menerbitkan ke topik ini saat tipe <i>thingTypeName</i> benda dibuat. Untuk informasi selengkapnya, lihat the section called “Acara registri” .
<code>\$ <i>thingTypeName</i> AWS/Event/ThingType//diperbarui</code>	Langganan	AWS IoT menerbitkan ke topik ini ketika jenis <i>thingTypeName</i> hal diperbarui. Untuk informasi selengkapnya, lihat the section called “Acara registri” .
<code>\$ <i>thingTypeName</i> AWS/Event/ThingType//dihapus</code>	Langganan	AWS IoT menerbitkan ke topik ini ketika jenis <i>thingTypeName</i> hal dihapus. Untuk informasi selengkapnya, lihat the section called “Acara registri” .
<code><i>\$aws/events/ / thing/ thingType Association ThingName/ thingType Name</i></code>	Langganan	AWS IoT menerbitkan ke topik ini ketika <i>thingName</i> dikaitkan dengan atau dipisahkan dari tipe benda. <i>thingType Name</i> Untuk informasi selengkapnya, lihat the section called “Acara registri” .
<code><i>\$aws/events/ / ThingGroup// thingName/ThingName /ditambahkan thingGroupMembership thingGroupName</i></code>	Langganan	AWS IoT menerbitkan ke topik ini ketika sesuatu <i>ThingName</i> <i>ditambahkan</i> ke grup benda. <i>thingGroupName</i> Untuk informasi selengkapnya, lihat the section called “Acara registri” .

Topik	Operasi klien diizinkan	Deskripsi
<code>\$aws/events/ / ThingGroup/ /thing/ thingGroupMembersh ip thingName / dihapus thingGrou pName</code>	Langganan	AWS IoT menerbitkan ke topik ini ketika hal <i>ThingName</i> <i>dihapus</i> dari grup benda. <i>thingGroupName</i> Untuk informasi selengkapnya, lihat the section called “Acara registri” .
<code>\$aws/events/ / ThingGroup>Nama// thingGroupHierarchy>Nama/ditambahkan parentThingGroup childThingGroup childThingGroup</code>	Langganan	AWS IoT menerbitkan ke topik ini ketika <i>childThingGroupName</i> grup ditambahkan ke grup hal <i>parentThingGroupName</i> . Untuk informasi selengkapnya, lihat the section called “Acara registri” .
<code>\$aws/events/ / ThingGroup/ Nama// Nama thingGrou pHierarchy / dihapus parentThingGroup childThingGroup childThingGroup</code>	Langganan	AWS IoT menerbitkan ke topik ini ketika <i>childThingGroupName</i> grup dihapus dari grup <i>parentThingGroupName</i> . Untuk informasi selengkapnya, lihat the section called “Acara registri” .

Topik penyediaan armada

Note

Operasi klien dicatat sebagai Terima dalam tabel ini menunjukkan topik yang AWS IoT diterbitkan langsung ke klien yang memintanya, apakah klien telah berlangganan topik atau tidak. Klien harus mengharapkan untuk menerima pesan respons ini bahkan jika mereka belum berlangganan. Pesan respons ini tidak melewati broker pesan dan mereka tidak dapat berlangganan oleh klien atau aturan lain.

Pesan-pesan ini mendukung buffer respons dalam format Concise Binary Object Representation (CBOR) dan JavaScript Object Notation (JSON), tergantung pada format payload topik.

<i>format muatan</i>	Jenis data format respons
cbor	Representasi Objek Biner Ringkas (CBOR)
json	JavaScript Notasi Objek (JSON)

Untuk informasi selengkapnya, lihat [Penyediaan perangkat MQTT API](#).

Topik	Operasi klien diizinkan	Deskripsi
<i>\$ aws/sertifikat/buat/format muatan</i>	Publikasikan	Publikasikan ke topik ini untuk membuat sertifikat dari permintaan penandatangan sertifikat (CSR).
<i>\$ aws/sertifikat/buat/format payload/diterima</i>	Berlangganan, Terima	AWS IoT <i>menerbitkan ke topik ini setelah panggilan berhasil ke \$aws/certificates/create/payload-format.</i>
<i>\$ aws/sertifikat/buat/format payload/ditolak</i>	Berlangganan, Terima	AWS IoT <i>menerbitkan ke topik ini setelah panggilan gagal ke format \$aws/certificates/create/payload-format.</i>
<i>\$ aws/sertifikat//format muatan create-from-csr</i>	Publikasikan	Menerbitkan topik ini untuk membuat sertifikat dari CSR.
<i>\$ aws/sertifikat//format payload/diterima create-from-csr</i>	Berlangganan, Terima	AWS IoT <i>menerbitkan ke topik ini panggilan yang berhasil ke create-from-csr \$aws/certificates//payload-format.</i>

Topik	Operasi klien diizinkan	Deskripsi
<code>\$ aws/sertifikat//format payload/ditolak create-from-csr</code>	Berlangganan, Terima	AWS IoT <i>menerbitkan ke topik ini panggilan yang gagal ke create-from-csr \$aws/certificates//payload-format.</i>
<code>\$ aws/provisioning-template/ templateName/penyediaan/format muatan</code>	Publikasikan	Publikasikan ke topik ini untuk mendaftarkan sesuatu.
<code>\$aws/provisioning-templates/template Name /penyediaan/format payload/diterima</code>	Berlangganan, Terima	AWS IoT <i>menerbitkan ke topik ini setelah panggilan berhasil ke \$aws/provisioning-templates/ templateName / provision/ payload-format.</i>
<code>\$aws/provisioning-templates/template Name /penyediaan/format payload/ditolak</code>	Berlangganan, Terima	AWS IoT <i>menerbitkan ke topik ini setelah panggilan gagal ke \$aws/provisioning-templates / templateName /provision/ payload-format.</i>

Topik Job

Note

Operasi klien dicatat sebagai Terima dalam tabel ini menunjukkan topik yang AWS IoT diterbitkan langsung ke klien yang memintanya, apakah klien telah berlangganan topik atau tidak. Klien harus mengharapkan untuk menerima pesan respons ini bahkan jika mereka belum berlangganan.

Pesan respons ini tidak melewati broker pesan dan mereka tidak dapat berlangganan oleh klien atau aturan lain. Untuk berlangganan pesan terkait aktivitas pekerjaan, gunakan `notify` dan `notify-next` topik.

Saat berlangganan topik pekerjaan dan `jobExecution` acara untuk solusi pemantauan armada Anda, Anda harus terlebih dahulu mengaktifkan [acara pelaksanaan pekerjaan dan pekerjaan untuk menerima acara](#) apa pun di sisi cloud.

Untuk informasi selengkapnya, lihat [Pekerjaan perangkat operasi MQTT API](#).

Topik	Operasi klien diizinkan	Deskripsi
<code>\$aws/things/thingName /jobs/get</code>	Publikasikan	Perangkat mempublikasikan pesan ke topik ini untuk membuat <code>GetPendingJobExecutions</code> permintaan. Untuk informasi selengkapnya, lihat Pekerjaan perangkat operasi MQTT API .
<code>\$aws/things/thingName /jobs/dapatkan/diterima</code>	Berlangganan, Terima	Perangkat berlangganan topik ini untuk menerima tanggapan yang berhasil dari <code>GetPendingJobExecutions</code> permintaan. Untuk informasi selengkapnya, lihat Pekerjaan perangkat operasi MQTT API .
<code>\$aws/things/ThingName /jobs/get/ditolak</code>	Berlangganan, Terima	Perangkat berlangganan topik ini untuk menerima tanggapan saat <code>GetPendingJobExecutions</code> permintaan ditolak. Untuk informasi selengkapnya, lihat Pekerjaan perangkat operasi MQTT API .
<code>\$aws/things/ThingName /jobs/start-next</code>	Publikasikan	Perangkat mempublikasikan pesan ke topik ini untuk membuat <code>StartNextPendingJobExecution</code> permintaan. Untuk informasi selengkapnya, lihat Pekerjaan perangkat operasi MQTT API .
<code>\$aws/things/thingName /jobs/start-next/diterima</code>	Berlangganan, Terima	Perangkat berlangganan topik ini untuk menerima tanggapan yang berhasil <code>StartNextPendingJobExecution</code>

Topik	Operasi klien diizinkan	Deskripsi
		n atas permintaan. Untuk informasi selengkapnya, lihat Pekerjaan perangkat operasi MQTT API .
<code>\$aws/things/ThingName /jobs/start-next/ditolak</code>	Berlangganan, Terima	Perangkat berlangganan topik ini untuk menerima tanggapan saat StartNext PendingJobExecution permintaan ditolak. Untuk informasi selengkapnya, lihat Pekerjaan perangkat operasi MQTT API .
<code>\$aws/things/thingName /jobs/jobId/dapatkan</code>	Publikasikan	Perangkat mempublikasikan pesan ke topik ini untuk membuat DescribeJobExecution permintaan. Untuk informasi selengkapnya, lihat Pekerjaan perangkat operasi MQTT API .
<code>\$aws/things/thingName /jobs/jobId/dapatkan/diterima</code>	Berlangganan, Terima	Perangkat berlangganan topik ini untuk menerima tanggapan yang berhasil DescribeJobExecution atas permintaan. Untuk informasi selengkapnya, lihat Pekerjaan perangkat operasi MQTT API .
<code>\$aws/things/thingName /jobs/jobId/dapatkan/ditolak</code>	Berlangganan, Terima	Perangkat berlangganan topik ini untuk menerima tanggapan saat DescribeJobExecution permintaan ditolak. Untuk informasi selengkapnya, lihat Pekerjaan perangkat operasi MQTT API .
<code>\$ aws/things/thingName /jobs/jobId/update</code>	Publikasikan	Perangkat mempublikasikan pesan ke topik ini untuk membuat UpdateJobExecution permintaan. Untuk informasi selengkapnya, lihat Pekerjaan perangkat operasi MQTT API .

Topik	Operasi klien diizinkan	Deskripsi
<code>\$aws/things/ ThingName /jobs/ joBid/update/diter ima</code>	Berlangganan, Terima	<p>Perangkat berlangganan topik ini untuk menerima tanggapan yang berhasil UpdateJobExecution atas permintaan. Untuk informasi selengkapnya, lihat Pekerjaan perangkat operasi MQTT API.</p> <div style="border: 1px solid #add8e6; border-radius: 10px; padding: 10px; margin-top: 10px;"> <p>ⓘ Catatan</p> <p><i>Hanya perangkat yang menerbitkan ke <code>\$aws/things/ThingName /jobs/joBid/update</code> yang menerima pesan tentang topik ini.</i></p> </div>
<code>\$aws/things/ ThingName /jobs/ joBid/update/ditol ak</code>	Berlangganan, Terima	<p>Perangkat berlangganan topik ini untuk menerima tanggapan saat UpdateJobExecution permintaan ditolak. Untuk informasi selengkapnya, lihat Pekerjaan perangkat operasi MQTT API.</p> <div style="border: 1px solid #add8e6; border-radius: 10px; padding: 10px; margin-top: 10px;"> <p>ⓘ Catatan</p> <p><i>Hanya perangkat yang menerbitkan ke <code>\$aws/things/ThingName /jobs/joBid/update</code> yang menerima pesan tentang topik ini.</i></p> </div>

Topik	Operasi klien diizinkan	Deskripsi
<code>\$aws/things/ ThingName /jobs/not ify</code>	Berlangganan, Terima	Perangkat berlangganan topik ini untuk menerima pemberitahuan saat eksekusi pekerjaan ditambahkan atau dihapus ke daftar eksekusi yang tertunda untuk suatu hal. Untuk informasi selengkapnya, lihat Pekerjaan perangkat operasi MQTT API .
<code>\$aws/things/ ThingName /jobs/not ify-next</code>	Berlangganan, Terima	Perangkat berlangganan topik ini untuk menerima pemberitahuan ketika eksekusi pekerjaan tertunda berikutnya untuk hal tersebut diubah. Untuk informasi selengkapnya, lihat Pekerjaan perangkat operasi MQTT API .
<code>\$ aws/acara/ pekerjaan/jobId/s elesai</code>	Langganan	Layanan Jobs menerbitkan acara tentang topik ini ketika pekerjaan selesai. Untuk informasi selengkapnya, lihat Acara Lowongan .
<code>\$ aws/acara/ pekerjaan/jobId/d ibatalan</code>	Langganan	Layanan Jobs menerbitkan acara tentang topik ini ketika pekerjaan dibatalkan. Untuk informasi selengkapnya, lihat Acara Lowongan .
<code>\$ aws/acara/ pekerjaan/jobId/d ihapus</code>	Langganan	Layanan Jobs menerbitkan acara tentang topik ini ketika pekerjaan dihapus. Untuk informasi selengkapnya, lihat Acara Lowongan .
<code>\$aws/acara/pekerja an/jobId/cancellat ion_in_progress</code>	Langganan	Layanan Jobs menerbitkan acara tentang topik ini ketika pembatalan pekerjaan dimulai. Untuk informasi selengkapnya, lihat Acara Lowongan .

Topik	Operasi klien diizinkan	Deskripsi
<i>\$aws/acara/pekerjaan/jobBid/deletion_in_progress</i>	Langganan	Layanan Jobs menerbitkan acara tentang topik ini ketika penghapusan pekerjaan dimulai. Untuk informasi selengkapnya, lihat Acara Lowongan .
<i>\$AWS/acara/Jobexecution/jobBid/berhasil</i>	Langganan	Layanan Jobs menerbitkan acara tentang topik ini ketika eksekusi pekerjaan berhasil. Untuk informasi selengkapnya, lihat Acara Lowongan .
<i>\$ AWS/acara/JobeXecution/jobBid/gagal</i>	Langganan	Layanan Jobs menerbitkan acara tentang topik ini ketika eksekusi pekerjaan gagal. Untuk informasi selengkapnya, lihat Acara Lowongan .
<i>\$AWS/acara/JobeXecution/jobBid/ditolak</i>	Langganan	Layanan Jobs menerbitkan acara tentang topik ini ketika eksekusi pekerjaan ditolak. Untuk informasi selengkapnya, lihat Acara Lowongan .
<i>\$AWS/acara/JobeXecution/jobID/dibatalkan</i>	Langganan	Layanan Jobs menerbitkan acara tentang topik ini ketika eksekusi pekerjaan dibatalkan. Untuk informasi selengkapnya, lihat Acara Lowongan .
<i>\$AWS/acara/JobeXecution/jobBid/timed_out</i>	Langganan	Layanan Jobs menerbitkan acara tentang topik ini ketika waktu eksekusi pekerjaan habis. Untuk informasi selengkapnya, lihat Acara Lowongan .
<i>\$ AWS/acara/JobeXecution/jobBid/dihapus</i>	Langganan	Layanan Jobs memublikasikan peristiwa tentang topik ini saat eksekusi pekerjaan dihapus. Untuk informasi selengkapnya, lihat Acara Lowongan .

Topik	Operasi klien diizinkan	Deskripsi
<code>\$ AWS/acara/ JobeXecution/jobId/dihapus</code>	Langganan	Layanan Jobs menerbitkan acara tentang topik ini ketika eksekusi pekerjaan dihapus. Untuk informasi selengkapnya, lihat Acara Lowongan .

Topik aturan

Topik	Operasi klien diizinkan	Deskripsi
<code>\$ aws/aturan/ RuleName</code>	Publikasikan	Perangkat atau aplikasi menerbitkan topik ini untuk memicu aturan secara langsung. Untuk informasi selengkapnya, lihat Mengurangi biaya pengiriman pesan dengan Basic Ingest .

Topik tunneling yang aman

Topik	Operasi klien diizinkan	Deskripsi
<code>\$aws/things/thing-name /tunnels/ notify</code>	Langganan	AWS IoT menerbitkan pesan ini untuk agen IoT untuk memulai proxy lokal di perangkat jarak jauh. Untuk informasi selengkapnya, lihat the section called “Cuplikan agen IoT” .

Topik bayangan

Topik di bagian ini digunakan oleh bayangan bernama dan tidak disebutkan namanya. Topik yang digunakan oleh masing-masing hanya berbeda dalam awalan topik. Tabel ini menunjukkan awalan topik yang digunakan oleh setiap jenis bayangan.

Nilai <i>ShadowTopicPrefix</i>	Jenis bayangan
<i>\$aws/things/thingName /shadow</i>	Bayangan tanpa nama (klasik)
<i>\$aws/things/thingName /shadow/nama/ ShadoName</i>	Bernama bayangan

Untuk membuat topik lengkap, pilih jenis bayangan yang ingin Anda rujuk, ganti ***ThingName*** dan jika berlaku, ***ShadoName***, dengan nilai yang sesuai, lalu tambahkan dengan topik rintisan seperti yang ditunjukkan pada tabel berikut. ***ShadowTopicPrefix*** Ingatlah bahwa topik peka huruf besar/kecil.

Topik	Operasi klien diizinkan	Deskripsi
<i>ShadowTopicPrefix /hapus</i>	Terbitkan/Berlangganan	Perangkat atau aplikasi menerbitkan topik ini untuk menghapus bayangan. Untuk informasi selengkapnya, lihat /delete .
<i>ShadowTopicPrefix /hapus/diterima</i>	Langganan	Layanan Device Shadow mengirimkan pesan ke topik ini saat bayangan dihapus. Untuk informasi selengkapnya, lihat /delete/accepted .
<i>ShadowTopicPrefix /hapus/ditolak</i>	Langganan	Layanan Device Shadow mengirimkan pesan ke topik ini saat permintaan untuk menghapus bayangan ditolak. Untuk informasi selengkapnya, lihat /hapus/ditolak .
<i>ShadowTopicPrefix /dapatkan</i>	Terbitkan/Berlangganan	Aplikasi atau sesuatu menerbitkan pesan kosong ke topik ini untuk mendapatkan bayangan. Untuk informasi selengkapnya, lihat Topik Device Shadow MQTT .

Topik	Operasi klien diizinkan	Deskripsi
<i>ShadowTopicPrefix</i> / dapatkan/diterima	Langganan	Layanan Device Shadow mengirim pesan ke topik ini saat permintaan bayangan berhasil dibuat. Untuk informasi selengkapnya, lihat /dapatkan/diterima .
<i>ShadowTopicPrefix</i> / dapatkan/ditolak	Langganan	Layanan Device Shadow mengirimkan pesan ke topik ini saat permintaan bayangan ditolak. Untuk informasi selengkapnya, lihat /get/ditolak .
<i>ShadowTopicPrefix</i> / perbarui	Terbitkan/Berlangganan	Sesuatu atau aplikasi menerbitkan topik ini untuk memperbarui bayangan. Untuk informasi selengkapnya, lihat /update .
<i>ShadowTopicPrefix</i> / perbarui/diterima	Langganan	Layanan Device Shadow mengirimkan pesan ke topik ini ketika pembaruan berhasil dibuat ke bayangan. Untuk informasi selengkapnya, lihat /update/accepted .
<i>ShadowTopicPrefix</i> / perbarui/ditolak	Langganan	Layanan Device Shadow mengirimkan pesan ke topik ini saat pembaruan ke bayangan ditolak. Untuk informasi selengkapnya, lihat /update/ditolak .
<i>ShadowTopicPrefix</i> / perbarui/delta	Langganan	Layanan Device Shadow mengirimkan pesan ke topik ini ketika perbedaan terdeteksi antara bagian bayangan yang dilaporkan dan yang diinginkan. Untuk informasi lebih lanjut, lihat /update/delta .

Topik	Operasi klien diizinkan	Deskripsi
<i>ShadowTopicPrefix</i> / pembaruan/dokumen	Langganan	AWS IoT menerbitkan dokumen status ke topik ini setiap kali pembaruan bayangan berhasil dilakukan. Untuk informasi selengkapnya, lihat /update/documents .

Topik pengiriman file berbasis MQTT

Note

Operasi klien dicatat sebagai Terima dalam tabel ini menunjukkan topik yang AWS IoT diterbitkan langsung ke klien yang memintanya, apakah klien telah berlangganan topik atau tidak. Klien harus mengharapkan untuk menerima pesan respons ini bahkan jika mereka belum berlangganan. Pesan respons ini tidak melewati broker pesan dan mereka tidak dapat berlangganan oleh klien atau aturan lain.

Pesan-pesan ini mendukung buffer respons dalam format Concise Binary Object Representation (CBOR) dan JavaScript Object Notation (JSON), tergantung pada format payload topik.

format muatan	Jenis data format respons
cbor	Representasi Objek Biner Ringkas (CBOR)
json	JavaScript Notasi Objek (JSON)

Topik	Operasi klien diizinkan	Deskripsi
<i>\$ aws/hal//stream//data/format muatan ThingName StreamId</i>	Berlangganan, Terima	AWS Pengiriman file berbasis MQTT diterbitkan ke topik ini jika permintaan "GetStream" dari perangkat diterima. Muatan berisi data aliran. Untuk informasi selengkapnya, lihat

Topik	Operasi klien diizinkan	Deskripsi
		Menggunakan pengiriman AWS IoT file berbasis MQTT di perangkat.
<code>\$ aws/hal//streams//get/payload-format ThingName StreamId</code>	Publikasikan	Perangkat menerbitkan topik ini untuk melakukan permintaan <code>GetStream ""</code> . Untuk informasi selengkapnya, lihat Menggunakan pengiriman AWS IoT file berbasis MQTT di perangkat.
<code>\$ aws/hal//streams//deskripsi/format muatan ThingName StreamId</code>	Berlangganan, Terima	AWS Pengiriman file berbasis MQTT diterbitkan ke topik ini jika permintaan <code>"DescribeStream"</code> dari perangkat diterima. Muatan berisi deskripsi aliran. Untuk informasi selengkapnya, lihat Menggunakan pengiriman AWS IoT file berbasis MQTT di perangkat.
<code>\$ aws/hal//streams//jelaskan/format muatan ThingName StreamId</code>	Publikasikan	Perangkat menerbitkan topik ini untuk melakukan permintaan <code>DescribeStream ""</code> . Untuk informasi selengkapnya, lihat Menggunakan pengiriman AWS IoT file berbasis MQTT di perangkat.
<code>\$ aws/hal//stream//ditolak/format muatan ThingName StreamId</code>	Berlangganan, Terima	AWS Pengiriman file berbasis MQTT diterbitkan ke topik ini jika permintaan <code>""</code> atau <code>DescribeStream "GetStream"</code> dari perangkat ditolak. Untuk informasi selengkapnya, lihat Menggunakan pengiriman AWS IoT file berbasis MQTT di perangkat.

Topik cadangan ARN

Semua ARN topik yang dicadangkan (Nama Sumber Daya Amazon) memiliki formulir berikut:

```
arn:aws:iot:aws-region:AWS-account-ID:topic/Topic
```

Misalnya, `arn:aws:iot:us-west-2:123EXAMPLE456:topic/$aws/things/thingName/jobs/get/accepted` adalah ARN untuk topik yang dicadangkan. `$aws/things/thingName/jobs/get/accepted`

Titik akhir yang dapat dikonfigurasi

Di AWS IoT Core, Anda dapat menggunakan konfigurasi domain untuk mengonfigurasi dan mengelola perilaku titik akhir data Anda. Dengan konfigurasi domain, Anda dapat menghasilkan beberapa titik akhir AWS IoT Core data, menyesuaikan titik akhir data ini dengan nama domain yang sepenuhnya memenuhi syarat (FQDN) dan sertifikat server terkait, dan juga mengaitkan otorisasi kustom. Untuk informasi selengkapnya, lihat [Otentikasi dan otorisasi khusus](#).

Note

Fitur ini tidak tersedia di GovCloud Wilayah AWS.

Kasus penggunaan konfigurasi domain

Anda dapat menggunakan konfigurasi domain untuk menyederhanakan tugas-tugas seperti berikut ini.

- Migrasikan perangkat ke AWS IoT Core.
- Mendukung armada perangkat heterogen dengan mempertahankan konfigurasi domain terpisah untuk jenis perangkat yang terpisah.
- Pertahankan identitas merek (misalnya, melalui nama domain) saat memigrasikan infrastruktur aplikasi ke AWS IoT Core.

Catatan penting untuk menggunakan konfigurasi domain di AWS IoT Core

AWS IoT Core menggunakan [ekstensi TLS indikasi nama server \(SNI\)](#) untuk menerapkan konfigurasi domain. Perangkat harus menggunakan ekstensi ini saat terhubung AWS IoT Core. Mereka juga harus melewati nama server yang identik dengan nama domain yang Anda tentukan dalam konfigurasi domain. Untuk menguji layanan ini, gunakan versi v2 [SDK AWS IoT Perangkat](#) di GitHub.

Jika Anda membuat beberapa titik akhir data di Anda Akun AWS, mereka akan berbagi AWS IoT Core sumber daya seperti topik MQTT, bayangan perangkat, dan aturan.

Ketika Anda memberikan sertifikat server untuk konfigurasi domain AWS IoT Core kustom, sertifikat memiliki maksimal empat nama domain. Untuk informasi lebih lanjut, lihat [AWS IoT Core kuota dan titik akhir](#).

Dalam Bab ini:

- [Membuat dan mengonfigurasi domain AWS terkelola](#)
- [Membuat dan mengonfigurasi domain khusus](#)
- [Mengelola konfigurasi domain](#)
- [Mengkonfigurasi pengaturan TLS dalam konfigurasi domain](#)
- [Konfigurasi sertifikat server untuk stapling OCSP](#)

Membuat dan mengonfigurasi domain AWS terkelola

Anda membuat titik akhir yang dapat dikonfigurasi pada domain AWS terkelola menggunakan API. [CreateDomainConfiguration](#) Konfigurasi domain untuk domain AWS terkelola terdiri dari yang berikut:

- `domainConfigurationName`

Nama yang ditentukan pengguna yang mengidentifikasi konfigurasi domain dan nilainya harus unik untuk Anda. Wilayah AWS Anda tidak dapat menggunakan nama konfigurasi domain yang dimulai IoT: karena nama tersebut dicadangkan untuk titik akhir default.

- `defaultAuthorizerName` (opsional)

Nama otorisasi khusus untuk digunakan pada titik akhir.

- `allowAuthorizerOverride` (opsional)

Nilai Boolean yang menentukan apakah perangkat dapat mengganti authorizer default dengan menentukan authorizer yang berbeda di header HTTP permintaan. Nilai ini diperlukan jika nilai untuk `defaultAuthorizerName` ditentukan.

- `serviceType` (opsional)

Jenis layanan yang diberikan endpoint. AWS IoT Core hanya mendukung jenis DATA layanan. Bila Anda menentukan DATA, AWS IoT Core mengembalikan endpoint dengan tipe endpoint.

`iot:Data-ATS` Anda tidak dapat membuat titik akhir yang dapat dikonfigurasi `iot:Data` (VeriSign).

- `TlsConfig` (opsional)

Objek yang menentukan konfigurasi TLS untuk domain. Untuk informasi selengkapnya, lihat [???](#).

Contoh AWS CLI perintah berikut membuat konfigurasi domain untuk `Data` endpoint.

```
aws iot create-domain-configuration --domain-configuration-name
  "myDomainConfigurationName" --service-type "DATA"
```

Output dari perintah dapat terlihat seperti berikut.

```
{
  "domainConfigurationName": "myDomainConfigurationName",
  "domainConfigurationArn": "arn:aws:iot:us-east-1:123456789012:domainconfiguration/
  myDomainConfigurationName/itihw"
}
```

Membuat dan mengonfigurasi domain khusus

Konfigurasi domain memungkinkan Anda menentukan nama domain kustom yang memenuhi syarat (FQDN) untuk terhubung. AWS IoT Core Ada banyak manfaat menggunakan domain khusus: Anda dapat mengekspos domain Anda sendiri atau domain perusahaan Anda sendiri kepada pelanggan untuk tujuan branding; Anda dapat dengan mudah mengubah domain Anda sendiri untuk menunjuk ke broker baru; Anda dapat mendukung multi-tenancy untuk melayani pelanggan dengan domain yang berbeda dalam hal yang sama Akun AWS; dan Anda dapat mengelola detail sertifikat server Anda sendiri, seperti otoritas sertifikat root (CA) yang digunakan untuk menandatangani sertifikat, algoritma tanda tangan, kedalaman rantai sertifikat, dan siklus hidup sertifikat.

Alur kerja untuk mengatur konfigurasi domain dengan domain kustom terdiri dari tiga tahap berikut.

1. [Mendaftarkan Sertifikat Server di AWS Certificate Manager](#)
2. [Membuat Konfigurasi Domain](#)
3. [Membuat Catatan DNS](#)

Mendaftarkan sertifikat server di manajer AWS sertifikat

Sebelum Anda membuat konfigurasi domain dengan domain kustom, Anda harus mendaftarkan rantai sertifikat server Anda di [AWS Certificate Manager \(ACM\)](#). Anda dapat menggunakan tiga jenis sertifikat server berikut.

- [ACM Menghasilkan Sertifikat Publik](#)
- [Sertifikat Eksternal Ditandatangani oleh CA Publik](#)
- [Sertifikat Eksternal Ditandatangani oleh CA Pribadi](#)

Note

AWS IoT Core menganggap sertifikat ditandatangani oleh CA publik jika disertakan dalam [ca-bundle tepercaya Mozilla](#).

Persyaratan sertifikat

Lihat [Prasyarat untuk Mengimpor Sertifikat untuk persyaratan untuk mengimpor sertifikat](#) ke ACM. Selain persyaratan ini, AWS IoT Core tambahkan persyaratan berikut.

- Sertifikat leaf harus menyertakan ekstensi Extended Key Usage x509 v3 dengan nilai ServerAuth (Otentikasi Server Web TLS). Jika Anda meminta sertifikat dari ACM, ekstensi ini ditambahkan secara otomatis.
- Kedalaman rantai sertifikat maksimum adalah 5 sertifikat.
- Ukuran rantai sertifikat maksimum adalah 16KB.
- Algoritma kriptografi dan ukuran kunci yang didukung termasuk RSA 2048 bit (RSA_2048) dan ECDSA 256 bit (EC_Prime256v1).

Menggunakan satu sertifikat untuk beberapa domain

Jika Anda berencana menggunakan satu sertifikat untuk mencakup beberapa subdomain, gunakan domain wildcard di bidang common name (CN) atau Subject Alternative Names (SAN). Misalnya, gunakan ***.iot.example.com** untuk menutupi dev.iot.example.com, qa.iot.example.com, dan prod.iot.example.com. Setiap FQDN memerlukan konfigurasi domainnya sendiri, tetapi lebih dari satu konfigurasi domain dapat menggunakan nilai wildcard yang sama. Baik CN atau SAN harus

mencakup FQDN yang ingin Anda gunakan sebagai domain khusus. Jika SAN hadir, CN diabaikan dan SAN harus mencakup FQDN yang ingin Anda gunakan sebagai domain khusus. Cakupan ini bisa berupa kecocokan persis atau pertandingan wildcard. Setelah sertifikat wildcard divalidasi dan didaftarkan ke akun, akun lain di wilayah tersebut diblokir agar tidak membuat domain khusus yang tumpang tindih dengan sertifikat.

Bagian berikut menjelaskan cara mendapatkan setiap jenis sertifikat. Setiap sumber daya sertifikat memerlukan Nama Sumber Daya Amazon (ARN) yang terdaftar dengan ACM yang Anda gunakan saat membuat konfigurasi domain.

Sertifikat publik yang dihasilkan ACM

Anda dapat membuat sertifikat publik untuk domain kustom Anda dengan menggunakan [RequestCertificate](#) API. Saat Anda membuat sertifikat dengan cara ini, ACM memvalidasi kepemilikan Anda atas domain kustom. Untuk informasi lebih lanjut, lihat [Permintaan Sertifikat Publik](#) dalam Panduan Pengguna AWS Certificate Manager .

Sertifikat eksternal yang ditandatangani oleh CA publik

Jika Anda sudah memiliki sertifikat server yang ditandatangani oleh CA publik (CA yang disertakan dalam ca-bundle terpercaya Mozilla), Anda dapat mengimpor rantai sertifikat langsung ke ACM dengan menggunakan API. [ImportCertificate Untuk mempelajari lebih lanjut tentang tugas ini dan prasyarat serta persyaratan format sertifikat, lihat Mengimpor Sertifikat.](#)

Sertifikat eksternal ditandatangani oleh CA pribadi

Jika Anda sudah memiliki sertifikat server yang ditandatangani oleh CA pribadi atau ditandatangani sendiri, Anda dapat menggunakan sertifikat untuk membuat konfigurasi domain Anda, tetapi Anda juga harus membuat sertifikat publik tambahan di ACM untuk memvalidasi kepemilikan domain Anda. Untuk melakukan ini, daftarkan rantai sertifikat server Anda di ACM menggunakan [ImportCertificate](#) API. [Untuk mempelajari lebih lanjut tentang tugas ini dan prasyarat serta persyaratan format sertifikat, lihat Mengimpor Sertifikat.](#)

Membuat sertifikat validasi

Setelah Anda mengimpor sertifikat ke ACM, buat sertifikat publik untuk domain kustom Anda dengan menggunakan [RequestCertificate](#) API. Saat Anda membuat sertifikat dengan cara ini, ACM memvalidasi kepemilikan Anda atas domain kustom. Untuk informasi selengkapnya, lihat [Meminta Sertifikat Publik](#). Saat Anda membuat konfigurasi domain, gunakan sertifikat publik ini sebagai sertifikat validasi Anda.

Membuat konfigurasi domain

Anda membuat titik akhir yang dapat dikonfigurasi pada domain kustom dengan menggunakan API. [CreateDomainConfiguration](#) Konfigurasi domain untuk domain kustom terdiri dari yang berikut:

- `domainConfigurationName`

Nama yang ditentukan pengguna yang mengidentifikasi konfigurasi domain. Nama konfigurasi domain IoT: yang dimulai dengan dicadangkan untuk titik akhir default dan tidak dapat digunakan. Juga, nilai ini harus unik untuk Anda Wilayah AWS.

- `domainName`

FQDN yang digunakan perangkat Anda untuk terhubung. AWS IoT Core AWS IoT Core memanfaatkan ekstensi TLS indikasi nama server (SNI) untuk menerapkan konfigurasi domain. Perangkat harus menggunakan ekstensi ini saat menghubungkan dan meneruskan nama server yang identik dengan nama domain yang ditentukan dalam konfigurasi domain.

- `serverCertificateArns`

ARN dari rantai sertifikat server yang Anda daftarkan dengan ACM. AWS IoT Core saat ini hanya mendukung satu sertifikat server.

- `validationCertificateArn`

ARN sertifikat publik yang Anda buat di ACM untuk memvalidasi kepemilikan domain kustom Anda. Argumen ini tidak diperlukan jika Anda menggunakan sertifikat server yang ditandatangani secara publik atau yang dibuat ACM.

- `defaultAuthorizerName` (optional)

Nama otorisasi khusus untuk digunakan pada titik akhir.

- `allowAuthorizerOverride`

Nilai Boolean yang menentukan apakah perangkat dapat mengganti authorizer default dengan menentukan authorizer yang berbeda di header HTTP permintaan. Nilai ini diperlukan jika nilai untuk `defaultAuthorizerName` ditentukan.

- `serviceType`

AWS IoT Core saat ini hanya mendukung jenis DATA layanan. Bila Anda menentukan DATA, AWS IoT mengembalikan endpoint dengan tipe endpoint. `iot:Data-ATS`

- `TlsConfig` (opsional)

Objek yang menentukan konfigurasi TLS untuk domain. Untuk informasi selengkapnya, lihat [???](#).

- `serverCertificateConfig` (opsional)

Objek yang menentukan konfigurasi sertifikat server untuk domain. Untuk informasi selengkapnya, lihat [???](#).

AWS CLI Perintah berikut membuat konfigurasi domain untuk `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

Setelah Anda membuat konfigurasi domain, mungkin diperlukan waktu hingga 60 menit hingga AWS IoT Core menyajikan sertifikat server kustom Anda.

Untuk informasi selengkapnya, lihat [???](#).

Membuat catatan DNS

Setelah Anda mendaftarkan rantai sertifikat server Anda dan membuat konfigurasi domain Anda, buat catatan DNS sehingga domain kustom Anda menunjuk ke AWS IoT domain. Catatan ini harus menunjuk ke AWS IoT titik akhir tipe `iot:Data-ATS`. Anda bisa mendapatkan endpoint Anda dengan menggunakan [DescribeEndpointAPI](#).

AWS CLI Perintah berikut menunjukkan cara mendapatkan endpoint Anda.

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

Setelah Anda mendapatkan `iot:Data-ATS` titik akhir Anda, buat CNAME catatan dari domain kustom Anda ke titik AWS IoT akhir ini. Jika Anda membuat beberapa domain kustom yang sama Akun AWS, alias mereka ke titik akhir yang sama `iot:Data-ATS` ini.

Pemecahan Masalah

Jika Anda mengalami kesulitan menghubungkan perangkat ke domain kustom, pastikan bahwa AWS IoT Core telah menerima dan menerapkan sertifikat server Anda. Anda dapat memverifikasi bahwa AWS IoT Core telah menerima sertifikat Anda dengan menggunakan AWS IoT Core konsol atau AWS CLI.

Untuk menggunakan AWS IoT Core konsol, navigasikan ke halaman Pengaturan dan pilih nama konfigurasi domain. Di bagian Detail sertifikat server, periksa detail status dan status. Jika sertifikat tidak valid, gantilah di ACM dengan sertifikat yang memenuhi [persyaratan sertifikat](#) yang tercantum di bagian sebelumnya. Jika sertifikat memiliki ARN yang sama, AWS IoT Core akan mengambilnya dan menerapkannya secara otomatis.

Untuk memeriksa status sertifikat dengan menggunakan AWS CLI, panggil [DescribeDomainConfiguration](#) API dan tentukan nama konfigurasi domain Anda.

Note

Jika sertifikat Anda tidak valid, AWS IoT Core akan terus melayani sertifikat terakhir yang valid.

Anda dapat memeriksa sertifikat mana yang sedang disajikan di titik akhir Anda dengan menggunakan perintah openssl berikut.

```
openssl s_client -connect custom-domain-name:8883 -showcerts -servername custom-domain-name
```

Mengelola konfigurasi domain

Anda dapat mengelola siklus hidup konfigurasi yang ada dengan menggunakan API berikut.

- [ListDomainConfigurations](#)
- [DescribeDomainConfiguration](#)
- [UpdateDomainConfiguration](#)
- [DeleteDomainConfiguration](#)

Melihat konfigurasi domain

Untuk mengembalikan daftar paginasi semua konfigurasi domain di Anda Akun AWS, gunakan API [ListDomainConfigurations](#). Anda dapat melihat detail konfigurasi domain tertentu menggunakan [DescribeDomainConfiguration](#) API. API ini mengambil satu `domainConfigurationName` parameter dan mengembalikan detail konfigurasi yang ditentukan.

Contoh

Memperbarui konfigurasi domain

Untuk memperbarui status atau otorisasi kustom konfigurasi domain Anda, gunakan [UpdateDomainConfiguration](#) API. Anda dapat mengatur status ke `ENABLED` atau `DISABLED`. Jika Anda menonaktifkan konfigurasi domain, perangkat yang terhubung ke domain tersebut akan menerima kesalahan otentikasi. Saat ini Anda tidak dapat memperbarui sertifikat server dalam konfigurasi domain Anda. Untuk mengubah sertifikat konfigurasi domain, Anda harus menghapus dan membuatnya kembali.

Contoh

Menghapus konfigurasi domain

Sebelum menghapus konfigurasi domain, gunakan [UpdateDomainConfiguration](#) API untuk menyetel statusnya `DISABLED`. Ini membantu Anda menghindari penghapusan titik akhir secara tidak sengaja. Setelah Anda menonaktifkan konfigurasi domain, hapus dengan menggunakan [DeleteDomainConfiguration](#) API. Anda harus menempatkan domain AWS-managed dalam `DISABLED` status selama 7 hari sebelum Anda dapat menghapusnya. Anda dapat menempatkan domain kustom dalam `DISABLED` status dan kemudian menghapusnya sekaligus.

Contoh

Setelah Anda menghapus konfigurasi domain, AWS IoT Core tidak lagi menyajikan sertifikat server yang terkait dengan domain kustom tersebut.

Memutar sertifikat di domain khusus

Anda mungkin perlu mengganti sertifikat server Anda secara berkala dengan sertifikat yang diperbarui. Tingkat di mana Anda melakukan ini tergantung pada masa berlaku sertifikat Anda. Jika Anda membuat sertifikat server dengan menggunakan AWS Certificate Manager (ACM), Anda dapat mengatur sertifikat untuk diperpanjang secara otomatis. Saat ACM memperbarui sertifikat Anda,

AWS IoT Core secara otomatis mengambil sertifikat baru. Anda tidak perlu melakukan tindakan tambahan apa pun. Jika Anda mengimpor sertifikat server dari sumber yang berbeda, Anda dapat memutarkannya dengan mengimpornya kembali ke ACM. Untuk informasi tentang mengimpor ulang sertifikat, lihat [Mengimpor ulang](#) sertifikat.

Note

AWS IoT Core hanya mengambil pembaruan sertifikat dalam kondisi berikut.

- Sertifikat baru memiliki ARN yang sama dengan yang lama.
- Sertifikat baru memiliki algoritma penandatanganan, nama umum, atau nama alternatif subjek yang sama dengan yang lama.

Mengkonfigurasi pengaturan TLS dalam konfigurasi domain

AWS IoT Core menyediakan [kebijakan keamanan yang telah ditentukan sebelumnya](#) bagi Anda untuk menyesuaikan pengaturan Transport Layer Security (TLS) Anda untuk [TLS 1.2 dan TLS 1.3](#) dalam konfigurasi domain. Kebijakan keamanan adalah kombinasi protokol TLS dan cipher mereka yang menentukan protokol dan cipher yang didukung selama negosiasi TLS antara klien dan server. Dengan kebijakan keamanan yang didukung, Anda dapat mengelola pengaturan TLS perangkat Anda dengan lebih fleksibel, menerapkan langkah-langkah up-to-date keamanan paling banyak saat menghubungkan perangkat baru, dan mempertahankan konfigurasi TLS yang konsisten untuk perangkat yang ada.

Tabel berikut menjelaskan kebijakan keamanan, versi TLS, dan wilayah yang didukung:

Nama kebijakan keamanan	Didukung Wilayah AWS
IoT_TLS13_1_3_2022_10 SecurityPolicy	Semua Wilayah AWS
IoT_TLS13_1_2_2022_10 SecurityPolicy	Semua Wilayah AWS

Nama kebijakan keamanan	Didukung Wilayah AWS
IoT_TLS12_1_2_2022_10 SecurityPolicy	Semua Wilayah AWS
IoT_TLS12_1_0_2016_01 SecurityPolicy	ap-east-1, ap-northeast-2, ap-selatan-1, ap-south-1, ap-southeast-2, ca-central-1, cn-utara-1, cn-utara-1, cn-barat laut-1, eu-north-1, eu-north-1, eu-west-2, eu-west-2, eu-west-2 3, me-south-1, sa-east-1, us-east-1, us-east-2, us-west-1
IoT_TLS12_1_0_2015_01 SecurityPolicy	ap-northeast-1, ap-southeast-1, eu-central-1, eu-central-1, eu-west-1, us-east-1, us-east-1, us-west-2

Nama-nama kebijakan keamanan AWS IoT Core termasuk informasi versi berdasarkan tahun dan bulan mereka dirilis. Jika Anda membuat konfigurasi domain baru, kebijakan keamanan akan defaultIoTSecurityPolicy_TLS13_1_2_2022_10. [Untuk tabel lengkap kebijakan keamanan dengan rincian protokol, port TCP, dan cipher, lihat Kebijakan keamanan.](#) AWS IoT Core tidak mendukung kebijakan keamanan khusus. Untuk informasi selengkapnya, lihat [???](#).

Untuk mengkonfigurasi pengaturan TLS dalam konfigurasi domain, Anda dapat menggunakan AWS IoT konsol atau. AWS CLI

Daftar Isi

- [Konfigurasi pengaturan TLS dalam konfigurasi domain \(konsol\)](#)
- [Konfigurasi pengaturan TLS dalam konfigurasi domain \(CLI\)](#)

Konfigurasi pengaturan TLS dalam konfigurasi domain (konsol)

Untuk mengkonfigurasi pengaturan TLS menggunakan konsol AWS IoT

1. Masuk ke AWS Management Console dan buka [AWS IoT konsol](#).
2. Untuk mengkonfigurasi pengaturan TLS saat Anda membuat konfigurasi domain baru, ikuti langkah-langkah ini.

1. Di panel navigasi kiri, pilih Pengaturan, dan kemudian, dari bagian Konfigurasi domain, pilih Buat konfigurasi domain.
 2. Di halaman Buat konfigurasi domain, di bagian Pengaturan domain khusus - opsional, pilih kebijakan keamanan dari Pilih kebijakan keamanan.
 3. Ikuti widget dan selesaikan langkah-langkah lainnya. Pilih Buat konfigurasi domain.
3. Untuk memperbarui setelan TLS dalam konfigurasi domain yang ada, ikuti langkah-langkah berikut.
1. Di panel navigasi kiri, pilih Pengaturan, dan kemudian, di bawah Konfigurasi domain, pilih konfigurasi domain.
 2. Di halaman detail konfigurasi Domain, pilih Edit. Kemudian, di bagian Pengaturan domain khusus - opsional, di bawah Pilih kebijakan keamanan, pilih kebijakan keamanan.
 3. Pilih Perbarui konfigurasi domain.

Untuk informasi selengkapnya, lihat [Membuat konfigurasi domain](#) dan [Mengelola konfigurasi domain](#).

Konfigurasi pengaturan TLS dalam konfigurasi domain (CLI)

Anda dapat menggunakan perintah [create-domain-configuration](#) dan [update-domain-configuration](#) CLI untuk mengonfigurasi pengaturan TLS Anda dalam konfigurasi domain.

1. Untuk menentukan pengaturan TLS menggunakan perintah [create-domain-configuration](#) CLI:

```
aws iot create-domain-configuration \  
  --domain-configuration-name domainConfigurationName \  
  --tls-config securityPolicy=IoTSecurityPolicy_TLS13_1_2_2022_10
```

Output dari perintah ini dapat terlihat seperti berikut:

```
{  
  "domainConfigurationName": "test",  
  "domainConfigurationArn": "arn:aws:iot:us-west-2:123456789012:domainconfiguration/  
test/34ga9"  
}
```

Jika Anda membuat konfigurasi domain baru tanpa menentukan kebijakan keamanan, nilai akan default ke: `IoTSecurityPolicy_TLS13_1_2_2022_10`.

2. Untuk menggambarkan pengaturan TLS menggunakan perintah [describe-domain-configuration](#) CLI:

```
aws iot describe-domain-configuration \
  --domain-configuration-name domainConfigurationName
```

Perintah ini dapat mengembalikan detail konfigurasi domain yang menyertakan pengaturan TLS seperti berikut:

```
{
  "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. Untuk memperbarui pengaturan TLS menggunakan perintah [update-domain-configuration](#) CLI:

```
aws iot update-domain-configuration \
  --domain-configuration-name domainConfigurationName \
  --tls-config securityPolicy=IoTSecurityPolicy_TLS13_1_2_2022_10
```

Output dari perintah ini dapat terlihat seperti berikut:

```
{
  "domainConfigurationName": "test",
  "domainConfigurationArn": "arn:aws:iot:us-west-2:123456789012:domainconfiguration/test/34ga9"
}
```

4. Untuk memperbarui pengaturan TLS untuk titik akhir ATS Anda, jalankan perintah CLI [update-domain-configuration](#). Nama konfigurasi domain untuk titik akhir ATS Anda adalah `iot:Data-ATS`.

```
aws iot update-domain-configuration \  
  --domain-configuration-name "iot:Data-ATS" \  
  --tls-config securityPolicy=IoTSecurityPolicy_TLS13_1_2_2022_10
```

Output dari perintah dapat terlihat seperti berikut:

```
{  
  "domainConfigurationName": "iot:Data-ATS",  
  "domainConfigurationArn": "arn:aws:iot:us-west-2:123456789012:domainconfiguration/  
iot:Data-ATS"  
}
```

Untuk informasi selengkapnya, lihat [CreateDomainConfiguration](#) dan [UpdateDomainConfiguration](#) di Referensi AWS API.

Konfigurasi sertifikat server untuk stapling OCSP

AWS IoT Core mendukung stapling [Online Certificate Status Protocol \(OCSP\)](#) untuk sertifikat server, juga dikenal sebagai stapling sertifikat server OCSP, atau stapling OCSP. Ini adalah mekanisme keamanan yang digunakan untuk memeriksa status pencabutan pada sertifikat server dalam jabat tangan Transport Layer Security (TLS). Stapling OCSP AWS IoT Core memungkinkan Anda menambahkan lapisan verifikasi tambahan ke validitas sertifikat server domain kustom Anda.

Anda dapat mengaktifkan sertifikat server OCSP stapling in AWS IoT Core untuk memeriksa validitas sertifikat dengan menanyakan responden OCSP secara berkala. Pengaturan stapling OCSP adalah bagian dari proses untuk membuat atau memperbarui konfigurasi domain dengan domain khusus. Stapling OCSP memeriksa status pencabutan pada sertifikat server secara terus menerus. Ini membantu memverifikasi bahwa sertifikat apa pun yang telah dicabut oleh CA tidak lagi dipercaya oleh klien yang terhubung ke domain kustom Anda. Untuk informasi selengkapnya, lihat [???](#).

Sertifikat server OCSP stapling menyediakan pemeriksaan status pencabutan waktu nyata, mengurangi latensi yang terkait dengan memeriksa status pencabutan, dan meningkatkan privasi dan keandalan koneksi aman. Untuk informasi lebih lanjut tentang manfaat menggunakan stapling OCSP, lihat [???](#)

Note

Fitur ini tidak tersedia di AWS GovCloud (US) Regions.

Dalam topik ini:

- [Apa itu OCSP?](#)
- [Cara kerja stapling OCSP](#)
- [Mengaktifkan sertifikat server OCSP stapling di AWS IoT Core](#)
- [Catatan penting untuk menggunakan sertifikat server OCSP stapling di AWS IoT Core](#)
- [Memecahkan masalah sertifikat server OCSP stapling di AWS IoT Core](#)

Apa itu OCSP?

Konsep utama

Konsep berikut memberikan rincian tentang OCSP dan konsep terkait.

OCSP

[OCSP](#) digunakan untuk memeriksa status pencabutan sertifikat selama jabat tangan Transport Layer Security (TLS). OCSP memungkinkan validasi sertifikat secara real-time. Ini menegaskan bahwa sertifikat belum dicabut atau kedaluwarsa sejak dikeluarkan. OCSP juga lebih skalabel dibandingkan dengan Daftar Pencabutan Sertifikat (CRL) tradisional. Respons OCSP lebih kecil dan dapat dihasilkan secara efisien, membuatnya lebih cocok untuk Infrastruktur Kunci Pribadi (PKI) skala besar.

Responden OCSP

Responder OCSP (juga dikenal sebagai server OCSP) menerima dan menanggapi permintaan OCSP dari klien yang berusaha memverifikasi status pencabutan sertifikat.

OCSP sisi klien

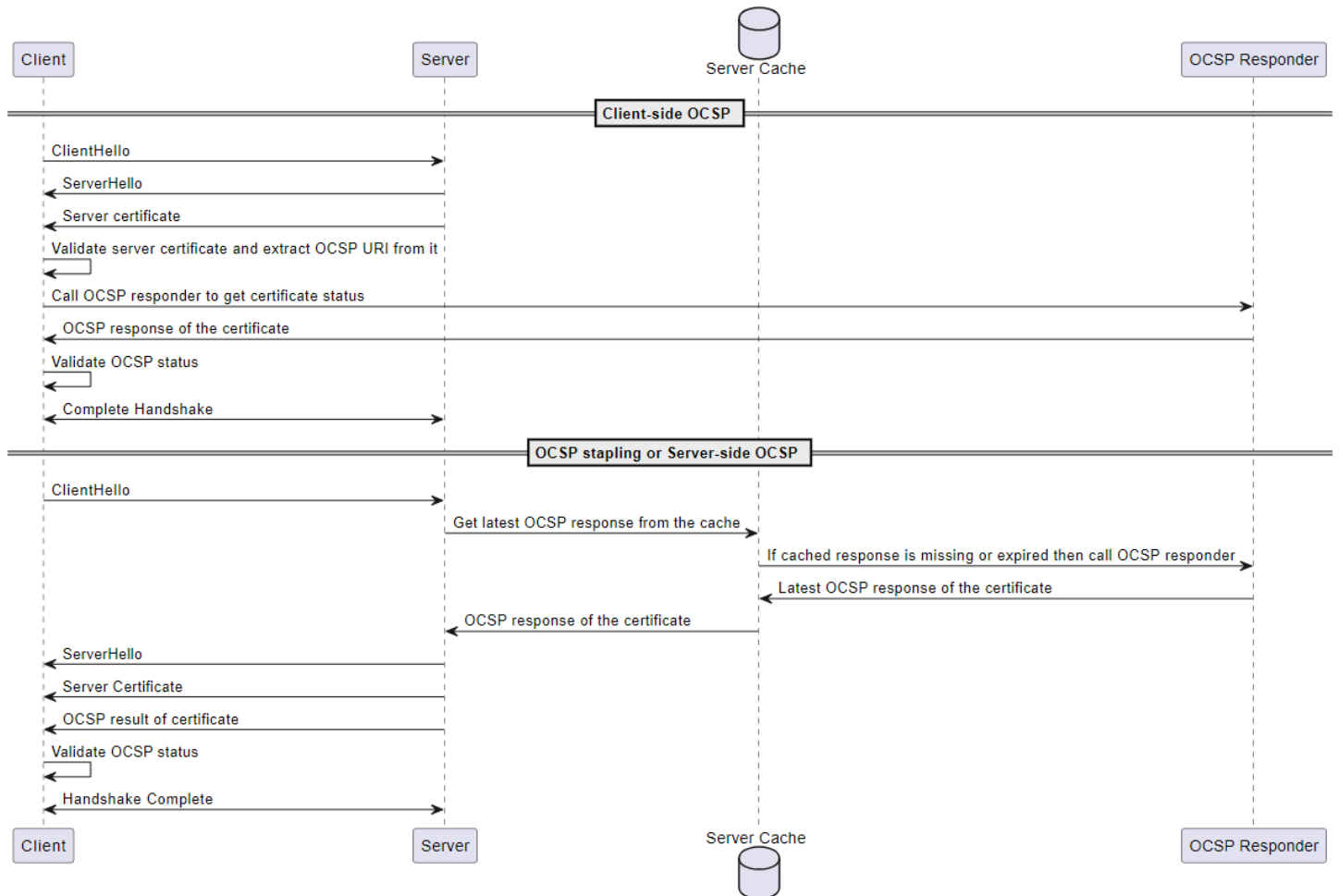
Di OCSP sisi klien, klien menggunakan OCSP untuk menghubungi responder OCSP untuk memeriksa status pencabutan sertifikat selama jabat tangan Transport Layer Security (TLS).

OCSP sisi server

Di OCSP sisi server (juga dikenal sebagai stapling OCSP), server diaktifkan (bukan klien) untuk membuat permintaan ke responder OCSP. Server menjepit respons OCSP ke sertifikat dan mengembalikannya ke klien selama jabat tangan TLS.

Diagram OCSP

Diagram berikut menggambarkan bagaimana OCSP sisi klien dan OCSP sisi server bekerja.



OCSP sisi klien

1. Klien mengirim `ClientHello` pesan untuk memulai jabat tangan TLS dengan server.
2. Server menerima pesan dan merespons dengan `ServerHello` pesan. Server juga mengirimkan sertifikat server ke klien.
3. Klien memvalidasi sertifikat server dan mengekstrak URI OCSP darinya.
4. Klien mengirimkan permintaan pemeriksaan pencabutan sertifikat ke responden OCSP.
5. Responden OCSP mengirimkan respons OCSP.

6. Klien memvalidasi status sertifikat dari respons OCSP.
7. Jabat tangan TLS selesai.

OCSP sisi server

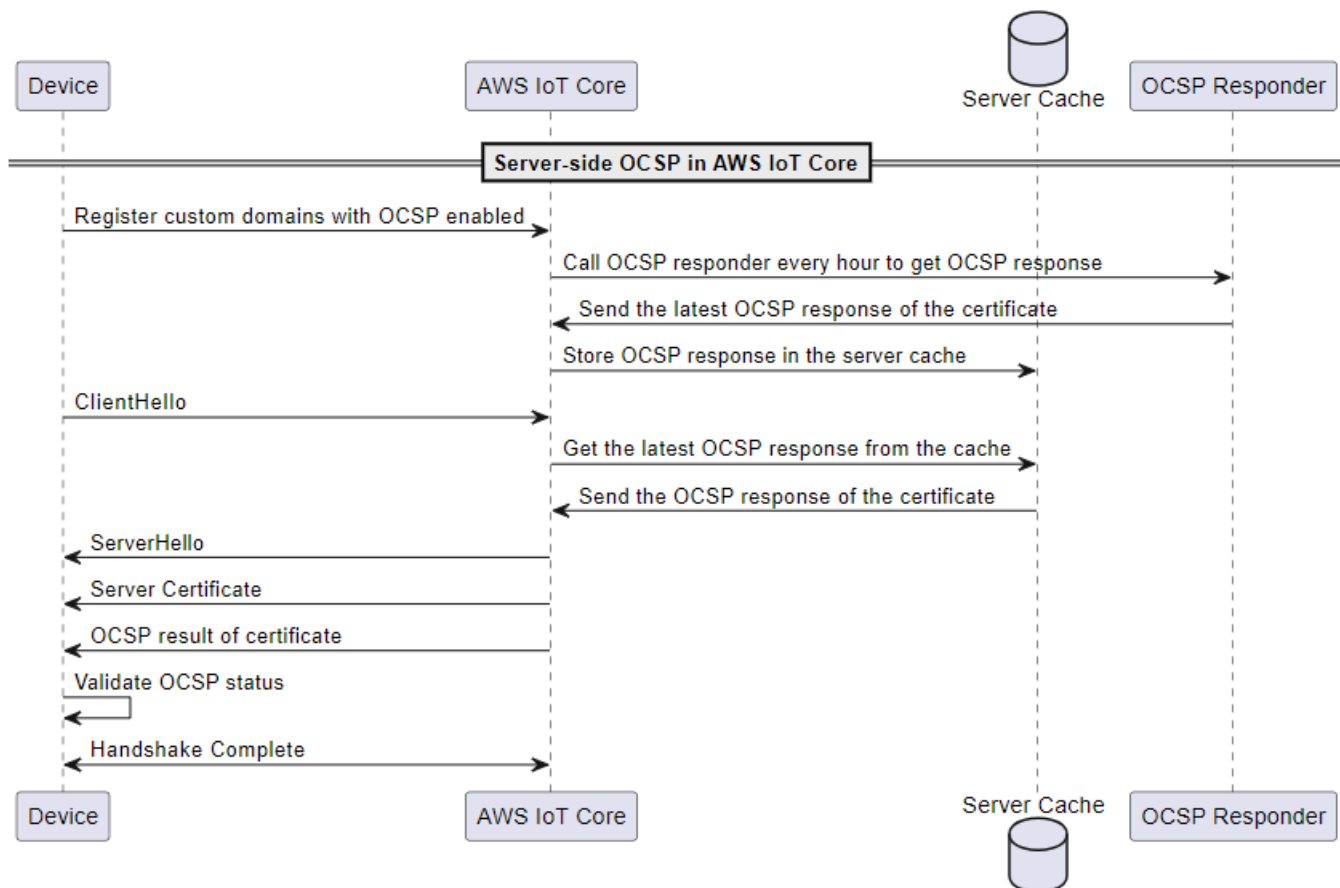
1. Klien mengirim `ClientHello` pesan untuk memulai jabat tangan TLS dengan server.
2. Server menerima pesan dan mendapatkan respons OCSP cache terbaru. Jika respons cache hilang atau kedaluwarsa, server akan memanggil responder OCSP untuk status sertifikat.
3. Responder OCSP mengirimkan respons OCSP ke server.
4. Server mengirim `ServerHello` pesan. Server juga mengirimkan sertifikat server dan status sertifikat ke klien.
5. Klien memvalidasi status sertifikat OCSP.
6. Jabat tangan TLS selesai.

Cara kerja stapling OCSP

Stapling OCSP digunakan selama jabat tangan Transport Layer Security (TLS) antara klien dan server untuk memeriksa status pencabutan sertifikat server. Server membuat permintaan OCSP ke responder OCSP dan staples tanggapan OCSP ke sertifikat yang dikembalikan ke klien. Dengan meminta server membuat permintaan ke responder OCSP, tanggapan dapat di-cache dan kemudian digunakan beberapa kali untuk banyak klien.

Bagaimana stapling OCSP bekerja di AWS IoT Core

Diagram berikut menunjukkan cara kerja stapling OCSP sisi server. AWS IoT Core



1. Perangkat harus terdaftar dengan domain khusus dengan stapling OCSP diaktifkan.
2. AWS IoT Core memanggil responden OCSP setiap jam untuk mendapatkan status sertifikat.
3. Responden OCSP menerima permintaan, mengirimkan respons OCSP terbaru, dan menyimpan respons OCSP yang di-cache.
4. Perangkat mengirimkan ClientHello pesan untuk memulai jabat tangan TLS dengan AWS IoT Core
5. AWS IoT Core mendapat respon OCSP terbaru dari cache server, yang merespons dengan respon OCSP sertifikat.
6. Server mengirim ServerHello pesan ke perangkat. Server juga mengirimkan sertifikat server dan status sertifikat ke klien.
7. Perangkat memvalidasi status sertifikat OCSP.
8. Jabat tangan TLS selesai.

Manfaat menggunakan stapling OCSP dibandingkan dengan pemeriksaan OCSP sisi klien

Beberapa keuntungan menggunakan sertifikat server OCSP stapling dirangkum sebagai berikut:

Privasi yang ditingkatkan

Tanpa stapling OCSP, perangkat klien dapat mengekspos informasi kepada responden OCSP pihak ketiga, yang berpotensi membahayakan privasi pengguna. Stapling OCSP mengurangi masalah ini dengan meminta server mendapatkan respons OCSP dan mengirimkannya langsung ke klien.

Peningkatan keandalan

Stapling OCSP dapat meningkatkan keandalan koneksi aman karena mengurangi risiko pemadaman server OCSP. Ketika tanggapan OCSP dijepit, server menyertakan respons terbaru dengan sertifikat. Ini agar klien memiliki akses ke status pencabutan meskipun responden OCSP sementara tidak tersedia. Stapling OCSP membantu mengurangi masalah ini karena server mengambil respons OCSP secara berkala dan menyertakan respons yang di-cache dalam jabat tangan TLS, mengurangi ketergantungan pada ketersediaan real-time responden OCSP.

Mengurangi beban server

Stapling OCSP menurunkan beban menanggapi permintaan OCSP dari responden OCSP ke server. Ini dapat membantu mendistribusikan beban secara lebih merata, membuat proses validasi sertifikat lebih efisien dan terukur.

Mengurangi latensi

Stapling OCSP mengurangi latensi yang terkait dengan memeriksa status pencabutan sertifikat selama jabat tangan TLS. Alih-alih klien harus menanyakan server OCSP secara terpisah, server mengirimkan permintaan dan melampirkan respons OCSP dengan sertifikat server selama jabat tangan.

Mengaktifkan sertifikat server OCSP stapling di AWS IoT Core

Untuk mengaktifkan sertifikat server OCSP stapling in AWS IoT Core, Anda harus membuat konfigurasi domain untuk domain kustom atau memperbarui konfigurasi domain kustom yang ada. Untuk informasi umum tentang membuat konfigurasi domain dengan domain kustom, lihat [???](#).

Gunakan petunjuk berikut untuk mengaktifkan stapling server OCSP menggunakan AWS Management Console atau. AWS CLI

Konsol

Untuk mengaktifkan stapling OCSP sertifikat server menggunakan AWS IoT konsol:

1. Pilih Pengaturan dari navigasi kiri menu, lalu pilih Buat konfigurasi domain atau konfigurasi domain yang ada untuk domain kustom.
2. Jika Anda memilih untuk membuat konfigurasi domain baru dari langkah sebelumnya, Anda akan melihat halaman Buat konfigurasi domain. Di bagian Properti konfigurasi Domain, pilih Domain kustom. Masukkan informasi untuk membuat konfigurasi domain.

Jika Anda memilih untuk memperbarui konfigurasi domain yang ada untuk domain kustom, Anda akan melihat halaman detail konfigurasi Domain. Pilih Edit.

3. Untuk mengaktifkan stapling server OCSP, pilih Aktifkan stapling sertifikat server OCSP di subbagian konfigurasi sertifikat Server.
4. Pilih Buat konfigurasi domain atau Perbarui konfigurasi domain.

AWS CLI

Untuk mengaktifkan stapling OCSP sertifikat server menggunakan: AWS CLI

1. Jika Anda membuat konfigurasi domain baru untuk domain kustom, perintah untuk mengaktifkan stapling server OCSP dapat terlihat seperti berikut:

```
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. Jika Anda memperbarui konfigurasi domain yang ada untuk domain kustom, perintah untuk mengaktifkan stapling server OCSP dapat terlihat seperti berikut:

```
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"
```

Untuk informasi selengkapnya, lihat [CreateDomainConfiguration](#) dan [UpdateDomainConfiguration](#) dari Referensi AWS IoT API.

Catatan penting untuk menggunakan sertifikat server OCSP stapling di AWS IoT Core

Saat Anda menggunakan sertifikat server OCSP AWS IoT Core, ingatlah hal berikut:

1. AWS IoT Core hanya mendukung responden OCSP yang dapat dijangkau melalui alamat IPv4 publik.
2. Fitur stapling OCSP di AWS IoT Core tidak mendukung responden resmi. Semua tanggapan OCSP harus ditandatangani oleh CA yang menandatangani sertifikat, dan CA harus menjadi bagian dari rantai sertifikat domain kustom.
3. Fitur stapling OCSP di AWS IoT Core tidak mendukung domain kustom yang dibuat menggunakan sertifikat yang ditandatangani sendiri.
4. AWS IoT Core memanggil responden OCSP setiap jam dan menyimpan respons dalam cache. Jika panggilan ke responden gagal, AWS IoT Core akan menjepit respons valid terbaru.
5. Jika `nextUpdateTime` tidak lagi valid, AWS IoT Core akan menghapus respons dari cache, dan jabat tangan TLS tidak akan menyertakan data respons OCSP sampai panggilan berhasil berikutnya ke responder OCSP. Hal ini dapat terjadi ketika respon cache telah kedaluwarsa sebelum server mendapat respon yang valid dari responder OCSP. Nilai `nextUpdateTime` menunjukkan bahwa respons OCSP akan valid hingga saat ini. Untuk informasi selengkapnya tentang `nextUpdateTime`, lihat [???](#).
6. Terkadang, AWS IoT Core gagal menerima respons OCSP atau menghapus respons OCSP yang ada karena sudah kedaluwarsa. Jika situasi seperti ini terjadi, AWS IoT Core akan terus menggunakan sertifikat server yang disediakan oleh domain kustom tanpa respon OCSP.
7. Ukuran respons OCSP tidak boleh melebihi 4 KiB.

Memecahkan masalah sertifikat server OCSP stapling di AWS IoT Core

AWS IoT Core memancarkan `RetrieveOCSPStapleData.Success` metrik dan entri `RetrieveOCSPStapleData` log ke CloudWatch Metrik dan entri log dapat membantu mendeteksi masalah yang terkait dengan pengambilan respons OCSP. Lihat informasi yang lebih lengkap di [???](#) dan [???](#).

Menghubungkan ke titik AWS IoT akhir FIPS

AWS IoT menyediakan titik akhir yang mendukung [Federal Information Processing Standard \(FIPS\) 140-2](#). Titik akhir yang sesuai dengan FIPS berbeda dari titik akhir standar. AWS Untuk berinteraksi dengan AWS IoT cara yang sesuai dengan FIPS, Anda harus menggunakan titik akhir yang dijelaskan di bawah ini dengan klien yang sesuai dengan FIPS Anda. AWS IoT Konsol tidak sesuai dengan FIPS.

Bagian berikut menjelaskan cara mengakses AWS IoT titik akhir yang sesuai dengan FIPS dengan menggunakan REST API, SDK, atau. AWS CLI

Topik

- [AWS IoT Core- titik akhir bidang kontrol](#)
- [AWS IoT Core- titik akhir bidang data](#)
- [AWS IoT Device Management- titik akhir data pekerjaan](#)
- [AWS IoT Device Management- Titik akhir Fleet Hub](#)
- [AWS IoT Device Management- titik akhir terowongan aman](#)

AWS IoT Core- titik akhir bidang kontrol

[Titik akhir bidang kontrol yang sesuai AWS IoT Core dengan FIPS yang mendukung AWS IoT operasi dan perintah CLI terkait tercantum dalam Titik Akhir FIPS menurut Layanan](#). Di [FIPS Endpoints by Service](#), temukan layanan AWS IoT Core- control plane, dan cari titik akhir untuk Anda. Wilayah AWS

Untuk menggunakan titik akhir yang sesuai dengan FIPS saat Anda mengakses [AWS IoT operasi](#), gunakan AWS SDK atau REST API dengan titik akhir yang sesuai untuk Anda. Wilayah AWS

Untuk menggunakan titik akhir yang sesuai dengan FIPS saat Anda menjalankan [perintah aws iot CLI](#), tambahkan `--endpoint` parameter dengan titik akhir yang sesuai untuk perintah Anda. Wilayah AWS

AWS IoT Core- titik akhir bidang data

Titik akhir bidang data yang sesuai AWS IoT Core dengan FIPS tercantum dalam Titik Akhir [FIPS](#) menurut Layanan. Di [FIPS Endpoints by Service](#), temukan layanan pesawat data, dan cari titik akhir untuk Anda. AWS IoT Core Wilayah AWS

Anda dapat menggunakan titik akhir yang sesuai dengan FIPS untuk klien yang sesuai Wilayah AWS dengan FIPS dengan menggunakan AWS IoT Device SDK dan menyediakan titik akhir ke fungsi koneksi SDK sebagai pengganti titik akhir bidang data default akun Anda. AWS IoT Core Fungsi koneksi khusus untuk AWS IoT Device SDK. Untuk contoh fungsi koneksi, lihat fungsi [Connection di AWS IoT Device SDK for Python](#).

Note

AWS IoT tidak mendukung Akun AWS-specific AWS IoT Core- titik akhir bidang data yang sesuai dengan FIPS. Fitur layanan yang memerlukan titik Akun AWS akhir spesifik di [Server Name Indication \(SNI\)](#) tidak dapat digunakan. [Sesuai FIPS AWS IoT Core- titik akhir bidang data tidak dapat mendukung Sertifikat Pendaftaran Multi-Akun, Domain Kustom, Otorisasi Kustom, dan Titik Akhir yang Dapat Dikonfigurasi \(termasuk kebijakan TLS yang didukung\)](#).

AWS IoT Device Management- titik akhir data pekerjaan

Titik akhir data pekerjaan yang sesuai AWS IoT Device Management dengan FIPS tercantum di Titik Akhir [FIPS](#) menurut Layanan. Di [FIPS Endpoints by Service](#), temukan layanan data pekerjaan, dan cari titik akhir untuk Anda. AWS IoT Device Management Wilayah AWS

Untuk menggunakan titik akhir data pekerjaan yang sesuai AWS IoT Device Management dengan FIPS saat Anda menjalankan [perintah aws iot-jobs-data CLI](#), tambahkan --endpoint parameter dengan titik akhir yang sesuai untuk perintah Anda. Wilayah AWS Anda juga dapat menggunakan REST API dengan endpoint ini.

Anda dapat menggunakan titik akhir yang sesuai dengan FIPS untuk klien yang sesuai Wilayah AWS dengan FIPS dengan menggunakan AWS IoT Device SDK dan menyediakan titik akhir ke fungsi koneksi SDK sebagai pengganti titik akhir data pekerjaan default akun Anda. AWS IoT Device Management Fungsi koneksi khusus untuk AWS IoT Device SDK. Untuk contoh fungsi koneksi, lihat fungsi [Connection di AWS IoT Device SDK for Python](#).

AWS IoT Device Management- Titik akhir Fleet Hub

[Titik akhir Fleet Hub yang sesuai AWS IoT Device Management dengan FIPS untuk digunakan dengan perintah CLIFleet Hub for AWS IoT Device Management tercantum dalam Titik Akhir FIPS menurut Layanan](#). Di [FIPS Endpoints by Service](#), temukan layanan AWS IoT Device Management- Fleet Hub, dan cari titik akhir untuk Anda. Wilayah AWS

Untuk menggunakan titik akhir FIPS compliant AWS IoT Device Management- Fleet Hub saat Anda menjalankan [perintah `aws iotfleethub CLI`](#), tambahkan `--endpoint` parameter dengan titik akhir yang sesuai untuk perintah Anda. Wilayah AWS Anda juga dapat menggunakan REST API dengan endpoint ini.

AWS IoT Device Management- titik akhir terowongan aman

[Titik akhir tunneling aman yang sesuai AWS IoT Device Management dengan FIPS untuk API tunneling AWS IoT aman dan perintah CLI yang sesuai tercantum di Titik Akhir FIPS menurut Layanan](#). Di [FIPS Endpoints by Service](#), temukan layanan tunneling yang aman, dan cari titik akhir untuk Anda. AWS IoT Device Management Wilayah AWS

Untuk menggunakan titik akhir tunneling yang sesuai AWS IoT Device Management dengan FIPS saat Anda menjalankan perintah [`aws iotsecuretunnelingCLI`](#), tambahkan `--endpoint parameter` dengan titik akhir yang sesuai untuk perintah Anda. Wilayah AWS Anda juga dapat menggunakan REST API dengan endpoint ini.

Tutorial AWS IoT

Parameter AWS IoT tutorial dibagi menjadi dua jalur pembelajaran untuk mendukung dua tujuan yang berbeda. Pilih jalur pembelajaran terbaik untuk tujuan Anda.

- Anda ingin membangun proof-of-concept untuk menguji atau menunjukkan AWS IoT solusi

Untuk menunjukkan tugas dan aplikasi IoT umum menggunakan AWS IoT Perangkat Client pada perangkat Anda, ikuti [the section called “Membangun demo dengan AWS IoT Perangkat”](#) jalur belajar. Parameter AWS IoT Device Client menyediakan perangkat lunak perangkat yang dengannya Anda dapat menerapkan sumber daya cloud Anda sendiri untuk mendemonstrasikan end-to-end solusi dengan pengembangan minimum.

Untuk informasi tentang AWS IoT Perangkat Client, lihat [AWS IoT Perangkat](#).

- Anda ingin mempelajari cara membangun perangkat lunak produksi untuk menyebarkan solusi Anda

Untuk membuat perangkat lunak solusi Anda sendiri yang memenuhi persyaratan spesifik Anda menggunakan AWS IoT Perangkat SDK, ikuti [the section called “Membangun solusi dengan AWS IoT Perangkat SDK”](#) jalur belajar.

Untuk informasi tentang AWS IoT SDK perangkat, lihat [???](#). Untuk informasi tentang AWS SDK, lihat [Alat untuk Dibangun AWS](#).

AWS IoT pilihan jalur pembelajaran tutorial

- [Membangun demo dengan AWS IoT Perangkat](#)
- [Membangun solusi dengan AWS IoT Perangkat SDK](#)

Membangun demo dengan AWS IoT Perangkat

Tutorial di jalur pembelajaran ini memandu Anda melalui langkah-langkah untuk mengembangkan perangkat lunak demonstrasi dengan menggunakan AWS IoT Perangkat. Parameter AWS IoT Device Client menyediakan perangkat lunak yang berjalan pada perangkat IoT Anda untuk menguji dan menunjukkan aspek solusi IoT yang dibangun di atas AWS IoT.

Tujuan dari tutorial ini adalah untuk memfasilitasi eksplorasi dan eksperimen sehingga Anda dapat merasa yakin bahwa AWS IoT mendukung solusi Anda sebelum mengembangkan perangkat lunak perangkat Anda.

Apa yang akan Anda pelajari dalam tutorial ini:

- Cara menyiapkan Raspberry Pi untuk digunakan sebagai perangkat IoT dengan AWS IoT
- Bagaimana cara mendemonstrasikan AWS IoT fitur dengan menggunakan AWS IoT Perangkat Client pada perangkat Anda

Di jalur pembelajaran ini, Anda akan menginstal AWS IoT Perangkat Client pada Raspberry Pi Anda sendiri dan membuat AWS IoT sumber daya di cloud untuk menunjukkan ide-ide solusi IoT. Sementara tutorial di jalur pembelajaran ini menunjukkan fitur dengan menggunakan Raspberry Pi, mereka menjelaskan tujuan dan prosedur untuk membantu Anda menyesuaikannya dengan perangkat lain.

Prasyarat untuk membangun demo dengan AWS IoT Perangkat

Bagian ini menjelaskan apa yang harus Anda miliki sebelum memulai tutorial di jalur pembelajaran ini.

Untuk menyelesaikan tutorial di jalur pembelajaran ini, Anda memerlukan:

- Sesi Akun AWS

Anda dapat menggunakan Akun AWS, jika Anda memilikinya, tetapi Anda mungkin perlu menambahkan peran atau izin tambahan untuk menggunakan AWS IoT fitur tutorial ini menggunakan.

Jika Anda perlu membuat Akun AWS, lihat [the section called “Siapkan Anda Akun AWS”](#).

- Raspberry Pi atau perangkat IoT yang kompatibel

Tutorial menggunakan [Raspberry Pi](#) karena datang dalam faktor bentuk yang berbeda, ada di mana-mana, dan itu adalah perangkat demonstrasi yang relatif murah. Tutorial telah diuji pada [Raspberry Pi 3 Model B +](#), yang [Raspberry Pi 4 Model B](#), dan pada instans Amazon EC2 yang menjalankan Ubuntu Server 20.04 LTS (HVM). Untuk menggunakan AWS CLI dan menjalankan perintah, Kami menyarankan Anda menggunakan versi terbaru dari OS Raspberry Pi ([OS Raspberry \(64-bit\)](#) atau OS Lite). Versi OS sebelumnya mungkin bekerja, tetapi kami belum mengujinya.

Note

Tutorial menjelaskan tujuan dari setiap langkah untuk membantu Anda menyesuaikannya dengan perangkat keras IoT yang belum kami coba; Namun, mereka tidak secara khusus menggambarkan cara menyesuaikannya dengan perangkat lain.

- Keakraban dengan sistem operasi perangkat IoT

Langkah-langkah dalam tutorial ini menganggap bahwa Anda sudah familiar dengan menggunakan perintah Linux dasar dan operasi dari antarmuka baris perintah didukung oleh Raspberry Pi. Jika Anda tidak terbiasa dengan operasi ini, Anda mungkin ingin memberi diri Anda lebih banyak waktu untuk menyelesaikan tutorial.

Untuk menyelesaikan tutorial ini, Anda seharusnya sudah mengerti bagaimana:

- Aman melakukan operasi perangkat dasar seperti merakit dan menghubungkan komponen, menghubungkan perangkat ke sumber daya yang diperlukan, dan menginstal dan menghapus kartu memori.
 - Unggah dan unduh perangkat lunak dan file sistem ke perangkat. Jika perangkat Anda tidak menggunakan perangkat penyimpanan yang dapat dilepas, seperti kartu microSD, Anda harus mengetahui cara menyambung ke perangkat Anda dan mengunggah dan mengunduh perangkat lunak dan file sistem ke perangkat.
 - Connect perangkat ke jaringan yang Anda rencanakan untuk menggunakannya.
 - Connect ke perangkat Anda dari komputer lain menggunakan terminal SSH atau program serupa.
 - Gunakan antarmuka baris perintah untuk membuat, menyalin, memindahkan, mengganti nama, dan mengatur izin file dan direktori pada perangkat.
 - Instal program baru di perangkat.
 - Mentransfer file ke dan dari perangkat Anda menggunakan alat seperti FTP atau SCP.
- Lingkungan pengembangan dan pengujian untuk solusi IoT Anda

Tutorial menggambarkan perangkat lunak dan perangkat keras yang diperlukan; Namun, tutorial menganggap bahwa Anda akan dapat melakukan operasi yang mungkin tidak dijelaskan secara eksplisit. Contoh perangkat keras dan operasi tersebut meliputi:

- Komputer host lokal untuk mengunduh dan menyimpan file

Untuk Raspberry Pi, ini biasanya komputer pribadi atau laptop yang dapat membaca dan menulis ke kartu memori microSD. Komputer host lokal harus:

- Terhubung ke Internet.
- Memiliki [AWS CLI](#) diinstal dan dikonfigurasi.
- Memiliki browser web yang mendukung AWS konsol.
- Cara untuk menghubungkan komputer host lokal ke perangkat Anda untuk berkomunikasi dengannya, memasukkan perintah, dan mentransfer file

Pada Raspberry Pi, ini sering dilakukan dengan menggunakan SSH dan SCP dari komputer host lokal.

- Monitor dan keyboard untuk terhubung ke perangkat IoT Anda

Ini dapat membantu, tetapi tidak diperlukan untuk menyelesaikan tutorial.

- Cara untuk komputer host lokal Anda dan perangkat IoT Anda untuk terhubung ke internet

Ini bisa berupa kabel atau koneksi jaringan nirkabel ke router atau gateway yang terhubung ke internet. Host lokal juga harus dapat terhubung ke Raspberry Pi. Ini mungkin mengharuskan mereka berada di jaringan area lokal yang sama. Tutorial tidak dapat menunjukkan cara mengatur ini untuk konfigurasi perangkat atau perangkat tertentu, tetapi mereka menunjukkan bagaimana Anda dapat menguji konektivitas ini.

- Akses ke router jaringan area lokal Anda untuk melihat perangkat yang terhubung

Untuk menyelesaikan tutorial di jalur pembelajaran ini, Anda harus dapat menemukan alamat IP perangkat IoT Anda.

Pada jaringan area lokal, ini dapat dilakukan dengan mengakses antarmuka admin router jaringan yang terhubung ke perangkat Anda. Jika Anda dapat menetapkan alamat IP tetap untuk perangkat Anda di router, Anda dapat menyederhanakan koneksi ulang setelah setiap kali perangkat dimulai ulang.

Jika Anda memiliki keyboard dan monitor yang terpasang pada perangkat, `ifconfig` dapat menampilkan alamat IP perangkat.

Jika tidak ada pilihan ini, Anda harus menemukan cara untuk mengidentifikasi alamat IP perangkat setelah setiap kali dimulai ulang.

Setelah Anda memiliki semua materi Anda, lanjutkan [the section called “Mempersiapkan perangkat Anda untuk AWS IoT Klien perangkat”](#).

Tutorial di jalur pembelajaran ini

- [Tutorial: Mempersiapkan perangkat Anda untuk AWS IoT Klien perangkat](#)
- [Tutorial: Menginstal dan mengkonfigurasi AWS IoT Perangkat klien](#)
- [Tutorial: Menunjukkan komunikasi pesan MQTT dengan AWS IoT Klien perangkat](#)
- [Tutorial: Menunjukkan tindakan jarak jauh \(pekerjaan\) dengan AWS IoT Perangkat perangkat perangkat perangkat perangkat](#)
- [Tutorial: Membersihkan setelah menjalankan tutorial AWS IoT Device Client](#)

Tutorial: Mempersiapkan perangkat Anda untuk AWS IoT Klien perangkat

Tutorial ini memandu Anda melalui inisialisasi Raspberry Pi Anda untuk mempersiapkannya untuk tutorial berikutnya di jalur pembelajaran ini.

Tujuan dari tutorial ini adalah untuk menginstal versi sistem operasi perangkat saat ini dan memastikan bahwa Anda dapat berkomunikasi dengan perangkat Anda dalam konteks lingkungan pengembangan Anda.

Untuk memulai tutorial ini:

- Memiliki item yang tercantum dalam [the section called “Prasyarat untuk membangun demo dengan AWS IoT Perangkat”](#) tersedia dan siap untuk digunakan.

Tutorial ini membutuhkan waktu sekitar 90 menit untuk menyelesaikannya.

Ketika Anda menyelesaikan tutorial ini:

- Perangkat IoT Anda akan memiliki sistem operasi terbaru.
- Perangkat IoT Anda akan memiliki perangkat lunak tambahan yang dibutuhkan untuk tutorial berikutnya.
- Anda akan tahu bahwa perangkat Anda memiliki konektivitas ke internet.
- Anda akan menginstal sertifikat yang diperlukan pada perangkat Anda.

Setelah Anda menyelesaikan tutorial ini, tutorial berikutnya mempersiapkan perangkat Anda untuk demo yang menggunakan AWS IoT CLI perangkat.

Prosedur dalam tutorial ini

- [Langkah 1: Instal dan perbarui sistem operasi perangkat](#)
- [Langkah 2: Instal dan verifikasi perangkat lunak yang diperlukan di perangkat Anda](#)
- [Langkah 3: Uji perangkat Anda dan simpan sertifikat Amazon CA](#)

Langkah 1: Instal dan perbarui sistem operasi perangkat

Prosedur di bagian ini menjelaskan cara menginisialisasi kartu microSD yang digunakan Raspberry Pi untuk drive sistem. Kartu microSD Raspberry Pi berisi perangkat lunak sistem operasi (OS) serta ruang untuk penyimpanan file aplikasinya. Jika Anda tidak menggunakan Raspberry Pi, ikuti petunjuk perangkat untuk menginstal dan memperbarui perangkat lunak sistem operasi perangkat.

Setelah Anda menyelesaikan bagian ini, Anda harus dapat memulai perangkat IoT Anda dan menghubungkannya dari program terminal di komputer host lokal Anda.

Peralatan yang dibutuhkan:

- Lingkungan pengembangan dan pengujian lokal Anda
- Raspberry Pi itu atau perangkat IoT Anda, yang dapat terhubung ke internet
- Kartu memori microSD dengan kapasitas minimal 8 GB atau penyimpanan yang cukup untuk OS dan perangkat lunak yang dibutuhkan.

Note

Saat memilih kartu microSD untuk latihan ini, pilih salah satu yang sebesar yang diperlukan namun sekecil mungkin.

Kartu SD kecil akan lebih cepat untuk mencadangkan dan memperbarui. Pada Raspberry Pi, Anda tidak memerlukan lebih dari kartu microSD 8-GB untuk tutorial ini. Jika Anda membutuhkan lebih banyak ruang untuk aplikasi spesifik Anda, file gambar yang lebih kecil yang Anda simpan dalam tutorial ini dapat mengubah ukuran sistem file pada kartu yang lebih besar untuk menggunakan semua ruang yang didukung dari kartu yang Anda pilih.

Peralatan opsional:


- Keyboard USB yang terhubung ke Raspberry Pi
- Monitor HDMI dan kabel untuk menghubungkan monitor ke Raspberry Pi

Prosedur di bagian ini:

- [Memuat sistem operasi perangkat ke kartu microSD](#)
- [Mulai perangkat IoT Anda dengan sistem operasi baru](#)
- [Connect komputer host lokal ke perangkat](#)

Memuat sistem operasi perangkat ke kartu microSD

Prosedur ini menggunakan komputer host lokal untuk memuat sistem operasi perangkat ke kartu microSD.

 Note

Jika perangkat Anda tidak menggunakan media penyimpanan yang dapat dilepas untuk sistem operasinya, instal sistem operasi menggunakan prosedur untuk perangkat tersebut dan lanjutkan [the section called “Mulai perangkat IoT Anda dengan sistem operasi baru”](#).

Untuk menginstal sistem operasi pada Raspberry Pi Anda

1. Di komputer host lokal Anda, unduh dan unzip gambar sistem operasi Raspberry Pi yang ingin Anda gunakan. Versi terbaru tersedia dari <https://www.raspberrypi.com/software/operating-systems/>

Memilih versi Raspberry Pi OS

Tutorial ini menggunakan Raspberry Pi OS versi karena itu adalah versi terkecil yang mendukung tutorial ini di jalur pembelajaran ini. Versi OS Raspberry Pi ini hanya memiliki antarmuka baris perintah dan tidak memiliki antarmuka pengguna grafis. Versi OS Raspberry Pi terbaru dengan antarmuka pengguna grafis juga akan bekerja dengan tutorial ini; Namun, prosedur yang dijelaskan dalam jalur pembelajaran ini hanya menggunakan antarmuka baris perintah untuk melakukan operasi pada Raspberry Pi.

2. Masukkan kartu microSD Anda ke komputer host lokal.

3. Menggunakan alat pencitraan kartu SD, tulis file gambar OS unzip ke kartu microSD.
4. Setelah menulis gambar Raspberry Pi OS ke kartu microSD:
 - a. Buka partisi BOOT pada kartu microSD di jendela baris perintah atau jendela file explorer.
 - b. Di partisi BOOT kartu microSD, di direktori root, buat file kosong bernama `sstana` tanpa ekstensi file dan tidak ada konten. Ini memberitahu Raspberry Pi untuk mengaktifkan komunikasi SSH pertama kali dimulai.
5. Keluarkan kartu microSD dan keluarkan dengan aman dari komputer host lokal.

Kartu microSD Anda siap [the section called “Mulai perangkat IoT Anda dengan sistem operasi baru”](#).

Mulai perangkat IoT Anda dengan sistem operasi baru

Prosedur ini menginstal kartu microSD dan memulai Raspberry Pi Anda untuk pertama kalinya menggunakan sistem operasi download.

Untuk memulai perangkat IoT Anda dengan sistem operasi baru

1. Dengan daya terputus dari perangkat, masukkan kartu microSD dari langkah sebelumnya, [the section called “Memuat sistem operasi perangkat ke kartu microSD”](#), ke Raspberry Pi.
2. Connect perangkat ke jaringan kabel.
3. Tutorial ini akan berinteraksi dengan Raspberry Pi Anda dari komputer host lokal Anda menggunakan terminal SSH.

Jika Anda juga ingin berinteraksi dengan perangkat secara langsung, Anda dapat:

- a. Connect monitor HDMI untuk menonton pesan konsol Raspberry Pi sebelum Anda dapat menghubungkan jendela terminal pada komputer host lokal Anda ke Raspberry Pi Anda.
 - b. Connect keyboard USB untuk itu jika Anda ingin berinteraksi langsung dengan Raspberry Pi.
4. Connect kekuatan ke Raspberry Pi dan tunggu sekitar satu menit untuk menginisialisasi.

Jika Anda memiliki monitor yang terhubung ke Raspberry Pi Anda, Anda dapat menonton proses start-up di atasnya.

5. Cari tahu alamat IP perangkat Anda:
 - Jika Anda menghubungkan monitor HDMI ke Raspberry Pi, alamat IP muncul di pesan yang ditampilkan pada monitor

- Jika Anda memiliki akses ke router Raspberry Pi Anda terhubung ke, Anda dapat melihat alamatnya di antarmuka admin router.

Setelah Anda memiliki alamat IP Raspberry Pi Anda, Anda siap [the section called “Connect komputer host lokal ke perangkat”](#).


Connect komputer host lokal ke perangkat

Prosedur ini menggunakan program terminal pada komputer host lokal Anda untuk terhubung ke Raspberry Pi Anda dan mengubah password default.

Untuk menghubungkan komputer host lokal ke perangkat

1. Pada komputer host lokal Anda, buka program terminal SSH:

- Windows: PuTTY
- Linux/macOS: Terminal

 Note

PuTTY tidak diinstal secara otomatis pada Windows. Jika tidak ada di komputer Anda, Anda mungkin perlu mengunduh dan menginstalnya.

2. Connect program terminal ke alamat IP Raspberry Pi Anda dan masuk menggunakan kredensi defaultnya.

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

3. Setelah Anda masuk ke Raspberry Pi Anda, ubah kata sandi untuk `pi` pengguna.

```
passwd
```

Ikuti petunjuk untuk mengubah kata sandi.

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

```
passwd: password updated successfully
```

Setelah Anda memiliki prompt baris perintah Raspberry Pi di jendela terminal dan mengubah kata sandinya, Anda siap untuk melanjutkan [the section called “Langkah 2: Instal dan verifikasi perangkat lunak yang diperlukan di perangkat Anda”](#).

Langkah 2: Instal dan verifikasi perangkat lunak yang diperlukan di perangkat Anda

Prosedur di bagian ini berlanjut dari [bagian sebelumnya](#) untuk membawa sistem operasi Raspberry Pi Anda up to date dan menginstal perangkat lunak pada Raspberry Pi yang akan digunakan di bagian berikutnya untuk membangun dan menginstal AWS IoT CLI perangkat.

Setelah Anda menyelesaikan bagian ini, Raspberry Pi Anda akan memiliki sistem operasi up-to-date, perangkat lunak yang diperlukan oleh tutorial di jalur pembelajaran ini, dan itu akan dikonfigurasi untuk lokasi Anda.

Peralatan yang dibutuhkan:

- Pengembangan lokal Anda dan lingkungan pengujian dari [bagian sebelumnya](#)
- Raspberry Pi yang Anda gunakan [bagian sebelumnya](#)
- Kartu memori microSD dari [bagian sebelumnya](#)

Note

Raspberry Pi Model 3+ dan Raspberry Pi Model 4 dapat melakukan semua perintah yang dijelaskan dalam jalur pembelajaran ini. Jika perangkat IoT Anda tidak dapat mengkompilasi perangkat lunak atau menjalankan AWS Command Line Interface, Anda mungkin perlu menginstal kompiler yang diperlukan pada komputer host lokal Anda untuk membangun perangkat lunak dan kemudian mentransfernya ke perangkat IoT Anda. Untuk informasi selengkapnya tentang cara menginstal dan membangun perangkat lunak untuk perangkat lunak Anda, lihat dokumentasi perangkat lunak perangkat lunak Anda.

Prosedur di bagian ini:

- [Perbarui perangkat lunak sistem operasi](#)
- [Menginstal aplikasi dan perpustakaan yang diperlukan](#)
- [\(Opsional\) Simpan gambar kartu microSD](#)

Perbarui perangkat lunak sistem operasi

Prosedur ini memperbarui perangkat lunak sistem operasi.

Untuk memperbarui perangkat lunak sistem operasi pada Raspberry Pi

Lakukan langkah-langkah ini di jendela terminal komputer host lokal Anda.

1. Masukkan perintah ini untuk memperbarui perangkat lunak sistem pada Raspberry Pi Anda.

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

2. Perbarui pengaturan lokal dan zona waktu Raspberry Pi (opsional).

Masukkan perintah ini untuk memperbarui pengaturan lokal dan zona waktu perangkat.

```
sudo raspi-config
```

- a. Untuk mengatur lokal perangkat:

- i. Di Alat Konfigurasi Perangkat Lunak Raspberry Pi (raspi-config) layar, pilih opsi 5.

5 Localisation Options Configure language and regional settings

Menggunakan Tab kunci untuk pindah ke <Select>, dan kemudian tekan space bar.

- ii. Pada menu opsi lokalisasi, pilih opsi L1.

L1 Locale Configure language and regional settings

Menggunakan Tab kunci untuk pindah ke <Select>, dan kemudian tekan space bar.

- iii. Dalam daftar pilihan lokal, pilih lokal yang ingin Anda instal pada Raspberry Pi Anda dengan menggunakan tombol panah untuk menggulir dan space bar untuk menandai orang-orang yang Anda inginkan.

Di Amerika Serikat, **en_US.UTF-8** adalah salah satu yang baik untuk memilih.

- iv. Setelah memilih lokal untuk perangkat Anda, gunakan Tab kunci untuk memilih <OK>, lalu tekan space bar untuk menampilkan Mengkonfigurasi lokal Halaman konfirmasi.

- b. Untuk mengatur zona waktu perangkat:

- i. Diraspi-configlayar, pilih opsi5.

5 Localisation Options Configure language and regional settings

MenggunakanTabkunci untuk pindah ke<Select>,dan kemudian tekanspace bar.

- ii. Di menu opsi lokalisasi, gunakan tombol panah untuk memilih opsiL2:

L2 time zone Configure time zone

MenggunakanTabkunci untuk pindah ke<Select>,dan kemudian tekanspace bar.

- iii. DiMengonfigurasi tzdatamenu, pilih area geografis Anda dari daftar.

MenggunakanTabkunci untuk pindah ke<OK>, lalu tekanspace bar.

- iv. Dalam daftar kota, gunakan tombol panah untuk memilih kota di zona waktu Anda.

Untuk mengatur zona waktu, gunakanTabkunci untuk pindah ke<OK>, lalu tekanspace bar.

- c. Setelah selesai memperbarui pengaturan, gunakanTabkunci untuk pindah ke<Finish>, lalu tekanspace baruntuk menutupraspi-configaplikasi.

3. Masukkan perintah ini untuk me-restart Raspberry Pi Anda.

```
sudo shutdown -r 0
```

4. Tunggu sampai Raspberry Pi Anda restart.
5. Setelah Raspberry Pi Anda dimulai ulang, sambungkan kembali jendela terminal di komputer host lokal Anda ke Raspberry Pi Anda.

Perangkat lunak sistem Raspberry Pi Anda sekarang dikonfigurasi dan Anda siap untuk melanjutkan[the section called “Menginstal aplikasi dan perpustakaan yang diperlukan”](#).

Menginstal aplikasi dan perpustakaan yang diperlukan

Prosedur ini menginstal perangkat lunak aplikasi dan perpustakaan yang digunakan oleh tutorial berikutnya.

Jika Anda menggunakan Raspberry Pi, atau jika Anda dapat mengkompilasi perangkat lunak yang diperlukan pada perangkat IoT Anda, lakukan langkah-langkah ini di jendela terminal pada komputer host lokal Anda. Jika Anda harus mengkompilasi perangkat lunak untuk perangkat IoT Anda di

komputer host lokal, tinjau dokumentasi perangkat lunak untuk perangkat IoT Anda untuk informasi tentang cara melakukan langkah-langkah ini di perangkat Anda.

Untuk menginstal perangkat lunak aplikasi dan perpustakaan di Raspberry Pi Anda

1. Masukkan perintah ini untuk menginstal perangkat lunak aplikasi dan perpustakaan.

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

2. Masukkan perintah ini untuk mengkonfirmasi bahwa versi perangkat lunak yang benar telah diinstal.

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

3. Konfirmasikan bahwa versi perangkat lunak aplikasi ini diinstal:

- gcc: 9.3.0 atau yang lebih baru
- cmake: 3.10.x atau yang lebih baru
- OpenSSL: 1.1.1 atau yang lebih baru
- git: 2.20.1 atau yang lebih baru

Jika Raspberry Pi Anda memiliki versi yang dapat diterima dari perangkat lunak aplikasi yang diperlukan, Anda siap untuk melanjutkan [the section called “\(Opsional\) Simpan gambar kartu microSD”](#).

(Opsional) Simpan gambar kartu microSD

Sepanjang tutorial di jalur pembelajaran ini, Anda akan menemukan prosedur ini untuk menyimpan salinan gambar kartu microSD Raspberry Pi ke file di komputer host lokal Anda. Sementara didorong, mereka tidak diperlukan tugas. Dengan menyimpan gambar kartu microSD di mana disarankan, Anda dapat melewati prosedur yang mendahului titik simpan di jalur pembelajaran ini, yang dapat menghemat waktu jika Anda menemukan kebutuhan untuk mencoba kembali sesuatu. Konsekuensi dari tidak menyimpan gambar kartu microSD secara berkala adalah Anda mungkin harus me-restart tutorial di jalur pembelajaran dari awal jika kartu microSD Anda rusak atau jika Anda secara tidak sengaja mengkonfigurasi aplikasi atau pengaturannya secara tidak benar.

Pada titik ini, kartu microSD Raspberry Pi Anda memiliki OS yang diperbarui dan perangkat lunak aplikasi dasar dimuat. Anda dapat menghemat waktu yang dibutuhkan untuk menyelesaikan langkah-langkah sebelumnya dengan menyimpan isi kartu microSD ke file sekarang. Memiliki gambar kartu microSD perangkat Anda saat ini memungkinkan Anda mulai dari titik ini untuk melanjutkan atau mencoba lagi tutorial atau prosedur tanpa perlu menginstal dan memperbarui perangkat lunak dari awal.

Untuk menyimpan gambar kartu microSD ke file

1. Masukkan perintah ini untuk mematikan Raspberry Pi.

```
sudo shutdown -h 0
```

2. Setelah Raspberry Pi mati sepenuhnya, lepaskan kekuatannya.
3. Lepaskan kartu microSD dari Raspberry Pi.
4. Di komputer host lokal Anda:
 - a. Masukkan kartu microSD.
 - b. Menggunakan alat pencitraan kartu SD Anda, simpan gambar kartu microSD ke file.
 - c. Setelah gambar kartu microSD disimpan, keluarkan kartu dari komputer host lokal.
5. Dengan daya terputus dari Raspberry Pi, masukkan kartu microSD ke Raspberry Pi.
6. Terapkan daya ke Raspberry Pi.
7. Setelah menunggu sekitar satu menit, pada komputer host lokal, hubungkan kembali jendela terminal pada komputer host lokal Anda yang terhubung ke Raspberry Pi Anda., dan kemudian masuk ke Raspberry Pi.

Langkah 3: Uji perangkat Anda dan simpan sertifikat Amazon CA

Prosedur di bagian ini berlanjut dari [bagian sebelumnya](#) untuk menginstal AWS Command Line Interfacedan sertifikat Otoritas Sertifikat yang digunakan untuk mengotentikasi koneksi Anda dengan AWS IoT Core.

Setelah Anda menyelesaikan bagian ini, Anda akan tahu bahwa Raspberry Pi Anda memiliki perangkat lunak sistem yang diperlukan untuk menginstal AWS IoT Perangkat Client dan bahwa ia memiliki koneksi kerja ke internet.

Peralatan yang dibutuhkan:

- Pengembangan lokal Anda dan lingkungan pengujian dari [bagian sebelumnya](#)
- Raspberry Pi yang Anda gunakan [bagian sebelumnya](#)
- Kartu memori microSD dari [bagian sebelumnya](#)

Prosedur di bagian ini:

- [Memasang AWS Command Line Interface](#)
- [Konfigurasi kredensial Akun AWS Anda](#)
- [Unduh sertifikat CA Amazon Root CA](#)
- [\(Opsional\) Simpan gambar kartu microSD](#)

Memasang AWS Command Line Interface

Prosedur ini menginstal AWS CLI ke Raspberry Pi Anda.

Jika Anda menggunakan Raspberry Pi atau jika Anda dapat mengkompilasi perangkat lunak pada perangkat IoT Anda, lakukan langkah-langkah ini di jendela terminal pada komputer host lokal Anda. Jika Anda harus mengkompilasi perangkat lunak untuk perangkat IoT Anda di komputer host lokal Anda, tinjau dokumentasi perangkat lunak untuk perangkat IoT Anda untuk informasi tentang perpustakaan yang dibutuhkannya.

Untuk menginstal AWS CLI di Raspberry Pi Anda

1. Jalankan perintah ini untuk mengunduh dan menginstal 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. Jalankan perintah ini untuk menginstal AWS CLI. Perintah ini dapat memakan waktu hingga 15 menit untuk menyelesaikannya.

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


3. Jalankan perintah ini untuk mengonfirmasi bahwa versi AWS CLI telah terpasang.

```
aws --version
```

Versi dari AWS CLI harus 2,2 atau lebih baru.

Jika AWS CLI ditampilkan versi saat ini, Anda siap untuk melanjutkan [the section called "Konfigurasi kredensial Akun AWS Anda"](#).

Konfigurasi kredensial Akun AWS Anda

Dalam prosedur ini, Anda akan mendapatkan Akun AWS kredensialnya dan menambahkannya untuk digunakan pada Raspberry Pi Anda.

Untuk menambahkan Akun AWS kredensi ke perangkat Anda

1. Mendapatkan Access key ID dan Secret Access dari Akun AWS untuk mengotentikasi AWS CLI di perangkat Anda.

Jika Anda baru AWS IAM, <https://aws.amazon.com/premiumsupport/knowledge-center/create-access-key/> menjelaskan proses untuk berjalan di AWS konsol untuk membuat AWS kredensi IAM untuk digunakan di perangkat Anda.

2. Di jendela terminal pada komputer host lokal Anda yang terhubung ke Raspberry Pi Anda. dan dengan Access key ID dan Secret Access kredensi untuk perangkat Anda:
 - a. Jalankan AWS konfigurasi aplikasi dengan perintah ini:

```
aws configure
```

- b. Masukkan informasi kredensial dan konfigurasi Anda saat diminta:

```
AWS Access Key ID: your Access Key ID  
AWS Secret Access Key: your Secret Access Key  
Default region name: your Wilayah AWS code  
Default output format: json
```

3. Jalankan perintah ini untuk menguji akses perangkat Anda ke Akun AWS dan AWS IoT Core titik akhir.

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

Ini harus mengembalikan Akun AWS-spesifik AWS IoT endpoint data, seperti contoh ini:

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

Jika Anda melihat Akun AWS-spesifik AWS IoT titik akhir data, Raspberry Pi Anda memiliki konektivitas dan izin untuk melanjutkan [the section called “Unduh sertifikat CA Amazon Root CA”](#).

Important

Klaster Akun AWS kredensial sekarang disimpan pada kartu microSD di Raspberry Pi Anda. Sementara ini membuat interaksi masa depan dengan AWS mudah bagi Anda dan perangkat lunak yang akan Anda buat dalam tutorial ini, mereka juga akan disimpan dan diduplikasi dalam gambar kartu microSD yang Anda buat setelah langkah ini secara default.

Untuk melindungi keamanan Akun AWS kredensial, sebelum Anda menyimpan lebih banyak gambar kartu microSD, pertimbangkan untuk menghapus kredensialnya dengan menjalankan `aws configure` lagi dan memasukkan karakter acak untuk Access key ID dan Secret Access untuk mencegah Akun AWS kredensial dari dikompromikan.

Jika Anda menemukan bahwa Anda telah menyimpan Akun AWS kredensial secara tidak sengaja, Anda dapat menonaktifkannya di AWS Konsol IAM.

Unduh sertifikat CA Amazon Root CA

Prosedur ini mengunduh dan menyimpan salinan sertifikat Amazon Root Certificate Authority (CA). Mengunduh sertifikat ini menyimpannya untuk digunakan dalam tutorial berikutnya dan juga menguji konektivitas perangkat Anda AWS layanan.

Untuk mengunduh dan menyimpan sertifikat Amazon Root CA

1. Jalankan perintah ini untuk membuat direktori untuk sertifikat.

```
mkdir ~/certs
```

2. Jalankan perintah ini untuk mengunduh sertifikat CA Amazon Root.

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

3. Jalankan perintah ini untuk mengatur akses ke direktori sertifikat dan file-nya.

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

4. Jalankan perintah ini untuk melihat file sertifikat CA di direktori baru.

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

Anda harus melihat entri seperti ini. Tanggal dan waktu akan berbeda; Namun, ukuran file dan semua info lainnya harus sama seperti yang ditunjukkan di sini.

```
-rw-r--r-- 1 pi pi 1188 Oct 28 13:02 AmazonRootCA1.pem
```

Jika ukuran file tidak 1188, periksa curl parameter perintah. Anda mungkin telah mengunduh file yang salah.

(Opsional) Simpan gambar kartu microSD

Pada titik ini, kartu microSD Raspberry Pi Anda memiliki OS yang diperbarui dan perangkat lunak aplikasi dasar dimuat.

Untuk menyimpan gambar kartu microSD ke file

1. Di jendela terminal di komputer host lokal Anda, kosongkan AWS kredensial.
 - a. Jalankan AWS konfigurasi aplikasi dengan perintah ini:

```
aws configure
```

- b. Ganti kredensialnya saat diminta. Anda dapat meninggalkan Nama wilayah default dan Format output default karena mereka dengan menekan ENTER.

```
AWS Access Key ID [*****YT2H]: XYXYXYXYX
```

```
AWS Secret Access Key [*****9p1H]: XYXYXYXYX
Default region name [us-west-2]:
Default output format [json]:
```

2. Masukkan perintah ini untuk mematikan Raspberry Pi.

```
sudo shutdown -h 0
```

3. Setelah Raspberry Pi mati sepenuhnya, lepaskan konektor daya.
4. Hapus kartu microSD dari perangkat Anda.
5. Di komputer host lokal Anda:
 - a. Masukkan kartu microSD.
 - b. Menggunakan alat pencitraan kartu SD Anda, simpan gambar kartu microSD ke file.
 - c. Setelah gambar kartu microSD disimpan, keluarkan kartu dari komputer host lokal.
6. Dengan daya terputus dari Raspberry Pi, masukkan kartu microSD ke Raspberry Pi.
7. Terapkan daya ke perangkat.
8. Setelah sekitar satu menit, di komputer host lokal, restart sesi jendela terminal dan masuk ke perangkat.

Jangan masukkan kembali Akun AWS kredensi belum.

Setelah Anda me-restart dan login ke Raspberry Pi Anda, Anda siap untuk melanjutkan [the section called “Menginstal dan mengkonfigurasi AWS IoT Perangkat klien”](#).

Tutorial: Menginstal dan mengkonfigurasi AWS IoT Perangkat klien

Tutorial ini memandu Anda melalui instalasi dan konfigurasi AWS IoT Perangkat Client dan penciptaan AWS IoT sumber daya yang akan Anda gunakan dalam demo ini dan lainnya.

Untuk memulai tutorial ini:

- Memiliki komputer host lokal Anda dan Raspberry Pi dari [tutorial sebelumnya](#) siap.

Tutorial ini bisa memakan waktu hingga 90 menit.

Setelah selesai dengan topik ini:

- Perangkat IoT Anda akan siap digunakan di lain AWS IoT Demo perangkat.

- Anda akan menyediakan perangkat IoT Anda di AWS IoT Core.
- Anda akan telah mengunduh dan menginstal AWS IoT Perangkat Client pada perangkat Anda.
- Anda akan menyimpan gambar kartu microSD perangkat Anda yang dapat digunakan dalam tutorial berikutnya.

Peralatan yang dibutuhkan:

- Pengembangan lokal Anda dan lingkungan pengujian dari [bagian sebelumnya](#)
- Raspberry Pi yang Anda gunakan di [bagian sebelumnya](#)
- Kartu memori microSD dari Raspberry Pi yang Anda gunakan di [bagian sebelumnya](#)

Prosedur dalam tutorial ini

- [Langkah 1: Download dan menyimpan AWS IoT Perangkat klien](#)
- [\(Opsional\) Simpan gambar kartu microSD](#)
- [Langkah 2: Menyediakan Raspberry Pi Anda AWS IoT](#)
- [Langkah 3: Mengonfigurasi AWS IoT Klien Perangkat untuk menguji konektivitas](#)

Langkah 1: Download dan menyimpan AWS IoT Perangkat klien

Prosedur di bagian ini men-download AWS IoT Perangkat Client, kompilasi, dan menginstalnya pada Raspberry Pi Anda. Setelah Anda menguji instalasi, Anda dapat menyimpan gambar kartu microSD Raspberry Pi untuk digunakan nanti ketika Anda ingin mencoba tutorial lagi.

Prosedur di bagian ini:

- [Download dan membangun AWS IoT Perangkat klien](#)
- [Buat direktori yang digunakan oleh tutorial](#)

Download dan membangun AWS IoT Perangkat klien

Prosedur ini menginstal AWS IoT Perangkat pada Raspberry Pi Anda.

Lakukan perintah ini di jendela terminal pada komputer host lokal Anda yang terhubung ke Raspberry Pi Anda.

Untuk menginstal AWS IoT Perangkat pada Raspberry Pi Anda

1. Masukkan perintah ini untuk mengunduh dan membangun AWS IoT Perangkat pada Raspberry Pi Anda.

```
cd ~
git clone https://github.com/aws-labs/aws-iot-device-client aws-iot-device-client
mkdir ~/aws-iot-device-client/build && cd ~/aws-iot-device-client/build
cmake ../
```

2. Jalankan perintah ini untuk membangun AWS IoT Perangkat klien. Perintah ini bisa memakan waktu hingga 15 menit.

```
cmake --build . --target aws-iot-device-client
```

Pesan peringatan ditampilkan sebagai AWS IoT Kompilasi Perangkat bisa diabaikan.

Tutorial ini telah diuji dengan AWS IoT Perangkat Client dibangun gcc, versi (Raspbian 10.2.1-6+rpi1) 10.2.1 20210110 pada tanggal 30 Oktober 2021 versi Raspberry Pi OS (Bullseye) pada gcc, versi (Raspbian 8.3.0-6+rpi1) 8.3.0 pada 7 Mei 2021 versi OS Raspberry Pi (buster).

3. Setelah AWS IoT Perangkat Client selesai membangun, mengujinya dengan menjalankan perintah ini.

```
./aws-iot-device-client --help
```

Jika Anda melihat bantuan baris perintah untuk AWS IoT Perangkat Client, AWS IoT Perangkat telah dibangun dengan sukses dan siap Anda gunakan.

Buat direktori yang digunakan oleh tutorial

Prosedur ini menciptakan direktori pada Raspberry Pi yang akan digunakan untuk menyimpan file yang digunakan oleh tutorial di jalur pembelajaran ini.

Untuk membuat direktori yang digunakan oleh tutorial di jalur pembelajaran ini:

1. Jalankan perintah ini untuk membuat direktori yang diperlukan.

```
mkdir ~/dc-configs
mkdir ~/policies
```

```
mkdir ~/messages
mkdir ~/certs/testconn
mkdir ~/certs/pubsub
mkdir ~/certs/jobs
```

2. Jalankan perintah ini untuk mengatur izin pada direktori baru.

```
chmod 745 ~
chmod 700 ~/certs/testconn
chmod 700 ~/certs/pubsub
chmod 700 ~/certs/jobs
```

Setelah Anda membuat direktori ini dan mengatur izin mereka, lanjutkan ke [the section called “\(Opsional\) Simpan gambar kartu microSD”](#).

(Opsional) Simpan gambar kartu microSD

Pada titik ini, kartu microSD Raspberry Pi Anda memiliki OS yang diperbarui, perangkat lunak aplikasi dasar, dan AWS IoT Perangkat klien.

Jika Anda ingin kembali untuk mencoba latihan dan tutorial ini lagi, Anda dapat melewati prosedur sebelumnya dengan menulis gambar kartu microSD yang Anda simpan dengan prosedur ini ke kartu microSD baru dan melanjutkan tutorial dari [the section called “Langkah 2: Menyediakan Raspberry Pi Anda AWS IoT”](#).

Untuk menyimpan gambar kartu microSD ke file:

Di jendela terminal di komputer host lokal Anda yang terhubung ke Raspberry Pi Anda:

1. Konfirmasi bahwa Akun AWS kredensial belum disimpan.
 - a. Jalankan AWS konfigurasi aplikasi dengan perintah ini:

```
aws configure
```

- b. Jika kredensial Anda telah disimpan (jika ditampilkan di prompt), kemudian masukkan `XXXXXXXXXX` string ketika diminta seperti yang ditunjukkan di sini. Meninggalkan Nama wilayah default dan Format output default kosong.

```
AWS Access Key ID [*****YXYX]: XXXXXYXYX
AWS Secret Access Key [*****YXYX]: XXXXXYXYX
```

```
Default region name:  
Default output format:
```

2. Masukkan perintah ini untuk mematikan Raspberry Pi.

```
sudo shutdown -h 0
```

3. Setelah Raspberry Pi mati sepenuhnya, lepaskan konektor daya.
4. Hapus kartu microSD dari perangkat Anda.
5. Di komputer host lokal Anda:
 - a. Masukkan kartu microSD.
 - b. Menggunakan alat pencitraan kartu SD Anda, simpan gambar kartu microSD ke file.
 - c. Setelah gambar kartu microSD disimpan, keluarkan kartu dari komputer host lokal.

Anda dapat melanjutkan dengan kartu microSD ini di [the section called “Langkah 2: Menyediakan Raspberry Pi Anda AWS IoT”](#).

Langkah 2: Menyediakan Raspberry Pi Anda AWS IoT

Prosedur di bagian ini dimulai dengan gambar microSD yang disimpan yang memiliki AWS CLI dan AWS IoT Perangkat Client diinstal dan membuat AWS IoT sumber daya dan sertifikat perangkat yang menyediakan Raspberry Pi Anda AWS IoT.

Instal kartu microSD di Raspberry Pi Anda

Prosedur ini menginstal kartu microSD dengan perangkat lunak yang diperlukan dimuat dan dikonfigurasi ke dalam Raspberry Pi dan mengkonfigurasi Akun AWS sehingga Anda dapat melanjutkan dengan tutorial di jalur pembelajaran ini.

Menggunakan kartu microSD dari [the section called “\(Opsional\) Simpan gambar kartu microSD”](#) yang memiliki perangkat lunak yang diperlukan untuk latihan dan tutorial di jalur pembelajaran ini.

Untuk menginstal kartu microSD di Raspberry Pi Anda

1. Dengan daya terputus dari Raspberry Pi, masukkan kartu microSD ke Raspberry Pi.
2. Menerapkan kekuatan ke Raspberry Pi.
3. Setelah sekitar satu menit, pada komputer host lokal, restart sesi jendela terminal dan masuk ke Raspberry Pi.

4. Pada komputer host lokal Anda, di jendela terminal, dan dengan Access key ID dan Secret access key kredensialnya untuk Raspberry Pi Anda:
 - a. Jalankan AWS untuk mengkonfigurasi aplikasi dengan perintah ini:

```
aws configure
```

- b. Masukkan Akun AWS kredensi dan informasi konfigurasi saat diminta:

```
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 Wilayah AWS code
Default output format [json]: json
```

Setelah Anda memulihkan Akun AWS kredensialnya, Anda siap untuk melanjutkan [the section called “Menyediakan perangkat Anda AWS IoT Core”](#).

Menyediakan perangkat Anda AWS IoT Core

Prosedur di bagian ini membuat AWS IoT sumber daya yang menyediakan Raspberry Pi Anda di AWS IoT. Saat Anda membuat sumber daya ini, Anda akan diminta untuk merekam berbagai informasi. Informasi ini digunakan oleh AWS IoT konfigurasi perangkat pada prosedur berikutnya.

Untuk Raspberry Pi Anda untuk bekerja dengan AWS IoT, itu harus disediakan. Penyediaan adalah proses pembuatan dan konfigurasi AWS IoT sumber daya yang diperlukan untuk mendukung Raspberry Pi Anda sebagai perangkat IoT.

Dengan Raspberry Pi Anda diaktifkan dan restart, menghubungkan jendela terminal pada komputer host lokal Anda ke Raspberry Pi dan menyelesaikan prosedur ini.

Prosedur di bagian ini:

- [Membuat dan mengunduh file sertifikat perangkat](#)
- [Buat AWS IoT sumber daya](#)

Membuat dan mengunduh file sertifikat perangkat

Prosedur ini membuat file sertifikat perangkat untuk demo ini.

Untuk membuat dan mengunduh file sertifikat perangkat untuk Raspberry Pi Anda

1. Di jendela terminal di komputer host lokal Anda, masukkan perintah ini untuk membuat file sertifikat perangkat untuk perangkat Anda.

```
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"
```

Perintah mengembalikan respons seperti berikut ini. Catat *certificateArn* nilai untuk digunakan nanti.

```
{
  "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. Masukkan perintah berikut untuk mengatur izin pada direktori sertifikat dan file-filenya.

```
chmod 745 ~
chmod 700 ~/certs/testconn
chmod 644 ~/certs/testconn/*
chmod 600 ~/certs/testconn/private.pem.key
```

3. Jalankan perintah ini untuk meninjau izin pada direktori dan file sertifikat Anda.

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

Output dari perintah harus sama dengan apa yang Anda lihat di sini, kecuali tanggal dan waktu file akan berbeda.

```
-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
```

Pada titik ini, Anda memiliki file sertifikat perangkat yang diinstal pada Raspberry Pi Anda dan Anda dapat melanjutkan [the section called “BuatAWS IoT sumber daya”](#).

BuatAWS IoT sumber daya

Prosedur ini menyediakan perangkat Anda diAWS IoT dengan membuat sumber daya yang perlu diakses perangkat AndaAWS IoT fitur dan layanan.

Untuk menyediakan perangkat Anda diAWS IoT

1. Di jendela terminal di komputer host lokal Anda, masukkan perintah berikut untuk mendapatkan alamat titik akhir data perangkat untuk AndaAkun AWS.

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

Perintah dari langkah-langkah sebelumnya mengembalikan respons seperti berikut ini. Catat *endpointAddress* nilai untuk digunakan nanti.

```
{
  "endpointAddress": "a3qjEXAMPLEffp-ats.iot.us-west-2.amazonaws.com"
}
```

2. Masukkan perintah ini untuk membuatAWS IoT sumber daya hal untuk Raspberry Pi Anda.

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

JikaAWS IoT sumber daya diciptakan, perintah mengembalikan respon seperti ini.

```
{
```

```
"thingName": "DevCliTestThing",  
"thingArn": "arn:aws:iot:us-west-2:57EXAMPLE833:thing/DevCliTestThing",  
"thingId": "8ea78707-32c3-4f8a-9232-14bEXAMPLEfd"  
}
```

3. Di jendela terminal:

- a. Buka editor teks, seperti nano.
- b. Salin dokumen kebijakan JSON ini dan tempelkan ke editor teks terbuka Anda.

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

Note

Dokumen kebijakan ini dengan murah hati memberikan setiap izin sumber daya untuk menghubungkan, menerima, mempublikasikan, dan berlangganan. Biasanya kebijakan hanya memberikan izin kepada sumber daya tertentu untuk melakukan tindakan tertentu. Namun, untuk tes konektivitas perangkat awal, kebijakan yang terlalu umum dan permisif ini digunakan untuk meminimalkan kemungkinan masalah akses selama pengujian ini. Dalam tutorial berikutnya, dokumen kebijakan yang lebih sempit akan digunakan untuk menunjukkan praktik yang lebih baik dalam desain kebijakan.

- c. Menyimpan file di editor teks Anda sebagai `~/policies/dev_cli_test_thing_policy.json`.
4. Jalankan perintah ini untuk menggunakan dokumen kebijakan dari langkah-langkah sebelumnya untuk membuat AWS IoT kebijakan.

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

Jika kebijakan dibuat, perintah mengembalikan respons seperti ini.

```
{
  "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}",
  "policyVersionId": "1"
}
```

5. Jalankan perintah ini untuk melampirkan kebijakan tersebut ke sertifikat perangkat. Ganti *certificateArn* dengan *certificateArn* nilai yang Anda simpan sebelumnya.

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

Jika berhasil, perintah ini tidak mengembalikan apa pun.

6. Jalankan perintah ini untuk melampirkan sertifikat perangkat ke AWS IoT sumber daya. Ganti *certificateArn* dengan *certificateArn* nilai yang Anda simpan sebelumnya.

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

Jika berhasil, perintah ini tidak mengembalikan apa pun.

Setelah Anda berhasil menyediakan perangkat Anda di AWS IoT, Anda siap untuk melanjutkan [the section called “Langkah 3: Mengonfigurasi AWS IoT Klien Perangkat untuk menguji konektivitas”](#).

Langkah 3: Mengonfigurasi AWS IoT Klien Perangkat untuk menguji konektivitas

Prosedur di bagian ini mengkonfigurasi AWS IoT Perangkat Client untuk mempublikasikan pesan MQTT dari Raspberry Pi Anda.

Prosedur di bagian ini:

- [Buat file konfigurasi](#)
- [Buka klien uji MQTT](#)
- [Jalankan AWS IoT Perangkat klien](#)

Buat file konfigurasi

Prosedur ini menciptakan file config untuk menguji AWS IoT Perangkat klien.

Untuk membuat file konfigurasi untuk menguji AWS IoT Perangkat klien

- Di jendela terminal di komputer host lokal Anda yang terhubung ke Raspberry Pi Anda:
 - a. Masukkan perintah ini untuk membuat direktori untuk file konfigurasi dan mengatur izin pada direktori:

```
mkdir ~/dc-configs
chmod 745 ~/dc-configs
```

- b. Buka editor teks, seperti nano.
- c. Salin dokumen JSON ini dan tempelkan ke editor teks terbuka Anda.

```
{
  "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. Ganti *titik akhir* nilai dengan titik akhir data perangkat untuk Akun AWS yang Anda temukan di [the section called “Menyediakan perangkat Anda AWS IoT Core”](#).
- e. Menyimpan file di editor teks Anda sebagai `~/dc-configs/dc-testconn-config.json`.

- f. Jalankan perintah ini untuk mengatur izin pada file konfigurasi baru.

```
chmod 644 ~/dc-configs/dc-testconn-config.json
```

Setelah menyimpan file, Anda siap untuk melanjutkan [the section called “Buka klien uji MQTT”](#).

Buka klien uji MQTT

Prosedur ini mempersiapkan Klien uji MQTT di AWS IoT konsol untuk berlangganan pesan MQTT bahwa AWS IoT Perangkat Client menerbitkan ketika berjalan.

Untuk menyiapkan Klien uji MQTT untuk berlangganan semua pesan MQTT

1. Di komputer host lokal Anda, di [AWS IoT konsol](#), pilih Klien uji MQTT.
2. Di Berlangganan topik tab, di Filter topik masukkan # (tanda pon tunggal), dan pilih Langganan untuk berlangganan setiap topik MQTT.
3. Di bawah Langganan label, konfirmasi bahwa Anda melihat # (tanda pon tunggal).

Tinggalkan jendela dengan Klien uji MQTT terbuka saat Anda melanjutkan [the section called “Jalankan AWS IoT Perangkat klien”](#).

Jalankan AWS IoT Perangkat klien

Prosedur ini menjalankan AWS IoT Perangkat Client sehingga menerbitkan pesan MQTT tunggal bahwa Klien uji MQTT menerima dan menampilkan.

Untuk mengirim pesan MQTT dari AWS IoT Perangkat klien

1. Pastikan bahwa kedua jendela terminal yang terhubung ke Raspberry Pi dan jendela dengan Klien uji MQTT terlihat saat Anda melakukan prosedur ini.
2. Di jendela terminal, masukkan perintah ini untuk menjalankan AWS IoT Perangkat Client menggunakan file konfigurasi yang dibuat di [the section called “Buat file konfigurasi”](#).

```
cd ~/aws-iot-device-client/build  
./aws-iot-device-client --config-file ~/dc-configs/dc-testconn-config.json
```

Di jendela terminal, AWS IoT Perangkat Client menampilkan pesan informasi dan kesalahan yang terjadi saat berjalan.

- Jika tidak ada kesalahan yang ditampilkan di jendela terminal, tinjauKlien uji MQTT.
3. DiKlien uji MQTT, di jendela Langganan, lihatHalo Dunia!pesan yang dikirim ketest/dc/pubtopicpesan.
 4. JikaAWS IoTPerangkat Client tidak menampilkan kesalahan dan Anda melihatHalo Dunia!dikirim ketest/dc/pubtopicpesan dalamKlien uji MQTT, Anda telah menunjukkan koneksi yang sukses.
 5. Di jendela terminal, masukkan`^C`(Ctrl-C) untuk menghentikanAWS IoTPerangkat klien.

Setelah Anda telah menunjukkan bahwaAWS IoTPerangkat Client berjalan dengan benar pada Raspberry Pi Anda dan dapat berkomunikasi denganAWS IoT, Anda dapat melanjutkan ke[the section called “Menunjukkan komunikasi pesan MQTT denganAWS IoTklien perangkat”](#).

Tutorial: Menunjukkan komunikasi pesan MQTT denganAWS IoTklien perangkat

Tutorial ini menunjukkan bagaimanaAWS IoTPerangkat Client dapat berlangganan dan mempublikasikan pesan MQTT, yang umumnya digunakan dalam solusi IoT.

Untuk memulai tutorial ini:

- Memiliki komputer host lokal Anda dan Raspberry Pi dikonfigurasi seperti yang digunakan dalam[bagian sebelumnya](#).

Jika Anda menyimpan gambar kartu microSD setelah menginstalAWS IoTPerangkat Client, Anda dapat menggunakan kartu microSD dengan gambar itu dengan Raspberry Pi Anda.

- Jika Anda telah menjalankan demo ini sebelumnya, tinjau[???](#)untuk menghapus semuaAWS IoTsumber daya yang Anda buat di jalankan sebelumnya untuk menghindari kesalahan sumber daya duplikat.

Tutorial ini membutuhkan waktu sekitar 45 menit untuk menyelesaikannya.

Setelah selesai dengan topik ini:

- Anda akan menunjukkan berbagai cara bahwa perangkat IoT Anda dapat berlangganan pesan MQTT dariAWS IoTdan mempublikasikan pesan MQTT keAWS IoT.

Peralatan yang dibutuhkan:

- Pengembangan lokal Anda dan lingkungan pengujian dari [bagian sebelumnya](#)
- Raspberry Pi yang Anda gunakan [bagian sebelumnya](#)
- Kartu memori microSD dari Raspberry Pi yang Anda gunakan [bagian sebelumnya](#)

Prosedur dalam tutorial ini

- [Langkah 1: Siapkan Raspberry Pi untuk menunjukkan komunikasi pesan MQTT](#)
- [Langkah 2: Menunjukkan penerbitan pesan dengan AWS IoT Client perangkat](#)
- [Langkah 3: Menunjukkan berlangganan pesan dengan AWS IoT Client perangkat](#)

Langkah 1: Siapkan Raspberry Pi untuk menunjukkan komunikasi pesan MQTT

Prosedur ini menciptakan sumber daya di AWS IoT dan di Raspberry Pi untuk menunjukkan komunikasi pesan MQTT menggunakan AWS IoT Client perangkat.

Prosedur di bagian ini:

- [Buat file sertifikat untuk menunjukkan komunikasi MQTT](#)
- [Menyediakan perangkat Anda untuk menunjukkan komunikasi MQTT](#)
- [Konfigurasi AWS IoT Perangkat Client file konfigurasi dan klien uji MQTT untuk menunjukkan komunikasi MQTT](#)

Buat file sertifikat untuk menunjukkan komunikasi MQTT

Prosedur ini membuat file sertifikat perangkat untuk demo ini.

Untuk membuat dan mengunduh file sertifikat perangkat untuk Raspberry Pi

1. Di jendela terminal di komputer host lokal Anda, masukkan perintah berikut untuk membuat file sertifikat perangkat untuk perangkat Anda.

```
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"
```

Perintah mengembalikan respons seperti berikut ini. Simpan *certificateArn* nilai untuk digunakan nanti.

```
{
  "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. Masukkan perintah berikut untuk mengatur izin pada direktori sertifikat dan file-filenya.

```
chmod 700 ~/certs/pubsub
chmod 644 ~/certs/pubsub/*
chmod 600 ~/certs/pubsub/private.pem.key
```

3. Jalankan perintah ini untuk meninjau izin pada direktori dan file sertifikat Anda.

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

Output dari perintah harus sama dengan apa yang Anda lihat di sini, kecuali tanggal dan waktu file akan berbeda.

```
-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. Masukkan perintah ini untuk membuat direktori untuk file log.

```
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/*
```

Menyediakan perangkat Anda untuk menunjukkan komunikasi MQTT

Bagian ini menciptakan AWS IoT sumber daya yang menyediakan Raspberry Pi Anda AWS IoT.

Untuk menyediakan perangkat Anda di AWS IoT:

1. Di jendela terminal di komputer host lokal Anda, masukkan perintah berikut untuk mendapatkan alamat titik akhir data perangkat untuk Anda Akun AWS.

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

Nilai endpoint tidak berubah sejak saat Anda menjalankan perintah ini untuk tutorial sebelumnya. Menjalankan perintah lagi di sini dilakukan untuk membuatnya mudah untuk menemukan dan menempelkan nilai endpoint data ke dalam file konfigurasi yang digunakan dalam tutorial ini.

Perintah dari langkah-langkah sebelumnya mengembalikan respons seperti berikut ini. Catat *endpointAddress* nilai untuk digunakan nanti.

```
{
  "endpointAddress": "a3qjEXAMPLEffp-ats.iot.us-west-2.amazonaws.com"
}
```

2. Masukkan perintah ini untuk membuat yang baru AWS IoT sumber daya hal untuk Raspberry Pi Anda.

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

Karena AWS IoT sumber daya hal adalah may representasi perangkat Anda di cloud, kita dapat membuat beberapa hal sumber daya di AWS IoT untuk digunakan untuk tujuan yang berbeda. Mereka semua dapat digunakan oleh perangkat IoT fisik yang sama untuk mewakili aspek yang berbeda dari perangkat.

Tutorial ini hanya akan menggunakan satu sumber daya pada suatu waktu untuk mewakili Raspberry Pi. Dengan cara ini, dalam tutorial ini, mereka mewakili demo yang berbeda sehingga setelah Anda membuat AWS IoT sumber daya untuk demo, Anda dapat kembali dan mengulangi demo menggunakan sumber daya yang Anda buat khusus untuk masing-masing.

Jika AWS IoT sumber daya diciptakan, mengembalikan respon seperti ini.

```
{
  "thingName": "PubSubTestThing",
  "thingArn": "arn:aws:iot:us-west-2:57EXAMPLE833:thing/PubSubTestThing",
  "thingId": "8ea78707-32c3-4f8a-9232-14bEXAMPLEfd"
}
```

3. Di jendela terminal:

- a. Buka editor teks, seperti nano.
- b. Salin dokumen JSON ini dan tempelkan ke editor teks terbuka Anda.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect"
      ],
      "Resource": [
        "arn:aws:iot:us-west-2:57EXAMPLE833:client/PubSubTestThing"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish"
      ],
      "Resource": [
        "arn:aws:iot:us-west-2:57EXAMPLE833:topic/test/dc/pubtopic"
      ]
    },
    {
      "Effect": "Allow",
```

```

    "Action": [
      "iot:Subscribe"
    ],
    "Resource": [
      "arn:aws:iot:us-west-2:57EXAMPLE833:topicfilter/test/dc/subtopic"
    ]
  },
  {
    "Effect": "Allow",
    "Action": [
      "iot:Receive"
    ],
    "Resource": [
      "arn:aws:iot:us-west-2:57EXAMPLE833:topic/test/dc/subtopic"
    ]
  }
]
}

```

- c. Di editor, di masing-masing Resource bagian dari dokumen kebijakan, ganti `us-west-2:57` dengan Wilayah AWS, karakter usus (:), dan 12-digit Akun AWS nomor.
 - d. Menyimpan file di editor teks Anda sebagai `~/policies/pubsub_test_thing_policy.json`.
4. Jalankan perintah ini untuk menggunakan dokumen kebijakan dari langkah-langkah sebelumnya untuk membuat AWS IoT kebijakan.

```

aws iot create-policy \
--policy-name "PubSubTestThingPolicy" \
--policy-document "file://~/policies/pubsub_test_thing_policy.json"

```

Jika kebijakan dibuat, perintah mengembalikan respons seperti ini.

```

{
  "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

```



```
{
  "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. Ganti *titik akhir* nilai dengan titik akhir data perangkat untuk Akun AWS yang Anda temukan di [the section called “Menyediakan perangkat Anda AWS IoT Core”](#).
- d. Menyimpan file di editor teks Anda sebagai `~/dc-configs/dc-pubsub-config.json`.
- e. Jalankan perintah ini untuk mengatur izin pada file konfigurasi baru.

```
chmod 644 ~/dc-configs/dc-pubsub-config.json
```

2. Untuk menyiapkan Klien uji MQTT untuk berlangganan semua pesan MQTT:
 - a. Di komputer host lokal Anda, di [AWS IoT konsol](#), pilih Klien uji MQTT.
 - b. Di Berlangganan topik tab, di Filter topik ENTER # (tanda pon tunggal), dan pilih Langganan.
 - c. Di bawah Langganan label, konfirmasi bahwa Anda melihat # (tanda pon tunggal).

Tinggalkan jendela dengan Klien uji MQTT terbuka saat Anda melanjutkan tutorial ini.

Setelah Anda menyimpan file dan mengkonfigurasi Klien uji MQTT, Anda siap untuk melanjutkan [the section called “Langkah 2: Menunjukkan penerbitan pesan dengan AWS IoT Klien perangkat”](#).

Langkah 2: Menunjukkan penerbitan pesan dengan AWS IoT Klien perangkat

Prosedur di bagian ini menunjukkan bagaimana AWS IoT Perangkat Client dapat mengirim pesan MQTT default dan kustom.

Pernyataan kebijakan ini dalam kebijakan yang Anda buat di langkah sebelumnya untuk latihan ini memberikan izin Raspberry Pi untuk melakukan tindakan ini:

- **iot:Connect**

Memberikan klien bernama `PubSubTestThing`, Pi Raspberry Anda menjalankan AWS IoT Perangkat Client, untuk terhubung.

```
{
```

```
"Effect": "Allow",
"Action": [
  "iot:Connect"
],
"Resource": [
  "arn:aws:iot:us-west-2:57EXAMPLE833:client/PubSubTestThing"
]
}
```

• **iot:Publish**

Memberikan izin Raspberry Pi untuk mempublikasikan pesan dengan topik MQTTtest/dc/pubtopic.

```
{
  "Effect": "Allow",
  "Action": [
    "iot:Publish"
  ],
  "Resource": [
    "arn:aws:iot:us-west-2:57EXAMPLE833:topic/test/dc/pubtopic"
  ]
}
```

Parameter `iot:Publish` tindakan memberikan izin untuk mempublikasikan ke topik MQTT yang tercantum dalam array `Resource`. Parameter `isi` pesan-pesan tersebut tidak dikendalikan oleh pernyataan kebijakan.

Mempublikasikan pesan default menggunakan AWS IoT Client perangkat

Prosedur ini menjalankan AWS IoT Client perangkat sehingga menerbitkan pesan MQTT default tunggal bahwa klien uji MQTT menerima dan menampilkan.

Untuk mengirim pesan MQTT default dari AWS IoT Client perangkat

1. Pastikan bahwa kedua jendela terminal pada komputer host lokal Anda yang terhubung ke Raspberry Pi dan jendela dengan klien uji MQTT terlihat saat Anda melakukan prosedur ini.
2. Pada jendela terminal, masukkan perintah ini untuk menjalankan AWS IoT Client perangkat menggunakan file konfigurasi yang dibuat di [the section called "Buat file konfigurasi"](#).

```
cd ~/aws-iot-device-client/build
```

```
./aws-iot-device-client --config-file ~/dc-configs/dc-pubsub-config.json
```

Dari jendela terminal, AWS IoT Perangkat Client menampilkan pesan informasi dan kesalahan yang terjadi saat berjalan.

Jika tidak ada kesalahan yang ditampilkan di jendela terminal, tinjau klien uji MQTT.

3. Di klien uji MQTT, di Langgan jendela, lihat Hello World! pesan yang dikirim ke test/dc/pubtopic topik pesan.
4. Jika AWS IoT Perangkat Client tidak menampilkan kesalahan dan Anda melihat Hello World! dikirim ke test/dc/pubtopic pesan dalam klien uji MQTT, Anda telah menunjukkan koneksi yang sukses.
5. Dari jendela terminal, masukkan **^C** (Ctrl-C) untuk menghentikan AWS IoT Klien perangkat.

Setelah Anda telah menunjukkan bahwa AWS IoT Perangkat Client menerbitkan pesan MQTT default, Anda dapat melanjutkan ke [the section called “Mempublikasikan pesan kustom menggunakan AWS IoT Klien perangkat.”](#).

Mempublikasikan pesan kustom menggunakan AWS IoT Klien perangkat.

Prosedur di bagian ini membuat pesan MQTT kustom dan kemudian menjalankan AWS IoT Perangkat Client sehingga menerbitkan pesan MQTT kustom satu kali untuk klien uji MQTT untuk menerima dan menampilkan.

Buat pesan MQTT khusus untuk AWS IoT Klien perangkat

Lakukan langkah-langkah ini di jendela terminal pada komputer host lokal yang terhubung ke Raspberry Pi Anda.

Membuat pesan khusus untuk AWS IoT Perangkat Client untuk mempublikasikan

1. Pada jendela terminal, buka editor teks, seperti nano.
2. Ke editor teks, salin dan tempel dokumen JSON berikut. Ini akan menjadi muatan pesan MQTT yang AWS IoT Perangkat Client menerbitkan.

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
```

```
"wind": {
  "velocity": 22,
  "bearing": 255
}
}
```

3. Menyimpan konten dari editor teks sebagai **~/messages/sample-ws-message.json**.
4. Masukkan perintah berikut untuk mengatur izin dari file pesan yang baru saja Anda buat.

```
chmod 600 ~/messages/*
```

Untuk membuat file konfigurasi untuk AWS IoT Perangkat Client yang digunakan untuk mengirim pesan kustom

1. Di jendela terminal, dalam editor teks seperti nano, buka yang ada AWS IoT Perangkat Client file konfigurasi: **~/dc-configs/dc-pubsub-config.json**.
2. Mengedit sampel terlihat seperti ini. Tidak ada bagian lain dari file ini yang perlu diubah.

```
"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. Menyimpan konten dari editor teks sebagai **~/dc-configs/dc-pubsub-custom-config.json**.
4. Jalankan perintah ini untuk mengatur izin pada file konfigurasi baru.

```
chmod 644 ~/dc-configs/dc-pubsub-custom-config.json
```

Publikasikan pesan MQTT kustom dengan menggunakan AWS IoT Klien perangkat

Perubahan ini hanya memengaruhi isi dari muatan pesan MQTT, sehingga kebijakan saat ini akan terus bekerja. Namun, jika topik MQTT (sebagaimana didefinisikan oleh `publish-topic` value `~/dc-configs/dc-pubsub-custom-config.json`) diubah, `iot::Publish` Pernyataan kebijakan juga perlu dimodifikasi untuk memungkinkan Raspberry Pi untuk mempublikasikan topik MQTT baru.

Untuk mengirim pesan MQTT dari AWS IoT Client perangkat

1. Pastikan bahwa jendela terminal dan jendela dengan Client uji MQTT terlihat saat Anda melakukan prosedur ini. Selain itu, pastikan bahwa Anda Client uji MQTT masih berlangganan #filter topik. Jika tidak, berlangganan #filter topik lagi.
2. Pada jendela terminal, masukkan perintah ini untuk menjalankan AWS IoT Client menggunakan file konfigurasi yang dibuat di [the section called “Buat file konfigurasi”](#).

```
cd ~/aws-iot-device-client/build
./aws-iot-device-client --config-file ~/dc-configs/dc-pubsub-custom-config.json
```

Dari jendela terminal, AWS IoT Client menampilkan pesan informasi dan kesalahan yang terjadi saat berjalan.

Jika tidak ada kesalahan yang ditampilkan di jendela terminal, tinjau Client uji MQTT.

3. Di Client uji MQTT, di Langganan jendela, lihat muatan pesan kustom yang dikirim ke test/dc/pubtopic topik pesan.
4. Jika AWS IoT Client Perangkat tidak menampilkan kesalahan dan Anda melihat muatan pesan kustom yang Anda publikasikan ke test/dc/pubtopic pesan dalam Client uji MQTT, Anda telah berhasil menerbitkan pesan kustom.
5. Dari jendela terminal, masukkan **^C** (Ctrl-C) untuk menghentikan AWS IoT Client perangkat.

Setelah Anda telah menunjukkan bahwa AWS IoT Client menerbitkan muatan pesan kustom, Anda dapat melanjutkan [the section called “Langkah 3: Menunjukkan berlangganan pesan dengan AWS IoT Client perangkat”](#).

Langkah 3: Menunjukkan berlangganan pesan dengan AWS IoT Client perangkat

Pada bagian ini, Anda akan menunjukkan dua jenis langganan pesan:

- Langganan topik tunggal
- Langganan topik Wild-card

Pernyataan kebijakan ini dalam kebijakan yang dibuat untuk latihan ini memberikan izin Raspberry Pi untuk melakukan tindakan ini:

- **iot:Receive**

Memberikan AWS IoT Klien Perangkat untuk menerima topik MQTT yang cocok dengan yang disebutkan dalam Resource objek.

```
{
  "Effect": "Allow",
  "Action": [
    "iot:Receive"
  ],
  "Resource": [
    "arn:aws:iot:us-west-2:57EXAMPLE833:topic/test/dc/subtopic"
  ]
}
```

- **iot:Subscribe**

Memberikan AWS IoT Klien untuk berlangganan filter topik MQTT yang cocok dengan yang disebutkan di Resource objek.

```
{
  "Effect": "Allow",
  "Action": [
    "iot:Subscribe"
  ],
  "Resource": [
    "arn:aws:iot:us-west-2:57EXAMPLE833:topicfilter/test/dc/subtopic"
  ]
}
```

Berlangganan satu topik pesan MQTT

Prosedur ini menunjukkan bagaimana AWS IoT Perangkat Client dapat berlangganan dan log pesan MQTT.

Di jendela terminal pada komputer host lokal Anda yang terhubung ke Raspberry Pi Anda, daftar isi `~/dc-configs/dc-pubsub-custom-config.json` atau buka file di editor teks untuk meninjau isinya. Temukan `samples` objek, yang akan terlihat seperti ini.

```
"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"
```

Perhatikan `subscribe-topic` adalah topik MQTT dimana AWS IoT Perangkat Client akan berlangganan ketika berjalan. Parameter AWS IoT Perangkat Client menulis muatan pesan yang diterima dari langganan ini ke file bernama `subscribe-file` ini.

Untuk berlangganan topik pesan MQTT dari AWS IoT Perangkat Client

1. Pastikan bahwa jendela terminal dan jendela dengan klien uji MQTT terlihat saat Anda melakukan prosedur ini. Selain itu, pastikan bahwa Anda klien uji MQTT masih berlangganan `#filter` topik. Jika tidak, berlangganan `#filter` topik lagi.
2. Pada jendela terminal, masukkan perintah ini untuk menjalankan AWS IoT Perangkat Client menggunakan file konfigurasi yang dibuat di [the section called "Buat file konfigurasi"](#).

```
cd ~/aws-iot-device-client/build
./aws-iot-device-client --config-file ~/dc-configs/dc-pubsub-custom-config.json
```

Dari jendela terminal, AWS IoT Perangkat Client menampilkan pesan informasi dan kesalahan yang terjadi saat berjalan.

Jika tidak ada kesalahan yang ditampilkan di jendela terminal, lanjutkan di jendela terminal AWS IoT konsol.

3. Di AWS IoT konsol, di klien uji MQTT, pilih **Menerbitkan** ke topik `tab`.
4. Masukkan nama topik `ENTER test/dc/subtopic`
5. Masukkan muatan pesan, meninjau isi pesan.
6. Pilih **Publikasikan** untuk mempublikasikan pesan MQTT.
7. Di jendela terminal, perhatikan pesan yang diterima dari AWS IoT Perangkat Client terlihat seperti ini.

```
2021-11-10T16:02:20.890Z [DEBUG] {samples/PubSubFeature.cpp}: Message received on
subscribe topic, size: 45 bytes
```

8. Setelah Anda melihat pesan yang diterima dari klien uji MQTT, masukkan `^C` (Ctrl-C) untuk menghentikan AWS IoT Perangkat Client.

9. Masukkan perintah ini untuk melihat akhir file log pesan dan melihat pesan yang Anda publikasikan dariKlien uji MQTT.

```
tail ~/.aws-iot-device-client/log/pubsub_rx_msgs.log
```

Dengan melihat pesan dalam file log, Anda telah menunjukkan bahwaAWS IoTClient Perangkat menerima pesan yang Anda publikasikan dari klien uji MQTT.

Berlangganan beberapa topik pesan MQTT menggunakan karakter wildcard

Prosedur ini menunjukkan bagaimanaAWS IoTPerangkat Client dapat berlangganan dan log pesan MQTT menggunakan karakter wildcard. Untuk melakukan ini, Anda akan:

1. Perbarui filter topik yangAWS IoTPerangkat Client menggunakan untuk berlangganan topik MQTT.
2. Perbarui kebijakan yang digunakan oleh perangkat untuk mengizinkan langganan baru.
3. JalankanAWS IoTPerangkat Client dan mempublikasikan pesan dari konsol uji MQTT.

Untuk membuat file konfigurasi untuk berlangganan beberapa topik pesan MQTT dengan menggunakan filter topik MQTT wildcard

1. Di jendela terminal di komputer host lokal Anda yang terhubung ke Raspberry Pi Anda, buka**~/dc-configs/dc-pubsub-custom-config.json**untuk mengedit dan menemukan**samples**objek.
2. Di editor teks, carib**samples**objek dan memperbaruis**subscribe-topic**terlihat seperti ini.

```
"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"
```

Yang baru**subscribe-topic**value adalah[Filter topik MQTT](#)dengan karakter kartu liar MQTT di akhir. Ini menjelaskan langganan ke semua topik MQTT yang dimulai dengantest/dc/. ParameterAWS IoTDevice Client menulis muatan pesan yang diterima dari langganan ini ke file bernama **subscribe-file**.

3. Simpan file konfigurasi yang dimodifikasi sebagai `~/dc-configs/dc-pubsub-wild-config.json`, dan keluar dari editor.

Untuk mengubah kebijakan yang digunakan oleh Raspberry Pi Anda untuk memungkinkan berlangganan dan menerima beberapa topik pesan MQTT

1. Di jendela terminal di komputer host lokal Anda yang terhubung ke Raspberry Pi Anda, di editor teks favorit Anda, buka `~/policies/pubsub_test_thing_policy.json` untuk mengedit, dan kemudian menemukan `iot::Subscribe` dan `iot::Receive` pernyataan kebijakan dalam file.
2. Di `iot::Subscribe` pernyataan kebijakan, memperbarui string dalam objek Sumber Daya untuk menggantikannya `subtopicbersama*`, sehingga terlihat seperti ini.

```
{
  "Effect": "Allow",
  "Action": [
    "iot:Subscribe"
  ],
  "Resource": [
    "arn:aws:iot:us-west-2:57EXAMPLE833:topicfilter/test/dc/*"
  ]
}
```

Note

Parameter [MQTT topik penyaring karakter kartu liar](#) adalah `+` (tanda plus) dan `#` (tanda pound). Permintaan berlangganan dengan `#` pada akhirnya berlangganan semua topik yang dimulai dengan string yang mendahului `#` karakter (misalnya, `test/dc/` dalam hal ini).

Nilai sumber daya dalam pernyataan kebijakan yang mengotorisasi langganan ini, bagaimanapun, harus menggunakan `*` (tanda bintang) di tempat `#` (tanda pound) di filter topik ARN. Hal ini karena prosesor kebijakan menggunakan karakter wild card yang berbeda dari MQTT menggunakan.

Untuk informasi selengkapnya tentang menggunakan karakter wild card untuk topik dan filter topik dalam kebijakan, lihat [Menggunakan karakter wildcard di MQTT dan kebijakan AWS IoT Core](#).

3. Diot::Receive pernyataan kebijakan, memperbarui string dalam objek Sumber Daya untuk menggantikansubtopicbersama*, sehingga terlihat seperti ini.

```
{
  "Effect": "Allow",
  "Action": [
    "iot:Receive"
  ],
  "Resource": [
    "arn:aws:iot:us-west-2:57EXAMPLE833:topic/test/dc/*"
  ]
}
```

4. Simpan dokumen kebijakan yang diperbarui sebagai `~/policies/pubsub_wild_test_thing_policy.json`, dan keluar dari editor.
5. Masukkan perintah ini untuk memperbarui kebijakan untuk tutorial ini untuk menggunakan definisi sumber daya baru.

```
aws iot create-policy-version \
--set-as-default \
--policy-name "PubSubTestThingPolicy" \
--policy-document "file://~/policies/pubsub_wild_test_thing_policy.json"
```

Jika perintah berhasil, ia mengembalikan respons seperti ini. Perhatikan bahwa `policyVersionId` sekarang 2, menunjukkan ini adalah versi kedua dari kebijakan ini.

Jika Anda berhasil memperbarui kebijakan, Anda dapat melanjutkan ke prosedur berikutnya.

```
{
  "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
}

```

Jika Anda mendapatkan kesalahan bahwa ada terlalu banyak versi kebijakan untuk menyimpan yang baru, masukkan perintah ini untuk mencantumkan versi kebijakan saat ini. Tinjau daftar yang dikembalikan perintah ini untuk menemukan versi kebijakan yang dapat Anda hapus.

```
aws iot list-policy-versions --policy-name "PubSubTestThingPolicy"
```

Masukkan perintah ini untuk menghapus versi yang tidak lagi Anda butuhkan. Perhatikan bahwa Anda tidak dapat menghapus versi kebijakan default. Versi kebijakan default adalah salah satu dengan `isDefaultVersion` nilai `true`.

```
aws iot delete-policy-version \
--policy-name "PubSubTestThingPolicy" \
--policy-version-id policyId
```

Setelah menghapus versi kebijakan, coba lagi langkah ini.

Dengan file konfigurasi yang diperbarui dan kebijakan, Anda siap untuk menunjukkan langganan wild card dengan AWS IoT CLI perangkat.

Untuk menunjukkan bagaimana AWS IoT CLI Perangkat berlangganan dan menerima beberapa topik pesan MQTT

1. Di CLI uji MQTT, periksa langganan. Jika CLI uji MQTT berlangganan ke dalam `#filter` topik, lanjutkan ke langkah berikutnya. Jika tidak, di CLI uji MQTT, di Berlangganan topik tab, di Filter topik ENTER `#` (karakter tanda pound), dan kemudian pilih Langganan untuk berlangganan itu.
2. Di jendela terminal pada komputer host lokal Anda yang terhubung ke Raspberry Pi Anda, masukkan perintah ini untuk memulai AWS IoT CLI perangkat.

```
cd ~/aws-iot-device-client/build
./aws-iot-device-client --config-file ~/dc-configs/dc-pubsub-wild-config.json
```

3. Sambil menonton AWS IoT Output perangkat Client di jendela terminal pada komputer host lokal, kembali ke Klien uji MQTT. Di Menerbitkan ke topik tab, di Nama topik ENTER **test/dc/subtopic**, dan kemudian pilih Publikasikan.
4. Di jendela terminal, konfirmasi bahwa pesan diterima dengan mencari pesan seperti:

```
2021-11-10T16:34:20.101Z [DEBUG] {samples/PubSubFeature.cpp}: Message received on
subscribe topic, size: 76 bytes
```

5. Sambil menonton AWS IoT Output perangkat Client di jendela terminal komputer host lokal, kembali ke Klien uji MQTT. Di Menerbitkan ke topik tab, di Nama topik ENTER **test/dc/subtopic2**, dan kemudian pilih Publikasikan.
6. Di jendela terminal, konfirmasi bahwa pesan diterima dengan mencari pesan seperti:

```
2021-11-10T16:34:32.078Z [DEBUG] {samples/PubSubFeature.cpp}: Message received on
subscribe topic, size: 77 bytes
```

7. Setelah Anda melihat pesan yang mengonfirmasi kedua pesan tersebut diterima, masukkan **^C** (Ctrl-C) untuk menghentikan AWS IoT Klien perangkat.
8. Masukkan perintah ini untuk melihat akhir file log pesan dan melihat pesan yang Anda publikasikan dari Klien uji MQTT.

```
tail -n 20 ~/.aws-iot-device-client/log/pubsub_rx_msgs.log
```

Note

File log hanya berisi muatan pesan. Topik pesan tidak direkam dalam file log pesan yang diterima.

Anda mungkin juga melihat pesan yang diterbitkan oleh AWS IoT Perangkat Client di log yang diterima. Hal ini karena filter topik wild card termasuk topik pesan itu dan, kadang-kadang, permintaan berlangganan dapat diproses oleh broker pesan sebelum pesan yang diterbitkan dikirim ke pelanggan.

Entri dalam file log menunjukkan bahwa pesan diterima. Anda dapat mengulangi prosedur ini menggunakan nama topik lainnya. Semua pesan yang memiliki nama topik yang dimulai dengan **test/dc/** harus diterima dan dicatat. Pesan dengan nama topik yang dimulai dengan teks lain diabaikan.

Setelah menunjukkan bagaimana AWS IoT Perangkat Client dapat mempublikasikan dan berlangganan pesan MQTT, lanjutkan ke [Tutorial: Menunjukkan tindakan jarak jauh \(pekerjaan\) dengan AWS IoT Perangkat perangkat perangkat perangkat perangkat](#).

Tutorial: Menunjukkan tindakan jarak jauh (pekerjaan) dengan AWS IoT Perangkat perangkat perangkat perangkat perangkat

Dalam tutorial ini, Anda akan mengkonfigurasi dan menyebarkan pekerjaan ke Raspberry Pi Anda untuk menunjukkan bagaimana Anda dapat mengirim operasi jarak jauh ke perangkat IoT Anda.

Untuk memulai tutorial ini:

- Minta komputer host lokal Anda yang dikonfigurasi Raspberry Pi seperti yang digunakan [bagian sebelumnya](#).
- Jika Anda belum menyelesaikan tutorial di bagian sebelumnya, Anda dapat mencoba tutorial ini dengan menggunakan Raspberry Pi dengan kartu microSD yang memiliki gambar yang Anda simpan setelah Anda instal AWS IoT Perangkat perangkat perangkat perangkat perangkat perangkat ([Opsional](#)) [Simpan gambar kartu microSD](#).
- Jika Anda telah menjalankan demo ini sebelumnya, tinjau [???](#) untuk menghapus semua AWS IoT sumber daya yang Anda buat di awal berjalan untuk menghindari kesalahan sumber daya duplikat.

Tutorial ini memakan waktu sekitar 45 menit untuk menyelesaikannya.

Setelah selesai dengan topik ini:

- Anda akan menunjukkan berbagai cara yang dapat digunakan perangkat IoT Anda AWS IoT Core untuk menjalankan operasi jarak jauh yang dikelola oleh AWS IoT.

Peralatan yang dibutuhkan:

- Lingkungan pengembangan dan pengujian lokal Anda yang Anda uji [bagian sebelumnya](#)
- Raspberry Pi yang Anda uji [bagian sebelumnya](#)
- Kartu memori microSD dari Raspberry Pi yang Anda uji [bagian sebelumnya](#)

Prosedur dalam tutorial ini

- [Langkah 1: Siapkan Raspberry Pi untuk menjalankan pekerjaan](#)

- [Langkah 2: Buat dan jalankan tugas AWS IoT](#)

Langkah 1: Siapkan Raspberry Pi untuk menjalankan pekerjaan

Prosedur di bagian ini menjelaskan cara mempersiapkan Raspberry Pi Anda untuk menjalankan tugas dengan menggunakan AWS IoT Perangkat perangkat perangkat perangkat perangkat perangkat

Note

Prosedur ini khusus perangkat. Jika Anda ingin melakukan prosedur di bagian ini dengan lebih dari satu perangkat pada saat yang sama, setiap perangkat akan memerlukan kebijakan sendiri dan unik, sertifikat khusus perangkat dan nama benda. Untuk memberi setiap perangkat sumber daya uniknya, lakukan prosedur ini satu kali untuk setiap perangkat sambil mengubah elemen khusus perangkat seperti yang dijelaskan dalam prosedur.

Prosedur dalam tutorial ini

- [Berikan Raspberry Pi Anda untuk mendemonstrasikan pekerjaan](#)
- [Konfigurasi AWS IoT Klien Perangkat untuk menjalankan agen pekerjaan](#)

Berikan Raspberry Pi Anda untuk mendemonstrasikan pekerjaan

Prosedur di bagian ini menyediakan Raspberry Pi Anda AWS IoT dengan menciptakan AWS IoT sumber daya dan sertifikat perangkat untuk itu.

Membuat dan mengunduh file sertifikat perangkat untuk diperagakan AWS IoT pekerjaan

Prosedur ini membuat file sertifikat perangkat untuk demo ini.

Jika Anda menyiapkan lebih dari satu perangkat, prosedur ini harus dilakukan pada setiap perangkat.

Untuk membuat dan mengunduh file sertifikat perangkat untuk Raspberry Pi Anda:

Di jendela terminal di komputer host lokal Anda yang terhubung ke Raspberry Pi Anda, masukkan perintah ini.

1. Masukkan perintah berikut untuk membuat file sertifikat perangkat untuk perangkat Anda.

```
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"
```

Perintah mengembalikan respons seperti berikut ini. Simpan `certificateArn` nilai untuk digunakan nanti.

```
{
  "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-----
\nMIIEowIBAACAQE_SHORTENED_FOR_EXAMPLE_T9RoDiukY\n-----END RSA PRIVATE KEY-----\n"
  }
}
```

2. Masukkan perintah berikut untuk mengatur izin pada direktori sertifikat dan file-file-nya.

```
chmod 700 ~/certs/jobs
chmod 644 ~/certs/jobs/*
chmod 600 ~/certs/jobs/private.pem.key
```

3. Jalankan perintah ini untuk meninjau izin pada direktori dan file sertifikat Anda.

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

Output dari perintah harus sama dengan apa yang Anda lihat di sini, kecuali tanggal dan waktu file akan berbeda.

```
-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
```

Setelah Anda mengunduh file sertifikat perangkat ke Raspberry Pi Anda, Anda siap untuk melanjutkan [the section called “Berikan Raspberry Pi Anda untuk mendemonstrasikan pekerjaan”](#).

Buat AWS IoT sumber daya untuk menunjukkan AWS IoT pekerjaan

Buat AWS IoT sumber daya untuk perangkat ini.

Jika Anda menyiapkan lebih dari satu perangkat, prosedur ini harus dilakukan untuk setiap perangkat.

Untuk menyediakan perangkat Anda AWS IoT:

Di jendela terminal di komputer host lokal Anda yang terhubung ke Raspberry Pi Anda:

1. Masukkan perintah berikut untuk mendapatkan alamat endpoint data perangkat perangkat perangkat perangkat perangkat perangkat perangkat perangkat perangkat perangkat perangkat perangkat Anda Akun AWS.

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

Nilai titik akhir tidak berubah sejak terakhir kali Anda menjalankan perintah ini. Menjalankan perintah lagi di sini membuatnya mudah untuk menemukan dan menempelkan nilai endpoint data ke file konfigurasi yang digunakan dalam tutorial ini.

Klaster `describe-endpoint` Perintah mengembalikan respons seperti berikut ini.

Catat *endpointAddress* nilai untuk digunakan nanti.

```
{
  "endpointAddress": "a3qjEXAMPLEffp-ats.iot.us-west-2.amazonaws.com"
}
```

2. Ganti *uniqueThingName* dengan nama unik perangkat perangkat perangkat perangkat perangkat perangkat perangkat perangkat perangkat perangkat perangkat Anda. Jika Anda ingin melakukan tutorial ini dengan beberapa perangkat, beri nama masing-masing perangkat sendiri. Misalnya, `TestDevice01`, `TestDevice02`, dan sebagainya.

Masukkan perintah ini untuk membuat AWS IoT sumber daya untuk Raspberry Pi Anda.


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

Karena AWS IoT hal sumber daya adalah representasi dari perangkat Anda di awan, kita dapat membuat beberapa sumber daya hal di AWS IoT untuk digunakan untuk tujuan yang berbeda. Semuanya dapat digunakan oleh perangkat IoT fisik yang sama untuk mewakili berbagai aspek perangkat.

Note

Saat Anda ingin mengamankan kebijakan untuk beberapa perangkat, Anda dapat menggunakannya `{iot:Thing.ThingName}` bukan nama hal statis, `uniqueThingName`.

Tutorial ini hanya akan menggunakan satu hal sumber daya pada satu waktu per perangkat. Dengan cara ini, dalam tutorial ini, mereka mewakili demo yang berbeda sehingga setelah Anda membuat AWS IoT sumber daya untuk demo, Anda dapat kembali dan mengulangi demo menggunakan sumber daya yang Anda buat khusus untuk masing-masing.

Jika AWS IoT sumber daya hal diciptakan, sistem menampilkan respon seperti ini. Catat `thingArn` nilai untuk digunakan nanti saat Anda membuat tugas untuk dijalankan di perangkat ini.

```
{
  "thingName": "uniqueThingName",
  "thingArn": "arn:aws:iot:us-west-2:57EXAMPLE833:thing/uniqueThingName",
  "thingId": "8ea78707-32c3-4f8a-9232-14bEXAMPLEfd"
}
```

3. Di jendela terminal:

- a. Buka editor teks, seperti nano.
- b. Salin dokumen JSON ini dan tempelkan ke editor teks terbuka Anda.

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

```

    "Effect": "Allow",
    "Action": [
      "iot:Connect"
    ],
    "Resource": [
      "arn:aws:iot:us-west-2:57EXAMPLE833:client/uniqueThingName"
    ]
  },
  {
    "Effect": "Allow",
    "Action": [
      "iot:Publish"
    ],
    "Resource": [
      "arn:aws:iot:us-west-2:57EXAMPLE833:topic/test/dc/pubtopic",
      "arn:aws:iot:us-west-2:57EXAMPLE833:topic/$aws/events/job/*",
      "arn:aws:iot:us-west-2:57EXAMPLE833:topic/$aws/events/jobExecution/*",
      "arn:aws:iot:us-west-2:57EXAMPLE833:topic/$aws/things/uniqueThingName/
jobs/*"
    ]
  },
  {
    "Effect": "Allow",
    "Action": [
      "iot:Subscribe"
    ],
    "Resource": [
      "arn:aws:iot:us-west-2:57EXAMPLE833:topicfilter/test/dc/subtopic",
      "arn:aws:iot:us-west-2:57EXAMPLE833:topic/$aws/events/jobExecution/*",
      "arn:aws:iot:us-west-2:57EXAMPLE833:topicfilter/$aws/
things/uniqueThingName/jobs/*"
    ]
  },
  {
    "Effect": "Allow",
    "Action": [
      "iot:Receive"
    ],
    "Resource": [
      "arn:aws:iot:us-west-2:57EXAMPLE833:topic/test/dc/subtopic",
      "arn:aws:iot:us-west-2:57EXAMPLE833:topic/$aws/things/uniqueThingName/
jobs/*"
    ]
  },
},

```

```

    {
      "Effect": "Allow",
      "Action": [
        "iot:DescribeJobExecution",
        "iot:GetPendingJobExecutions",
        "iot:StartNextPendingJobExecution",
        "iot:UpdateJobExecution"
      ],
      "Resource": [
        "arn:aws:iot:us-west-2:57EXAMPLE833:topic/$aws/things/uniqueThingName"
      ]
    }
  ]
}

```

- c. Di editor, diResource bagian dari setiap pernyataan kebijakan, ganti *us-west-2:57EXAMPLE833* dengan Wilayah AWS, karakter titik dua (:), dan 12 digit Akun AWS Angka.
- d. Di editor, di setiap pernyataan kebijakan, ganti *uniqueThingName* dengan nama hal yang Anda berikan sumber daya hal ini.
- e. Simpan file dalam editor teks Anda sebagai **~/policies/jobs_test_thing_policy.json**.

Jika Anda menjalankan prosedur ini untuk beberapa perangkat, simpan file ke nama file ini di setiap perangkat.

4. Ganti *uniqueThingName* dengan nama benda untuk perangkat, dan kemudian jalankan perintah ini untuk membuat AWS IoT kebijakan yang disesuaikan untuk perangkat itu.

```

aws iot create-policy \
--policy-name "JobTestPolicyForuniqueThingName" \
--policy-document "file://~/policies/jobs_test_thing_policy.json"

```

Jika kebijakan dibuat, sistem menampilkan respon seperti ini.

```

{
  "policyName": "JobTestPolicyForuniqueThingName",
  "policyArn": "arn:aws:iot:us-west-2:57EXAMPLE833:policy/JobTestPolicyForuniqueThingName",
  "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\"
```

- Ganti *uniqueThingName* dengan nama benda untuk perangkat dan *certificateArn* dengan *certificateArn* nilai yang Anda simpan sebelumnya di bagian ini untuk perangkat ini, lalu jalankan perintah ini untuk melampirkan kebijakan ke sertifikat perangkat.

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

Jika berhasil, perintah ini tidak mengembalikan apa pun.

- Ganti *uniqueThingName* dengan nama benda untuk perangkat, ganti *certificateArn* dengan *certificateArn* nilai yang Anda simpan sebelumnya di bagian ini, lalu jalankan perintah ini untuk melampirkan sertifikat perangkat ke AWS IoT sumber daya hal.

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

Jika berhasil, perintah ini tidak mengembalikan apa pun.

Setelah Anda berhasil menyediakan Raspberry Pi Anda, Anda siap untuk mengulangi bagian ini untuk Raspberry Pi lain dalam pengujian Anda atau, jika semua perangkat telah disediakan, lanjutkan [the section called “Konfigurasi AWS IoT Klien Perangkat untuk menjalankan agen pekerjaan”](#).

Konfigurasi AWS IoT Klien Perangkat untuk menjalankan agen pekerjaan

Prosedur ini membuat file konfigurasi untuk AWS IoT Perangkat Client untuk menjalankan agen pekerjaan:.

Catatan: jika Anda menyiapkan lebih dari satu perangkat, prosedur ini harus dilakukan pada setiap perangkat.

Untuk membuat file konfigurasi untuk mengujiAWS IoTPerangkat perangkat perangkat perangkat perangkat perangkat perangkat

1. Di jendela terminal di komputer host lokal Anda yang terhubung ke Raspberry Pi Anda:
 - a. Buka editor teks, seperti nano.
 - b. Salin dokumen JSON ini dan tempelkan ke editor teks terbuka Anda.

```
{
  "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. Ganti *titik akhir* nilai dengan nilai endpoint data perangkat untuk Akun AWS yang kamu temukan [the section called “Menyediakan perangkat Anda AWS IoT Core”](#).
 - d. Ganti *unique ThingName* dengan nama benda yang Anda gunakan untuk perangkat ini.
 - e. Simpan file dalam editor teks Anda sebagai `~/dc-configs/dc-jobs-config.json`.
2. Jalankan perintah ini untuk mengatur izin file dari file konfigurasi baru.

```
chmod 644 ~/dc-configs/dc-jobs-config.json
```

Anda tidak akan menggunakan klien uji uji MQTT untuk tes ini. Sementara perangkat akan bertukar pesan MQTT terkait pekerjaan dengan AWS IoT, pesan kemajuan pekerjaan hanya dipertukarkan dengan perangkat yang menjalankan pekerjaan. Karena pesan kemajuan pekerjaan hanya dipertukarkan dengan perangkat yang menjalankan pekerjaan, Anda tidak dapat berlangganan pesan tersebut dari perangkat lain, seperti AWS IoT konsol.

Setelah menyimpan file konfigurasi, Anda siap untuk melanjutkan [the section called “Langkah 2: Buat dan jalankan tugas AWS IoT”](#).

Langkah 2: Buat dan jalankan tugas AWS IoT

Prosedur di bagian ini membuat dokumen pekerjaan dan AWS IoT sumber daya pekerjaan. Setelah Anda membuat sumber daya pekerjaan, AWS IoT mengirimkan dokumen pekerjaan ke target pekerjaan yang ditentukan di mana agen pekerjaan menerapkan dokumen pekerjaan ke perangkat atau klien.

Prosedur di bagian ini

- [Membuat dan menyimpan dokumen pekerjaan](#)
- [Jalankan tugasAWS IoTuntuk satu perangkat IoT](#)

Membuat dan menyimpan dokumen pekerjaan

Prosedur ini membuat dokumen pekerjaan sederhana untuk dimasukkan dalamAWS IoTsumber daya pekerjaan. Dokumen pekerjaan ini menampilkan “Hello world!” pada target pekerjaan.

Untuk membuat dan menyimpan dokumen pekerjaan:

1. Pilih bucket Amazon S3 tempat Anda akan menyimpan dokumen pekerjaan Anda. Jika Anda tidak memiliki sebuah bucket Amazon S3 yang belum ada untuk digunakan untuk hal ini, Anda harus membuatnya. Untuk informasi tentang cara membuat bucket Amazon S3, lihat topik di[Memulai dengan Amazon S3](#).
2. Buat dan simpan dokumen tugas untuk tugas ini
 - a. Di komputer host lokal Anda, buka editor teks.
 - b. Salin dan tempel teks ini ke editor.

```
{
  "operation": "echo",
  "args": ["Hello world!"]
}
```

- c. Di komputer host lokal, simpan isi editor ke file bernama**hello-world-job.json**.
 - d. Konfirmasikan bahwa file telah disimpan dengan benar. Beberapa editor teks secara otomatis menambahkan .txtke nama file saat mereka menyimpan file teks. Jika editor Anda ditambahkan .txtke nama file, perbaiki nama file sebelum melanjutkan.
3. Gantipath_to_filedengan jalan menuju**hello-world-job.json**, jika tidak ada di direktori Anda saat ini, gantis3_bucket_namedengan jalur bucket Amazon S3 ke bucket yang Anda pilih, lalu jalankan perintah ini untuk memasukkan dokumen pekerjaan Anda ke bucket Amazon S3.

```
aws s3api put-object \
--key hello-world-job.json \
--body path_to_file/hello-world-job.json --bucket s3_bucket_name
```

URL dokumen pekerjaan yang mengidentifikasi dokumen pekerjaan yang Anda simpan di Amazon S3 ditentukan dengan mengganti `s3_bucket_name` dan `AWS_Wilayah` dalam URL berikut ini. Rekam URL yang dihasilkan untuk digunakan nanti sebagai `job_document_path`

```
https://s3_bucket_name.s3.AWS_Region.amazonaws.com/hello-world-job.json
```

Note

AWS keamanan mencegah Anda untuk dapat membuka URL ini di luar Akun AWS, misalnya dengan menggunakan browser. URL digunakan oleh AWS IoT pekerjaan mesin, yang memiliki akses ke file, secara default. Dalam lingkungan produksi, Anda harus memastikan bahwa AWS IoT layanan memiliki izin untuk mengakses dokumen tugas yang disimpan dalam Amazon S3.

Setelah Anda menyimpan URL dokumen pekerjaan, lanjutkan [the section called “Jalankan tugas AWS IoT untuk satu perangkat IoT”](#).

Jalankan tugas AWS IoT untuk satu perangkat IoT

Prosedur di bagian ini memulai AWS IoT Klien Perangkat di Raspberry Pi Anda untuk menjalankan agen pekerjaan di perangkat untuk menunggu pekerjaan berjalan. Hal ini juga menciptakan sumber daya pekerjaan di AWS IoT, yang akan mengirim pekerjaan ke dan berjalan di perangkat IoT Anda.

Note

Prosedur ini menjalankan pekerjaan hanya pada satu perangkat.

Untuk memulai agen pekerjaan di Raspberry Pi Anda:

1. Di jendela terminal di komputer host lokal Anda yang terhubung ke Raspberry Pi Anda, jalankan perintah ini untuk memulai AWS IoT Perangkat perangkat perangkat perangkat perangkat perangkat

```
cd ~/aws-iot-device-client/build
./aws-iot-device-client --config-file ~/dc-configs/dc-jobs-config.json
```


2. Di jendela terminal terminal, konfirmasi bahwa AWS IoT Client Perangkat dan menampilkan pesan-pesan ini

```
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. Di jendela terminal, setelah Anda melihat pesan ini, lanjutkan ke prosedur berikutnya dan buat sumber daya pekerjaan. Perhatikan bahwa itu mungkin bukan entri terakhir dalam daftar.

```
2021-11-15T18:45:56.785Z [INFO] {JobsFeature.cpp}: No pending jobs are scheduled,
waiting for the next incoming job
```

Untuk membuat AWS IoT sumber daya pekerjaan

1. Di komputer host lokal Anda:
 - a. Ganti `job_document_url` dengan URL dokumen pekerjaan dari [the section called "Membuat dan menyimpan dokumen pekerjaan"](#).
 - b. Ganti `thing_id` dengan ARN dari sumber daya hal yang Anda buat untuk perangkat Anda dan kemudian jalankan perintah ini.

```
aws iot create-job \  
--job-id hello-world-job-1 \  
--document-source "job_document_url" \  
--targets "thing_arn" \  
--target-selection SNAPSHOT
```

Jika berhasil, sistem menampilkan hasil seperti ini.

```
{  
  "jobArn": "arn:aws:iot:us-west-2:57EXAMPLE833:job/hello-world-job-1",  
  "jobId": "hello-world-job-1"  
}
```

2. Di jendela terminal, Anda akan melihat output dari AWS IoT Klien Perangkat seperti ini.

```
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 EphemeralPromise 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 EphemeralPromise 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. Sementara AWS IoT Device Client sedang berjalan dan menunggu pekerjaan, Anda dapat mengirimkan pekerjaan lain dengan mengubah `job-id` nilai dan kembali menjalankan `create-job` dari Langkah 1:

Setelah selesai menjalankan pekerjaan, di jendela terminal, masukkan `^C` (Control-c) untuk menghentikan AWS IoT Perangkat perangkat perangkat perangkat perangkat perangkat perangkat

Tutorial: Membersihkan setelah menjalankan tutorial AWS IoT Device Client

Prosedur dalam tutorial ini memandu Anda melalui menghapus file dan sumber daya yang Anda buat saat menyelesaikan tutorial di jalur pembelajaran ini.

Prosedur dalam tutorial ini

- [Langkah 1: Membersihkan perangkat Anda setelah membuat demo dengan Klien AWS IoT Perangkat](#)
- [Langkah 2: Membersihkan demo Akun AWS setelah membangun Anda dengan Klien AWS IoT Perangkat](#)

Langkah 1: Membersihkan perangkat Anda setelah membuat demo dengan Klien AWS IoT Perangkat

Tutorial ini menjelaskan dua opsi untuk cara membersihkan kartu microSD setelah Anda membuat demo di jalur pembelajaran ini. Pilih opsi yang menyediakan tingkat keamanan yang Anda butuhkan.

Perhatikan bahwa membersihkan kartu microSD perangkat tidak menghapus AWS IoT sumber daya apa pun yang Anda buat. Untuk membersihkan AWS IoT sumber daya setelah Anda membersihkan kartu microSD perangkat, Anda harus meninjau tutorialnya. [the section called “Membersihkan setelah membangun demo dengan Klien AWS IoT Perangkat”](#)

Opsi 1: Membersihkan dengan menulis ulang kartu microSD

Cara termudah dan paling menyeluruh untuk membersihkan kartu microSD setelah menyelesaikan tutorial di jalur pembelajaran ini adalah dengan menimpa kartu microSD dengan file gambar tersimpan yang Anda buat saat menyiapkan perangkat Anda untuk pertama kalinya.

Prosedur ini menggunakan komputer host lokal untuk menulis gambar kartu microSD yang disimpan ke kartu microSD.

Note

Jika perangkat Anda tidak menggunakan media penyimpanan yang dapat dilepas untuk sistem operasinya, lihat prosedur untuk perangkat tersebut.

Untuk menulis gambar baru ke kartu microSD

1. Di komputer host lokal Anda, cari gambar kartu microSD yang disimpan yang ingin Anda tulis ke kartu microSD Anda.
2. Masukkan kartu microSD Anda ke komputer host lokal.
3. Menggunakan alat pencitraan kartu SD, tulis file gambar yang dipilih ke kartu microSD.
4. Setelah menulis gambar Raspberry Pi OS ke kartu microSD, keluarkan kartu microSD dan lepaskan dengan aman dari komputer host lokal.

Kartu microSD Anda siap digunakan.

Opsi 2: Membersihkan dengan menghapus direktori pengguna

Untuk membersihkan kartu microSD setelah menyelesaikan tutorial tanpa menulis ulang gambar kartu microSD, Anda dapat menghapus direktori pengguna satu per satu. Ini tidak selengkap menulis ulang kartu microSD dari gambar yang disimpan karena tidak menghapus file sistem yang mungkin telah diinstal.

Jika menghapus direktori pengguna cukup menyeluruh untuk kebutuhan Anda, Anda dapat mengikuti prosedur ini.

Untuk menghapus direktori pengguna jalur pembelajaran ini dari perangkat

1. Jalankan perintah ini untuk menghapus direktori pengguna, subdirektori, dan semua file mereka yang dibuat di jalur pembelajaran ini, di jendela terminal yang terhubung ke perangkat Anda.

Note

Setelah Anda menghapus direktori dan file ini, Anda tidak akan dapat menjalankan demo tanpa menyelesaikan tutorial lagi.

```
rm -Rf ~/dc-configs
rm -Rf ~/policies
rm -Rf ~/messages
rm -Rf ~/certs
rm -Rf ~/.aws-iot-device-client
```

2. Jalankan perintah ini untuk menghapus direktori dan file sumber aplikasi, di jendela terminal yang terhubung ke perangkat Anda.

Note

Perintah ini tidak menghapus program apa pun. Mereka hanya menghapus file sumber yang digunakan untuk membangun dan menginstalnya. Setelah Anda menghapus file-file ini, AWS CLI dan Klien AWS IoT Perangkat mungkin tidak berfungsi.

```
rm -Rf ~/aws-cli
rm -Rf ~/aws
rm -Rf ~/aws-iot-device-client
```

Langkah 2: Membersihkan demo Akun AWS setelah membangun Anda dengan Klien AWS IoT Perangkat

Prosedur ini membantu Anda mengidentifikasi dan menghapus AWS sumber daya yang Anda buat saat menyelesaikan tutorial di jalur pembelajaran ini.

Bersihkan AWS IoT sumber daya

Prosedur ini membantu Anda mengidentifikasi dan menghapus AWS IoT sumber daya yang Anda buat saat menyelesaikan tutorial di jalur pembelajaran ini.

AWS IoT sumber daya yang dibuat di jalur pembelajaran ini

Tutorial	Hal sumber daya	Sumber daya kebijakan
the section called “Menginstal dan mengkonfigurasi AWS IoT Perangkat klien”	DevCliTestThing	DevCliTestThingPolicy
the section called “Menunjukkan komunikasi pesan MQTT dengan AWS IoT Klien perangkat”	PubSubTestThing	PubSubTestThingPolicy

Tutorial	Hal sumber daya	Sumber daya kebijakan
the section called “Menunjukkan tindakan jarak jauh (pekerjaan) denganAWS IoTPerangkat perangkat perangkat perangkat”	didefinisikan pengguna (mungkin ada lebih dari satu)	didefinisikan pengguna (mungkin ada lebih dari satu)

Untuk menghapus AWS IoT sumber daya, ikuti prosedur ini untuk setiap sumber daya hal yang Anda buat

1. Ganti *thing_name* dengan nama sumber daya hal yang ingin Anda hapus, dan kemudian jalankan perintah ini untuk mencantumkan sertifikat yang dilampirkan ke sumber daya, dari komputer host lokal.

```
aws iot list-thing-principals --thing-name thing_name
```

Perintah ini mengembalikan respon seperti ini yang berisi daftar sertifikat yang dilampirkan *thing_name*. Dalam kebanyakan kasus, hanya akan ada satu sertifikat dalam daftar.

```
{
  "principals": [
    "arn:aws:iot:us-west-2:57EXAMPLE833:cert/23853eea3cf0edc7f8a69c74abeafa27b2b52823cab5b3e156295e94b26ae8ac"
  ]
}
```

2. Untuk setiap sertifikat yang dicantumkan oleh perintah sebelumnya:
 - a. Ganti *certificate_ID* dengan ID sertifikat dari perintah sebelumnya. ID sertifikat adalah karakter alfanumerik yang mengikuti cert/ ARN yang dikembalikan oleh perintah sebelumnya. Kemudian jalankan perintah ini untuk menonaktifkan sertifikat.

```
aws iot update-certificate --new-status INACTIVE --certificate-id certificate_ID
```

Jika berhasil, perintah ini tidak mengembalikan apa pun.

- b. Ganti *certificate_ARN* dengan ARN sertifikat dari daftar sertifikat yang dikembalikan sebelumnya, lalu jalankan perintah ini untuk mencantumkan kebijakan yang dilampirkan ke sertifikat ini.

```
aws iot list-attached-policies --target certificate_ARN
```

Perintah ini mengembalikan respon seperti ini yang mencantumkan kebijakan yang dilampirkan ke sertifikat. Dalam kebanyakan kasus, hanya akan ada satu kebijakan dalam daftar.

```
{
  "policies": [
    {
      "policyName": "DevCliTestThingPolicy",
      "policyArn": "arn:aws:iot:us-west-2:57EXAMPLE833:policy/DevCliTestThingPolicy"
    }
  ]
}
```

- c. Untuk setiap kebijakan yang dilampirkan pada sertifikat:
 - i. Ganti *policy_name* dengan `policyName` nilai dari perintah sebelumnya, ganti *certificate_ARN* dengan ARN sertifikat, dan kemudian jalankan perintah ini untuk melepaskan kebijakan dari sertifikat.

```
aws iot detach-policy --policy-name policy_name --target certificate_ARN
```

Jika berhasil, perintah ini tidak mengembalikan apa pun.

- ii. Ganti *policy_name* dengan `policyName` nilai, dan kemudian jalankan perintah ini untuk melihat apakah kebijakan dilampirkan ke sertifikat lainnya.

```
aws iot list-targets-for-policy --policy-name policy_name
```

Jika perintah mengembalikan daftar kosong seperti ini, kebijakan tidak dilampirkan ke sertifikat apa pun dan Anda terus mencantumkan versi kebijakan. Jika masih ada

sertifikat yang dilampirkan pada kebijakan, lanjutkan dengan detach-thing-principal langkahnya.

```
{
  "targets": []
}
```

- iii. Ganti *policy_name* dengan `policyName` nilai, dan kemudian jalankan perintah ini untuk memeriksa versi kebijakan. Untuk menghapus kebijakan, itu harus memiliki hanya satu versi.

```
aws iot list-policy-versions --policy-name policy_name
```

Jika kebijakan hanya memiliki satu versi, seperti contoh ini, Anda dapat melompat ke delete-policy langkah dan menghapus kebijakan sekarang.

```
{
  "policyVersions": [
    {
      "versionId": "1",
      "isDefaultVersion": true,
      "createDate": "2021-11-18T01:02:46.778000+00:00"
    }
  ]
}
```

Jika kebijakan memiliki lebih dari satu versi, seperti contoh ini, versi kebijakan dengan `isDefaultVersion` nilai `false` harus dihapus sebelum kebijakan dapat dihapus.

```
{
  "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"
    }
  ]
}
```

```
]
}
```

Jika Anda perlu menghapus versi kebijakan, ganti *policy_name* dengan `policyName` nilai, ganti *version_ID* dengan `versionId` nilai dari perintah sebelumnya, dan kemudian jalankan perintah ini untuk menghapus versi kebijakan.

```
aws iot delete-policy-version --policy-name policy_name --policy-version-id version_ID
```

Jika berhasil, perintah ini tidak mengembalikan apa pun.

Setelah Anda menghapus versi kebijakan, ulangi langkah ini hingga kebijakan hanya memiliki satu versi kebijakan.

- iv. Ganti *policy_name* dengan `policyName` nilai, dan kemudian jalankan perintah ini untuk menghapus kebijakan.

```
aws iot delete-policy --policy-name policy_name
```

- d. Ganti *thing_name* dengan nama benda itu, ganti *certificate_ARN* dengan ARN sertifikat, dan kemudian jalankan perintah ini untuk melepaskan sertifikat dari sumber daya benda.

```
aws iot detach-thing-principal --thing-name thing_name --principal certificate_ARN
```

Jika berhasil, perintah ini tidak mengembalikan apa pun.

- e. Ganti *certificate_ID* dengan ID sertifikat dari perintah sebelumnya. ID sertifikat adalah karakter alfanumerik yang mengikuti `cert/` ARN yang dikembalikan oleh perintah sebelumnya. Kemudian jalankan perintah ini untuk menghapus sumber daya sertifikat.

```
aws iot delete-certificate --certificate-id certificate_ID
```

Jika berhasil, perintah ini tidak mengembalikan apa pun.

3. Ganti *thing_name* dengan nama benda itu, lalu jalankan perintah ini untuk menghapusnya.

```
aws iot delete-thing --thing-name thing_name
```

Jika berhasil, perintah ini tidak mengembalikan apa pun.

Bersihkan AWS sumber daya

Prosedur ini membantu Anda mengidentifikasi dan menghapus AWS sumber daya lain yang Anda buat saat menyelesaikan tutorial di jalur pembelajaran ini.

AWS Sumber daya lain yang dibuat di jalur pembelajaran ini

Tutorial	Tipe sumber daya	Nama sumber daya atau ID
the section called “Menunjukkan tindakan jarak jauh (pekerjaan) dengan AWS IoT Perangkat perangkat perangkat perangkat”	Objek Amazon S3	hello-world-job.json
the section called “Menunjukkan tindakan jarak jauh (pekerjaan) dengan AWS IoT Perangkat perangkat perangkat perangkat”	AWS IoT sumber daya pekerjaan	pengguna didefinisikan

Untuk menghapus AWS sumber daya yang dibuat di jalur pembelajaran ini

1. Untuk menghapus pekerjaan yang dibuat di jalur pembelajaran ini
 - a. Jalankan perintah ini untuk daftar pekerjaan di Anda Akun AWS.

```
aws iot list-jobs
```

Perintah mengembalikan daftar AWS IoT pekerjaan di Anda Akun AWS dan Wilayah AWS yang terlihat seperti ini.

```
{
  "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. Untuk setiap pekerjaan yang Anda kenali dari daftar sebagai pekerjaan yang Anda buat di jalur pembelajaran ini, ganti *jobId* dengan *jobId* nilai pekerjaan yang akan dihapus, lalu jalankan perintah ini untuk menghapus AWS IoT pekerjaan.

```
aws iot delete-job --job-id jobId
```

Jika perintah berhasil, ia tidak mengembalikan apa-apa.

2. Untuk menghapus dokumen pekerjaan yang Anda simpan di bucket Amazon S3 di jalur pembelajaran ini.
- a. Ganti *bucket* dengan nama bucket yang Anda gunakan, lalu jalankan perintah ini untuk mencantumkan objek di bucket Amazon S3 yang Anda gunakan.

```
aws s3api list-objects --bucket bucket
```

Perintah mengembalikan daftar objek Amazon S3 dalam bucket yang terlihat seperti ini.

```
{
```

```

"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. Untuk setiap objek yang Anda kenali dari daftar sebagai objek yang Anda buat di jalur pembelajaran ini, ganti *bucket* dengan nama bucket dan *key* dengan nilai kunci objek yang akan dihapus, lalu jalankan perintah ini untuk menghapus objek Amazon S3.

```
aws s3api delete-object --bucket bucket --key key
```

Jika perintah berhasil, ia tidak mengembalikan apa-apa.

Setelah Anda menghapus semua AWS sumber daya dan objek yang Anda buat saat menyelesaikan jalur pembelajaran ini, Anda dapat memulai kembali dan mengulangi tutorial.

Membangun solusi denganAWS IoTPerangkat SDK

Tutorial di bagian ini membantu memandu Anda melalui langkah-langkah untuk mengembangkan solusi IoT yang dapat digunakan untuk lingkungan produksi menggunakanAWS IoT.

Tutorial ini dapat mengambil lebih banyak waktu untuk diselesaikan daripada yang ada di bagian[the section called “Membangun demo denganAWS IoTPerangkat”](#)karena mereka menggunakanAWS IoTSDK perangkat dan menjelaskan konsep yang diterapkan secara lebih rinci untuk membantu Anda menciptakan solusi yang aman dan andal.

Mulai membangun solusi denganAWS IoTPerangkat SDK

Tutorial ini memandu Anda melalui yang berbedaAWS IoTskenario. Jika tepat, tutorial menggunakanAWS IoTPerangkat SDK.

Topik

- [Tutorial: Menghubungkan perangkatAWS IoT Core dengan menggunakanAWS IoT Device SDK](#)
- [Membuat AWS IoT aturan untuk merutekan data perangkat ke layanan lain](#)
- [Mempertahankan status perangkat saat perangkat offline dengan Device Shadows](#)
- [Tutorial: Membuat otorisasi khusus untuk AWS IoT Core](#)
- [Tutorial: Memantau kelembaban tanah dengan AWS IoT dan Raspberry Pi](#)

Tutorial: Menghubungkan perangkatAWS IoT Core dengan menggunakanAWS IoT Device SDK

Tutorial ini menunjukkan bagaimana menghubungkan perangkat keAWS IoT Core sehingga dapat mengirim dan menerima data ke dan dariAWS IoT. Setelah Anda menyelesaikan tutorial

ini, perangkat Anda akan dikonfigurasi untuk terhubung AWS IoT Core dan Anda akan memahami bagaimana perangkat berkomunikasi dengan AWS IoT.

Dalam tutorial ini, Anda akan:

1. [the section called “Siapkan perangkat Anda AWS IoT”](#)
2. [the section called “Tinjau protokol MQTT”](#)
3. [the section called “Meninjau aplikasi sampel SDK Perangkat pubsub.py”](#)
4. [the section called “Connect perangkat Anda dan berkomunikasi dengan AWS IoT Core”](#)
5. [the section called “Tinjau hasilnya”](#)

Tutorial ini memakan waktu sekitar satu jam untuk menyelesaikannya.

Sebelum Anda mulai tutorial ini, pastikan bahwa Anda memilikinya:

- Selesai [Memulai dengan AWS IoT Core](#)

Di bagian tutorial di mana Anda harus [the section called “Konfigurasi perangkat Anda”](#), pilih [the section called “Connect Raspberry Pi atau perangkat lain”](#) opsi untuk perangkat Anda dan gunakan opsi bahasa Python untuk mengonfigurasi perangkat Anda.

Terus buka jendela terminal yang Anda gunakan dalam tutorial itu karena Anda juga akan menggunakannya dalam tutorial ini.

- Perangkat yang dapat menjalankan AWS IoT Device SDK v2 untuk Python.

Tutorial ini menunjukkan bagaimana menghubungkan perangkat AWS IoT Core dengan menggunakan contoh kode Python, yang memerlukan perangkat yang relatif kuat.

Jika Anda bekerja dengan perangkat yang dibatasi sumber daya, contoh kode ini mungkin tidak berfungsi. Dalam hal ini, Anda mungkin memiliki lebih sukses dengan [the section called “Menggunakan AWS IoT Device SDK for Embedded C”](#) tutorial.

Siapkan perangkat Anda AWS IoT

Di [Memulai dengan AWS IoT Core](#), Anda menyiapkan perangkat dan AWS akun Anda sehingga mereka dapat berkomunikasi. Bagian ini mengulas aspek-aspek persiapan yang berlaku untuk koneksi perangkat apa pun AWS IoT Core.

Agar perangkat dapat terhubung ke AWS IoT Core:

1. Anda harus memiliki Akun AWS.

Prosedur di [Siapkan Anda Akun AWS](#) menjelaskan cara membuat Akun AWS jika Anda belum memilikinya.

2. Di akun itu, Anda harus memiliki AWS IoT sumber daya berikut yang ditentukan untuk perangkat di Akun AWS dan Wilayah Anda.

Prosedur dalam [Buat AWS IoT sumber daya](#) menjelaskan cara membuat sumber daya ini untuk perangkat di Wilayah Anda Akun AWS dan.

- Sertifikat perangkat yang terdaftar AWS IoT dan diaktifkan untuk mengautentikasi perangkat.

Sertifikat sering dibuat dengan, dan melekat pada, objek AWS IoT benda. Sementara objek benda tidak diperlukan untuk perangkat untuk terhubung AWS IoT, itu membuat AWS IoT fitur tambahan tersedia untuk perangkat.

- Kebijakan yang dilampirkan pada sertifikat perangkat yang mengizinkannya untuk terhubung AWS IoT Core dan melakukan semua tindakan yang Anda inginkan.

3. Koneksi internet yang dapat mengakses titik akhir perangkat Anda Akun AWS.

Titik akhir perangkat dijelaskan [AWS IoT data perangkat dan titik akhir layanan](#) dan dapat dilihat di [halaman pengaturan AWS IoT konsol](#).

4. Perangkat lunak komunikasi seperti SDK AWS IoT Perangkat menyediakan. Tutorial ini menggunakan [AWS IoT Device SDK v2 untuk Python](#).

Tinjau protokol MQTT

Sebelum kita berbicara tentang aplikasi sampel, ada baiknya memahami protokol MQTT. Protokol MQTT menawarkan beberapa keunggulan dibandingkan protokol komunikasi jaringan lainnya, seperti HTTP, yang menjadikannya pilihan populer untuk perangkat IoT. Bagian ini mengulas aspek-aspek kunci dari MQTT yang berlaku untuk tutorial ini. Untuk informasi tentang bagaimana MQTT dibandingkan dengan HTTP, lihat [Memilih protokol untuk komunikasi perangkat Anda](#).

MQTT menggunakan model komunikasi publish/subscribe

Protokol MQTT menggunakan model komunikasi publish/subscribe dengan hostnya. Model ini berbeda dari model permintaan/respons yang digunakan HTTP. Dengan MQTT, perangkat membuat sesi dengan host yang diidentifikasi oleh ID klien unik. Untuk mengirim data, perangkat mempublikasikan pesan yang diidentifikasi berdasarkan topik ke broker pesan di host. Untuk

menerima pesan dari broker pesan, perangkat berlangganan topik dengan mengirim filter topik dalam permintaan berlangganan ke broker pesan.

MQTT mendukung sesi persisten

Pialang pesan menerima pesan dari perangkat dan menerbitkan pesan ke perangkat yang telah berlangganan. Dengan [sesi persisten](#)—session yang tetap aktif bahkan saat perangkat yang memulai terputus—perangkat dapat mengambil pesan yang dipublikasikan saat terputus. Di sisi perangkat, MQTT mendukung tingkat Kualitas Layanan ([QoS](#)) yang memastikan host menerima pesan yang dikirim oleh perangkat.

Meninjau aplikasi sampel SDK Perangkat pubsub.py

Bagian ini meninjau `pubsub.py` contoh aplikasi dari AWS IoT Device SDK v2 untuk Python yang digunakan dalam tutorial ini. Di sini, kami akan meninjau bagaimana menghubungkan AWS IoT Core untuk mempublikasikan dan berlangganan pesan MQTT. Bagian selanjutnya menyajikan beberapa latihan untuk membantu Anda menjelajahi cara perangkat terhubung dan berkomunikasi AWS IoT Core.

`pubsub.py` Contoh aplikasi menunjukkan aspek-aspek koneksi MQTT ini dengan AWS IoT Core:

- [Protokol komunikasi](#)
- [sesi persisten](#)
- [Kualitas Layanan](#)
- [Memublikasikan pesan](#)
- [Langganan pesan](#)
- [Pemutusan dan penyambungan ulang perangkat](#)

Protokol komunikasi

`pubsub.py` Sampel menunjukkan koneksi MQTT menggunakan MQTT dan MQTT melalui protokol WSS. Pustaka [runtime AWS umum \(AWSCRT\)](#) menyediakan dukungan protokol komunikasi tingkat rendah dan disertakan dengan AWS IoT Device SDK v2 untuk Python.

MQTT

Panggilan `pubsub.py` sampel `mqtt_connection_builder` (ditampilkan di sini) di [mqtt_connection_builder](#) untuk membuat koneksi dengan AWS IoT Core dengan menggunakan

protokol MQTT. `mtls_from_path` menggunakan sertifikat X.509 dan TLS v1.2 untuk mengotentikasi perangkat. Pustaka AWS CRT menangani rincian tingkat yang lebih rendah dari koneksi itu.

```
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

Titik akhir perangkat IoT Anda Akun AWS

Di aplikasi contoh, nilai ini diteruskan dari baris perintah.

cert_filepath

Jalur ke file sertifikat perangkat

Di aplikasi contoh, nilai ini diteruskan dari baris perintah.

pri_key_filepath

Path ke file kunci pribadi perangkat yang dibuat dengan file sertifikatnya

Di aplikasi contoh, nilai ini diteruskan dari baris perintah.

ca_filepath

Path ke file CA Root. Diperlukan hanya jika server MQTT menggunakan sertifikat yang belum ada di toko kepercayaan Anda.

Di aplikasi contoh, nilai ini diteruskan dari baris perintah.

client_bootstrap

Objek runtime umum yang menangani kegiatan komunikasi socket

Dalam aplikasi sampel, objek ini dipakai sebelum panggilan `mqtt_connection_builder.mtls_from_path`.

`on_connection_interrupted`, `on_connection_resumed`

Fungsi callback yang akan dipanggil saat koneksi perangkat terputus dan dilanjutkan `client_id`

ID yang secara unik mengidentifikasi perangkat ini di Wilayah AWS

Di aplikasi contoh, nilai ini diteruskan dari baris perintah.

`clean_session`

Apakah akan memulai sesi persisten baru, atau, jika ada, sambungkan kembali ke yang sudah ada

`keep_alive_secs`

Nilai keep live, dalam hitungan detik, untuk mengirim CONNECT permintaan. Ping akan secara otomatis dikirim pada interval ini. Jika server tidak menerima ping setelah 1,5 kali nilai ini, diasumsikan bahwa koneksi hilang.

MQTT atas WSS

Panggilan `pubsub.py` sampel `websockets_with_default_aws_signing` (ditampilkan di sini) di [mqtt_connection_builder](#) untuk membuat koneksi dengan AWS IoT Core menggunakan protokol MQTT melalui WSS. `websockets_with_default_aws_signing` menciptakan koneksi MQTT melalui WSS menggunakan [Signature V4](#) untuk mengotentikasi perangkat.

```
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

Titik akhir perangkat IoT Anda Akun AWS

Di aplikasi contoh, nilai ini diteruskan dari baris perintah.

client_bootstrap

Objek runtime umum yang menangani kegiatan komunikasi socket

Dalam aplikasi sampel, objek ini dipakai sebelum panggilan `kemqtt_connection_builder.websockets_with_default_aws_signing`.

region

Region AWS penandatanganan digunakan oleh otentikasi Signature V4. Di `pubsub.py`, ia melewati parameter yang dimasukkan dalam baris perintah.

Di aplikasi contoh, nilai ini diteruskan dari baris perintah.

credentials_provider

AWS Kredensi yang disediakan untuk digunakan untuk otentikasi

Dalam aplikasi sampel, objek ini dipakai sebelum panggilan `kemqtt_connection_builder.websockets_with_default_aws_signing`.

websocket_proxy_options

Opsi proxy HTTP, jika menggunakan host proxy

Di aplikasi contoh, nilai ini diinisialisasi sebelum panggilan `kemqtt_connection_builder.websockets_with_default_aws_signing`.

ca_filepath

Path ke file CA Root. Diperlukan hanya jika server MQTT menggunakan sertifikat yang belum ada di toko kepercayaan Anda.

Di aplikasi contoh, nilai ini diteruskan dari baris perintah.

on_connection_interrupted, on_connection_resumed

Fungsi callback yang akan dipanggil saat koneksi perangkat terputus dan dilanjutkan

`client_id`

ID yang secara unik mengidentifikasi perangkat ini di Wilayah AWS.

Di aplikasi contoh, nilai ini diteruskan dari baris perintah.

`clean_session`

Apakah akan memulai sesi persisten baru, atau, jika ada, sambungkan kembali ke yang sudah ada

`keep_alive_secs`

Nilai keep live, dalam hitungan detik, untuk mengirim CONNECT permintaan. Ping akan secara otomatis dikirim pada interval ini. Jika server tidak menerima ping setelah 1,5 kali nilai ini, itu mengasumsikan koneksi hilang.

HTTPS

Bagaimana HTTPS? AWS IoT Core mendukung perangkat yang mempublikasikan permintaan HTTPS. Dari perspektif pemrograman, perangkat mengirim permintaan HTTPS ke AWS IoT Core seperti halnya aplikasi lain. Untuk contoh program Python yang mengirimkan pesan HTTP dari perangkat, lihat [contoh kode HTTPS](#) menggunakan `requests` pustaka Python. Contoh ini mengirimkan pesan untuk AWS IoT Core menggunakan HTTPS sehingga AWS IoT Core menafsirkannya sebagai pesan MQTT.

Sementara AWS IoT Core mendukung permintaan HTTPS dari perangkat, pastikan untuk meninjau informasi tentang [Memilih protokol untuk komunikasi perangkat Anda](#) sehingga Anda dapat membuat keputusan yang tepat tentang protokol mana yang akan digunakan untuk komunikasi perangkat Anda.

sesi persisten

Di aplikasi contoh, setelah `clean_session` parameter untuk `False` menunjukkan bahwa koneksi harus persisten. Dalam prakteknya, ini berarti bahwa koneksi yang dibuka oleh panggilan ini terhubung kembali ke sesi persisten yang ada, jika ada. Jika tidak, itu menciptakan dan menghubungkan ke sesi persisten baru.

Dengan sesi persisten, pesan yang dikirim ke perangkat disimpan oleh broker pesan saat perangkat tidak terhubung. Saat perangkat terhubung kembali ke sesi persisten, broker pesan mengirimkan ke perangkat pesan apa pun yang tersimpan yang telah berlangganannya.

Tanpa sesi persisten, perangkat tidak akan menerima pesan yang dikirim saat perangkat tidak tersambung. Opsi mana yang akan digunakan tergantung pada aplikasi Anda dan apakah pesan yang terjadi saat perangkat tidak terhubung harus dikomunikasikan. Untuk informasi selengkapnya, lihat [Sesi persisten MQTT](#).

Kualitas Layanan

Saat perangkat menerbitkan dan berlangganan pesan, Quality of Service (QoS) pilihan dapat diatur. AWS IoT mendukung level QoS 0 dan 1 untuk mempublikasikan dan berlangganan operasi. Untuk informasi lebih lanjut tentang tingkat QoS di AWS IoT, lihat [Opsi Kualitas Layanan \(QoS\) MQTT](#).

Runtime AWS CRT untuk Python mendefinisikan konstanta ini untuk level QoS yang didukungnya:

Tingkat Kualitas Layanan Python

Tingkat MQTT QoS	Nilai simbolis Python yang digunakan oleh SDK	Deskripsi
Tingkat QoS 0	<code>mqtt.QoS.AT_MOST_ONCE</code>	Hanya satu upaya untuk mengirim pesan yang akan dilakukan, apakah itu diterima atau tidak. Pesan mungkin tidak dikirim sama sekali, misalnya, jika perangkat tidak terhubung atau ada kesalahan jaringan.
QoS tingkat 1	<code>mqtt.QoS.AT_LEAST_ONCE</code>	Pesan dikirim berulang kali sampai PUBLISH pengakuan diterima.

Di aplikasi sampel, permintaan publikasi dan berlangganan dibuat dengan level QoS 1 (`mqtt.QoS.AT_LEAST_ONCE`).

- QoS pada publikasi

Saat perangkat menerbitkan pesan dengan QoS level 1, perangkat akan mengirim pesan berulang kali hingga menerima PUBLISH respons dari broker pesan. Jika perangkat tidak tersambung, pesan akan diantrekan untuk dikirim setelah tersambung kembali.

- QoS saat berlangganan

Ketika perangkat berlangganan pesan dengan QoS level 1, broker pesan menyimpan pesan yang berlangganan perangkat hingga dapat dikirim ke perangkat. Pihak pesan mengirim ulang pesan sampai menerima PUBLISH respons dari perangkat.

Memublikasikan pesan

Setelah berhasil membuat koneksi ke AWS IoT Core, perangkat dapat memublikasikan pesan. `pubsub.py` Sampel melakukan ini dengan memanggil `publish` operasi `mqtt_connection` objek.

```
mqtt_connection.publish(  
    topic=args.topic,  
    payload=message,  
    qos=mqtt.QoS.AT_LEAST_ONCE  
)
```

topic

Nama topik pesan yang mengidentifikasi pesan

Di aplikasi contoh, ini diteruskan dari baris perintah.

payload

Payload pesan diformat sebagai string (misalnya, dokumen JSON)

Di aplikasi contoh, ini diteruskan dari baris perintah.

Dokumen JSON adalah format payload umum, dan yang diakui oleh AWS IoT layanan lain; namun, format data payload pesan dapat berupa apa saja yang disetujui oleh penerbit dan pelanggan. Namun, AWS IoT layanan lain hanya mengenali JSON, dan CBOR, dalam beberapa kasus, untuk sebagian besar operasi.

qos

Level QoS untuk pesan ini

Langganan pesan

Untuk menerima pesan dari AWS IoT dan layanan dan perangkat lain, perangkat berlangganan pesan tersebut dengan nama topik mereka. Perangkat dapat berlangganan pesan individual dengan

menentukan [nama topik](#), dan ke sekelompok pesan dengan menentukan [filter topik](#), yang dapat menyertakan karakter wild card. `pubsub.py` Contoh menggunakan kode yang ditampilkan di sini untuk berlangganan pesan dan mendaftarkan fungsi callback untuk memproses pesan setelah diterima.

```
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

Topik untuk dilanggan. Ini bisa berupa nama topik atau filter topik.

Di aplikasi contoh, ini diteruskan dari baris perintah.

qos

Apakah broker pesan harus menyimpan pesan-pesan ini saat perangkat terputus.

Nilai `mqtt.QoS.AT_LEAST_ONCE` (QoS level 1), membutuhkan sesi persisten yang akan ditentukan (`clean_session=False`) saat koneksi dibuat.

callback

Fungsi untuk memanggil untuk memproses pesan berlangganan.

`mqtt_connection.subscribe` Fungsi mengembalikan future dan ID paket. Jika permintaan berlangganan berhasil dimulai, ID paket yang dikembalikan lebih besar dari 0. Untuk memastikan bahwa langganan diterima dan didaftarkan oleh broker pesan, Anda harus menunggu hasil operasi asinkron kembali, seperti yang ditunjukkan pada contoh kode.

Fungsi panggilan balik

Callback dalam `pubsub.py` sampel memproses pesan berlangganan saat perangkat menerimanya.

```
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

Topik pesan

Ini adalah nama topik khusus dari pesan yang diterima, bahkan jika Anda berlangganan filter topik.

payload

Muatan pesan

Format untuk ini adalah aplikasi spesifik.

kwargs

Kemungkinan argumen tambahan seperti yang dijelaskan dalam [mqtt.Connection.subscribe](#).

Dalam `pubsub.py` sampel, `on_message_received` hanya menampilkan topik dan muatannya. Ini juga menghitung pesan yang diterima untuk mengakhiri program setelah batas tercapai.

Aplikasi Anda akan mengevaluasi topik dan payload untuk menentukan tindakan apa yang harus dilakukan.

Pemutusan dan penyambungan ulang perangkat

`pubsub.py` Sampel menyertakan fungsi callback yang dipanggil saat perangkat terputus dan saat koneksi dibuat kembali. Tindakan apa yang dilakukan perangkat Anda pada peristiwa ini adalah aplikasi khusus.

Saat perangkat terhubung untuk pertama kalinya, perangkat harus berlangganan topik untuk diterima. Jika sesi perangkat hadir saat tersambung kembali, langganannya dipulihkan, dan setiap pesan yang disimpan dari langganan tersebut akan dikirim ke perangkat setelah tersambung kembali.

Jika sesi perangkat tidak ada lagi saat tersambung kembali, sesi tersebut harus berlangganan kembali ke langganannya. Sesi persisten memiliki masa pakai terbatas dan dapat kedaluwarsa saat perangkat terputus terlalu lama.

Connect perangkat Anda dan berkomunikasi dengan AWS IoT Core

Bagian ini menyajikan beberapa latihan untuk membantu Anda menjelajahi berbagai aspek menghubungkan perangkat Anda dengan AWS IoT Core. Untuk latihan ini, Anda akan menggunakan [klien uji MQTT](#) di AWS IoT konsol untuk melihat apa yang diterbitkan perangkat Anda dan mempublikasikan pesan ke perangkat Anda. Latihan ini menggunakan [pubsub.py](#) sampel dari [AWS IoT Device SDK v2 untuk Python](#) dan membangun pengalaman Anda dengan [Memulai dengan AWS IoT Core](#) tutorial.

Di bagian ini, Anda akan:

- [Berlangganan filter topik wild card](#)
- [Proses langganan filter topik](#)
- [Memublikasikan pesan dari perangkat](#)

Untuk latihan ini, Anda akan mulai dari program `pubsub.py` sampel.

Note

Latihan ini mengasumsikan bahwa Anda menyelesaikan [Memulai dengan AWS IoT Core](#) tutorial dan menggunakan jendela terminal untuk perangkat Anda dari tutorial itu.

Berlangganan filter topik wild card

Dalam latihan ini, Anda akan memodifikasi baris perintah yang digunakan `pubsub.py` untuk menelepon untuk berlangganan filter topik wild card dan memproses pesan yang diterima berdasarkan topik pesan.

Prosedur Latihan

Untuk latihan ini, bayangkan perangkat Anda berisi kontrol suhu dan kontrol cahaya. Menggunakan nama-nama topik ini untuk mengidentifikasi pesan tentang mereka.

1. Sebelum memulai latihan, coba jalankan perintah ini dari [Memulai dengan AWS IoT Core](#) tutorial di perangkat Anda untuk memastikan semuanya siap untuk latihan.

```
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
```

Anda akan melihat output yang sama seperti yang Anda lihat di [tutorial Memulai](#).

- Untuk latihan ini, ubah parameter baris perintah ini.

Action	Parameter baris perintah	Efek
tambahkan	<code>--message ""</code>	Konfigurasi <code>pubsub.py</code> untuk mendengarkan saja
tambahkan	<code>--count 2</code>	Akhiri program setelah menerima dua pesan
perubahan	<code>--topic device/+/ details</code>	Tentukan filter topik untuk berlangganan

Membuat perubahan ini pada baris perintah awal menghasilkan baris perintah ini. Masukkan perintah ini di jendela terminal untuk perangkat Anda.

```
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
```

Program harus menampilkan tampilan seperti ini:

```
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...
```

Jika Anda melihat sesuatu seperti ini di terminal Anda, perangkat Anda siap dan mendengarkan pesan di mana nama topik dimula `device` dan diakhiri dengan `/detail`. Jadi, mari kita uji itu.

- Berikut adalah beberapa pesan yang mungkin diterima perangkat Anda.

Nama topik	Muatan pesan
device/temp/details	{ "desiredTemp": 20, "currentTemp": 15 }
device/light/details	{ "desiredLight": 100, "currentLight": 50 }

4. Menggunakan klien uji MQTT diAWS IoT konsol, kirim pesan yang dijelaskan pada langkah sebelumnya ke perangkat Anda.
 - a. Buka [klien uji MQTT](#) diAWS IoT konsol.
 - b. Di Berlangganan topik, di bidang Topik berlangganan, masukkan filter topik:**device/+/
details**, lalu pilih Berlangganan topik.
 - c. Di kolom Langganan klien pengujian MQTT, pilih perangkat/+/detail.
 - d. Untuk setiap topik dalam tabel sebelumnya, lakukan hal berikut di klien uji MQTT:
 1. Di Publikasikan, masukkan nilai dari kolom Nama topik di tabel.
 2. Di bidang payload pesan di bawah nama topik, masukkan nilai dari kolom Message payload di tabel.
 3. Perhatikan jendela terminal tempatpubsub . py sedang berjalan dan, di klien uji MQTT, pilih Publikasikan ke topik.

Anda akan melihat bahwa pesan diterima olehpubsub . py di jendela terminal.

Hasil Latihan

Dengan ini,pubsub . py, berlangganan pesan menggunakan filter topik kartu liar, menerimanya, dan menampilkannya di jendela terminal. Perhatikan bagaimana Anda berlangganan filter topik tunggal, dan fungsi callback dipanggil untuk memproses pesan yang memiliki dua topik berbeda.

Proses langganan filter topik

Membangun latihan sebelumnya, memodifikasi aplikasipubsub . py sampel untuk mengevaluasi topik pesan dan memproses pesan berlangganan berdasarkan topik.

Prosedur Latihan

Untuk mengevaluasi topik pesan

1. Salin `pubsub.py` ke `pubsub2.py`.
2. Bukapubsub2.py di editor teks favorit Anda atau IDE.
3. Dipubsub2.py, temukan `on_message_received` fungsinya.
4. `on_message_received`, masukkan kode berikut setelah baris yang dimulai dengan `print("Received message dan sebelum baris yang dimulai dengan 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. Simpan perubahan Anda dan jalankan program yang dimodifikasi dengan menggunakan baris perintah ini.

```
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. DiAWS IoT konsol, buka [klien uji MQTT](#).
7. Di Berlangganan topik, di bidang Topik berlangganan, masukkan filter topik:**device/+/details**, lalu pilih Berlangganan topik.
8. Di kolom Langganan klien pengujian MQTT, pilih perangkat/+/detail.
9. Untuk setiap topik dalam tabel ini, lakukan hal berikut di klien uji MQTT:

Nama topik	Muatan pesan
device/temp/details	{ "desiredTemp": 20, "currentTemp": 15 }
device/light/details	{ "desiredLight": 100, "currentLight": 50 }

1. Di Publikasikan, masukkan nilai dari kolom Nama topik di tabel.
2. Di bidang payload pesan di bawah nama topik, masukkan nilai dari kolom Message payload di tabel.
3. Perhatikan jendela terminal tempatpubsub . py sedang berjalan dan, di klien uji MQTT, pilih Publikasikan ke topik.

Anda akan melihat bahwa pesan diterima olehpubsub . py di jendela terminal.

Anda akan melihat tampilan yang serupa dengan ini di jendela terminal Anda.

```
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!
```

Hasil Latihan

Dalam latihan ini, Anda menambahkan kode sehingga aplikasi sampel akan mengenali dan memproses beberapa pesan dalam fungsi callback. Dengan ini, perangkat Anda dapat menerima pesan dan menindaklanjuti pesan tersebut.

Cara lain bagi perangkat Anda untuk menerima dan memproses beberapa pesan adalah berlangganan pesan yang berbeda secara terpisah dan menetapkan setiap langganan ke fungsi callbacknya sendiri.

Memublikasikan pesan dari perangkat

Anda dapat menggunakan contoh aplikasi pubsub.py untuk memublikasikan pesan dari perangkat Anda. Meskipun akan memublikasikan pesan apa adanya, pesan tidak dapat dibaca sebagai dokumen JSON. Latihan ini memodifikasi aplikasi contoh untuk dapat memublikasikan dokumen JSON dalam payload pesan yang dapat dibaca oleh AWS IoT Core.

Prosedur Latihan

Dalam latihan ini, pesan berikut akan dikirim dengan `device/data` topik.

```
{
  "timestamp": 1601048303,
  "sensorId": 28,
  "sensorData": [
    {
      "sensorName": "Wind speed",
      "sensorValue": 34.2211224
    }
  ]
}
```

Untuk mempersiapkan klien uji MQTT Anda untuk memantau pesan dari latihan ini

1. Di Berlangganan topik, di bidang Topik berlangganan, masukkan filter topik: **device/data**, lalu pilih Berlangganan topik.
2. Di kolom Langganan klien pengujian MQTT, pilih perangkat/data.
3. Biarkan jendela klien uji MQTT terbuka untuk menunggu pesan dari perangkat Anda.

Untuk mengirim dokumen JSON dengan aplikasi contoh pubsub.py

1. Di perangkat Anda, salin `pubsub.py` ke `pubsub3.py`.
2. Edit `pubsub3.py` untuk mengubah cara memformat pesan yang dipublikasikan.

a. Buka `pubsub3.py` di editor teks.

b. Temukan baris kode ini:

```
message = "{} [{}]".format(message_string, publish_count)
```

c. Ubah ke:

```
message = "{}".format(message_string)
```

d. Temukan baris kode ini:

```
message_json = json.dumps(message)
```

e. Ubah ke:

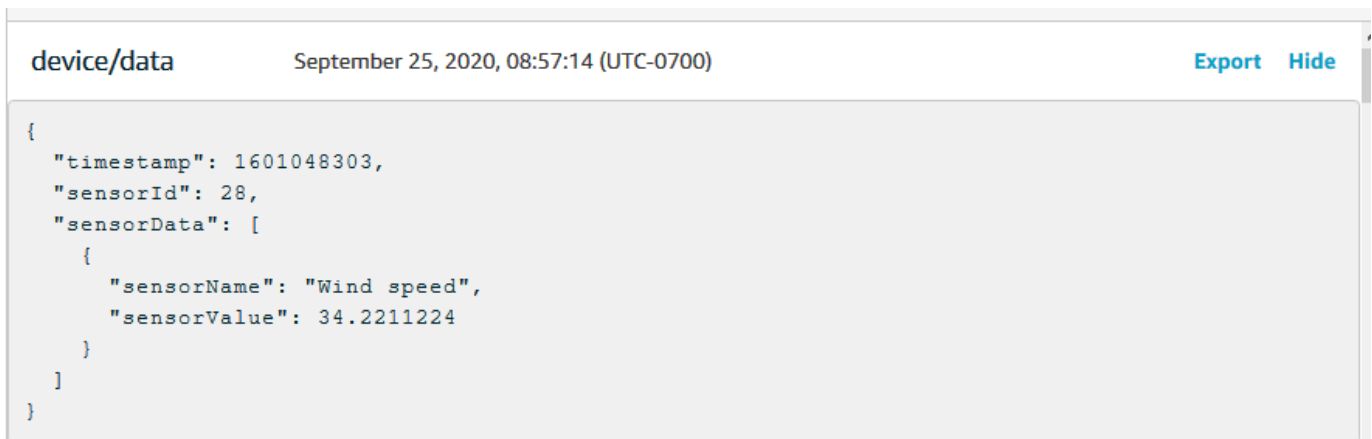
```
message = "{}".json.dumps(json.loads(message))
```

f. Simpan perubahan Anda.

3. Di perangkat Anda, jalankan perintah ini untuk mengirim pesan dua kali.

```
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. Di klien pengujian MQTT, periksa untuk melihat bahwa ia telah menafsirkan dan memformat dokumen JSON dalam payload pesan, seperti ini:



The screenshot shows a MQTT client interface with the following details:

- Topic: `device/data`
- Timestamp: `September 25, 2020, 08:57:14 (UTC-0700)`
- Buttons: `Export` and `Hide`
- Message Content (JSON):

```
{  "timestamp": 1601048303,  "sensorId": 28,  "sensorData": [    {      "sensorName": "Wind speed",      "sensorValue": 34.2211224    }  ]}
```


Secara default, `pubsub3.py` juga berlangganan pesan yang dikirimnya. Anda akan melihat bahwa itu menerima pesan dalam output aplikasi. Jendela terminal akan terlihat seperti ini.

```
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!
```

Hasil Latihan

Dengan ini, perangkat Anda dapat membuat pesan untuk dikirim AWS IoT Core untuk menguji konektivitas dasar dan menyediakan pesan perangkat AWS IoT Core untuk diproses. Misalnya, Anda dapat menggunakan aplikasi ini untuk mengirim data pengujian dari perangkat untuk menguji tindakan AWS IoT aturan.

Tinjau hasilnya

Contoh dalam tutorial ini memberi Anda pengalaman langsung dengan dasar-dasar bagaimana perangkat dapat berkomunikasi dengan AWS IoT Core —bagian mendasar dari AWS IoT solusi Anda. Ketika perangkat Anda dapat berkomunikasi AWS IoT Core, mereka dapat mengirimkan pesan ke AWS layanan dan perangkat lain tempat mereka dapat bertindak. Demikian juga, AWS layanan dan perangkat lain dapat memproses informasi yang menghasilkan pesan yang dikirim kembali ke perangkat Anda.

Ketika Anda siap untuk mengeksplorasi AWS IoT Core lebih lanjut, coba tutorial ini:

- [the section called “Mengirim notifikasi Amazon SNS”](#)
- [the section called “Menyimpan data perangkat dalam tabel DynamoDB”](#)
- [the section called “Memformat notifikasi dengan menggunakan fungsi AWS Lambda ”](#)

Tutorial: Menggunakan AWS IoT Device SDK for Embedded C

Bagian ini menjelaskan cara menjalankan file AWS IoT Device SDK for Embedded C.

Prosedur di bagian ini

- [Langkah1: Instal AWS IoT Device SDK for Embedded C](#)
- [Langkah 2: Konfigurasi aplikasi sampel](#)
- [Langkah 3: Bangun dan jalankan aplikasi sampel](#)

Langkah1: Instal AWS IoT Device SDK for Embedded C

Umumnya AWS IoT Device SDK for Embedded C ditargetkan pada perangkat terbatas sumber daya yang memerlukan runtime bahasa C yang dioptimalkan. Anda dapat menggunakan SDK pada sistem operasi apa pun dan menghostingnya pada semua jenis prosesor (misalnya, MCU dan MPU). Jika Anda memiliki lebih banyak memori dan sumber daya pemrosesan yang tersedia, kami sarankan Anda menggunakan salah satu SDK AWS IoT Perangkat dan Seluler tingkat tinggi (misalnya, C ++, Java JavaScript, dan Python).

Secara umum, AWS IoT Device SDK for Embedded C ini ditujukan untuk sistem yang menggunakan MCU atau MPU low-end yang menjalankan sistem operasi tertanam. Untuk contoh pemrograman di bagian ini, kami menganggap perangkat Anda menggunakan Linux.

Example

1. Unduh AWS IoT Device SDK for Embedded C ke perangkat Anda dari [GitHub](#).

```
git clone https://github.com/aws/aws-iot-device-sdk-embedded-c.git --recurse-submodules
```

Ini membuat direktori bernama `aws-iot-device-sdk-embedded-c` dalam direktori saat ini.

2. Arahkan ke direktori itu dan periksa rilis terbaru. Silakan lihat [github.com/aws/ aws-iot-device-sdk-embedded-c/tags](https://github.com/aws/aws-iot-device-sdk-embedded-c/tags) untuk tag rilis terbaru.

```
cd aws-iot-device-sdk-embedded-c
git checkout latest-release-tag
```

3. Instal OpenSSL versi 1.1.0 atau yang lebih baru. Library pengembangan OpenSSL biasanya disebut “libssl-dev” atau “openssl-devel” ketika diinstal melalui manajer paket.

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

Langkah 2: Konfigurasi aplikasi sampel

AWS IoT Device SDK for Embedded C Termasuk contoh aplikasi untuk Anda coba. Untuk kesederhanaan, tutorial ini menggunakan `mqtt_demo_mutual_auth` aplikasi, yang menggambarkan cara terhubung ke broker AWS IoT Core pesan dan berlangganan dan mempublikasikan ke topik MQTT.

1. Salin sertifikat dan kunci pribadi yang Anda buat [Memulai dengan AWS IoT Core](#) ke dalam `build/bin/certificates` direktori.

Note

Sertifikat CA perangkat dan root dapat kedaluwarsa atau dicabut. Jika sertifikat ini kedaluwarsa atau dicabut, Anda harus menyalin sertifikat CA baru atau kunci pribadi dan sertifikat perangkat ke perangkat Anda.

2. Anda harus mengonfigurasi sampel dengan AWS IoT Core titik akhir pribadi, kunci pribadi, sertifikat, dan sertifikat root CA Anda. Buka direktori `aws-iot-device-sdk-embedded-c/demos/mqtt/mqtt_demo_mutual_auth` tersebut.

Jika Anda telah AWS CLI menginstal, Anda dapat menggunakan perintah ini untuk menemukan URL titik akhir akun Anda.

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

Jika Anda belum AWS CLI menginstal, buka [AWS IoT konsol](#) Anda. Dari panel navigasi, pilih Kelola, lalu pilih Things. Pilih IoT untuk perangkat Anda, lalu pilih Interact. Titik akhir Anda ditampilkan di bagian HTTPS dari halaman detail hal.

3. Buka `demo_config.h` file dan perbarui nilai untuk yang berikut:

AWS_IOT_TITIK AKHIR

Titik akhir pribadi Anda.

CLIENT_CERT_PATH

Jalur file sertifikat Anda, misalnya `certificates/device.pem.crt`".

CLIENT_PRIVATE_KEY_PATH

Nama file kunci pribadi Anda, misalnya `certificates/private.pem.key`.

Sebagai contoh:

```
// 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. Periksa untuk melihat apakah Anda telah menginstal CMake di perangkat Anda dengan menggunakan perintah ini.

```
cmake --version
```

Jika Anda melihat informasi versi untuk kompiler, Anda dapat melanjutkan ke bagian berikutnya.

Jika Anda mendapatkan kesalahan atau tidak melihat informasi apa pun, maka Anda harus menginstal paket cmake menggunakan perintah ini.

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

Jalankan `cmake --version` perintah lagi dan konfirmasikan bahwa CMake telah diinstal dan Anda siap untuk melanjutkan.

5. Periksa untuk melihat apakah Anda memiliki alat pengembangan yang diinstal pada perangkat Anda dengan menggunakan perintah ini.

```
gcc --version
```

Jika Anda melihat informasi versi untuk kompiler, Anda dapat melanjutkan ke bagian berikutnya.

Jika Anda mendapatkan kesalahan atau tidak melihat informasi kompiler, Anda harus menginstal `build-essential` paket menggunakan perintah ini.

```
sudo apt-get install build-essential
```

Jalankan `gcc --version` perintah lagi dan konfirmasikan bahwa alat build telah diinstal dan Anda siap untuk melanjutkan.

Langkah 3: Bangun dan jalankan aplikasi sampel

Untuk menjalankan aplikasi AWS IoT Device SDK for Embedded C sampel

1. Arahkan ke `aws-iot-device-sdk-embedded-c` dan buat direktori build.

```
mkdir build && cd build
```

2. Masukkan perintah CMake berikut untuk menghasilkan Makefiles yang diperlukan untuk membangun.

```
cmake ..
```

3. Masukkan perintah berikut untuk membangun file aplikasi yang dapat dieksekusi.

```
make
```

4. Jalankan `mqtt_demo_mutual_auth` aplikasi dengan perintah ini.

```
cd bin  
./mqtt_demo_mutual_auth
```

Anda akan melihat output yang serupa dengan yang berikut:

```
[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.
```

Perangkat Anda sekarang terhubung untuk AWS IoT menggunakan file AWS IoT Device SDK for Embedded C.

Anda juga dapat menggunakan AWS IoT konsol untuk melihat pesan MQTT yang dipublikasikan oleh aplikasi sampel. Untuk informasi tentang cara menggunakan klien MQTT di [AWS IoT konsol](#), lihat [the section called “Lihat pesan MQTT dengan klien MQTT AWS IoT”](#)

Membuat AWS IoT aturan untuk merutekan data perangkat ke layanan lain

Tutorial ini menunjukkan cara membuat dan menguji AWS IoT aturan menggunakan beberapa tindakan aturan yang lebih umum.

AWS IoT aturan mengirim data dari perangkat Anda ke AWS layanan lain. Mereka mendengarkan pesan MQTT tertentu, memformat data dalam muatan pesan, dan mengirim hasilnya ke layanan lain. **AWS**

Kami menyarankan Anda mencoba ini dalam urutan yang ditampilkan di sini, bahkan jika tujuan Anda adalah membuat aturan yang menggunakan fungsi Lambda atau sesuatu yang lebih kompleks. Tutorial disajikan secara berurutan dari dasar hingga kompleks. Mereka menyajikan konsep baru

secara bertahap untuk membantu Anda mempelajari konsep yang dapat Anda gunakan untuk membuat tindakan aturan yang tidak memiliki tutorial khusus.

Note

AWS IoT aturan membantu Anda mengirim data dari perangkat IoT Anda ke layanan lain AWS . Namun, untuk melakukannya dengan sukses, Anda memerlukan pengetahuan tentang layanan lain tempat Anda ingin mengirim data. Meskipun tutorial ini memberikan informasi yang diperlukan untuk menyelesaikan tugas, Anda mungkin merasa terbantu untuk mempelajari lebih lanjut tentang layanan yang ingin Anda kirim data sebelum Anda menggunakannya dalam solusi Anda. Penjelasan rinci tentang AWS layanan lain berada di luar cakupan tutorial ini.

Ikhtisar skenario tutorial

Skenario untuk tutorial ini adalah perangkat sensor cuaca yang secara berkala menerbitkan datanya. Ada banyak perangkat sensor seperti itu dalam sistem imajiner ini. Tutorial di bagian ini, bagaimanapun, fokus pada satu perangkat sambil menunjukkan bagaimana Anda dapat mengakomodasi beberapa sensor.

Tutorial di bagian ini menunjukkan cara menggunakan AWS IoT aturan untuk melakukan tugas-tugas berikut dengan sistem imajiner perangkat sensor cuaca ini.

- [Tutorial: Menerbitkan ulang pesan MQTT](#)

Tutorial ini menunjukkan cara mempublikasikan ulang pesan MQTT yang diterima dari sensor cuaca sebagai pesan yang hanya berisi ID sensor dan nilai suhu. Ini hanya menggunakan AWS IoT Core layanan dan menunjukkan kueri SQL sederhana dan cara menggunakan klien MQTT untuk menguji aturan Anda.

- [Tutorial: Mengirim notifikasi Amazon SNS](#)

Tutorial ini menunjukkan cara mengirim pesan SNS ketika nilai dari perangkat sensor cuaca melebihi nilai tertentu. Ini dibangun di atas konsep yang disajikan dalam tutorial sebelumnya dan menambahkan cara bekerja dengan AWS layanan lain, [Amazon Simple Notification Service](#) (Amazon SNS).

Jika Anda baru mengenal Amazon SNS, tinjau latihan [Memulai](#) sebelum memulai tutorial ini.

- [Tutorial: Menyimpan data perangkat dalam tabel DynamoDB](#)

Tutorial ini menunjukkan cara menyimpan data dari perangkat sensor cuaca dalam tabel database. Ini menggunakan pernyataan kueri aturan dan template substitusi untuk memformat data pesan untuk layanan tujuan, [Amazon DynamoDB](#).

Jika Anda baru mengenal DynamoDB, tinjau latihan [Memulai](#) sebelum memulai tutorial ini.

- [Tutorial: Memformat notifikasi dengan menggunakan fungsi AWS Lambda](#)

Tutorial ini menunjukkan cara memanggil fungsi Lambda untuk memformat ulang data perangkat dan kemudian mengirimkannya sebagai pesan teks. Ia menambahkan skrip Python dan fungsi AWS SDK dalam [AWS Lambda](#) fungsi untuk memformat dengan data payload pesan dari perangkat sensor cuaca dan mengirim pesan teks.

Jika Anda baru mengenal Lambda, tinjau latihan [Memulai](#) sebelum memulai tutorial ini.

AWS IoT ikhtisar aturan

Semua tutorial ini membuat AWS IoT aturan.

Untuk AWS IoT aturan untuk mengirim data dari perangkat ke AWS layanan lain, ia menggunakan:

- Pernyataan kueri aturan yang terdiri dari:
 - Klausula SQL SELECT yang memilih dan memformat data dari payload pesan
 - Filter topik (objek FROM dalam pernyataan kueri aturan) yang mengidentifikasi pesan yang akan digunakan
 - Pernyataan kondisional opsional (klausa SQL WHERE) yang menentukan kondisi tertentu untuk bertindak
- Setidaknya satu tindakan aturan

Perangkat mempublikasikan pesan ke topik MQTT. Filter topik dalam pernyataan SQL SELECT mengidentifikasi topik MQTT untuk menerapkan aturan. Bidang yang ditentukan dalam pernyataan SQL SELECT memformat data dari payload pesan MQTT yang masuk untuk digunakan oleh tindakan aturan. Untuk daftar lengkap tindakan aturan, lihat [Tindakan AWS IoT aturan](#).

Tutorial di bagian ini

- [Tutorial: Menerbitkan ulang pesan MQTT](#)
- [Tutorial: Mengirim notifikasi Amazon SNS](#)

- [Tutorial: Menyimpan data perangkat dalam tabel DynamoDB](#)
- [Tutorial: Memformat notifikasi dengan menggunakan fungsi AWS Lambda](#)

Tutorial: Menerbitkan ulang pesan MQTT

Tutorial ini menunjukkan cara membuat AWS IoT aturan yang menerbitkan pesan MQTT ketika pesan MQTT tertentu diterima. Payload pesan yang masuk dapat dimodifikasi oleh aturan sebelum dipublikasikan. Ini memungkinkan untuk membuat pesan yang disesuaikan dengan aplikasi tertentu tanpa perlu mengubah perangkat Anda atau firmware-nya. Anda juga dapat menggunakan aspek pemfilteran aturan untuk mempublikasikan pesan hanya ketika kondisi tertentu terpenuhi.

Pesan yang diterbitkan ulang oleh aturan bertindak seperti pesan yang dikirim oleh AWS IoT perangkat atau klien lain. Perangkat dapat berlangganan pesan yang diterbitkan ulang dengan cara yang sama seperti mereka dapat berlangganan topik pesan MQTT lainnya.

Apa yang akan Anda pelajari dalam tutorial ini:

- Cara menggunakan query SQL sederhana dan fungsi dalam pernyataan query aturan
- Cara menggunakan klien MQTT untuk menguji aturan AWS IoT

Tutorial ini membutuhkan waktu sekitar 30 menit untuk menyelesaikannya.

Dalam tutorial ini, Anda akan:

- [Tinjau topik dan aturan MQTT AWS IoT](#)
- [Langkah 1: Buat AWS IoT aturan untuk menerbitkan ulang pesan MQTT](#)
- [Langkah 2: Uji aturan baru Anda](#)
- [Langkah 3: Tinjau hasil dan langkah selanjutnya](#)

Sebelum Anda memulai tutorial ini, pastikan Anda memiliki:

- [Siapkan Anda Akun AWS](#)

Anda akan membutuhkan Akun AWS dan AWS IoT konsol Anda untuk menyelesaikan tutorial ini.

- Diulas [Lihat pesan MQTT dengan klien MQTT AWS IoT](#)

Pastikan Anda dapat menggunakan klien MQTT untuk berlangganan dan mempublikasikan ke suatu topik. Anda akan menggunakan klien MQTT untuk menguji aturan baru Anda dalam prosedur ini.

Tinjau topik dan aturan MQTT AWS IoT

Sebelum berbicara tentang AWS IoT aturan, ada baiknya untuk memahami protokol MQTT. Dalam solusi IoT, protokol MQTT menawarkan beberapa keunggulan dibandingkan protokol komunikasi jaringan lainnya, seperti HTTP, yang menjadikannya pilihan populer untuk digunakan oleh perangkat IoT. Bagian ini mengulas aspek-aspek kunci dari MQTT karena mereka berlaku untuk tutorial ini. Untuk informasi tentang bagaimana MQTT dibandingkan dengan HTTP, lihat. [Memilih protokol untuk komunikasi perangkat Anda](#)

Protokol MQTT

Protokol MQTT menggunakan model komunikasi terbitkan/berlangganan dengan host-nya. Untuk mengirim data, perangkat mempublikasikan pesan yang diidentifikasi berdasarkan topik ke broker AWS IoT pesan. Untuk menerima pesan dari broker pesan, perangkat berlangganan topik yang akan mereka terima dengan mengirimkan filter topik dalam permintaan berlangganan ke broker pesan. Mesin AWS IoT aturan menerima pesan MQTT dari broker pesan.

AWS IoT aturan

AWS IoT aturan terdiri dari pernyataan kueri aturan dan satu atau lebih tindakan aturan. Ketika mesin AWS IoT aturan menerima pesan MQTT, elemen-elemen ini bertindak pada pesan sebagai berikut.

- Pernyataan kueri aturan

Pernyataan kueri aturan menjelaskan topik MQTT yang akan digunakan, menafsirkan data dari muatan pesan, dan memformat data seperti yang dijelaskan oleh pernyataan SQL yang mirip dengan pernyataan yang digunakan oleh database SQL populer. Hasil dari pernyataan query adalah data yang dikirim ke tindakan aturan.

- Tindakan aturan

Setiap tindakan aturan dalam aturan bertindak pada data yang dihasilkan dari pernyataan kueri aturan. AWS IoT mendukung [banyak tindakan aturan](#). Namun, dalam tutorial ini, Anda akan berkonsentrasi pada tindakan [Publikasikan ulang](#) aturan, yang menerbitkan hasil pernyataan kueri sebagai pesan MQTT dengan topik tertentu.

Langkah 1: Buat AWS IoT aturan untuk menerbitkan ulang pesan MQTT

AWS IoT Aturan yang akan Anda buat dalam tutorial ini berlangganan topik `device/device_id/` data MQTT di mana *device_id* adalah ID perangkat yang mengirim pesan. Topik-topik ini dijelaskan oleh [filter topik](#) sebagai `device/+/data`, di mana `+` adalah karakter wildcard yang cocok dengan string apa pun di antara dua karakter garis miring ke depan.

Ketika aturan menerima pesan dari topik yang cocok, aturan akan menerbitkan kembali `device_id` dan `temperature` nilai sebagai pesan MQTT baru dengan topik tersebut. `device/data/temp`

Misalnya, muatan pesan MQTT dengan `device/22/data` topik terlihat seperti ini:

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

Aturan mengambil `temperature` nilai dari payload pesan, dan `device_id` dari topik, dan menerbitkannya kembali sebagai pesan MQTT dengan `device/data/temp` topik dan muatan pesan yang terlihat seperti ini:

```
{
  "device_id": "22",
  "temperature": 28
}
```

Dengan aturan ini, perangkat yang hanya memerlukan ID perangkat dan data suhu berlangganan `device/data/temp` topik untuk hanya menerima informasi itu.

Untuk membuat aturan yang menerbitkan kembali pesan MQTT

1. Buka [hub Aturan AWS IoT konsol](#).
2. Di Aturan, pilih Buat dan mulai membuat aturan baru Anda.
3. Di bagian atas Buat aturan:

- a. Di Nama, masukkan nama aturan. Untuk tutorial ini, beri nama **republish_temp**.

Ingat bahwa nama aturan harus unik dalam Akun dan Wilayah Anda, dan tidak dapat memiliki spasi apa pun. Kami telah menggunakan karakter garis bawah dalam nama ini untuk memisahkan dua kata dalam nama aturan.

- b. Dalam Deskripsi, jelaskan aturannya.

Deskripsi yang bermakna membantu Anda mengingat apa yang dilakukan aturan ini dan mengapa Anda membuatnya. Deskripsi bisa selama dibutuhkan, jadi sedetail mungkin.

4. Dalam pernyataan kueri Aturan Buat aturan:

- a. Dalam Menggunakan versi SQL, pilih **2016-03-23**.
- b. Dalam kotak edit pernyataan kueri aturan, masukkan pernyataan:

```
SELECT topic(2) as device_id, temperature FROM 'device/+/data'
```

Pernyataan ini:

- Mendengarkan pesan MQTT dengan topik yang cocok dengan filter topik. `device/+/data`
- Memilih elemen kedua dari string topik dan menugaskannya ke bidang. `device_id`
- Memilih `temperature` bidang nilai dari payload pesan dan menetapkannya ke bidang. `temperature`

5. Di Tetapkan satu atau lebih tindakan:

- a. Untuk membuka daftar tindakan aturan untuk aturan ini, pilih Tambah tindakan.
- b. Di Pilih tindakan, pilih Menerbitkan ulang pesan ke AWS IoT topik.
- c. Di bagian bawah daftar tindakan, pilih Konfigurasi tindakan untuk membuka halaman konfigurasi tindakan yang dipilih.

6. Dalam tindakan Konfigurasi:

- a. Di Topik, masukkan **device/data/temp**. Ini adalah topik MQTT dari pesan yang akan dipublikasikan oleh aturan ini.
- b. Dalam Kualitas Layanan, pilih 0 - Pesan dikirim nol kali atau lebih.
- c. Di Pilih atau buat peran untuk memberikan AWS IoT akses untuk melakukan tindakan ini:

- i. Pilih Buat Peran. Kotak dialog Buat peran baru terbuka.
- ii. Masukkan nama yang menggambarkan peran baru. Dalam tutorial ini, gunakan **republish_role**.

Saat Anda membuat peran baru, kebijakan yang benar untuk melakukan tindakan aturan dibuat dan dilampirkan ke peran baru. Jika Anda mengubah topik tindakan aturan ini atau menggunakan peran ini dalam tindakan aturan lain, Anda harus memperbarui kebijakan untuk peran tersebut guna mengotorisasi topik atau tindakan baru. Untuk memperbarui peran yang ada, pilih Perbarui peran di bagian ini.

- iii. Pilih Buat Peran untuk membuat peran dan menutup kotak dialog.
 - d. Pilih Tambahkan tindakan untuk menambahkan tindakan ke aturan dan kembali ke halaman Buat aturan.
7. Menerbitkan ulang pesan ke tindakan AWS IoT topik sekarang tercantum dalam Tetapkan satu atau beberapa tindakan.

Di ubin tindakan baru, di bawah Publikasikan ulang pesan ke suatu AWS IoT topik, Anda dapat melihat topik yang akan dipublikasikan oleh tindakan penerbitan ulang Anda.

Ini adalah satu-satunya tindakan aturan yang akan Anda tambahkan ke aturan ini.

8. Di Buat aturan, gulir ke bawah ke bawah dan pilih Buat aturan untuk membuat aturan dan selesaikan langkah ini.

Langkah 2: Uji aturan baru Anda

Untuk menguji aturan baru Anda, Anda akan menggunakan klien MQTT untuk mempublikasikan dan berlangganan pesan MQTT yang digunakan oleh aturan ini.

Buka [klien MQTT di AWS IoT konsol di jendela](#) baru. Ini akan memungkinkan Anda mengedit aturan tanpa kehilangan konfigurasi klien MQTT Anda. Klien MQTT tidak menyimpan langganan atau log pesan apa pun jika Anda membiarkannya pergi ke halaman lain di konsol.

Untuk menggunakan klien MQTT untuk menguji aturan Anda

1. Di [klien MQTT di AWS IoT konsol](#), berlangganan topik input, dalam hal ini, `device+/data`
 - a. Di klien MQTT, di bawah Langganan, pilih Berlangganan topik.
 - b. Dalam topik Langganan, masukkan topik filter topik masukan, **device+/data**.

- c. Simpan sisa bidang di pengaturan defaultnya.
- d. Pilih Berlangganan topik.

Di kolom Langganan, di bawah Publikasikan ke topik, **device/+/data** muncul.

2. Berlangganan topik yang akan diterbitkan oleh aturan Anda:device/data/temp.
 - a. Di bawah Langganan, pilih Berlangganan topik lagi, dan dalam topik Langganan, masukkan topik pesan yang diterbitkan ulang,. **device/data/temp**
 - b. Simpan sisa bidang di pengaturan defaultnya.
 - c. Pilih Berlangganan topik.

Di kolom Langganan, di bawah perangkat+/data, muncul. **device/data/temp**

3. Publikasikan pesan ke topik input dengan ID perangkat tertentu,**device/22/data**. Anda tidak dapat mempublikasikan ke topik MQTT yang berisi karakter wildcard.
 - a. Di klien MQTT, di bawah Langganan, pilih Publikasikan ke topik.
 - b. Di bidang Publikasikan, masukkan nama topik masukan,**device/22/data**.
 - c. Salin data sampel yang ditampilkan di sini dan, di kotak edit di bawah nama topik, tempel data sampel.

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

- d. Untuk mengirim pesan MQTT Anda, pilih Publikasikan ke topik.
4. Tinjau pesan yang dikirim.
 - a. Di klien MQTT, di bawah Langganan, ada titik hijau di sebelah dua topik yang Anda berlangganan sebelumnya.

Titik-titik hijau menunjukkan bahwa satu atau lebih pesan baru telah diterima sejak terakhir kali Anda melihatnya.

- b. Di bawah Langganan, pilih perangkat/+/data untuk memeriksa apakah muatan pesan cocok dengan apa yang baru saja Anda terbitkan dan terlihat seperti ini:

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

- c. Di bawah Langganan, pilih perangkat/data/temp untuk memeriksa apakah payload pesan yang dipublikasikan ulang terlihat seperti ini:

```
{
  "device_id": "22",
  "temperature": 28
}
```

Perhatikan bahwa `device_id` nilainya adalah string yang dikutip dan `temperature` nilainya numerik. Hal ini karena `topic()` fungsi mengekstrak string dari nama topik pesan masukan sementara `temperature` nilai menggunakan nilai numerik dari payload pesan masukan.

Jika Anda ingin membuat nilai `device_id` nilai numerik, ganti `topic(2)` dalam pernyataan kueri aturan dengan:

```
cast(topic(2) AS DECIMAL)
```

Perhatikan bahwa `topic(2)` mentransmisikan nilai ke nilai numerik hanya akan berfungsi jika bagian topik tersebut hanya berisi karakter numerik.

5. Jika Anda melihat bahwa pesan yang benar telah dipublikasikan ke topik perangkat/data/temp, maka aturan Anda berfungsi. Lihat apa lagi yang dapat Anda pelajari tentang tindakan aturan Publikasikan ulang di bagian berikutnya.

Jika Anda tidak melihat bahwa pesan yang benar telah dipublikasikan ke topik perangkat/+/data atau perangkat/data/temp, periksa tips pemecahan masalah.

Memecahkan masalah aturan pesan Republish

Berikut adalah beberapa hal untuk diperiksa jika Anda tidak melihat hasil yang Anda harapkan.

- Anda mendapat spanduk kesalahan

Jika kesalahan muncul saat Anda mempublikasikan pesan input, perbaiki kesalahan itu terlebih dahulu. Langkah-langkah berikut dapat membantu Anda memperbaiki kesalahan itu.

- Anda tidak melihat pesan masukan di klien MQTT

Setiap kali Anda mempublikasikan pesan masukan ke `device/22/data` topik, pesan itu akan muncul di klien MQTT jika Anda berlangganan filter `device/+/data` topik seperti yang dijelaskan dalam prosedur.

Hal-hal yang harus diperiksa

- Periksa filter topik yang Anda langgani

Jika Anda berlangganan topik pesan input seperti yang dijelaskan dalam prosedur, Anda akan melihat salinan pesan input setiap kali Anda mempublikasikannya.

Jika Anda tidak melihat pesan, periksa nama topik yang Anda langgani dan bandingkan dengan topik yang Anda terbitkan. Nama topik peka huruf besar/kecil dan topik yang Anda langgani harus identik dengan topik yang Anda publikasikan payload pesan.

- Periksa fungsi publikasi pesan

Di klien MQTT, di bawah Langganan, pilih perangkat/`+/data`, periksa topik pesan terbitkan, lalu pilih Publikasikan ke topik. Anda akan melihat payload pesan dari kotak edit di bawah topik yang muncul di daftar pesan.

- Anda tidak melihat pesan yang diterbitkan ulang di klien MQTT

Agar aturan Anda berfungsi, ia harus memiliki kebijakan yang benar yang mengizinkannya untuk menerima dan menerbitkan ulang pesan dan harus menerima pesan.

Hal-hal yang harus diperiksa

- Periksa klien MQTT Anda dan aturan yang Anda buat Wilayah AWS

Konsol tempat Anda menjalankan klien MQTT harus berada di AWS Wilayah yang sama dengan aturan yang Anda buat.

- Periksa topik pesan masukan dalam pernyataan kueri aturan

Agar aturan berfungsi, aturan harus menerima pesan dengan nama topik yang cocok dengan filter topik dalam klausa FROM dari pernyataan kueri aturan.

Periksa ejaan filter topik dalam pernyataan kueri aturan dengan topik di klien MQTT. Nama topik peka huruf besar/kecil dan topik pesan harus cocok dengan filter topik dalam pernyataan kueri aturan.

- Periksa isi payload pesan masukan

Agar aturan berfungsi, ia harus menemukan bidang data di payload pesan yang dideklarasikan dalam pernyataan SELECT.

Periksa ejaan temperature bidang dalam pernyataan kueri aturan dengan payload pesan di klien MQTT. Nama bidang peka huruf besar/kecil dan temperature bidang dalam pernyataan kueri aturan harus identik dengan temperature bidang di payload pesan.

Pastikan dokumen JSON di payload pesan diformat dengan benar. Jika JSON memiliki kesalahan, seperti koma yang hilang, aturan tidak akan dapat membacanya.

- Periksa topik pesan yang diterbitkan ulang dalam tindakan aturan

Topik di mana tindakan aturan Republish menerbitkan pesan baru harus sesuai dengan topik yang Anda berlangganan di klien MQTT.

Buka aturan yang Anda buat di konsol dan periksa topik di mana tindakan aturan akan menerbitkan ulang pesan.

- Periksa peran yang digunakan oleh aturan

Tindakan aturan harus memiliki izin untuk menerima topik asli dan mempublikasikan topik baru.

Kebijakan yang mengizinkan aturan untuk menerima data pesan dan mempublikasikannya khusus untuk topik yang digunakan. Jika mengubah topik yang digunakan untuk mempublikasikan ulang data pesan, Anda harus memperbarui peran tindakan aturan untuk memperbarui kebijakannya agar sesuai dengan topik saat ini.

Jika Anda mencurigai ini masalahnya, edit tindakan aturan Republish dan buat peran baru. Peran baru yang dibuat oleh tindakan aturan menerima otorisasi yang diperlukan untuk melakukan tindakan ini.

Langkah 3: Tinjau hasil dan langkah selanjutnya

Dalam tutorial ini

- Anda menggunakan query SQL sederhana dan beberapa fungsi dalam pernyataan query aturan untuk menghasilkan pesan MQTT baru.
- Anda membuat aturan yang menerbitkan ulang pesan baru itu.
- Anda menggunakan klien MQTT untuk menguji aturan Anda. AWS IoT

Langkah selanjutnya

Setelah Anda menerbitkan ulang beberapa pesan dengan aturan ini, cobalah bereksperimen dengannya untuk melihat bagaimana mengubah beberapa aspek tutorial memengaruhi pesan yang diterbitkan ulang. Berikut adalah beberapa ide untuk Anda mulai.

- Ubah *device_id* dalam topik pesan masukan dan amati efeknya dalam payload pesan yang diterbitkan ulang.
- Ubah bidang yang dipilih dalam pernyataan kueri aturan dan amati efeknya dalam payload pesan yang diterbitkan ulang.
- Coba tutorial berikutnya dalam seri ini dan pelajari caranya [Tutorial: Mengirim notifikasi Amazon SNS](#).

Tindakan aturan Republish yang digunakan dalam tutorial ini juga dapat membantu Anda men-debug pernyataan kueri aturan. Misalnya, Anda dapat menambahkan tindakan ini ke aturan untuk melihat bagaimana pernyataan kueri aturannya memformat data yang digunakan oleh tindakan aturannya.

Tutorial: Mengirim notifikasi Amazon SNS

Tutorial ini menunjukkan cara membuat AWS IoT aturan yang mengirimkan data pesan MQTT ke topik Amazon SNS sehingga dapat dikirim sebagai pesan teks SMS.

Dalam tutorial ini, Anda membuat aturan yang mengirimkan data pesan dari sensor cuaca ke semua pelanggan topik Amazon SNS, setiap kali suhu melebihi nilai yang ditetapkan dalam aturan. Aturan mendeteksi ketika suhu yang dilaporkan melebihi nilai yang ditetapkan oleh aturan, dan kemudian membuat muatan pesan baru yang hanya mencakup ID perangkat, suhu yang dilaporkan, dan batas suhu yang terlampaui. Aturan mengirimkan payload pesan baru sebagai dokumen JSON ke topik SNS, yang memberi tahu semua pelanggan ke topik SNS.

Apa yang akan Anda pelajari dalam tutorial ini:

- Cara membuat dan menguji notifikasi Amazon SNS
- Cara memanggil notifikasi Amazon SNS dari suatu aturan AWS IoT
- Cara menggunakan kueri dan fungsi SQL sederhana dalam pernyataan kueri aturan
- Cara menggunakan klien MQTT untuk menguji aturan AWS IoT

Tutorial ini membutuhkan waktu sekitar 30 menit untuk menyelesaikannya.

Dalam tutorial ini, Anda akan:

- [Langkah 1: Buat topik Amazon SNS yang mengirim pesan teks SMS](#)
- [Langkah 2: Buat AWS IoT aturan untuk mengirim pesan teks](#)
- [Langkah 3: Uji AWS IoT aturan dan pemberitahuan Amazon SNS](#)
- [Langkah 4: Tinjau hasil dan langkah selanjutnya](#)

Sebelum Anda memulai tutorial ini, pastikan Anda memiliki:

- [Siapkan Anda Akun AWS](#)

Anda akan membutuhkan Akun AWS dan AWS IoT konsol Anda untuk menyelesaikan tutorial ini.

- Diulas [Lihat pesan MQTT dengan klien MQTT AWS IoT](#)

Pastikan Anda dapat menggunakan klien MQTT untuk berlangganan dan mempublikasikan ke suatu topik. Anda akan menggunakan klien MQTT untuk menguji aturan baru Anda dalam prosedur ini.

- Meninjau [Layanan Pemberitahuan Sederhana Amazon](#)

Jika Anda belum pernah menggunakan Amazon SNS sebelumnya, tinjau [Menyiapkan akses untuk Amazon SNS](#). Jika Anda sudah menyelesaikan AWS IoT tutorial lain, Anda Akun AWS harus sudah dikonfigurasi dengan benar.

Langkah 1: Buat topik Amazon SNS yang mengirim pesan teks SMS

Untuk membuat topik Amazon SNS yang mengirim pesan teks SMS

1. Buat topik Amazon SNS.

- a. Masuk ke [konsol Amazon SNS](#).
- b. Di panel navigasi kiri, pilih Topics (Topik).
- c. Di halaman Topics (Topik), pilih Create topic (Buat topik).
- d. Di Detail, pilih tipe Standar. Secara default, konsol membuat topik FIFO.
- e. Di Nama, masukkan nama topik SNS. Untuk tutorial ini, masukkan **high_temp_notice**.
- f. Gulir ke akhir halaman dan pilih Buat topik.

Konsol membuka halaman Detail topik baru.

2. Buat langganan Amazon SNS.

Note

Nomor telepon yang Anda gunakan dalam langganan ini mungkin dikenakan biaya pesan teks dari pesan yang akan Anda kirim dalam tutorial ini.

- a. Di halaman detail topik `high_temp_notice`, pilih Buat langganan.
- b. Di Buat langganan, di bagian Detail, dalam daftar Protokol, pilih SMS.
- c. Di Endpoint, masukkan nomor telepon yang dapat menerima pesan teks. Pastikan untuk memasukkannya sedemikian rupa sehingga dimulai dengan+, termasuk kode negara dan area, dan tidak menyertakan karakter tanda baca lainnya.
- d. Pilih Buat langganan.

3. Uji notifikasi Amazon SNS.

- a. Di [konsol Amazon SNS](#), di panel navigasi kiri, pilih Topik.
- b. Untuk membuka halaman detail topik, di Topik, dalam daftar topik, pilih `high_temp_notice`.
- c. Untuk membuka halaman Publikasikan pesan ke topik, di halaman detail `high_temp_notice`, pilih Publikasikan pesan.
- d. Di Publikasikan pesan ke topik, di bagian Isi pesan, di Badan pesan untuk dikirim ke titik akhir, masukkan pesan singkat.
- e. Gulir ke bawah ke bagian bawah halaman dan pilih Publikasikan pesan.
- f. Di telepon dengan nomor yang Anda gunakan sebelumnya saat membuat langganan, konfirmasi bahwa pesan telah diterima.

Jika Anda tidak menerima pesan tes, periksa kembali nomor telepon dan pengaturan telepon Anda.

Pastikan Anda dapat mempublikasikan pesan pengujian dari [konsol Amazon SNS](#) sebelum melanjutkan tutorial.

Langkah 2: Buat AWS IoT aturan untuk mengirim pesan teks

AWS IoT Aturan yang akan Anda buat dalam tutorial ini berlangganan topik `device/device_id/` data MQTT di `device_id` mana ID perangkat yang mengirim pesan. Topik-topik ini dijelaskan dalam filter topik sebagai `device/+/data`, di mana `+` adalah karakter wildcard yang cocok dengan string apa pun di antara dua karakter garis miring ke depan. Aturan ini juga menguji nilai `temperature` bidang dalam payload pesan.

Saat aturan menerima pesan dari topik yang cocok, aturan akan mengambil `device_id` dari nama topik, `temperature` nilai dari payload pesan, dan menambahkan nilai konstan untuk batas pengujian, dan mengirimkan nilai ini sebagai dokumen JSON ke topik notifikasi Amazon SNS.

Misalnya, pesan MQTT dari perangkat sensor cuaca nomor 32 menggunakan `device/32/data` topik dan memiliki muatan pesan yang terlihat seperti ini:

```
{
  "temperature": 38,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

Pernyataan kueri aturan mengambil `temperature` nilai dari payload pesan, `device_id` dari nama topik, dan menambahkan `max_temperature` nilai konstan untuk mengirim payload pesan yang terlihat seperti ini ke topik Amazon SNS:

```
{
  "device_id": "32",
  "reported_temperature": 38,
```

```
"max_temperature": 30
}
```

Untuk membuat AWS IoT aturan untuk mendeteksi nilai suhu over-limit dan membuat data untuk dikirim ke topik Amazon SNS

1. Buka [hub AturanAWS IoT konsol](#).
2. Jika ini adalah aturan pertama Anda, pilih Buat, atau Buat aturan.
3. Di Buat aturan:

- a. Di Nama, masukkan **temp_limit_notify**.

Ingat bahwa nama aturan harus unik di dalam Anda Akun AWS dan Wilayah, dan tidak dapat memiliki spasi apa pun. Kami telah menggunakan karakter garis bawah dalam nama ini untuk memisahkan kata-kata dalam nama aturan.

- b. Dalam Deskripsi, jelaskan aturannya.

Deskripsi yang bermakna membuatnya lebih mudah untuk mengingat apa yang dilakukan aturan ini dan mengapa Anda membuatnya. Deskripsi bisa selama dibutuhkan, jadi sedetail mungkin.

4. Dalam pernyataan kueri Aturan Buat aturan:

- a. Dalam Menggunakan versi SQL, pilih 2016-03-23.
- b. Dalam kotak edit pernyataan kueri aturan, masukkan pernyataan:

```
SELECT topic(2) as device_id,
       temperature as reported_temperature,
       30 as max_temperature
FROM 'device/+/data'
WHERE temperature > 30
```

Pernyataan ini:

- Mendengarkan pesan MQTT dengan topik yang cocok dengan filter `device/+/data` topik dan yang memiliki `temperature` nilai lebih besar dari 30.
- Memilih elemen kedua dari string topik dan menetapkannya ke bidang. `device_id`
- Memilih `temperature` bidang nilai dari payload pesan dan menetapkannya ke bidang. `reported_temperature`

- Menciptakan nilai konstan 30 untuk mewakili nilai batas dan menetapkannya ke `max_temperature` bidang.
5. Untuk membuka daftar tindakan aturan untuk aturan ini, di Tetapkan satu atau beberapa tindakan, pilih Tambah tindakan.
 6. Di Pilih tindakan, pilih Kirim pesan sebagai pemberitahuan push SNS.
 7. Untuk membuka halaman konfigurasi tindakan yang dipilih, di bagian bawah daftar tindakan, pilih Konfigurasi tindakan.
 8. Dalam tindakan Konfigurasi:
 - a. Di target SNS, pilih Pilih, temukan topik SNS Anda bernama `high_temp_notice`, dan pilih Select.
 - b. Dalam format Pesan, pilih RAW.
 - c. Di Pilih atau buat peran untuk memberikan AWS IoT akses untuk melakukan tindakan ini, pilih Buat Peran.
 - d. Di Buat peran baru, di Nama, masukkan nama unik untuk peran baru. Untuk tutorial ini, gunakan **`sns_rule_role`**.
 - e. Pilih Buat peran.

Jika Anda mengulangi tutorial ini atau menggunakan kembali peran yang ada, pilih Perbarui peran sebelum melanjutkan. Ini memperbarui dokumen kebijakan peran agar berfungsi dengan target SNS.

9. Pilih Tambah tindakan dan kembali ke halaman Buat aturan.

Di ubin tindakan baru, di bawah Kirim pesan sebagai pemberitahuan push SNS, Anda dapat melihat topik SNS yang akan dipanggil aturan Anda.

Ini adalah satu-satunya tindakan aturan yang akan Anda tambahkan ke aturan ini.

10. Untuk membuat aturan dan menyelesaikan langkah ini, di Buat aturan, gulir ke bawah ke bawah dan pilih Buat aturan.

Langkah 3: Uji AWS IoT aturan dan pemberitahuan Amazon SNS

Untuk menguji aturan baru Anda, Anda akan menggunakan klien MQTT untuk mempublikasikan dan berlangganan pesan MQTT yang digunakan oleh aturan ini.

Buka [klien MQTT di AWS IoT konsol di jendela](#) baru. Ini akan memungkinkan Anda mengedit aturan tanpa kehilangan konfigurasi klien MQTT Anda. Jika Anda meninggalkan klien MQTT untuk pergi ke halaman lain di konsol, itu tidak akan menyimpan langganan atau log pesan apa pun.

Untuk menggunakan klien MQTT untuk menguji aturan Anda

1. Di [klien MQTT di AWS IoT konsol](#), berlangganan topik input, dalam hal ini, `device/+/data`
 - a. Di klien MQTT, di bawah Langganan, pilih Berlangganan topik.
 - b. Dalam topik Langganan, masukkan topik filter topik masukan, **`device/+/data`**.
 - c. Simpan sisa bidang di pengaturan defaultnya.
 - d. Pilih Berlangganan topik.

Di kolom Langganan, di bawah Publikasikan ke topik, **`device/+/data`** muncul.

2. Publikasikan pesan ke topik input dengan ID perangkat tertentu, **`device/32/data`**. Anda tidak dapat mempublikasikan ke topik MQTT yang berisi karakter wildcard.
 - a. Di klien MQTT, di bawah Langganan, pilih Publikasikan ke topik.
 - b. Di bidang Publikasikan, masukkan nama topik masukan, **`device/32/data`**.
 - c. Salin data sampel yang ditampilkan di sini dan, di kotak edit di bawah nama topik, tempel data sampel.

```
{
  "temperature": 38,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

- d. Pilih Publikasikan ke topik untuk mempublikasikan pesan MQTT Anda.
3. Konfirmasikan bahwa pesan teks telah dikirim.
 - a. Di klien MQTT, di bawah Langganan, ada titik hijau di sebelah topik yang Anda berlangganan sebelumnya.

Titik hijau menunjukkan bahwa satu atau lebih pesan baru telah diterima sejak terakhir kali Anda melihatnya.

- b. Di bawah Langganan, pilih perangkat+/data untuk memeriksa apakah muatan pesan cocok dengan apa yang baru saja Anda terbitkan dan terlihat seperti ini:

```
{
  "temperature": 38,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

- c. Periksa ponsel yang Anda gunakan untuk berlangganan topik SNS dan konfirmasi isi payload pesan terlihat seperti ini:

```
{"device_id":"32","reported_temperature":38,"max_temperature":30}
```

Perhatikan bahwa `device_id` nilainya adalah string yang dikutip dan `temperature` nilainya numerik. Hal ini karena `topic()` fungsi mengekstrak string dari nama topik pesan masukan sementara `temperature` nilai menggunakan nilai numerik dari payload pesan masukan.

Jika Anda ingin membuat nilai `device_id` nilai numerik, ganti `topic(2)` dalam pernyataan kueri aturan dengan:

```
cast(topic(2) AS DECIMAL)
```

Perhatikan bahwa `topic(2)` mentransmisikan nilai ke numerik, `DECIMAL` nilai hanya akan berfungsi jika bagian topik tersebut hanya berisi karakter numerik.

4. Coba kirim pesan MQTT di mana suhunya tidak melebihi batas.
 - a. Di klien MQTT, di bawah Langganan, pilih Publikasikan ke topik.
 - b. Di bidang Publikasikan, masukkan nama topik masukan, **device/33/data**.
 - c. Salin data sampel yang ditampilkan di sini dan, di kotak edit di bawah nama topik, tempel data sampel.

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

- d. Untuk mengirim pesan MQTT Anda, pilih Publikasikan ke topik.

Anda akan melihat pesan yang Anda kirim dalam **device/+data** langganan. Namun, karena nilai suhu di bawah suhu maks dalam pernyataan kueri aturan, Anda seharusnya tidak menerima pesan teks.

Jika Anda tidak melihat perilaku yang benar, periksa tips pemecahan masalah.

Memecahkan masalah aturan pesan SNS Anda

Berikut adalah beberapa hal untuk diperiksa, jika Anda tidak melihat hasil yang Anda harapkan.

- Anda mendapat spanduk kesalahan

Jika kesalahan muncul saat Anda mempublikasikan pesan input, perbaiki kesalahan itu terlebih dahulu. Langkah-langkah berikut dapat membantu Anda memperbaiki kesalahan itu.

- Anda tidak melihat pesan masukan di klien MQTT

Setiap kali Anda mempublikasikan pesan masukan Anda ke `device/22/data` topik, pesan itu akan muncul di klien MQTT, jika Anda berlangganan filter `device/+data` topik seperti yang dijelaskan dalam prosedur.

Hal-hal yang harus diperiksa

- Periksa filter topik yang Anda langgani

Jika Anda berlangganan topik pesan input seperti yang dijelaskan dalam prosedur, Anda akan melihat salinan pesan input setiap kali Anda mempublikasikannya.

Jika Anda tidak melihat pesan, periksa nama topik yang Anda langgani dan bandingkan dengan topik yang Anda terbitkan. Nama topik peka huruf besar/kecil dan topik yang Anda langgani harus identik dengan topik yang Anda publikasikan payload pesan.

- Periksa fungsi publikasi pesan

Di klien MQTT, di bawah Langganan, pilih perangkat/+/data, periksa topik pesan terbitkan, lalu pilih Publikasikan ke topik. Anda akan melihat payload pesan dari kotak edit di bawah topik yang muncul di daftar pesan.

- Anda tidak menerima pesan SMS

Agar aturan Anda berfungsi, itu harus memiliki kebijakan yang benar yang mengizinkannya untuk menerima pesan dan mengirim pemberitahuan SNS, dan itu harus menerima pesan.

Hal-hal yang harus diperiksa

- Periksa klien MQTT Anda dan aturan yang Anda buat Wilayah AWS

Konsol tempat Anda menjalankan klien MQTT harus berada di AWS Wilayah yang sama dengan aturan yang Anda buat.

- Periksa apakah nilai suhu dalam muatan pesan melebihi ambang uji

Jika nilai suhu kurang dari atau sama dengan 30, seperti yang didefinisikan dalam pernyataan kueri aturan, aturan tidak akan melakukannya.

- Periksa topik pesan masukan dalam pernyataan kueri aturan

Agar aturan berfungsi, aturan harus menerima pesan dengan nama topik yang cocok dengan filter topik dalam klausa FROM dari pernyataan kueri aturan.

Periksa ejaan filter topik dalam pernyataan kueri aturan dengan topik di klien MQTT. Nama topik peka huruf besar/kecil dan topik pesan harus cocok dengan filter topik dalam pernyataan kueri aturan.

- Periksa isi payload pesan masukan

Agar aturan berfungsi, ia harus menemukan bidang data di payload pesan yang dideklarasikan dalam pernyataan SELECT.

Periksa ejaan temperature bidang dalam pernyataan kueri aturan dengan payload pesan di klien MQTT. Nama bidang peka huruf besar/kecil dan temperature bidang dalam pernyataan kueri aturan harus identik dengan temperature bidang di payload pesan.

Pastikan dokumen JSON di payload pesan diformat dengan benar. Jika JSON memiliki kesalahan, seperti koma yang hilang, aturan tidak akan dapat membacanya.

- Periksa topik pesan yang diterbitkan ulang dalam tindakan aturan

Topik di mana tindakan aturan Republish menerbitkan pesan baru harus sesuai dengan topik yang Anda berlangganan di klien MQTT.

Buka aturan yang Anda buat di konsol dan periksa topik di mana tindakan aturan akan menerbitkan ulang pesan.

- Periksa peran yang digunakan oleh aturan

Tindakan aturan harus memiliki izin untuk menerima topik asli dan mempublikasikan topik baru.

Kebijakan yang mengizinkan aturan untuk menerima data pesan dan mempublikasikannya khusus untuk topik yang digunakan. Jika mengubah topik yang digunakan untuk mempublikasikan ulang data pesan, Anda harus memperbarui peran tindakan aturan untuk memperbarui kebijakannya agar sesuai dengan topik saat ini.

Jika Anda mencurigai ini masalahnya, edit tindakan aturan Republish dan buat peran baru. Peran baru yang dibuat oleh tindakan aturan menerima otorisasi yang diperlukan untuk melakukan tindakan ini.

Langkah 4: Tinjau hasil dan langkah selanjutnya

Dalam tutorial ini:

- Anda membuat dan menguji topik dan langganan notifikasi Amazon SNS.
- Anda menggunakan kueri dan fungsi SQL sederhana dalam pernyataan kueri aturan untuk membuat pesan baru untuk notifikasi Anda.
- Anda membuat AWS IoT aturan untuk mengirim notifikasi Amazon SNS yang menggunakan payload pesan yang disesuaikan.
- Anda menggunakan klien MQTT untuk menguji aturan Anda. AWS IoT

Langkah selanjutnya

Setelah Anda mengirim beberapa pesan teks dengan aturan ini, coba bereksperimen dengannya untuk melihat bagaimana mengubah beberapa aspek tutorial memengaruhi pesan dan kapan dikirim. Berikut adalah beberapa ide untuk Anda mulai.

- Ubah `device_id` dalam topik pesan masukan dan amati efeknya dalam isi pesan teks.
- Ubah bidang yang dipilih dalam pernyataan kueri aturan dan amati efeknya dalam isi pesan teks.
- Ubah pengujian dalam pernyataan kueri aturan untuk menguji suhu minimum, bukan suhu maksimum. Ingatlah untuk mengubah `namamax_temperature`!
- Tambahkan tindakan aturan penerbitan ulang untuk mengirim pesan MQTT saat pemberitahuan SNS dikirim.
- Coba tutorial berikutnya dalam seri ini dan pelajari caranya [Tutorial: Menyimpan data perangkat dalam tabel DynamoDB](#).

Tutorial: Menyimpan data perangkat dalam tabel DynamoDB

Tutorial ini menunjukkan cara membuat AWS IoT aturan yang mengirimkan data pesan ke tabel DynamoDB.

Dalam tutorial ini, Anda membuat aturan yang mengirimkan data pesan dari perangkat sensor cuaca imajiner ke tabel DynamoDB. Aturan memformat data dari banyak sensor cuaca sehingga dapat ditambahkan ke tabel database tunggal.

Apa yang akan Anda pelajari dalam tutorial ini

- Cara membuat tabel DynamoDB
- Cara mengirim data pesan ke tabel DynamoDB dari aturan AWS IoT
- Cara menggunakan templat substitusi dalam aturan AWS IoT
- Cara menggunakan kueri dan fungsi SQL sederhana dalam pernyataan kueri aturan
- Cara menggunakan klien MQTT untuk menguji aturan AWS IoT

Tutorial ini membutuhkan waktu sekitar 30 menit untuk menyelesaikannya.

Dalam tutorial ini, Anda akan:

- [Langkah 1: Buat tabel DynamoDB untuk tutorial ini](#)

- [Langkah 2: Buat AWS IoT aturan untuk mengirim data ke tabel DynamoDB](#)
- [Langkah 3: Uji AWS IoT aturan dan tabel DynamoDB](#)
- [Langkah 4: Tinjau hasil dan langkah selanjutnya](#)

Sebelum Anda memulai tutorial ini, pastikan Anda memiliki:

- [Siapkan Anda Akun AWS](#)

Anda akan membutuhkan Akun AWS dan AWS IoT konsol Anda untuk menyelesaikan tutorial ini.

- Diulas [Lihat pesan MQTT dengan klien MQTT AWS IoT](#)

Pastikan Anda dapat menggunakan klien MQTT untuk berlangganan dan mempublikasikan ke suatu topik. Anda akan menggunakan klien MQTT untuk menguji aturan baru Anda dalam prosedur ini.

- Meninjau ikhtisar [Amazon DynamoDB](#)

Jika Anda belum pernah menggunakan DynamoDB sebelumnya, [tinjau Memulai dengan DynamoDB](#) agar terbiasa dengan konsep dasar dan operasi DynamoDB.

Langkah 1: Buat tabel DynamoDB untuk tutorial ini

Dalam tutorial ini, Anda akan membuat tabel DynamoDB dengan atribut ini untuk merekam data dari perangkat sensor cuaca imajiner:

- `sample_time` adalah kunci utama dan menjelaskan waktu sampel direkam.
- `device_id` adalah kunci sortir dan menjelaskan perangkat yang menyediakan sampel
- `device_data` adalah data yang diterima dari perangkat dan diformat oleh pernyataan kueri aturan

Untuk membuat tabel DynamoDB untuk tutorial ini

1. Buka konsol [DynamoDB](#), lalu pilih Buat tabel.
2. Di Buat tabel:
 - a. Dalam nama Tabel, masukkan nama tabel: **wx_data**.
 - b. Di kunci Partisi **sample_time**, masukkan, dan dalam daftar opsi di sebelah bidang, pilih **Number**.

- c. Di Sortir kunci **device_id**, masukkan, dan dalam daftar opsi di sebelah bidang, pilih **Number**.
- d. Di bagian bawah halaman, pilih **Buat**.

Anda akan menentukan `device_data` nanti, ketika Anda mengkonfigurasi tindakan aturan DynamoDB.

Langkah 2: Buat AWS IoT aturan untuk mengirim data ke tabel DynamoDB

Pada langkah ini, Anda akan menggunakan pernyataan kueri aturan untuk memformat data dari perangkat sensor cuaca imajiner untuk menulis ke tabel database.

Contoh payload pesan yang diterima dari perangkat sensor cuaca terlihat seperti ini:

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

Untuk entri database, Anda akan menggunakan pernyataan kueri aturan untuk meratakan struktur payload pesan agar terlihat seperti ini:

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind_velocity": 22,
  "wind_bearing": 255
}
```

Dalam aturan ini, Anda juga akan menggunakan beberapa [Templat substitusi](#). Templat substitusi adalah ekspresi yang memungkinkan Anda menyisipkan nilai dinamis dari fungsi dan data pesan.

Untuk membuat AWS IoT aturan untuk mengirim data ke tabel DynamoDB

1. Buka [hub Aturan AWS IoT konsol](#). Atau, Anda dapat membuka AWS IoT beranda di dalam AWS Management Console dan menavigasi ke Perutean Pesan > Aturan.
2. Untuk mulai membuat aturan baru Anda di Aturan, pilih Buat aturan.
3. Dalam properti Aturan:
 - a. Dalam nama Aturan, masukkan `wx_data_ddb`.

Ingat bahwa nama aturan harus unik di dalam Anda Akun AWS dan Wilayah, dan tidak dapat memiliki spasi apa pun. Kami telah menggunakan karakter garis bawah dalam nama ini untuk memisahkan dua kata dalam nama aturan.

- b. Dalam Deskripsi aturan, jelaskan aturannya.

Deskripsi yang bermakna membuatnya lebih mudah untuk mengingat apa yang dilakukan aturan ini dan mengapa Anda membuatnya. Deskripsi bisa selama dibutuhkan, jadi sedetail mungkin.

4. Pilih Next untuk melanjutkan.
5. Dalam pernyataan SQL:
 - a. Dalam versi SQL, pilih **2016-03-23**.
 - b. Dalam kotak edit pernyataan SQL, masukkan pernyataan:


```
SELECT temperature, humidity, barometer,  
       wind.velocity as wind_velocity,  
       wind.bearing as wind_bearing,  
FROM 'device/+/data'
```

Pernyataan ini:

- Mendengarkan pesan MQTT dengan topik yang cocok dengan filter topik. `device/+/data`
- Memformat elemen `wind` atribut sebagai atribut individual.
- Melewati `temperature`, `humidity`, dan `barometer` atribut tidak berubah.

6. Pilih Next untuk melanjutkan.
7. Dalam tindakan Aturan:

- a. Untuk membuka daftar tindakan aturan untuk aturan ini, di Tindakan 1, pilih **DynamoDB**.

 Note

Pastikan Anda memilih DynamoDB dan bukan DynamoDBv2 sebagai tindakan aturan.

- b. Dalam nama Tabel, pilih nama tabel DynamoDB yang Anda buat pada langkah sebelumnya: **wx_data**

Tipe kunci Partition dan kolom tipe kunci Sort diisi dengan nilai-nilai dari tabel DynamoDB Anda.

- c. Di kunci Partisi, masukkan **sample_time**.
- d. Dalam nilai kunci partisi, masukkan **`${timestamp()}`**.

Ini adalah yang pertama dari yang akan [Templat substitusi](#) Anda gunakan dalam aturan ini. Alih-alih menggunakan nilai dari payload pesan, itu akan menggunakan nilai yang dikembalikan dari fungsi stempel waktu. Untuk mempelajari selengkapnya, lihat [stempel waktu](#) di Panduan AWS IoT Core Pengembang.

- e. Di tombol Sortir, masukkan **device_id**.
- f. Di Sortir nilai kunci, masukkan **`${cast(topic(2) AS DECIMAL)}`**.

Ini adalah yang kedua dari yang akan [Templat substitusi](#) Anda gunakan dalam aturan ini. Ini menyisipkan nilai elemen kedua dalam nama topik, yang merupakan ID perangkat, setelah itu melemparkan ke nilai DECIMAL untuk mencocokkan format numerik kunci. Untuk mempelajari lebih lanjut tentang topik, lihat [topik](#) di Panduan AWS IoT Core Pengembang. Atau untuk mempelajari lebih lanjut tentang casting, lihat [pemeran](#) di Panduan AWS IoT Core Pengembang.

- g. Di Tulis data pesan ke kolom ini, masukkan **device_data**.

Ini akan membuat `device_data` kolom dalam tabel DynamoDB.

- h. Biarkan Operasi kosong.
- i. Dalam peran IAM, pilih Buat peran baru.
- j. Dalam kotak dialog Buat peran, untuk nama Peran, masukkan `wx_ddb_role`. Peran baru ini secara otomatis akan berisi kebijakan dengan awalan "aws-iot-rule" yang akan

memungkinkan **wx_data_ddb** aturan untuk mengirim data ke tabel **wx_data** DynamoDB yang Anda buat.

- k. Dalam peran IAM, pilih **wx_ddb_role**.
 - l. Di bagian bawah halaman, pilih Selanjutnya.
8. Di bagian bawah halaman Tinjau dan buat, pilih Buat untuk membuat aturan.

Langkah 3: Uji AWS IoT aturan dan tabel DynamoDB

Untuk menguji aturan baru, Anda akan menggunakan klien MQTT untuk mempublikasikan dan berlangganan pesan MQTT yang digunakan dalam pengujian ini.

Buka [klien MQTT di AWS IoT konsol di jendela](#) baru. Ini akan memungkinkan Anda mengedit aturan tanpa kehilangan konfigurasi klien MQTT Anda. Klien MQTT tidak menyimpan langganan atau log pesan apa pun jika Anda membiarkannya pergi ke halaman lain di konsol. Anda juga ingin jendela konsol terpisah terbuka ke [hub DynamoDB Tables di AWS IoT](#) konsol untuk melihat entri baru yang dikirim aturan Anda.

Untuk menggunakan klien MQTT untuk menguji aturan Anda

1. Di [klien MQTT di AWS IoT konsol](#), berlangganan topik input, `device/+ /data`
 - a. Di klien MQTT, pilih Berlangganan topik.
 - b. Untuk filter Topik, masukkan topik filter topik masukan, **`device/+ /data`**.
 - c. Pilih Langganan.
2. Sekarang, publikasikan pesan ke topik input dengan ID perangkat tertentu, **`device/22/data`**. Anda tidak dapat mempublikasikan ke topik MQTT yang berisi karakter wildcard.
 - a. Di klien MQTT, pilih Publikasikan ke topik.
 - b. Untuk nama Topik, masukkan nama topik masukan, **`device/22/data`**.
 - c. Untuk payload Pesan, masukkan contoh data berikut.

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

```
}  
}
```

- d. Untuk mempublikasikan pesan MQTT, pilih Publish.
 - e. Sekarang, di klien MQTT, pilih Berlangganan topik. Di kolom Berlangganan, pilih **device/+data** langganan. Konfirmasikan bahwa data sampel dari langkah sebelumnya muncul di sana.
3. Periksa untuk melihat baris dalam tabel DynamoDB yang aturan Anda buat.

- a. Di [hub DynamoDB Tables di AWS IoT](#) konsol, pilih wx_data, lalu pilih tab Items.

Jika Anda sudah berada di tab Item, Anda mungkin perlu menyegarkan tampilan dengan memilih ikon penyegaran di sudut kanan atas header tabel.

- b. Perhatikan bahwa nilai sample_time dalam tabel adalah link dan buka satu. Jika Anda baru saja mengirim pesan pertama Anda, itu akan menjadi satu-satunya dalam daftar.

Tautan ini menampilkan semua data di baris tabel tersebut.

- c. Perluas entri device_data untuk melihat data yang dihasilkan dari pernyataan kueri aturan.
- d. Jelajahi berbagai representasi data yang tersedia di tampilan ini. Anda juga dapat mengedit data di tampilan ini.
- e. Setelah Anda selesai meninjau baris data ini, untuk menyimpan perubahan apa pun yang Anda buat, pilih Simpan, atau keluar tanpa menyimpan perubahan apa pun, pilih Batal.

Jika Anda tidak melihat perilaku yang benar, periksa tips pemecahan masalah.

Memecahkan masalah aturan DynamoDB Anda

Berikut adalah beberapa hal untuk diperiksa jika Anda tidak melihat hasil yang Anda harapkan.

- Anda mendapat spanduk kesalahan

Jika kesalahan muncul saat Anda mempublikasikan pesan input, perbaiki kesalahan itu terlebih dahulu. Langkah-langkah berikut dapat membantu Anda memperbaiki kesalahan itu.

- Anda tidak melihat pesan masukan di klien MQTT

Setiap kali Anda mempublikasikan pesan masukan ke device/22/data topik, pesan itu akan muncul di klien MQTT jika Anda berlangganan filter device/+data topik seperti yang dijelaskan [dalam prosedur](#).

Hal-hal yang harus diperiksa

- Periksa filter topik yang Anda langgani

Jika Anda berlangganan topik pesan input seperti yang dijelaskan dalam prosedur, Anda akan melihat salinan pesan input setiap kali Anda mempublikasikannya.

Jika Anda tidak melihat pesan, periksa nama topik yang Anda langgani dan bandingkan dengan topik yang Anda terbitkan. Nama topik peka huruf besar/kecil dan topik yang Anda langgani harus identik dengan topik yang Anda publikasikan payload pesan.

- Periksa fungsi publikasi pesan

Di klien MQTT, di bawah Langganan, pilih perangkat/+/data, periksa topik pesan terbitkan, lalu pilih Publikasikan ke topik. Anda akan melihat payload pesan dari kotak edit di bawah topik yang muncul di daftar pesan.

- Anda tidak melihat data Anda di tabel DynamoDB

Hal pertama yang harus dilakukan adalah menyegarkan tampilan dengan memilih ikon penyegaran di sudut kanan atas header tabel. Jika itu tidak menampilkan data yang Anda cari, periksa yang berikut ini.

Hal-hal yang harus diperiksa

- Periksa klien MQTT Anda dan aturan yang Anda buat Wilayah AWS

Konsol tempat Anda menjalankan klien MQTT harus berada di AWS Wilayah yang sama dengan aturan yang Anda buat.

- Periksa topik pesan masukan dalam pernyataan kueri aturan

Agar aturan berfungsi, aturan harus menerima pesan dengan nama topik yang cocok dengan filter topik dalam klausa FROM dari pernyataan kueri aturan.

Periksa ejaan filter topik dalam pernyataan kueri aturan dengan topik di klien MQTT. Nama topik peka huruf besar/kecil dan topik pesan harus cocok dengan filter topik dalam pernyataan kueri aturan.

- Periksa isi payload pesan masukan

Agar aturan berfungsi, ia harus menemukan bidang data di payload pesan yang dideklarasikan dalam pernyataan SELECT.

Periksa ejaan temperature bidang dalam pernyataan kueri aturan dengan payload pesan di klien MQTT. Nama bidang peka huruf besar/kecil dan temperature bidang dalam pernyataan kueri aturan harus identik dengan temperature bidang di payload pesan.

Pastikan dokumen JSON di payload pesan diformat dengan benar. Jika JSON memiliki kesalahan, seperti koma yang hilang, aturan tidak akan dapat membacanya.

- Periksa nama kunci dan bidang yang digunakan dalam tindakan aturan

Nama bidang yang digunakan dalam aturan topik harus cocok dengan nama yang ditemukan di payload pesan JSON dari pesan yang dipublikasikan.

Buka aturan yang Anda buat di konsol dan periksa nama bidang dalam konfigurasi tindakan aturan dengan yang digunakan dalam klien MQTT.

- Periksa peran yang digunakan oleh aturan

Tindakan aturan harus memiliki izin untuk menerima topik asli dan mempublikasikan topik baru.

Kebijakan yang mengotorisasi aturan untuk menerima data pesan dan memperbarui tabel DynamoDB khusus untuk topik yang digunakan. Jika mengubah topik atau nama tabel DynamoDB yang digunakan oleh aturan, Anda harus memperbarui peran tindakan aturan untuk memperbarui kebijakannya agar sesuai.

Jika Anda mencurigai ini masalahnya, edit tindakan aturan dan buat peran baru. Peran baru yang dibuat oleh tindakan aturan menerima otorisasi yang diperlukan untuk melakukan tindakan ini.

Langkah 4: Tinjau hasil dan langkah selanjutnya

Setelah Anda mengirim beberapa pesan ke tabel DynamoDB dengan aturan ini, coba bereksperimen dengannya untuk melihat bagaimana mengubah beberapa aspek dari tutorial memengaruhi data yang ditulis ke tabel. Berikut adalah beberapa ide untuk Anda mulai.

- Ubah *device_id* dalam topik pesan masukan dan amati efeknya pada data. Anda dapat menggunakan ini untuk mensimulasikan penerimaan data dari beberapa sensor cuaca.
- Ubah bidang yang dipilih dalam pernyataan kueri aturan dan amati efeknya pada data. Anda dapat menggunakan ini untuk memfilter data yang disimpan dalam tabel.

- Tambahkan tindakan aturan penerbitan ulang untuk mengirim pesan MQTT untuk setiap baris yang ditambahkan ke tabel. Anda dapat menggunakan ini untuk debugging.

Setelah Anda menyelesaikan tutorial ini, periksa [the section called “Memformat notifikasi dengan menggunakan fungsi AWS Lambda”](#).

Tutorial: Memformat notifikasi dengan menggunakan fungsi AWS Lambda

Tutorial ini menunjukkan cara mengirim data pesan MQTT ke AWS Lambda tindakan untuk memformat dan mengirim ke layanan lain. AWS Dalam tutorial ini, AWS Lambda tindakan menggunakan AWS SDK untuk mengirim pesan yang diformat ke topik Amazon SNS yang Anda buat dalam tutorial tentang cara melakukannya. [the section called “Mengirim notifikasi Amazon SNS”](#)

Dalam tutorial tentang bagaimana caranya [the section called “Mengirim notifikasi Amazon SNS”](#), dokumen JSON yang dihasilkan dari pernyataan kueri aturan dikirim sebagai badan pesan teks. Hasilnya adalah pesan teks yang terlihat seperti contoh ini:

```
{"device_id":"32","reported_temperature":38,"max_temperature":30}
```

Dalam tutorial ini, Anda akan menggunakan tindakan AWS Lambda aturan untuk memanggil AWS Lambda fungsi yang memformat data dari pernyataan kueri aturan ke dalam format yang lebih ramah, seperti contoh ini:

```
Device 32 reports a temperature of 38, which exceeds the limit of 30.
```

AWS Lambda Fungsi yang akan Anda buat dalam tutorial ini memformat string pesan dengan menggunakan data dari pernyataan kueri aturan dan memanggil fungsi [publikasi SNS](#) AWS SDK untuk membuat notifikasi.

Apa yang akan Anda pelajari dalam tutorial ini

- Cara membuat dan menguji suatu AWS Lambda fungsi
- Cara menggunakan AWS SDK dalam AWS Lambda fungsi untuk mempublikasikan notifikasi Amazon SNS
- Cara menggunakan kueri dan fungsi SQL sederhana dalam pernyataan kueri aturan
- Cara menggunakan klien MQTT untuk menguji aturan AWS IoT

Tutorial ini membutuhkan waktu sekitar 45 menit untuk menyelesaikannya.

Dalam tutorial ini, Anda akan:

- [Langkah 1: Buat AWS Lambda fungsi yang mengirim pesan teks](#)
- [Langkah 2: Buat AWS IoT aturan dengan tindakan AWS Lambda aturan](#)
- [Langkah 3: Uji AWS IoT aturan dan tindakan AWS Lambda aturan](#)
- [Langkah 4: Tinjau hasil dan langkah selanjutnya](#)

Sebelum Anda memulai tutorial ini, pastikan Anda memiliki:

- [Siapkan Anda Akun AWS](#)

Anda akan membutuhkan Akun AWS dan AWS IoT konsol Anda untuk menyelesaikan tutorial ini.

- Diulas [Lihat pesan MQTT dengan klien MQTT AWS IoT](#)

Pastikan Anda dapat menggunakan klien MQTT untuk berlangganan dan mempublikasikan ke suatu topik. Anda akan menggunakan klien MQTT untuk menguji aturan baru Anda dalam prosedur ini.

- Menyelesaikan tutorial aturan lain di bagian ini

Tutorial ini membutuhkan topik notifikasi SNS yang Anda buat di tutorial tentang cara [the section called “Mengirim notifikasi Amazon SNS”](#) melakukannya. Ini juga mengasumsikan bahwa Anda telah menyelesaikan tutorial terkait aturan lainnya di bagian ini.

- Meninjau [AWS Lambda](#)ikhtisar

Jika Anda belum pernah menggunakannya AWS Lambda sebelumnya, tinjau [AWS Lambda](#) dan [Mulai dengan Lambda](#) untuk mempelajari istilah dan konsepnya.

Langkah 1: Buat AWS Lambda fungsi yang mengirim pesan teks

AWS Lambda Fungsi dalam tutorial ini menerima hasil dari pernyataan kueri aturan, menyisipkan elemen ke dalam string teks, dan mengirimkan string yang dihasilkan ke Amazon SNS sebagai pesan dalam notifikasi.

Berbeda dengan tutorial tentang bagaimana caranya [the section called “Mengirim notifikasi Amazon SNS”](#), yang menggunakan tindakan AWS IoT aturan untuk mengirim notifikasi, tutorial ini mengirimkan notifikasi dari fungsi Lambda dengan menggunakan fungsi SDK. AWS Topik notifikasi Amazon SNS yang sebenarnya digunakan dalam tutorial ini, bagaimanapun, adalah topik yang sama

dengan yang Anda gunakan dalam tutorial tentang cara melakukannya. [the section called “Mengirim notifikasi Amazon SNS”](#)

Untuk membuat AWS Lambda fungsi yang mengirim pesan teks

1. Buat AWS Lambda fungsi baru.
 - a. Di [AWS Lambda konsol](#), pilih Buat fungsi.
 - b. Dalam fungsi Buat, pilih Gunakan cetak biru.

Cari dan pilih **hello-world-python** cetak biru, lalu pilih Konfigurasi.
 - c. Dalam informasi Dasar:
 - i. Dalam nama Fungsi, masukkan nama fungsi ini, **format-high-temp-notification**.
 - ii. Di Peran eksekusi, pilih Buat peran baru dari templat AWS kebijakan.
 - iii. Di Nama peran, masukkan nama peran baru, **format-high-temp-notification-role**.
 - iv. Di Templat kebijakan - opsional, cari dan pilih kebijakan publikasi Amazon SNS.
 - v. Pilih Buat fungsi.
2. Ubah kode cetak biru untuk memformat dan mengirim pemberitahuan Amazon SNS.
 - a. Setelah Anda membuat fungsi Anda, Anda akan melihat halaman format-high-temp-notificationdetail. Jika tidak, buka dari halaman [Fungsi Lambda](#).
 - b. Di halaman format-high-temp-notificationdetail, pilih tab Konfigurasi dan gulir ke panel Kode fungsi.
 - c. Di jendela Function code, di panel Environment, pilih file Python, `lambda_function.py`
 - d. Di jendela Function code, hapus semua kode program asli dari cetak biru dan ganti dengan kode ini.

```
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. Pilih Deploy.
3. Di jendela baru, cari Nama Sumber Daya Amazon (ARN) dari topik Amazon SNS Anda dari tutorial tentang cara melakukannya. [the section called “Mengirim notifikasi Amazon SNS”](#)
 - a. Di jendela baru, buka [halaman Topik konsol Amazon SNS](#).
 - b. Di halaman Topik, temukan topik notifikasi high_temp_notice dalam daftar topik Amazon SNS.

- c. Temukan ARN dari topik notifikasi `high_temp_notice` untuk digunakan pada langkah berikutnya.
4. Buat kasus uji untuk fungsi Lambda Anda.
 - a. Di halaman [Fungsi Lambda](#) konsol, pada halaman `format-high-temp-notificationdetail`, pilih `Pilih acara pengujian` di sudut kanan atas halaman (meskipun terlihat dinonaktifkan), lalu pilih `Konfigurasi peristiwa pengujian`.
 - b. Di `Configure test event`, pilih `Create new test event`.
 - c. Dalam nama acara, masukkan **`SampleRuleOutput`**.
 - d. Di editor JSON di bawah nama Acara, tempel contoh dokumen JSON ini. Ini adalah contoh dari apa yang akan dikirim AWS IoT aturan Anda ke fungsi Lambda.

```
{
  "device_id": "32",
  "reported_temperature": 38,
  "max_temperature": 30,
  "notify_topic_arn": "arn:aws:sns:us-east-1:57EXAMPLE833:high_temp_notice"
}
```

- e. Lihat jendela yang memiliki ARN dari topik notifikasi `high_temp_notice` dan salin nilai ARN.
 - f. Ganti `notify_topic_arn` nilai di editor JSON dengan ARN dari topik notifikasi Anda.

Biarkan jendela ini tetap terbuka sehingga Anda dapat menggunakan nilai ARN ini lagi saat Anda membuat aturan. AWS IoT
 - g. Pilih `Buat`.
5. Uji fungsi dengan data sampel.
 - a. Di halaman `format-high-temp-notificationdetail`, di sudut kanan atas halaman, konfirmasi yang `SampleRuleOutput` muncul di sebelah tombol `Uji`. Jika tidak, pilih dari daftar acara pengujian yang tersedia.
 - b. Untuk mengirim pesan keluaran aturan sampel ke fungsi Anda, pilih `Uji`.

Jika fungsi dan notifikasi keduanya berfungsi, Anda akan mendapatkan pesan teks di ponsel yang berlangganan notifikasi.

Jika Anda tidak mendapatkan pesan teks di telepon, periksa hasil operasi. Di panel Kode fungsi, di tab Hasil eksekusi, tinjau respons untuk menemukan kesalahan yang terjadi. Jangan melanjutkan ke langkah berikutnya sampai fungsi Anda dapat mengirim notifikasi ke ponsel Anda.

Langkah 2: Buat AWS IoT aturan dengan tindakan AWS Lambda aturan

Pada langkah ini, Anda akan menggunakan pernyataan kueri aturan untuk memformat data dari perangkat sensor cuaca imajiner untuk dikirim ke fungsi Lambda, yang akan memformat dan mengirim pesan teks.

Contoh payload pesan yang diterima dari perangkat cuaca terlihat seperti ini:

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

Dalam aturan ini, Anda akan menggunakan pernyataan kueri aturan untuk membuat payload pesan untuk fungsi Lambda yang terlihat seperti ini:

```
{
  "device_id": "32",
  "reported_temperature": 38,
  "max_temperature": 30,
  "notify_topic_arn": "arn:aws:sns:us-east-1:57EXAMPLE833:high_temp_notice"
}
```

Ini berisi semua informasi yang diperlukan fungsi Lambda untuk memformat dan mengirim pesan teks yang benar.

Untuk membuat AWS IoT aturan untuk memanggil fungsi Lambda

1. Buka [hub Aturan AWS IoT konsol](#).
2. Untuk mulai membuat aturan baru Anda di Aturan, pilih Buat.
3. Di bagian atas Buat aturan:

- a. Dalam Nama, masukkan nama aturan, **wx_friendly_text**.

Ingat bahwa nama aturan harus unik di dalam Anda Akun AWS dan Wilayah, dan tidak dapat memiliki spasi apa pun. Kami telah menggunakan karakter garis bawah dalam nama ini untuk memisahkan dua kata dalam nama aturan.

- b. Dalam Deskripsi, jelaskan aturannya.

Deskripsi yang bermakna membuatnya lebih mudah untuk mengingat apa yang dilakukan aturan ini dan mengapa Anda membuatnya. Deskripsi bisa selama dibutuhkan, jadi sedetail mungkin.

4. Dalam pernyataan kueri Aturan Buat aturan:

- a. Dalam Menggunakan versi SQL, pilih **2016-03-23**.
- b. Dalam kotak edit pernyataan kueri aturan, masukkan pernyataan:

```
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
```

Pernyataan ini:

- Mendengarkan pesan MQTT dengan topik yang cocok dengan filter `device/+/data` topik dan yang memiliki `temperature` nilai lebih besar dari 30.
 - Memilih elemen kedua dari string topik, mengubahnya menjadi angka desimal, dan kemudian menentukannya ke bidang `device_id`
 - Memilih nilai `temperature` bidang dari payload pesan dan menentukannya ke bidang `reported_temperature`
 - Menciptakan nilai konstan `30`, untuk mewakili nilai batas dan menentukannya ke `max_temperature` bidang.
 - Menciptakan nilai konstan untuk `notify_topic_arn` bidang.
- c. Lihat jendela yang memiliki ARN dari topik notifikasi `high_temp_notice` dan salin nilai ARN.
 - d. Ganti nilai ARN (`arn:aws:sns:us-east-1:57 Example833:HIGH_TEMP_NOTICE`) di editor pernyataan kueri aturan dengan ARN topik notifikasi Anda.

5. Di Tetapkan satu atau lebih tindakan:
 - a. Untuk membuka daftar tindakan aturan untuk aturan ini, pilih Tambah tindakan.
 - b. Di Pilih tindakan, pilih Kirim pesan ke fungsi Lambda.
 - c. Untuk membuka halaman konfigurasi tindakan yang dipilih, di bagian bawah daftar tindakan, pilih Konfigurasi tindakan.
6. Dalam tindakan Konfigurasi:
 - a. Di Nama fungsi, pilih Pilih.
 - b. Pilih format-high-temp-notification.
 - c. Di bagian bawah Configure action, pilih Add action.
 - d. Untuk membuat aturan, di bagian bawah Buat aturan, pilih Buat aturan.

Langkah 3: Uji AWS IoT aturan dan tindakan AWS Lambda aturan

Untuk menguji aturan baru Anda, Anda akan menggunakan klien MQTT untuk mempublikasikan dan berlangganan pesan MQTT yang digunakan oleh aturan ini.

Buka [klien MQTT di AWS IoT konsol di jendela](#) baru. Sekarang Anda dapat mengedit aturan tanpa kehilangan konfigurasi klien MQTT Anda. Jika Anda meninggalkan klien MQTT untuk pergi ke halaman lain di konsol, Anda akan kehilangan langganan atau log pesan Anda.

Untuk menggunakan klien MQTT untuk menguji aturan Anda

1. Di [klien MQTT di AWS IoT konsol](#), berlangganan topik input, dalam hal ini, `device/+ /data`
 - a. Di klien MQTT, di bawah Langganan, pilih Berlangganan topik.
 - b. Dalam topik Langganan, masukkan topik filter topik masukan, **`device/+ /data`**.
 - c. Simpan sisa bidang di pengaturan defaultnya.
 - d. Pilih Berlangganan topik.

Di kolom Langganan, di bawah Publikasikan ke topik, **`device/+ /data`** muncul.

2. Publikasikan pesan ke topik input dengan ID perangkat tertentu, **`device/32/data`**. Anda tidak dapat mempublikasikan ke topik MQTT yang berisi karakter wildcard.
 - a. Di klien MQTT, di bawah Langganan, pilih Publikasikan ke topik.
 - b. Di bidang Publikasikan, masukkan nama topik masukan, **`device/32/data`**.

- c. Salin data sampel yang ditampilkan di sini dan, di kotak edit di bawah nama topik, tempel data sampel.

```
{
  "temperature": 38,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

- d. Untuk mempublikasikan pesan MQTT Anda, pilih Publikasikan ke topik.
3. Konfirmasikan bahwa pesan teks telah dikirim.
 - a. Di klien MQTT, di bawah Langganan, ada titik hijau di sebelah topik yang Anda berlangganan sebelumnya.

Titik hijau menunjukkan bahwa satu atau lebih pesan baru telah diterima sejak terakhir kali Anda melihatnya.
 - b. Di bawah Langganan, pilih perangkat+/data untuk memeriksa apakah muatan pesan cocok dengan apa yang baru saja Anda terbitkan dan terlihat seperti ini:

```
{
  "temperature": 38,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

- c. Periksa ponsel yang Anda gunakan untuk berlangganan topik SNS dan konfirmasikan isi payload pesan terlihat seperti ini:

```
Device 32 reports a temperature of 38, which exceeds the limit of 30.
```

Jika Anda mengubah elemen ID topik dalam topik pesan, ingatlah bahwa `topic(2)` mentransmisikan nilai ke nilai numerik hanya akan berfungsi jika elemen dalam topik pesan hanya berisi karakter numerik.

4. Coba kirim pesan MQTT di mana suhunya tidak melebihi batas.
 - a. Di klien MQTT, di bawah Langganan, pilih Publikasikan ke topik.
 - b. Di bidang Publikasikan, masukkan nama topik masukan, **device/33/data**.
 - c. Salin data sampel yang ditampilkan di sini dan, di kotak edit di bawah nama topik, tempel data sampel.

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

- d. Untuk mengirim pesan MQTT Anda, pilih Publikasikan ke topik.

Anda akan melihat pesan yang Anda kirim dalam **device/+data** langganan; Namun, karena nilai suhu di bawah suhu maks dalam pernyataan kueri aturan, Anda seharusnya tidak menerima pesan teks.

Jika Anda tidak melihat perilaku yang benar, periksa tips pemecahan masalah.

Memecahkan masalah AWS Lambda aturan dan pemberitahuan

Berikut adalah beberapa hal untuk diperiksa, jika Anda tidak melihat hasil yang Anda harapkan.

- Anda mendapat spanduk kesalahan

Jika kesalahan muncul saat Anda mempublikasikan pesan input, perbaiki kesalahan itu terlebih dahulu. Langkah-langkah berikut dapat membantu Anda memperbaiki kesalahan itu.

- Anda tidak melihat pesan masukan di klien MQTT

Setiap kali Anda mempublikasikan pesan masukan Anda ke `device/32/data` topik, pesan itu akan muncul di klien MQTT, jika Anda berlangganan filter `device/+/data` topik seperti yang dijelaskan dalam prosedur.

Hal-hal yang harus diperiksa

- Periksa filter topik yang Anda langgani

Jika Anda berlangganan topik pesan input seperti yang dijelaskan dalam prosedur, Anda akan melihat salinan pesan input setiap kali Anda mempublikasikannya.

Jika Anda tidak melihat pesan, periksa nama topik yang Anda langgani dan bandingkan dengan topik yang Anda terbitkan. Nama topik peka huruf besar/kecil dan topik yang Anda langgani harus identik dengan topik yang Anda publikasikan payload pesan.

- Periksa fungsi publikasi pesan

Di klien MQTT, di bawah Langganan, pilih perangkat/`+/data`, periksa topik pesan terbitkan, lalu pilih Publikasikan ke topik. Anda akan melihat payload pesan dari kotak edit di bawah topik yang muncul di daftar pesan.

- Anda tidak menerima pesan SMS

Agar aturan Anda berfungsi, itu harus memiliki kebijakan yang benar yang mengizinkannya untuk menerima pesan dan mengirim pemberitahuan SNS, dan itu harus menerima pesan.

Hal-hal yang harus diperiksa

- Periksa klien MQTT Anda dan aturan yang Anda buat Wilayah AWS

Konsol tempat Anda menjalankan klien MQTT harus berada di AWS Wilayah yang sama dengan aturan yang Anda buat.

- Periksa apakah nilai suhu dalam muatan pesan melebihi ambang uji

Jika nilai suhu kurang dari atau sama dengan 30, seperti yang didefinisikan dalam pernyataan kueri aturan, aturan tidak akan melakukannya.

- Periksa topik pesan masukan dalam pernyataan kueri aturan

Agar aturan berfungsi, aturan harus menerima pesan dengan nama topik yang cocok dengan filter topik dalam klausa FROM dari pernyataan kueri aturan.

Periksa ejaan filter topik dalam pernyataan kueri aturan dengan topik di klien MQTT. Nama topik peka huruf besar/kecil dan topik pesan harus cocok dengan filter topik dalam pernyataan kueri aturan.

- Periksa isi payload pesan masukan

Agar aturan berfungsi, ia harus menemukan bidang data di payload pesan yang dideklarasikan dalam pernyataan SELECT.

Periksa ejaan `temperature` bidang dalam pernyataan kueri aturan dengan payload pesan di klien MQTT. Nama bidang peka huruf besar/kecil dan `temperature` bidang dalam pernyataan kueri aturan harus identik dengan `temperature` bidang di payload pesan.

Pastikan dokumen JSON di payload pesan diformat dengan benar. Jika JSON memiliki kesalahan, seperti koma yang hilang, aturan tidak akan dapat membacanya.

- Periksa notifikasi Amazon SNS

Di [Langkah 1: Buat topik Amazon SNS yang mengirim pesan teks SMS](#), lihat langkah 3 yang menjelaskan cara menguji notifikasi Amazon SNS dan menguji notifikasi untuk memastikan notifikasi berfungsi.

- Periksa fungsi Lambda

Di [Langkah 1: Buat AWS Lambda fungsi yang mengirim pesan teks](#), lihat langkah 5 yang menjelaskan cara menguji fungsi Lambda menggunakan data uji dan menguji fungsi Lambda.

- Periksa peran yang digunakan oleh aturan

Tindakan aturan harus memiliki izin untuk menerima topik asli dan mempublikasikan topik baru.

Kebijakan yang mengizinkan aturan untuk menerima data pesan dan mempublikasikannya khusus untuk topik yang digunakan. Jika mengubah topik yang digunakan untuk mempublikasikan ulang data pesan, Anda harus memperbarui peran tindakan aturan untuk memperbarui kebijakannya agar sesuai dengan topik saat ini.

Jika Anda mencurigai ini masalahnya, edit tindakan aturan `Republish` dan buat peran baru. Peran baru yang dibuat oleh tindakan aturan menerima otorisasi yang diperlukan untuk melakukan tindakan ini.

Langkah 4: Tinjau hasil dan langkah selanjutnya

Dalam tutorial ini:

- Anda membuat AWS IoT aturan untuk memanggil fungsi Lambda yang mengirim notifikasi Amazon SNS yang menggunakan payload pesan yang disesuaikan.
- Anda menggunakan kueri dan fungsi SQL sederhana dalam pernyataan kueri aturan untuk membuat payload pesan baru untuk fungsi Lambda Anda.
- Anda menggunakan klien MQTT untuk menguji aturan Anda. AWS IoT

Langkah selanjutnya

Setelah Anda mengirim beberapa pesan teks dengan aturan ini, coba bereksperimen dengannya untuk melihat bagaimana mengubah beberapa aspek tutorial memengaruhi pesan dan kapan dikirim. Berikut adalah beberapa ide untuk Anda mulai.

- Ubah *device_id* dalam topik pesan masukan dan amati efeknya dalam isi pesan teks.
- Ubah bidang yang dipilih dalam pernyataan kueri aturan, perbarui fungsi Lambda untuk menggunakannya dalam pesan baru, dan amati efeknya dalam isi pesan teks.
- Ubah pengujian dalam pernyataan kueri aturan untuk menguji suhu minimum, bukan suhu maksimum. Perbarui fungsi Lambda untuk memformat pesan baru dan ingat untuk mengubah nama. `max_temperature`
- Untuk mempelajari lebih lanjut tentang cara menemukan kesalahan yang mungkin terjadi saat Anda mengembangkan dan menggunakan AWS IoT aturan, lihat [Pemantauan AWS IoT](#).

Mempertahankan status perangkat saat perangkat offline dengan Device Shadows

Tutorial ini menunjukkan kepada Anda cara menggunakan AWS IoT Layanan Device Shadow untuk menyimpan dan memperbarui informasi status perangkat. Dokumen Shadow, yang merupakan dokumen JSON, menunjukkan perubahan status perangkat berdasarkan pesan yang diterbitkan oleh perangkat, aplikasi lokal, atau layanan. Dalam tutorial ini, dokumen Shadow menunjukkan perubahan warna bola lampu. Tutorial ini juga menunjukkan bagaimana bayangan menyimpan informasi ini bahkan ketika perangkat terputus dari internet, dan meneruskan informasi status terbaru kembali ke perangkat ketika kembali online dan meminta informasi ini.

Sebaiknya Anda mencoba tutorial ini sesuai urutan yang ditampilkan di sini, dimulai dengan AWS IoT sumber daya yang Anda butuhkan untuk membuat dan pengaturan perangkat keras yang diperlukan, yang juga membantu Anda mempelajari konsep secara bertahap. Tutorial ini menunjukkan cara mengkonfigurasi dan menghubungkan perangkat Raspberry Pi untuk digunakan dengan AWS IoT. Jika Anda tidak memiliki perangkat keras yang diperlukan, Anda dapat mengikuti tutorial ini dengan mengadaptasinya ke perangkat pilihan Anda atau oleh [membuat perangkat virtual dengan Amazon EC2](#).

Ikhtisar skenario tutorial

Skenario untuk tutorial ini adalah aplikasi atau layanan lokal yang mengubah warna bola lampu dan yang menerbitkan datanya ke topik bayangan yang dipesan. Tutorial ini mirip dengan fungsi Device Shadow yang dijelaskan dalam [tutorial memulai interaktif](#) dan diimplementasikan pada perangkat Raspberry Pi. Tutorial di bagian ini berfokus pada bayangan klasik tunggal sambil menunjukkan bagaimana Anda dapat mengakomodasi bayangan bernama atau beberapa perangkat.

Tutorial berikut akan membantu Anda mempelajari cara menggunakan AWS IoT Layanan Device Shadow.

- [Tutorial: Mempersiapkan Raspberry Pi Anda untuk menjalankan aplikasi bayangan](#)

Tutorial ini menunjukkan cara mengatur perangkat Raspberry Pi untuk menghubungkan dengan AWS IoT. Anda juga akan membuat AWS IoT dokumen kebijakan dan sumber daya hal, men-download sertifikat, dan kemudian melampirkan kebijakan untuk sumber daya hal itu. Tutorial ini memakan waktu sekitar 30 menit untuk menyelesaikannya.

- [Tutorial: Menginstal Device SDK dan menjalankan aplikasi sampel untuk Device Shadows](#)

Tutorial ini menunjukkan cara menginstal alat yang diperlukan, perangkat lunak, dan AWS IoT Perangkat SDK untuk Python, dan kemudian jalankan aplikasi bayangan sampel. Tutorial ini dibangun berdasarkan konsep-konsep yang disajikan dalam [Connect Raspberry Pi atau perangkat lain](#) dan membutuhkan waktu 20 menit untuk menyelesaikannya.

- [Tutorial: Berinteraksi dengan Device Shadow menggunakan aplikasi sampel dan klien uji MQTT](#)

Tutorial ini menunjukkan bagaimana Anda menggunakan `shadow.py` aplikasi sampel dan AWS IoT konsol untuk mengamati interaksi antara AWS IoT Device Shadows dan perubahan keadaan bola lampu. Tutorial ini juga menunjukkan cara mengirim pesan MQTT ke topik yang dipesan Device Shadow. Tutorial ini dapat memakan waktu 45 menit untuk menyelesaikannya.

AWS IoT Ringkasan Device Shadow

Device Shadow adalah representasi virtual persisten dari perangkat yang dikelola oleh [sumber daya hal](#). Anda membuat AWS IoT Registry. Dokumen Shadow adalah JSON atau JavaScript notasi doc yang digunakan untuk menyimpan dan mengambil informasi status saat ini untuk perangkat. Anda dapat menggunakan bayangan untuk mendapatkan dan mengatur status perangkat melalui topik MQTT atau HTTP REST API, terlepas dari apakah perangkat terhubung ke internet.

Sebuah dokumen Shadow berisikan properti yang menggambarkan aspek-aspek keadaan perangkat.

- **desired:** Aplikasi menentukan status properti perangkat yang diinginkan dengan memperbarui `desired` objek.
- **reported:** Perangkat melaporkan keadaan mereka saat ini di `reported` objek.
- **delta:** AWS IoT melaporkan perbedaan antara yang diinginkan dan keadaan yang dilaporkan di `delta` objek.

Berikut adalah contoh dokumen status Shadow.

```
{
  "state": {
    "desired": {
      "color": "green"
    },
    "reported": {
      "color": "blue"
    },
    "delta": {
      "color": "green"
    }
  }
}
```

Untuk memperbarui dokumen Shadow perangkat, Anda dapat menggunakan [topik MQTT yang dipesan](#), yang [API SIST Perangkat](#) yang mendukung GET, UPDATE, dan DELETE operasi dengan HTTP, dan [AWS IoT CLI](#).

Pada contoh sebelumnya, katakanlah Anda ingin mengubah `desired` warna `yellow`. Untuk melakukan ini, kirim permintaan ke [UpdateThingShadow](#) API atau mempublikasikan pesan ke [Perbarui](#) topik, `$aws/things/THING_NAME/shadow/update`.

```
{
  "state": {
    "desired": {
      "color": yellow
    }
  }
}
```

Pembaruan hanya mempengaruhi bidang yang ditentukan dalam permintaan. Setelah berhasil memperbarui Device Shadow, AWS IoT menerbitkan yang baru `desired` negara ke `delta` topik, `$aws/things/THING_NAME/shadow/delta`. Dokumen Shadow dalam kasus ini terlihat seperti ini:

```
{
  "state": {
    "desired": {
      "color": yellow
    },
    "reported": {
      "color": green
    },
    "delta": {
      "color": yellow
    }
  }
}
```

Negara baru kemudian dilaporkan ke AWS IoT Device Shadow menggunakan `update` tema `$aws/things/THING_NAME/shadow/update` dengan pesan JSON berikut:

```
{
  "state": {
    "reported": {
      "color": yellow
    }
  }
}
```

Jika Anda ingin mendapatkan informasi status saat ini, kirim permintaan ke [GetThingShadow](#) API atau mempublikasikan pesan MQTT ke [Dapatkan](#) topik, `$aws/things/THING_NAME/shadow/get`.

Untuk informasi selengkapnya tentang cara menggunakan layanan Device Shadow, lihat [AWS IoT Layanan Device Shadow](#).

Untuk informasi selengkapnya tentang menggunakan Device Shadows di perangkat, aplikasi, dan layanan, lihat [Menggunakan bayangan di perangkat](#) dan [Menggunakan bayangan di aplikasi dan layanan](#).

Untuk informasi tentang berinteraksi dengan AWS IoT bayangan, lihat [Berinteraksi dengan bayangan](#).

Untuk informasi tentang topik yang dipesan MQTT dan HTTP REST API, lihat [Topik Device Shadow MQTT](#) dan [Device Shadow REST API](#).

Tutorial: Mempersiapkan Raspberry Pi Anda untuk menjalankan aplikasi bayangan

Tutorial ini menunjukkan cara mengatur dan mengkonfigurasi perangkat Raspberry Pi dan membuat AWS IoT sumber daya yang diperlukan perangkat untuk menghubungkan dan bertukar pesan MQTT.

Note

Jika Anda berencana [the section called “Buat perangkat virtual dengan Amazon EC2”](#), Anda dapat melewati halaman ini dan melanjutkan [the section called “Konfigurasi perangkat Anda”](#). Anda akan membuat sumber daya ini ketika Anda membuat hal virtual Anda. Jika Anda ingin menggunakan perangkat yang berbeda daripada Raspberry Pi, Anda dapat mencoba untuk mengikuti tutorial ini dengan mengadaptasinya ke perangkat pilihan Anda.

Dalam tutorial ini, Anda akan belajar cara:

- Siapkan perangkat Raspberry Pi dan konfigurasi untuk digunakan dengan AWS IoT.
- Membuat AWS IoT dokumen kebijakan, yang mengotorisasi perangkat Anda untuk berinteraksi AWS IoT layanan.
- Buat sumber daya hal di AWS IoT sertifikat perangkat X.509, dan kemudian melampirkan dokumen kebijakan.

Masalahnya adalah representasi virtual perangkat Anda di AWS IoT registri. Sertifikat mengautentikasi perangkat Anda AWS IoT inti, dan dokumen kebijakan mengizinkan perangkat Anda untuk berinteraksi AWS IoT.

Cara menjalankan tutorial ini

Untuk menjalankanshadow .pycontoh aplikasi untuk Device Shadows, Anda akan memerlukan perangkat Raspberry Pi yang terhubung keAWS IoT. Kami menyarankan Anda mengikuti tutorial ini dalam urutan yang disajikan di sini, dimulai dengan menyiapkan Raspberry Pi dan itu aksesoris, dan kemudian membuat kebijakan dan melampirkan kebijakan untuk sumber daya hal yang Anda buat. Anda kemudian dapat mengikuti tutorial ini dengan menggunakan antarmuka pengguna grafis (GUI) didukung oleh Raspberry Pi untuk membukaAWS IoTkonsol pada browser web perangkat, yang juga membuatnya lebih mudah untuk men-download sertifikat langsung ke Raspberry Pi Anda untuk menghubungkan keAWS IoT.

Sebelum Anda memulai tutorial ini, pastikan bahwa Anda memiliki:

- Sesi Akun AWS. Jika Anda belum memilikinya, selesaikan langkah-langkah yang dijelaskan di[Siapkan Anda Akun AWS](#)sebelum Anda melanjutkan. Anda akan membutuhkanAkun AWSdanAWS IoTkonsol untuk menyelesaikan tutorial ini.
- Raspberry Pi dan aksesoris yang diperlukan. Anda akan membutuhkan:
 - SEBUAH[Raspberry Pi 3 Model B](#)atau model yang lebih baru. Tutorial ini mungkin bekerja pada versi sebelumnya dari Raspberry Pi, tetapi kami belum mengujinya.
 - [OS Pi Raspberry \(32-bit\)](#)atau nanti. Sebaiknya gunakan versi terbaru dari OS Raspberry. Versi OS sebelumnya mungkin bekerja, tetapi kami belum mengujinya.
 - Koneksi Ethernet atau Wi-Fi.
 - Keyboard, mouse, monitor, kabel, dan catu daya.

Tutorial ini memakan waktu sekitar 30 menit untuk menyelesaikannya.

Langkah 1: Mengatur dan mengkonfigurasi perangkat Raspberry Pi

Pada bagian ini, kita akan mengkonfigurasi perangkat Raspberry Pi untuk digunakan denganAWS IoT.

Important

Mengadaptasi instruksi ini ke perangkat lain dan sistem operasi dapat menjadi tantangan. Anda harus memahami perangkat Anda dengan cukup baik untuk dapat menafsirkan instruksi ini dan menerapkannya ke perangkat Anda. Jika Anda mengalami kesulitan, Anda mungkin mencoba salah satu opsi perangkat lain sebagai alternatif, seperti[Buat perangkat](#)

[virtual dengan Amazon EC2](#) atau [Gunakan Windows atau Linux PC atau Mac sebagai AWS IoT perangkat](#).

Anda harus mengkonfigurasi Raspberry Pi Anda sehingga dapat memulai sistem operasi (OS), terhubung ke internet, dan memungkinkan Anda untuk berinteraksi dengan itu pada antarmuka baris perintah. Anda juga dapat menggunakan antarmuka pengguna grafis (GUI) didukung dengan Raspberry Pi untuk membuka AWS IoT konsol dan jalankan tutorial ini.

Untuk mengatur Raspberry Pi

1. Masukkan kartu SD ke slot kartu microSD pada Raspberry Pi. Beberapa kartu SD datang pra-dimuat dengan manajer instalasi yang meminta Anda dengan menu untuk menginstal OS setelah booting up papan. Anda juga dapat menggunakan imager Raspberry Pi untuk menginstal OS pada kartu Anda.
2. Connect TV HDMI atau monitor ke kabel HDMI yang terhubung ke port HDMI dari Raspberry Pi.
3. Connect keyboard dan mouse ke port USB dari Raspberry Pi dan kemudian pasang adaptor daya untuk boot papan.

Setelah Raspberry Pi boot up, jika kartu SD datang pra-dimuat dengan manajer instalasi, menu muncul untuk menginstal sistem operasi. Jika Anda mengalami masalah saat menginstal OS, Anda dapat mencoba langkah-langkah berikut. Untuk informasi selengkapnya tentang pengaturan Raspberry Pi, lihat [Menyiapkan Raspberry Pi Anda](#).

Jika Anda mengalami masalah saat menyiapkan Raspberry Pi:

- Periksa apakah Anda memasukkan kartu SD sebelum boot papan. Jika Anda memasang kartu SD setelah booting papan, menu instalasi mungkin tidak muncul.
- Pastikan TV atau monitor dinyalakan dan input yang benar dipilih.
- Pastikan bahwa Anda menggunakan perangkat lunak yang kompatibel dengan Raspberry Pi.

Setelah Anda menginstal dan mengkonfigurasi OS Raspberry Pi, buka browser web Raspberry Pi dan arahkan ke AWS IoT Core konsol untuk melanjutkan langkah-langkah yang ada di tutorial ini.

Jika Anda dapat membuka AWS IoT Core konsol, Anda Raspberry Pi sudah siap dan Anda dapat terus [the section called “Menyediakan perangkat Anda AWS IoT”](#).

Jika Anda mengalami masalah atau memerlukan bantuan tambahan, lihat [Mendapatkan bantuan untuk Raspberry Pi Anda](#).

Tutorial: Menyediakan perangkat Anda AWS IoT

Bagian ini menciptakan AWS IoT Core sumber daya yang akan digunakan oleh tutorial Anda.

Langkah-langkah untuk menyediakan perangkat Anda AWS IoT

- [Langkah 1: Membuat AWS IoT kebijakan untuk Device Shadow](#)
- [Langkah 2: Membuat sumber daya hal dan melampirkan kebijakan untuk hal](#)
- [Langkah 3: Tinjau hasil dan langkah selanjutnya](#)

Langkah 1: Membuat AWS IoT kebijakan untuk Device Shadow

Sertifikat X.509 mengautentikasi perangkat Anda AWS IoT Core. AWS IoT kebijakan dilampirkan ke sertifikat yang memungkinkan perangkat untuk melakukan AWS IoT operasi, seperti berlangganan atau menerbitkan topik cadangan MQTT yang digunakan oleh layanan Device Shadow. Perangkat Anda menampilkan sertifikat saat menghubungkan dan mengirim pesan ke AWS IoT Core.

Dalam prosedur ini, Anda akan membuat kebijakan yang memungkinkan perangkat Anda melakukan AWS IoT operasi yang diperlukan untuk menjalankan program contoh. Sebaiknya buat kebijakan yang hanya memberikan izin yang diperlukan untuk melakukan tugas. Anda membuat AWS IoT kebijakan terlebih dahulu, lalu lampirkan ke sertifikat perangkat yang akan Anda buat nanti.

Membuat AWS IoT kebijakan

1. Di menu sebelah kiri, pilih **Aman**, dan kemudian pilih **Kebijakan**. Jika akun Anda memiliki kebijakan yang ada, pilih **Buat**, jika tidak, pada **Anda belum memiliki kebijakan halaman**, pilih **Membuat kebijakan**.
2. Pada **Membuat kebijakan laman**:
 - a. Masukkan nama untuk kebijakan di **Nama bidang** (misalnya, **My_Device_Shadow_policy**). Jangan gunakan informasi identitas pribadi dalam nama kebijakan Anda.
 - b. Dalam dokumen kebijakan, Anda menjelaskan menghubungkan, berlangganan, menerima, dan mempublikasikan tindakan yang memberikan izin perangkat untuk mempublikasikan dan berlangganan topik cadangan MQTT.

Salin contoh kebijakan berikut dan tempelkan di dokumen kebijakan Anda. Gantithingnamedengan nama benda yang akan Anda buat (misalnya,My_light_bulb),regiondenganAWS IoTWilayah di mana Anda menggunakan layanan, danaccountdenganAkun AWSnomor. Untuk informasi lebih lanjut tentangAWS IoTkebijakan, lihat[AWS IoT Core kebijakan](#).

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish"
      ],
      "Resource": [
        "arn:aws:iot:region:account:topic/$aws/things/thingname/shadow/get",
        "arn:aws:iot:region:account:topic/$aws/things/thingname/shadow/update"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:region:account:topic/$aws/things/thingname/shadow/get/
accepted",
        "arn:aws:iot:region:account:topic/$aws/things/thingname/shadow/get/
rejected",
        "arn:aws:iot:region:account:topic/$aws/things/thingname/shadow/update/
accepted",
        "arn:aws:iot:region:account:topic/$aws/things/thingname/shadow/update/
rejected",
        "arn:aws:iot:region:account:topic/$aws/things/thingname/shadow/update/
delta"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Subscribe"
      ],
    },
  ]
}
```

```

    "Resource": [
      "arn:aws:iot:region:account:topicfilter/$aws/things/thingname/shadow/get/accepted",
      "arn:aws:iot:region:account:topicfilter/$aws/things/thingname/shadow/get/rejected",
      "arn:aws:iot:region:account:topicfilter/$aws/things/thingname/shadow/update/accepted",
      "arn:aws:iot:region:account:topicfilter/$aws/things/thingname/shadow/update/rejected",
      "arn:aws:iot:region:account:topicfilter/$aws/things/thingname/shadow/update/delta"
    ],
    {
      "Effect": "Allow",
      "Action": "iot:Connect",
      "Resource": "arn:aws:iot:region:account:client/test-*"
    }
  ]
}

```

Langkah 2: Membuat sumber daya hal dan melampirkan kebijakan untuk hal

Perangkat yang terhubung ke AWS IoT dapat diwakili oleh sumber daya hal di dalam AWS IoT Registry. SEBUAH sumber daya hal mewakili perangkat tertentu atau entitas logis, seperti bola lampu dalam tutorial ini.

Untuk mempelajari cara membuat sesuatu di AWS IoT, ikuti langkah-langkah yang dijelaskan dalam [Buat objek benda](#). Berikut adalah beberapa hal penting saat Anda mengikuti langkah-langkah dalam tutorial tersebut:

1. Pilih Buat satu hal, dan di Namabidang, masukkan nama untuk hal yang sama dengan `thingname` (misalnya, `My_light_bulb`) Anda menentukan ketika Anda membuat kebijakan sebelumnya.

Anda tidak dapat mengubah nama setelah dibuat. Jika Anda memberikan nama yang berbeda selain `thingname`, membuat hal baru dengan nama sebagai `thingname` dan menghapus hal lama.

Note

Jangan gunakan informasi yang dapat diidentifikasi secara pribadi dalam nama barang Anda. Nama benda dapat muncul dalam komunikasi dan laporan yang tidak terenkripsi.

2. Kami menyarankan Anda mengunduh masing-masing file sertifikat di halaman ke lokasi di mana Anda dapat dengan mudah menemukan mereka. Anda harus menginstal file-file ini untuk menjalankan aplikasi sampel.

Sebaiknya unduh file ke dalam `certs` subdirektori di home direktori pada Raspberry Pi dan nama masing-masing dengan nama sederhana seperti yang disarankan dalam tabel berikut.

Nama berkas sertifikat

File	Jalur file
Sertifikat CA Akar	<code>~/certs/Amazon-root-CA-1.pem</code>
Sertifikat perangkat lunak	<code>~/certs/device.pem.crt</code>
Kunci privat	<code>~/certs/private.pem.key</code>

3. Setelah Anda mengaktifkan sertifikat untuk mengaktifkan koneksi ke AWS IoT, pilih kebijakan pastikan Anda melampirkan kebijakan yang Anda buat sebelumnya (misalnya, `My_Device_Shadow_policy`) untuk hal itu.

Setelah Anda membuat sesuatu, Anda dapat melihat sumber daya Anda ditampilkan dalam daftar hal-hal di AWS IoT konsol.

Langkah 3: Tinjau hasil dan langkah selanjutnya

Dalam tutorial ini, Anda belajar bagaimana:

- Mengatur dan mengkonfigurasi perangkat Raspberry Pi.
- Membuat AWS IoT dokumen kebijakan yang mengotorisasi perangkat Anda untuk berinteraksi AWS IoT layanan.
- Membuat sumber daya hal dan sertifikat perangkat X.509 terkait, dan melampirkan dokumen kebijakan untuk itu.

Langkah selanjutnya

Sekarang Anda dapat menginstal AWS IoT perangkat SDK untuk Python, jalankan `shadow.py` contoh aplikasi, dan menggunakan Device Shadows untuk mengontrol keadaan. Untuk informasi selengkapnya tentang cara menjalankan tutorial ini, lihat [Tutorial: Menginstal Device SDK dan menjalankan aplikasi sampel untuk Device Shadows](#).

Tutorial: Menginstal Device SDK dan menjalankan aplikasi sampel untuk Device Shadows

Bagian ini menunjukkan bagaimana Anda dapat menginstal perangkat lunak yang diperlukan dan AWS IoT Perangkat SDK untuk Python dan jalankan `shadow.py` contoh aplikasi untuk mengedit dokumen Shadow dan mengontrol status bayangan.

Dalam tutorial ini, Anda akan belajar cara:

- Gunakan perangkat lunak yang diinstal dan AWS IoT Perangkat SDK untuk Python untuk menjalankan aplikasi sampel.
- Pelajari cara memasukkan nilai menggunakan aplikasi sampel menerbitkan nilai yang diinginkan di AWS IoT konsol.
- Tinjau `shadow.py` contoh aplikasi dan bagaimana menggunakan protokol MQTT untuk memperbarui status bayangan.

Sebelum Anda menjalankan tutorial ini:

Anda harus telah mengatur Akun AWS, mengkonfigurasi perangkat Raspberry Pi Anda, dan membuat AWS IoT Role dan kebijakan yang memberikan izin perangkat untuk mempublikasikan dan berlangganan topik yang dipesan MQTT dari layanan Device Shadow. Untuk informasi selengkapnya, lihat [Tutorial: Mempersiapkan Raspberry Pi Anda untuk menjalankan aplikasi bayangan](#).

Anda juga harus menginstal Git, Python, dan AWS IoT SDK for Python. Tutorial ini dibangun di atas konsep-konsep yang disajikan dalam tutorial [Connect Raspberry Pi atau perangkat lain](#). Jika Anda belum mencoba tutorial itu, kami sarankan Anda mengikuti langkah-langkah yang dijelaskan dalam tutorial itu untuk menginstal file sertifikat dan Device SDK dan kemudian kembali ke tutorial ini untuk menjalankan `shadow.py` aplikasi sampel.

Dalam tutorial ini, Anda akan:

- [Langkah 1: Jalankan aplikasi contoh `shadow.py`](#)

- [Langkah 2: Tinjau aplikasi sampel shadow.py Device SDK](#)
- [Langkah 3: Memecahkan masalah denganshadow.pyaplikasi sampel](#)
- [Langkah 4: Tinjau hasil dan langkah selanjutnya](#)

Tutorial ini memakan waktu sekitar 20 menit untuk menyelesaikannya.

Langkah 1: Jalankan aplikasi contoh shadow.py

Sebelum Anda menjalankanshadow . pycontoh aplikasi, Anda akan memerlukan informasi berikut selain nama dan lokasi file sertifikat yang Anda instal.

Nilai parameter aplikasi

Parameter	Dimana menemukan nilainya
<i>your-iot-thing-nama</i>	<p>NamaAWS IoT yang Anda buat sebelumnya at the section called “Langkah 2: Membuat sumber daya hal dan melampirkan kebijakan untuk hal”.</p> <p>Untuk menemukan nilai ini, di AWS IoT Konsol, pilih Kelola, dan kemudian pilih Things.</p>
<i>your-iot-endpoint</i>	<p>Parameter <i>your-iot-endpoint</i> memiliki format: <i>endpoint_id</i> -ats.iot. <i>region</i>.amazonaws.com , misalnya, a3qj468EXAMPLE-ats.iot.us-west-2.amazonaws.com . Untuk menemukan nilai ini:</p> <ol style="list-style-type: none"> 1. Di AWS IoT Konsol, pilih Kelola, dan kemudian pilih Things. 2. Pilih hal IoT yang Anda buat untuk perangkat Anda, MY_LIGHT_Bulb, yang Anda gunakan sebelumnya, dan kemudian pilih Berinteraksi. Pada halaman detail objek, titik akhir Anda ditampilkan di halaman HTTPS bagian.

Instal dan jalankan aplikasi sampel

1. Buka direktori aplikasi sampel.

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

2. Di jendela baris perintah, ganti *your-iot-endpoint* dan *your-iot-thing-name* seperti yang ditunjukkan dan jalankan perintah ini.

```
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. Amati bahwa aplikasi sampel:

1. Terhubung ke AWS Layanan IoT untuk akun Anda.
2. Berlangganan `Delta` peristiwa dan `Update` dan `Get` tanggapan.
3. Meminta Anda untuk memasukkan nilai yang diinginkan di terminal.
4. Menampilkan output yang serupa dengan yang berikut ini:

```
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

Jika Anda mengalami kesulitan menjalankan `shadow.py` contoh aplikasi, review [the section called “Langkah 3: Memecahkan masalah dengan `shadow.py` aplikasi sampel”](#). Untuk mendapatkan informasi tambahan yang dapat membantu Anda memperbaiki masalah,

tambahkan `--verbosity debug` parameter ke baris perintah sehingga aplikasi sampel menampilkan pesan rinci tentang apa yang dilakukannya.

Masukkan nilai dan amati pembaruan dalam dokumen Shadow

Anda dapat memasukkan nilai di terminal untuk menentukan `desired` nilai, yang juga memperbarui `reported` nilai. Katakanlah Anda memasukkan warna `yellow` di terminal. Parameter `reported` nilai juga diperbarui untuk warna `yellow`. Berikut ini menunjukkan pesan yang ditampilkan di 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'.
```

Ketika Anda mempublikasikan permintaan pemutakhiran ini, AWS IoT menciptakan default, bayangan klasik untuk sumber daya hal. Anda dapat mengamati permintaan pembaruan yang Anda publikasikan ke `reported` dan `desired` nilai-nilai dalam AWS IoT konsol dengan melihat dokumen Shadow untuk sumber daya hal yang Anda buat (misalnya, `My_light_bulb`). Untuk melihat pembaruan dalam dokumen Shadow:

1. Di AWS IoT konsol, pilih **Kelola** dan kemudian pilih **Things**.
2. Dalam daftar hal-hal yang ditampilkan, pilih hal yang Anda buat, pilih **Bayangan**, dan kemudian pilih **Bayangan klasik**.

Dokumen Shadow akan serupa dengan yang berikut ini, menunjukkan `reported` dan `desired` nilai diatur ke warna `yellow`. Anda melihat nilai-nilai ini di `Status` bayangan bagian dari dokumen.

```
{
  "desired": {
    "welcome": "aws-iot",
    "color": "yellow"
  },
  "reported": {
    "welcome": "aws-iot",
    "color": "yellow"
  }
}
```



```
}  
}
```

Anda juga melihat `Metadata` bagian yang berisi informasi timestamp dan nomor versi permintaan.

Anda dapat menggunakan versi dokumen negara untuk memastikan Anda memperbarui versi terbaru dari dokumen Shadow perangkat. Jika Anda mengirim permintaan pembaruan lain, nomor versi bertambah 1. Ketika Anda menyediakan versi dengan permintaan pembaruan, layanan menolak permintaan dengan kode respons konflik HTTP 409 jika versi dokumen status saat ini tidak cocok dengan versi yang disediakan.

```
{  
  "metadata": {  
    "desired": {  
      "welcome": {  
        "timestamp": 1620156892  
      },  
      "color": {  
        "timestamp": 1620156893  
      }  
    },  
    "reported": {  
      "welcome": {  
        "timestamp": 1620156892  
      },  
      "color": {  
        "timestamp": 1620156893  
      }  
    }  
  },  
  "version": 10  
}
```

Untuk mempelajari lebih lanjut tentang dokumen Shadow dan mengamati perubahan pada informasi negara, lanjutkan ke tutorial berikutnya [Tutorial: Berinteraksi dengan Device Shadow menggunakan aplikasi sampel dan klien uji MQTT](#) seperti yang dijelaskan dalam [Langkah 4: Tinjau hasil dan langkah selanjutnya](#) bagian dari tutorial ini. Opsional, Anda juga dapat mempelajari tentang `shadow.py` kode sampel dan bagaimana menggunakan protokol MQTT di bagian berikut.

Langkah 2: Tinjau aplikasi sampel shadow.py Device SDK

Bagian ini mengulas `shadow.py` contoh aplikasi dari AWS IoT SDK v2 for Python digunakan dalam tutorial ini. Di sini, kami akan meninjau bagaimana menghubungkannya dengan AWS IoT Core dengan menggunakan MQTT dan MQTT lebih protokol WSS. Parameter [AWS waktu aktif umum \(AWS-CRT\)](#) perpustakaan menyediakan dukungan protokol komunikasi tingkat rendah dan disertakan dengan AWS IoT SDK v2 for Python.

Sementara tutorial ini menggunakan MQTT dan MQTT lebih WSS, AWS IoT mendukung perangkat yang mempublikasikan permintaan HTTPS. Untuk contoh program Python yang mengirimkan pesan HTTP dari perangkat, lihat [Contoh kode HTTPS](#) menggunakan `Python requests` perpustakaan.

Untuk informasi tentang bagaimana Anda dapat membuat keputusan berdasarkan informasi tentang protokol mana yang akan digunakan untuk komunikasi perangkat Anda, tinjau [Memilih protokol untuk komunikasi perangkat Anda](#).

MQTT

Parameter `shadow.py` panggilan sampel `mtls_from_path` (ditampilkan di sini) [`mqtt_connection_builder`](#) untuk membangun koneksi dengan AWS IoT Core dengan menggunakan protokol MQTT. `mtls_from_path` menggunakan sertifikat X.509 dan TLS v1.2 untuk mengotentikasi perangkat. Parameter AWS perpustakaan `-CRT` menangani rincian tingkat yang lebih rendah dari koneksi itu.

```
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` adalah AWS IoT endpoint yang Anda lewatkan dari baris perintah dan `client_id` adalah ID yang mengidentifikasi perangkat ini secara unik di Wilayah AWS.
- `cert_filepath`, `pri_key_filepath`, dan `ca_filepath` adalah jalur ke sertifikat perangkat dan file kunci pribadi, dan file root CA.

- `client_bootstrap` adalah objek runtime umum yang menangani kegiatan komunikasi soket, dan dipakai sebelum panggilan `mqtt_connection_builder.mtls_from_path`.
- `on_connection_interrupted` dan `on_connection_resumed` adalah fungsi callback untuk memanggil ketika koneksi perangkat terganggu dan dilanjutkan.
- `clean_session` adalah apakah akan memulai sesi baru, persisten, atau jika ada, sambungkan kembali ke sesi yang sudah ada. `keep_alive_secs` adalah nilai tetap hidup, dalam hitungan detik, untuk mengirim CONNECT permintaan. Ping akan secara otomatis dikirim pada interval ini. Server mengasumsikan bahwa koneksi hilang jika tidak menerima ping setelah 1,5 kali nilai ini.

Parameter `shadow.pysampel` juga panggilan `websockets_with_default_aws_signing` di dalam [mqtt_connection_builder](#) untuk membangun koneksi dengan AWS IoT Core menggunakan protokol MQTT lebih WSS. MQTT lebih WSS juga menggunakan parameter yang sama seperti MQTT dan mengambil parameter tambahan:

- `region` adalah AWS penandatanganan Wilayah digunakan oleh otentikasi Signature V4, `credentials_provider` adalah AWS kredensi yang disediakan untuk digunakan untuk otentikasi. Wilayah dilewatkan dari baris perintah, dan `credentials_provider` objek dipakai sesaat sebelum panggilan `mqtt_connection_builder.websockets_with_default_aws_signing`.
- `websocket_proxy_options` adalah opsi proxy HTTP, jika menggunakan host proxy. Di `shadow.py` contoh aplikasi, nilai ini dipakai sesaat sebelum panggilan `mqtt_connection_builder.websockets_with_default_aws_signing`.

Berlangganan topik dan acara Shadow

Parameter `shadow.pysampel` mencoba untuk membangun koneksi dan menunggu untuk sepenuhnya terhubung. Jika tidak terhubung, perintah akan antri. Setelah terhubung, sampel berlangganan peristiwa delta dan memperbarui dan mendapatkan pesan, dan menerbitkan pesan dengan kualitas layanan (QoS) tingkat 1 (`mqtt.QoS.AT_LEAST_ONCE`).

Ketika perangkat berlangganan pesan dengan QoS level 1, broker pesan menyimpan pesan bahwa perangkat berlangganan sampai mereka dapat dikirim ke perangkat. Broker pesan mengirim ulang pesan sampai menerima PUBLISH respon dari perangkat.

Untuk informasi selengkapnya tentang protokol MQTT, lihat [Tinjau protokol MQTT](#) dan [MQTT](#).

Untuk informasi lebih lanjut tentang bagaimana MQTT, MQTT lebih WSS, sesi persisten, dan tingkat QoS yang digunakan dalam tutorial ini, lihat [Meninjau aplikasi sampel SDK Perangkat pubsub.py](#).

Langkah 3: Memecahkan masalah dengan `shadow.py` aplikasi sampel

Ketika Anda menjalankanshadow.py contoh aplikasi, Anda akan melihat beberapa pesan yang ditampilkan di terminal dan prompt untuk memasukkandesired nilai. Jika program melempar kesalahan, maka untuk men-debug kesalahan, Anda dapat mulai dengan memeriksa apakah Anda menjalankan perintah yang benar untuk sistem Anda.

Dalam beberapa kasus, pesan kesalahan mungkin menunjukkan masalah koneksi dan terlihat mirip dengan: `Host name was invalid for dns resolution` atau `Connection was closed unexpectedly`. Dalam kasus tersebut, berikut adalah beberapa hal yang dapat Anda periksa:

- Periksa alamat endpoint dalam perintah

Tinjau endpoint argumen dalam perintah yang Anda masukkan untuk menjalankan aplikasi sampel, (misalnya, `a3qEXAMPLEffp-ats.iot.us-west-2.amazonaws.com`) dan periksa nilai ini di AWS IoT konsol.

Untuk memeriksa apakah Anda menggunakan nilai yang benar:

1. Di AWS IoT konsol, pilih **Kelola** dan kemudian pilih **Things**.
2. Pilih hal yang Anda buat untuk aplikasi sampel Anda (misalnya, `MY_LIGHT_Bulb`) dan kemudian pilih **Interaksi**.

Pada halaman detail objek, titik akhir Anda ditampilkan di halaman **HTTPS** bagian. Anda juga harus melihat pesan yang mengatakan: `This thing already appears to be connected`.

- Periksa aktivasi sertifikat

Sertifikat mengautentikasi perangkat Anda AWS IoT Core.

Untuk memeriksa apakah sertifikat Anda aktif:

1. Di AWS IoT konsol, pilih **Kelola** dan kemudian pilih **Things**.
2. Pilih hal yang Anda buat untuk aplikasi sampel Anda (misalnya, `MY_LIGHT_Bulb`) dan kemudian pilih **Keamanan**.
3. Pilih sertifikat dan kemudian, dari halaman rincian sertifikat, pilih **Pilih sertifikat** dan kemudian, dari halaman rincian sertifikat, pilih **Tindakan**.

Jika dalam daftar dropdown **Aktif** tidak tersedia dan Anda hanya dapat memilih **Nonaktifkan**, sertifikat Anda aktif. Jika tidak, pilih **Aktif** dan jalankan kembali program sampel.

Jika program masih tidak berjalan, periksa nama berkas sertifikat di `certs` folder.

- Periksa kebijakan yang dilampirkan pada sumber daya hal

Sementara sertifikat mengautentikasi perangkat Anda, AWS IoT kebijakan mengizinkan perangkat untuk melakukan AWS IoT operasi, seperti berlangganan atau menerbitkan topik cadangan MQTT.

Untuk memeriksa apakah kebijakan yang benar terlampir:

1. Temukan sertifikat seperti yang dijelaskan sebelumnya, lalu pilih **Kebijakan**.
2. Pilih kebijakan yang ditampilkan dan periksa apakah kebijakan tersebut menjelaskan `connect`, `subscribe`, `receive`, dan `publish` tindakan yang memberikan izin perangkat untuk mempublikasikan dan berlangganan topik yang dipesan MQTT.

Untuk kebijakan contoh, lihat [Langkah 1: Membuat AWS IoT kebijakan untuk Device Shadow](#).

Jika Anda melihat pesan kesalahan yang menunjukkan masalah saat terhubung ke AWS IoT, bisa jadi karena izin yang Anda gunakan untuk kebijakan. Jika demikian, kami menyarankan Anda untuk memulai dengan kebijakan yang menyediakan akses penuh ke AWS IoT sumber daya dan kemudian menjalankan kembali program sampel. Anda dapat mengedit kebijakan saat ini, atau memilih kebijakan saat ini, pilih **Lepaskan**, dan kemudian membuat kebijakan lain yang menyediakan akses penuh dan melampirkannya ke sumber daya hal Anda. Anda kemudian dapat membatasi kebijakan untuk hanya tindakan dan kebijakan yang Anda butuhkan untuk menjalankan program.

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

- Periksa instalasi SDK Perangkat

Jika program masih belum berjalan, Anda dapat menginstal ulang Device SDK untuk memastikan bahwa instalasi SDK Anda selesai dan benar.

Langkah 4: Tinjau hasil dan langkah selanjutnya

Dalam tutorial ini, Anda belajar bagaimana:

- Menginstal perangkat lunak, alat, dan perangkat lunak yang diperlukan AWS IoT SDK for Python.
- Memahami bagaimana aplikasi sampel, `shadow.py`, menggunakan protokol MQTT untuk mengambil dan memperbarui status bayangan saat ini.
- Jalankan contoh aplikasi untuk Device Shadows dan amati pembaruan ke dokumen Shadow di AWS IoT Konsol. Anda juga belajar memecahkan masalah dan memperbaiki kesalahan saat menjalankan program.

Langkah selanjutnya

Sekarang Anda dapat menjalankannya `shadow.py` contoh aplikasi dan menggunakan Device Shadows untuk mengontrol negara. Anda dapat mengamati pembaruan pada dokumen Shadow di AWS IoT Konsol dan amati peristiwa delta yang merespons aplikasi sampel. Dengan menggunakan klien uji MQTT, Anda dapat berlangganan topik bayangan yang dipesan dan mengamati pesan yang diterima oleh topik saat menjalankan program sampel. Untuk informasi selengkapnya tentang cara menjalankan tutorial ini, lihat [Tutorial: Berinteraksi dengan Device Shadow menggunakan aplikasi sampel dan klien uji MQTT](#).

Tutorial: Berinteraksi dengan Device Shadow menggunakan aplikasi sampel dan klien uji MQTT

Untuk berinteraksi dengan `shadow.py` contoh aplikasi, masukkan nilai di terminal untuk `desired` nilai. Misalnya, Anda dapat menentukan warna yang menyerupai lampu lalu lintas dan AWS IoT menanggapi permintaan dan memperbarui nilai yang dilaporkan.

Dalam tutorial ini, Anda akan belajar cara:

- Menggunakan `shadow.py` contoh aplikasi untuk menentukan status yang diinginkan dan memperbarui status bayangan saat ini.

- Mengedit dokumen Shadow untuk mengamati peristiwa delta dan bagaimana shadow.py aplikasi sampel merespon itu.
- Gunakan klien uji MQTT untuk berlangganan topik bayangan dan mengamati pembaruan saat Anda menjalankan program sampel.

Sebelum Anda menjalankan tutorial ini, Anda harus memiliki:

Menyiapkan Akun AWS, mengkonfigurasi perangkat Raspberry Pi Anda, dan membuat AWS IoT Thing dan kebijakan. Anda juga harus menginstal perangkat lunak yang diperlukan, Device SDK, file sertifikat, dan menjalankan program sampel di terminal. Untuk informasi selengkapnya, lihat tutorial sebelumnya [Tutorial: Mempersiapkan Raspberry Pi Anda untuk menjalankan aplikasi bayangan](#) dan [Langkah 1: Jalankan aplikasi contoh shadow.py](#). Anda harus menyelesaikan tutorial ini jika belum.

Dalam tutorial ini, Anda akan:

- [Langkah 1: Perbarui nilai yang diinginkan dan dilaporkan menggunakan shadow.py aplikasi sampel](#)
- [Langkah 2: Melihat pesan dari shadow.py contoh aplikasi di klien uji MQTT](#)
- [Langkah 3: Memecahkan masalah kesalahan dengan interaksi Device Shadow](#)
- [Langkah 4: Tinjau hasil dan langkah selanjutnya](#)

Tutorial ini memakan waktu sekitar 45 menit untuk menyelesaikannya.

Langkah 1: Perbarui nilai yang diinginkan dan dilaporkan menggunakan **shadow.py** aplikasi sampel

Dalam tutorial sebelumnya [Langkah 1: Jalankan aplikasi contoh shadow.py](#), Anda belajar bagaimana mengamati pesan yang diterbitkan ke dokumen Shadow di AWS IoT konsol saat Anda memasukkan nilai yang diinginkan seperti yang dijelaskan di bagian [Tutorial: Menginstal Device SDK dan menjalankan aplikasi sampel untuk Device Shadows](#).

Pada contoh sebelumnya, kita mengatur warna yang diinginkan `yellow`. Setelah Anda memasukkan setiap nilai, terminal meminta Anda untuk memasukkan yang lain `desired` nilai. Jika Anda lagi memasukkan nilai yang sama (`yellow`), aplikasi mengenali ini dan meminta Anda untuk memasukkan yang baru `desired` nilai.

```
Enter desired value:
yellow
Local value is already 'yellow'.
```

```
Enter desired value:
```

Sekarang, katakan bahwa Anda memasukkan `green`. AWS IoT menanggapi permintaan dan memperbarui `reported` nilai `green`. Ini adalah bagaimana pembaruan terjadi ketika `desired` berbeda dari `reported` negara, menyebabkan delta.

Bagaimana `shadow.py` contoh aplikasi mensimulasikan interaksi Device Shadow:

1. Masukkan `desired` nilai (katakanlah `yellow`) di terminal untuk mempublikasikan keadaan yang diinginkan.
2. Sebagai `desired` berbeda dari `reported` negara (mengatakan `green`), delta terjadi, dan aplikasi yang berlangganan delta menerima pesan ini.
3. Aplikasi merespons pesan dan memperbarui statusnya ke `desired` nilai `yellow`.
4. Aplikasi kemudian menerbitkan pesan pembaruan dengan nilai baru yang dilaporkan dari status perangkat, `yellow`.

Berikut ini menunjukkan pesan yang ditampilkan di terminal yang menunjukkan bagaimana permintaan pembaruan dipublikasikan.

```
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'.
```

Di AWS IoT console, dokumen Shadow mencerminkan nilai diperbarui untuk `green` untuk kedua `reported` dan `desired` bidang, dan nomor versi bertambah 1. Misalnya, jika nomor versi sebelumnya ditampilkan sebagai 10, nomor versi saat ini akan ditampilkan sebagai 11.

Note

Menghapus bayangan tidak mengatur ulang nomor versi ke 0. Anda akan melihat bahwa versi bayangan bertambah 1 ketika Anda mempublikasikan permintaan pembaruan atau membuat bayangan lain dengan nama yang sama.

Mengedit dokumen Shadow untuk mengamati peristiwa delta

Parametershadow.py aplikasi sampel juga berlangganandelta peristiwa, dan merespon ketika ada perubahan padadesirednilai. Misalnya, Anda dapat mengantidesirednilai untuk warnared. Untuk melakukannya, diAWS IoTkonsol, edit dokumen Shadow dengan mengklikMengeditdan kemudian mengatordesirednilaireddi JSON, sambil menjagareportednilaigreen. Sebelum Anda menyimpan perubahan, menjaga terminal pada Raspberry Pi terbuka karena Anda akan melihat pesan yang ditampilkan di terminal ketika perubahan terjadi.

```
{
  "desired": {
    "welcome": "aws-iot",
    "color": "red"
  },
  "reported": {
    "welcome": "aws-iot",
    "color": "green"
  }
}
```

Setelah Anda menyimpan nilai baru,shadow.py aplikasi sampel merespons perubahan ini dan menampilkan pesan di terminal yang menunjukkan delta. Anda kemudian akan melihat pesan berikut muncul di bawah prompt untuk memasukkandesirednilai.

```
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'.
```

Langkah 2: Melihat pesan dari shadow.py contoh aplikasi di klien uji MQTT

Anda dapat menggunakan Klien uji MQTT di dalam AWS IoT konsol untuk memonitor pesan MQTT yang diteruskan dalam Akun AWS. Dengan berlanggan topik MQTT yang dipesan yang digunakan oleh layanan Device Shadow, Anda dapat mengamati pesan yang diterima oleh topik saat menjalankan aplikasi sampel.

Jika Anda belum menggunakan klien uji MQTT, Anda dapat meninjau [Lihat pesan MQTT dengan klien MQTT AWS IoT](#). Ini membantu Anda mempelajari cara menggunakan Klien uji MQTT di dalam AWS IoT konsol untuk melihat pesan MQTT saat mereka melewati broker pesan.

1. Buka klien uji MQTT

Buka [Klien uji MQTT di AWS IoT konsol](#) di jendela baru sehingga Anda dapat mengamati pesan yang diterima oleh topik MQTT tanpa kehilangan konfigurasi klien uji MQTT Anda. Klien uji MQTT tidak menyimpan langganan atau log pesan jika Anda membiarkannya pergi ke halaman lain di konsol. Untuk bagian tutorial ini, Anda dapat memiliki dokumen Shadow dari Anda AWS IoT dan klien uji MQTT terbuka di jendela terpisah untuk lebih mudah mengamati interaksi dengan Device Shadows.

2. Berlangganan topik Shadow yang dipesan MQTT

Anda dapat menggunakan klien uji MQTT untuk memasukkan nama topik yang dipesan MQTT Device Shadow dan berlangganan ke mereka untuk menerima pembaruan saat menjalankan `shadow.py` aplikasi sampel. Untuk berlangganan topik:

- a. Di Klien uji MQTT di dalam AWS IoT konsol, pilih Berlangganan topik.
- b. Di Filter topik bagian, masukkan: `$ aws/hal/thingname/bayangan/update/ #`. Di sini, *thingname* adalah nama sumber daya yang Anda buat sebelumnya (misalnya, `My_light_bulb`).
- c. Menjaga nilai default untuk pengaturan konfigurasi tambahan, lalu pilih Langganan.

Dengan menggunakan `#wildcard` dalam langganan topik, Anda dapat berlangganan beberapa topik MQTT pada saat yang sama dan mengamati semua pesan yang dipertukarkan antara perangkat dan Shadow dalam satu jendela. Untuk informasi selengkapnya tentang karakter wildcard dan penggunaannya, lihat [Topik MQTT](#).

3. Jalankan `shadow.py` program sampel dan mengamati pesan

Di jendela baris perintah Raspberry Pi Anda, jika Anda telah memutuskan program, jalankan aplikasi sampel lagi dan tonton pesan di Klien uji MQTT di dalam AWS IoT konsol.

- a. Jalankan perintah berikut untuk me-restart program sampel. Ganti *your-iot-thing-name* dan *your-iot-endpoint* dengan nama-nama AWS IoT yang Anda buat sebelumnya (misalnya, `My_light_bulb`), dan titik akhir untuk berinteraksi dengan perangkat.

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

Parameters `shadow.py` contoh aplikasi kemudian berjalan dan mengambil status bayangan saat ini. Jika Anda telah menghapus bayangan atau menghapus status saat ini, program menetapkan nilai saat ini ke `off` dan kemudian meminta Anda untuk memasukkan `desired` nilai.

```
Connecting to a3qEXAMPLEffp-atl.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:
```

Di sisi lain, jika program berjalan dan Anda me-restart, Anda akan melihat nilai warna terbaru yang dilaporkan di terminal. Di klien uji MQTT, Anda akan melihat pembaruan ke topik `aws/hal/thingname/bayangan/dapatkan` dan `aws/hal/thingname/bayangan/mendapatkan/diterima`.

Misalkan warna terbaru yang dilaporkan `green`. Berikut ini menunjukkan isi dari `aws/hal/thingname/bayangan/mendapatkan/diterima` File JSON.

```
{
  "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. Masukkan `desired` nilai di terminal, seperti `yellow`. `Parametershadow.py` contoh aplikasi merespon dan menampilkan pesan berikut di terminal yang menunjukkan perubahan dalam `reported` nilai `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'.

```

Di Klien uji MQTT di dalam AWS IoT konsol, di bawah `Langganan`, Anda melihat bahwa topik berikut menerima pesan:

- `$ aws/hal/thingname/bayangan/update`: menunjukkan bahwa kedua `desired` dan `updated` nilai berubah menjadi `yellow`.
- `$ aws/hal/thingname/shadow/update/accepted`: menunjukkan nilai-nilai saat `desired` dan `reported` negara dan metadata dan informasi versi mereka.
- `$ aws/hal/thingname/bayangan/perbaruan/dokumen`: menunjukkan nilai sebelumnya dan saat ini dari `desired` dan `reported` negara dan metadata dan informasi versi mereka.

Sebagai dokumen `$ aws/hal/thingname/bayangan/perbaruan/dokumen` juga berisi informasi yang terkandung dalam dua topik lainnya, kita dapat meninjau untuk melihat informasi negara. Keadaan sebelumnya menunjukkan nilai yang dilaporkan diatur ke `green`, metadata dan informasi versinya, dan keadaan saat ini yang menunjukkan nilai dilaporkan diperbarui untuk `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. Sekarang, jika Anda memasukkan yang lain `desired` nilai, Anda melihat perubahan lebih lanjut ke `reported` nilai dan pembaruan pesan yang diterima oleh topik ini. Nomor versi juga

bertambah sebesar 1. Misalnya, jika Anda memasukkan nilai `green`, keadaan sebelumnya melaporkan nilainya `yellow` dan keadaan saat ini melaporkan nilainya `green`.

4. Edit dokumen Shadow untuk mengamati peristiwa delta

Untuk mengamati perubahan pada topik delta, edit dokumen Shadow di AWS IoT konsol. Misalnya, Anda dapat mengganti nilai untuk warna `red`. Untuk melakukannya, di AWS IoT konsol, pilih **Mengedit** dan kemudian mengatur nilai merah di JSON, sambil menjaga nilai `reported` diatur ke `green`. Sebelum menyimpan perubahan, buka terminal karena Anda akan melihat pesan delta yang dilaporkan di terminal.

```
{
  "desired": {
    "welcome": "aws-iot",
    "color": "red"
  },
  "reported": {
    "welcome": "aws-iot",
    "color": "green"
  }
}
```

Parameter `shadow.py` aplikasi sampel merespons perubahan ini dan menampilkan pesan di terminal yang menunjukkan delta. Dalam klien uji MQTT, update topik akan menerima pesan yang menunjukkan perubahan pada `desired` dan `reported` nilai.

Anda juga melihat bahwa topik `$ aws/hal/thingname/shadow/update/delta` menerima pesan. Untuk melihat pesan, pilih topik ini, yang tercantum di bawah **Langganan**.

```
{
  "version": 13,
  "timestamp": 1617318480,
  "state": {
    "color": "red"
  },
  "metadata": {
    "color": {
      "timestamp": 1617318480
    }
  }
}
```

Langkah 3: Memecahkan masalah kesalahan dengan interaksi Device Shadow

Saat menjalankan aplikasi contoh Shadow, Anda mungkin mengalami masalah dengan mengamati interaksi dengan layanan Device Shadow.

Jika program berjalan dengan sukses dan meminta Anda untuk memasukkan `desired` nilai, Anda harus dapat mengamati interaksi Device Shadow dengan menggunakan dokumen Shadow dan klien uji MQTT seperti yang dijelaskan sebelumnya. Namun, jika Anda tidak dapat melihat interaksinya, berikut adalah beberapa hal yang dapat Anda periksa:

- Periksa nama benda dan bayangannya di AWS IoT Konsol

Jika Anda tidak melihat pesan dalam dokumen Shadow, tinjau perintahnya dan pastikan pesan tersebut sesuai dengan nama benda di AWS IoT Konsol. Anda juga dapat memeriksa apakah Anda memiliki bayangan klasik dengan memilih sumber daya Anda dan kemudian memilih Bayangan. Tutorial ini berfokus terutama pada interaksi dengan bayangan klasik.

Anda juga dapat mengonfirmasi bahwa perangkat yang Anda gunakan terhubung ke internet. Di AWS IoT Konsol, pilih hal yang Anda buat sebelumnya, lalu pilih Interaksi. Pada halaman detail objek, Anda akan melihat pesan di sini yang mengatakan: `This thing already appears to be connected.`

- Periksa topik cadangan MQTT yang Anda langgani

Jika Anda tidak melihat pesan muncul di klien uji MQTT, periksa apakah topik yang Anda langgani diformat dengan benar. Topik MQTT Device Shadow memiliki format `$ aws/hal/thingname/ bayangan/` dan mungkin memiliki `update`, `get`, atau `delete` mengikutinya tergantung pada tindakan yang ingin Anda lakukan pada bayangan. Tutorial ini menggunakan topik `$ aws/hal/thingname/ bayangan/ #` jadi pastikan Anda memasukkannya dengan benar saat berlangganan topik di Filter topik bagian dari klien uji.

Saat Anda memasukkan nama topik, pastikan bahwa **thingname** sama dengan nama AWS IoT Thing yang Anda buat sebelumnya. Anda juga dapat berlangganan topik MQTT tambahan untuk melihat apakah pembaruan telah berhasil dilakukan. Misalnya, Anda dapat berlangganan topik `$ aws/hal/thingname/shadow/update/` ditolak untuk menerima pesan setiap kali permintaan pembaruan gagal sehingga Anda dapat men-debug masalah koneksi. Untuk informasi selengkapnya tentang topik yang dipesan, lihat [the section called "Topik bayangan"](#) dan [Topik Device Shadow MQTT](#).

Langkah 4: Tinjau hasil dan langkah selanjutnya

Dalam tutorial ini, Anda belajar bagaimana:

- Menggunakan `shadow.py` contoh aplikasi untuk menentukan status yang diinginkan dan memperbarui status bayangan saat ini.
- Mengedit dokumen Shadow untuk mengamati peristiwa delta dan bagaimana `shadow.py` aplikasi sampel merespon itu.
- Gunakan klien uji MQTT untuk berlangganan topik bayangan dan mengamati pembaruan saat Anda menjalankan program sampel.

Langkah selanjutnya

Anda dapat berlangganan topik tambahan yang dipesan MQTT untuk mengamati pembaruan pada aplikasi bayangan. Misalnya, jika Anda hanya berlangganan topik `$aws/hal/thingname/shadow/update/accepted`, Anda hanya akan melihat informasi status saat ini ketika pembaruan berhasil dilakukan.

Anda juga dapat berlangganan topik bayangan tambahan untuk men-debug masalah atau mempelajari lebih lanjut tentang interaksi Device Shadow dan juga men-debug masalah apa pun dengan interaksi Device Shadow. Untuk informasi selengkapnya, lihat [the section called “Topik bayangan”](#) dan [Topik Device Shadow MQTT](#).

Anda juga dapat memilih untuk memperluas aplikasi Anda dengan menggunakan bayangan bernama atau dengan menggunakan perangkat keras tambahan yang terhubung dengan Raspberry Pi untuk LED dan mengamati perubahan keadaan mereka menggunakan pesan yang dikirim dari terminal.

Untuk informasi selengkapnya tentang layanan Device Shadow dan menggunakan layanan di perangkat, aplikasi, dan layanan, lihat [AWS IoT Layanan Device Shadow](#), [Menggunakan bayangan di perangkat](#), dan [Menggunakan bayangan di aplikasi dan layanan](#).

Tutorial: Membuat otorisasi khusus untuk AWS IoT Core

Tutorial ini menunjukkan langkah-langkah untuk membuat, memvalidasi, dan menggunakan Custom Authentication dengan menggunakan AWS CLI Secara opsional, menggunakan tutorial ini, Anda dapat menggunakan Postman untuk mengirim data AWS IoT Core dengan menggunakan HTTP Publish API.

Tutorial ini menunjukkan cara membuat contoh fungsi Lambda yang mengimplementasikan logika otorisasi dan otentikasi dan otorisasi khusus menggunakan panggilan dengan penandatanganan token diaktifkan. `create-authorizer` Authorizer kemudian divalidasi menggunakan `test-invoke-authorizer`, dan akhirnya Anda dapat mengirim data AWS IoT Core dengan menggunakan HTTP Publish API ke topik MQTT pengujian. Permintaan sampel akan menentukan otorisasi untuk dipanggil dengan menggunakan `x-amz-customauthorizer-name` header dan meneruskan header `token-key-name` and `x-amz-customauthorizer-signature` in request.

Apa yang akan Anda pelajari dalam tutorial ini:

- Cara membuat fungsi Lambda menjadi penanganan otorisasi khusus
- Cara membuat otorisasi khusus menggunakan penandatanganan token AWS CLI dengan diaktifkan
- Cara menguji otorisasi khusus Anda menggunakan perintah `test-invoke-authorizer`
- Cara mempublikasikan topik MQTT dengan menggunakan [Postman](#) dan memvalidasi permintaan dengan otorisasi khusus Anda

Tutorial ini membutuhkan waktu sekitar 60 menit untuk menyelesaikannya.

Dalam tutorial ini, Anda akan:

- [Langkah 1: Buat fungsi Lambda untuk otorisasi kustom Anda](#)
- [Langkah 2: Buat key pair publik dan pribadi untuk otorisasi kustom Anda](#)
- [Langkah 3: Buat sumber otorisasi pelanggan dan otorisasi](#)
- [Langkah 4: Uji otorisasi dengan menelepon `test-invoke-authorizer`](#)
- [Langkah 5: Uji penerbitan pesan MQTT menggunakan Postman](#)
- [Langkah 6: Lihat pesan di klien pengujian MQTT](#)
- [Langkah 7: Tinjau hasil dan langkah selanjutnya](#)
- [Langkah 8: Membersihkan](#)

Sebelum Anda memulai tutorial ini, pastikan Anda memiliki:

- [Siapkan Anda Akun AWS](#)

Anda akan membutuhkan Akun AWS dan AWS IoT konsol Anda untuk menyelesaikan tutorial ini.

Akun yang Anda gunakan untuk tutorial ini berfungsi paling baik jika menyertakan setidaknya kebijakan AWS terkelola ini:

- [IAMFullAccess](#)
- [AWSIoTFullAccess](#)
- [AWSLambda_FullAccess](#)

 Important

Kebijakan IAM yang digunakan dalam tutorial ini lebih permisif daripada yang harus Anda ikuti dalam implementasi produksi. Di lingkungan produksi, pastikan kebijakan akun dan sumber daya Anda hanya memberikan izin yang diperlukan.

Saat Anda membuat kebijakan IAM untuk produksi, tentukan akses apa yang dibutuhkan pengguna dan peran, lalu rancang kebijakan yang memungkinkan mereka hanya melakukan tugas tersebut.

Untuk informasi selengkapnya, lihat [Praktik terbaik keamanan di IAM](#)

- Menginstal AWS CLI

Untuk informasi tentang cara menginstal AWS CLI, lihat [Menginstal AWS CLI](#). Tutorial ini membutuhkan AWS CLI versi `aws-cli/2.1.3 Python/3.7.4 Darwin/18.7.0 exe/x86_64` atau yang lebih baru.

- Alat OpenSSL

Contoh dalam tutorial ini menggunakan [LibreSSL](#) 2.6.5. Anda juga dapat menggunakan alat [OpenSSL](#) v1.1.1i untuk tutorial ini.

- Meninjau [AWS Lambda](#)ikhtisar

Jika Anda belum pernah menggunakannya AWS Lambda sebelumnya, tinjau [AWS Lambda](#) dan [Mulai dengan Lambda](#) untuk mempelajari istilah dan konsepnya.

- Meninjau cara membuat permintaan di Postman

Untuk informasi selengkapnya, lihat [Permintaan bangunan](#).

- Dihapus otorisasi kustom dari tutorial sebelumnya

Anda hanya Akun AWS dapat memiliki sejumlah otorisasi khusus yang dikonfigurasi pada satu waktu. Untuk informasi tentang cara menghapus otorisasi kustom, lihat [the section called “Langkah 8: Membersihkan”](#).

Langkah 1: Buat fungsi Lambda untuk otorisasi kustom Anda

Otentikasi kustom AWS IoT Core menggunakan [sumber daya otorisasi](#) yang Anda buat untuk mengautentikasi dan mengotorisasi klien. Fungsi yang akan Anda buat di bagian ini mengautentikasi dan mengotorisasi klien saat mereka terhubung AWS IoT Core dan mengakses AWS IoT sumber daya.

Fungsi Lambda melakukan hal berikut:

- Jika permintaan berasal dari `test-invoke-authorizer`, ia mengembalikan kebijakan IAM dengan `Deny` tindakan.
- Jika permintaan berasal dari `Passport` menggunakan `HTTP` dan `actionToken` parameter memiliki nilai `allow`, ia mengembalikan kebijakan IAM dengan `Allow` tindakan. Jika tidak, ia mengembalikan kebijakan IAM dengan `Deny` tindakan.

Untuk membuat fungsi Lambda untuk otorisasi kustom Anda

1. [Di konsol Lambda, buka Fungsi.](#)
2. Pilih `Buat fungsi`.
3. Konfirmasi Penulis dari awal dipilih.
4. Di bawah Informasi dasar:
 - a. Dalam nama Fungsi, masukkan **`custom-auth-function`**.
 - b. Di Runtime, konfirmasi `Node.js 18.x`.
5. Pilih `Buat fungsi`.

Lambda membuat fungsi `Node.js` dan [peran eksekusi](#) yang memberikan izin fungsi untuk mengunggah log. Fungsi Lambda mengasumsikan peran eksekusi saat Anda menjalankan fungsi dan menggunakan peran eksekusi untuk membuat kredensial AWS SDK dan membaca data dari sumber peristiwa.

6. Untuk melihat kode dan konfigurasi fungsi di [AWS Cloud9](#) editor, pilih `custom-auth-function` di jendela desainer, lalu pilih `index.js` di panel navigasi editor.

Untuk bahasa scripting seperti `Node.js`, Lambda menyertakan fungsi dasar yang mengembalikan respons sukses. Anda dapat menggunakan [AWS Cloud9](#) editor untuk mengedit fungsi Anda selama kode sumber Anda tidak melebihi 3 MB.

7. Ganti kode `index.js` di editor dengan kode berikut:

```
// 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. Pilih Deploy.
9. Setelah Perubahan diterapkan muncul di atas editor:
 - a. Gulir ke bagian Ikhtisar fungsi di atas editor.
 - b. Salin Fungsi ARN dan simpan untuk digunakan nanti dalam tutorial ini.
10. Uji fungsi Anda.
 - a. Pilih tab Uji.
 - b. Menggunakan pengaturan pengujian default, pilih Invoke.
 - c. Jika tes berhasil, dalam hasil Eksekusi, buka tampilan Detail. Anda akan melihat dokumen kebijakan yang dikembalikan fungsi.

Jika pengujian gagal atau Anda tidak melihat dokumen kebijakan, tinjau kode untuk menemukan dan memperbaiki kesalahan.

Langkah 2: Buat key pair publik dan pribadi untuk otorisasi kustom Anda

Authorizer kustom Anda memerlukan kunci publik dan pribadi untuk mengotentikasi itu. Perintah di bagian ini menggunakan alat OpenSSL untuk membuat key pair ini.

Untuk membuat key pair publik dan pribadi untuk otorisasi kustom Anda

1. Buat file kunci pribadi.

```
openssl genrsa -out private-key.pem 4096
```

2. Verifikasi file kunci pribadi yang baru saja Anda buat.

```
openssl rsa -check -in private-key.pem -noout
```

Jika perintah tidak menampilkan kesalahan apa pun, file kunci pribadi valid.

3. Buat file kunci publik.

```
openssl rsa -in private-key.pem -pubout -out public-key.pem
```

4. Verifikasi file kunci publik.

```
openssl pkey -inform PEM -pubin -in public-key.pem -noout
```

Jika perintah tidak menampilkan kesalahan apa pun, file kunci publik valid.

Langkah 3: Buat sumber otorisasi pelanggan dan otorisasi

Authorizer AWS IoT kustom adalah sumber daya yang mengikat semua elemen yang dibuat pada langkah sebelumnya. Di bagian ini, Anda akan membuat sumber otorisasi khusus dan memberinya izin untuk menjalankan fungsi Lambda yang Anda buat sebelumnya. Anda dapat membuat resource otorisasi kustom dengan menggunakan AWS IoT konsol, API AWS CLI, atau AWS API.

Untuk tutorial ini, Anda hanya perlu membuat satu otorisasi khusus. Bagian ini menjelaskan cara membuat dengan menggunakan AWS IoT konsol dan AWS CLI, sehingga Anda dapat menggunakan metode yang paling nyaman bagi Anda. Tidak ada perbedaan antara sumber daya otorisasi khusus yang dibuat oleh salah satu metode.

Buat sumber daya otorisasi pelanggan

Pilih salah satu opsi ini untuk membuat sumber otorisasi kustom Anda

- [Buat otorisasi khusus dengan menggunakan konsol AWS IoT](#)
- [Buat otorisasi khusus menggunakan AWS CLI](#)

Untuk membuat otorisasi kustom (konsol)

1. Buka [halaman Custom Authorizer AWS IoT konsol](#), dan pilih Create Authorizer.
2. Di Create Authorizer:
 - a. Dalam nama Authorizer, masukkan **my-new-authorizer**.
 - b. Dalam status Authorizer, periksa Aktif.
 - c. Dalam fungsi Authorizer, pilih fungsi Lambda yang Anda buat sebelumnya.
 - d. Dalam validasi Token - opsional:
 - i. Beralih pada validasi Token.
 - ii. Dalam nama kunci Token, masukkan **tokenKeyName**.
 - iii. Pilih Tambah kunci.

- iv. Dalam Nama kunci, masukkan **FirstKey**.
- v. Dalam kunci Publik, masukkan isi `public-key.pem` file. Pastikan untuk menyertakan baris dari file dengan `-----BEGIN PUBLIC KEY-----` dan `-----END PUBLIC KEY-----` dan jangan menambahkan atau menghapus umpan baris, pengembalian carriage, atau karakter lain dari konten file. String yang Anda masukkan akan terlihat seperti contoh ini.

```
-----BEGIN PUBLIC KEY-----
MIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEAvEBz0k4vhN+3Lgs1vEWt
sLCqNmt5Damas3bmiTRvq2gjRJ6KXGTGQChqArAJwL1a9dkS9+maaXC3vc6xxz9z
QPu/vQ0e5tyzz1MsKdmtFGxMqQ3qjEXAMPLE0mqyUKPP5mff58k6ePSfXAnzBH0q
lg2Hioefrpu50SAnpuRAjYKofKjbc2Vrn6N2G7hV+IfTBvCE1f0csa1S/Rk4phD5
oa4Y0GHISRnevyppg5C8n9Rrz91PWGqP6M/q5DNJjXjMy1eG92hQgu1N696bn5Dw8
FhedszFa6b2x6xrItZFzewNQkPMLMFhNrQIIyvshtT/F1LVCS5+v8AQ8UGGDfZmv
QeqAMAF7WgagDMXcfgKSVU8yid2sIm56qsCLMvD2Sq8Lgzpey9N50N1o1Cv1dwvc
KrJJtgwW6hVqRGuShnownLpgG86M6neZ5sRMbVNZ080zcobLngJ0Ibw9KkcUdklW
gvZ6HEJqBY2XE70iEXAMPLETPHzhqvK6Ei1HGxpHsXx6BNft582J1VpgYjXha8oa
/NN717Zbj/euAb41IVtmX8JrD9z613d1iM5L8HluJlUzn62Q+VeNV2tdA7MfPfMC
8btGYladFAnitThaz6+F0VSBJPu7pZQoLnqyEp5zLMtF+kF12y0BmGAP0RBivRd9
JWBUCG0bqcLQPeQyjbXS0fUCAwEAAQ==
-----END PUBLIC KEY-----
```

3. Pilih Buat Authorizer.
4. Jika sumber otorisasi kustom dibuat, Anda akan melihat daftar otorisasi kustom dan otorisasi kustom baru Anda akan muncul dalam daftar dan Anda dapat melanjutkan ke bagian berikutnya untuk mengujinya.

Jika Anda melihat kesalahan, tinjau kesalahan dan coba buat otorisasi khusus Anda lagi dan periksa kembali entri. Perhatikan bahwa setiap sumber otorisasi kustom harus memiliki nama yang unik.

Untuk membuat otorisasi kustom () AWS CLI

1. Gantikan nilai Anda untuk `authorizer-function-arn` dan `token-signing-public-keys`, lalu jalankan perintah berikut:

```
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-----
MIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEAvEBz0k4vhN+3LgslvEWt
sLCqNmt5Damas3bmiTRvq2gjRJ6KXGTGQChqArAJwL1a9dkS9+maaXC3vc6xzx9z
QPu/vQ0e5tyzz1MsKdmtFGxMqQ3qjEXAMPLE0mqyUKPP5mff58k6ePSfXAnzBH0q
lg2HioefrpU50SANpuRAjYKofKjbc2Vrn6N2G7hV+IfTBvCElf0csa1S/Rk4phD5
oa4Y0GHISRnevyppg5C8n9Rrz91PWGqP6M/q5DNJJXjMyLeG92hQgu1N696bn5Dw8
FhedszFa6b2x6xrItZFzewNqkPMLMFhNrQIIyvshtT/F1LVCS5+v8AQ8UGGDFZmv
QeqAMAF7WgagDMXcfGKSVU8yid2sIm56qsCLMvD2Sg8Lgzpey9N50N1o1Cvldwvc
KrJJtgwW6hVqRGUShnownLpgG86M6neZ5sRMbVNZ080zcobLngJ0Ibw9KkcUdk1W
gvZ6HEJqBY2XE70iEXAMPLETPHzhqvK6Ei1HGxpHsXx6BNft582J1VpgYjXha8oa
/NN7L7Zbj/euAb41IVtmX8JrD9z613d1iM5L8HluJlUzn62Q+VeNV2tdA7MfPFMC
8btGYladFAnitThaz6+F0VSBJPu7pZQoLnqyEp5zLMtF+kFL2y0BmGAP0RBivRd9
JWBUCG0bqcLQPeQyjbXS0FUCAwEAAQ==
-----END PUBLIC KEY-----"
```

Di mana:

- `authorizer-function-arn` Nilainya adalah Nama Sumber Daya Amazon (ARN) dari fungsi Lambda yang Anda buat untuk otorisasi khusus Anda.
- `token-signing-public-keys` Nilai termasuk nama kunci, **FirstKey**, dan isi `public-key.pem` file. Pastikan untuk menyertakan baris dari file dengan `-----BEGIN PUBLIC KEY-----` dan `-----END PUBLIC KEY-----` dan jangan menambahkan atau menghapus umpan baris, pengembalian carriage, atau karakter lain dari konten file.

Catatan: hati-hati memasukkan kunci publik karena setiap perubahan pada nilai kunci publik membuatnya tidak dapat digunakan.

2. Jika otorisasi kustom dibuat, perintah mengembalikan nama dan ARN dari sumber daya baru, seperti berikut ini.

```
{
  "authorizerName": "my-new-authorizer",
  "authorizerArn": "arn:aws:iot:Region:57EXAMPLE833:authorizer/my-new-authorizer"
}
```

Simpan `authorizerArn` nilai untuk digunakan pada langkah berikutnya.

Ingat bahwa setiap sumber otorisasi kustom harus memiliki nama yang unik.

Otorisasi sumber otorisasi kustom

Di bagian ini, Anda akan memberikan izin sumber otorisasi khusus yang baru saja Anda buat izin untuk menjalankan fungsi Lambda. Untuk memberikan izin, Anda dapat menggunakan perintah CLI [izin tambahan](#).

Berikan izin ke fungsi Lambda Anda menggunakan AWS CLI

1. Setelah memasukkan nilai Anda, masukkan perintah berikut. Perhatikan bahwa `statement-id` nilainya harus unik. Ganti `Id-1234` dengan nilai lain jika Anda telah menjalankan tutorial ini sebelumnya atau jika Anda mendapatkan `ResourceConflictException` kesalahan.

```
aws lambda add-permission \
--function-name "custom-auth-function" \
--principal "iot.amazonaws.com" \
--action "lambda:InvokeFunction" \
--statement-id "Id-1234" \
--source-arn authorizerArn
```

2. Jika perintah berhasil, ia mengembalikan pernyataan izin, seperti contoh ini. Anda dapat melanjutkan ke bagian berikutnya untuk menguji otorisasi khusus.

```
{
  "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\"}}}"
}
```

Jika perintah tidak berhasil, ia mengembalikan kesalahan, seperti contoh ini. Anda harus meninjau dan memperbaiki kesalahan sebelum melanjutkan.

```
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
```

Langkah 4: Uji otorisasi dengan menelepon test-invoke-authorizer

Dengan semua sumber daya yang ditentukan, di bagian ini, Anda akan memanggil test-invoke-authorizer dari baris perintah untuk menguji izin otorisasi.

Perhatikan bahwa ketika memanggil otorisasi dari baris perintah, tidak protokolData ditentukan, sehingga otorisasi akan selalu mengembalikan dokumen DENY. Namun, pengujian ini mengonfirmasi bahwa otorisasi khusus dan fungsi Lambda Anda dikonfigurasi dengan benar - bahkan jika itu tidak sepenuhnya menguji fungsi Lambda.

Untuk menguji otorisasi kustom Anda dan fungsi Lambda dengan menggunakan AWS CLI

1. Di direktori yang memiliki private-key.pem file yang Anda buat pada langkah sebelumnya, jalankan perintah berikut.

```
echo -n "tokenKeyValue" | openssl dgst -sha256 -sign private-key.pem | openssl base64 -A
```

Perintah ini membuat string tanda tangan untuk digunakan pada langkah berikutnya. String tanda tangan terlihat seperti ini:

```
dBwykz1b+fo+JmSGdwoGr8dyC2qB/IyLefJJr+rbCvmu9Jl4KHAA9DG+V
+MMWu09YSA86+64Y3Gt4t0ykpZqn9mn
VB1wyxp+0bDZ8hmqUAUH3fwi3fPjBvCa4cwNuLQNqBZzbCvs1uv7i2IMjEg
+CPY0zrWt1jr9BikgGPDxWkjaeeh
bQHHTo357TegKs9pP30Uf4TrxypNmFswA5k7QIc01n4bIyRTm900yZ94R4bdJsHNig1JePgnu0BvMGCEFE09jGjj
szEHfgAUAQIWXiVGQj16BU1xKpTGSiTawheLKUjITOEXAMPLECK3aHKYKY
+d1vTvdthKtYHBq8MjhzJ0kggbt29V
QJCb8Ri1N/P5+vcVniSXWPplyB5jkYs9UvG08REoy64AtizfUhvSul/r/F3VV8ITtQp3aXiUtcspACi6ca
+tsDuX
f3LzCwQQF/YSUy02u5Xkwn
+sto6KCKpNlkD0wU8g13+k0zxrthnQ8gEajd5Iy1x230iqcXo3osjPha7JDyWM5o+K
EWckTe91I1mokDr5sJ4JXixvnJTVSx1li49IalW4en1DAkc1a0s2U2UNm236EXAMPLELotyh7h
+f1Fe1oZ1AWQFH
xR1XsPqiVKS1ZIUC1aZWprh/orDJplpiWfBgBIOgokJIDGP9gwhXIIk7zWtGmWpMK9o=
```

Salin string tanda tangan ini untuk digunakan pada langkah berikutnya. Berhati-hatilah untuk tidak menyertakan karakter tambahan atau meninggalkannya.

2. Dalam perintah ini, ganti token-signature nilai dengan string tanda tangan dari langkah sebelumnya dan jalankan perintah ini untuk menguji otorisasi Anda.

```
aws iot test-invoke-authorizer \
--authorizer-name my-new-authorizer \
--token tokenKeyValue \
--token-signature dBwykzLb+fo+JmSGdwoGr8dyC2qB/IyLefJJr
+rbCvmu9JL4KHAA9DG+V+MMWu09YSA86+64Y3Gt4t0ykpZqn9mnVB1wyxp
+0bDZh8hmqUAUH3fwi3fPjBvCa4cwNuLQNqBZzbCvsluv7i2IMjEg
+CPY0zrWt1jr9BikgGPDxWkjaeehbQHHTo357TegKs9pP30Uf4TrxypNmFswA5k7QIc01n4bIyRTm900yZ94R4bdJsh
+d1vTvdthKtYHBq8MjhzJ0kggbt29VQJCb8RiLN/
P5+vcVniSXWpPlyB5jkYs9UvG08REoy64AtizfUhvSul/r/F3VV8ITtQp3aXiUtcspACi6ca
+tsDuXf3LzCwQQF/YsUy02u5Xkwn
+sto6KCKpNlkD0wU8gl3+k0zxrthnQ8gEajd5IyLx230iqcXo3osjPha7JDyWM5o
+KEWckTe91I1mokDr5sJ4JXixvnJTVSx1li49IalW4en1DAkc1a0s2U2UNm236EXAMPLELotyh7h
+f1FeLoZLAWQFHxRLXsPqiVKS1ZIUClaZWprh/orDJplpiWfBgBIOgokJIDGP9gwhXIik7zWrGmWpMK9o=
```

Jika perintah berhasil, ia mengembalikan informasi yang dihasilkan oleh fungsi otorisasi pelanggan Anda, seperti contoh ini.

```
{
  "isAuthenticated": true,
  "principalId": "principalId",
  "policyDocuments": [
    "{\"Version\":\"2012-10-17\",\"Statement\": [{\"Action\":\"iot:*\",\"Effect\":\"Deny\",\"Resource\":\"arn:aws:iot:Region:57EXAMPLE833:*\"}]}"
  ],
  "refreshAfterInSeconds": 600,
  "disconnectAfterInSeconds": 3600
}
```

Jika perintah mengembalikan kesalahan, tinjau kesalahan dan periksa kembali perintah yang Anda gunakan di bagian ini.

Langkah 5: Uji penerbitan pesan MQTT menggunakan Postman

1. Untuk mendapatkan titik akhir data perangkat Anda dari baris perintah, panggil [describe-endpoint](#) seperti yang ditunjukkan di sini

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

Simpan alamat ini untuk digunakan sebagai *device_data_endpoint_address* di langkah selanjutnya.

2. Buka jendela Postman baru dan buat permintaan HTTP POST baru.
 - a. Dari komputer Anda, buka aplikasi Postman.
 - b. Di Postman, di menu File, pilih New... .
 - c. Di kotak dialog Baru, pilih Permintaan.
 - d. Dalam permintaan Simpan,
 - i. Dalam nama Permintaan masukkan **Custom authorizer test request**.
 - ii. Di Pilih koleksi atau folder untuk disimpan: pilih atau buat koleksi untuk menyimpan permintaan ini.
 - iii. Pilih Simpan ke **collection_name**.
3. Buat permintaan POST untuk menguji otorisasi kustom Anda.
 - a. Dalam pemilih metode permintaan di sebelah bidang URL, pilih POST.
 - b. Di bidang URL, buat URL untuk permintaan Anda dengan menggunakan URL berikut dengan *device_data_endpoint_address* dari perintah *describe-endpoint* pada langkah sebelumnya.

```
https://device_data_endpoint_address:443/topics/test/cust-auth/topic?  
qos=0&actionToken=allow
```

Perhatikan bahwa URL ini menyertakan parameter `actionToken=allow` kueri yang akan memberi tahu fungsi Lambda Anda untuk mengembalikan dokumen kebijakan yang memungkinkan akses ke AWS IoT. Setelah Anda memasukkan URL, parameter kueri juga muncul di tab Params Postman.

- c. Di tab Auth, di bidang Type, pilih No Auth.

d. Di tab Header:

- i. Jika ada kunci Host yang dicentang, hapus centang yang satu ini.
- ii. Di bagian bawah daftar header tambahkan header baru ini dan konfirmasikan bahwa mereka diperiksa. Ganti **Host** nilai dengan *device_data_endpoint_address* Anda dan **x-amz-customauthorizer-signature** nilainya dengan string tanda tangan yang Anda gunakan dengan perintah di bagian sebelumnya. `test-invoke-authorize`

Kunci	Nilai
x-amz-customauthorizer-name	my-new-authorizer
Host	<i>device_data_endpoint_address</i>
tokenKeyName	tokenKeyValue

Kunci	Nilai
x-amz-customauthorizer-signature	<i>dbwykzlb+fo+jms 8dYc2qb/ jjr +rbcvmu9jl4khaa9dg+v+mmw u09ysa86+64y3gt4 9mnb1wyx p+0bdzh8hmquauh3fwi3 ca4cwnulqnb 7i2i +pyc2i 0zrwt1jr9bikggp qhhto357 9pp30uf4 a5k7qic01n4biyrtm9 0oyz94r4bdjshnig1 GdwoGr obvmgcefe09jgjszehfgauaqiw xivgqj16bu1xkptgsitawhelkuj itoexampleCk3ahaqiwxivgqj16 bu1xkptgsitawhelkujitoexam leCk3ahaqiwxivgqj16bu1xkptg sitawhelkujitoexampleCk3aha qiwxivgqj16bu1xkptgsitawhel kujitoexampleCk3ahaqiwxivgq j16bu1xkptgsitaKY+d1 yhbq8mjhzy0kggbt29vqjcb8ril n/p5+vcvnisxwplyb5jkys9uvq0 8reoy64 /r/f3vv8i 3 aci6ca+ 3 IyLef qqf/ysuy02u5 +Sto6K t0ykpZqn fPjBv ZzbCvsluv MjEg DxWkjaeehb TegKs TrxypNmFsw JePgnu vTvdthKt AtizfUhvSul TtQp aXiUtcsp tsDuXf LzCw XkWn CkpNlkd0wu8gl3+kozxrthnq8ge aJd5iylx230iqcxo3osjpha7jdy wm5o+ke 91i1mokdr5sj4jxixv njtvsx1li49ialw4en1dakc1a0s 2u2u2unm236Examplelotyh7h7+ AWQF vks1ziuclazwprh/Ord WckTe BiogokjidGp9gwHxiik7 wpmk9o= fIFeloZI HxRIXsPqi JplpiWfBg zWrGm</i>

- e. Di tab Tubuh:
 - i. Di kotak opsi format data, pilih Raw.
 - ii. Dalam daftar tipe data, pilih JavaScript.
 - iii. Di bidang teks, masukkan payload pesan JSON ini untuk pesan pengujian Anda:

```
{
  "data_mode": "test",
  "vibration": 200,
  "temperature": 40
}
```

4. Pilih Kirim untuk mengirim permintaan.

Jika permintaan berhasil, ia mengembalikan:

```
{
  "message": "OK",
  "traceId": "ff35c33f-409a-ea90-b06f-fbEXAMPLE25c"
}
```

Respons yang berhasil menunjukkan bahwa otorisasi khusus Anda mengizinkan koneksi ke AWS IoT dan bahwa pesan pengujian dikirimkan ke broker di AWS IoT Core.

Jika mengembalikan kesalahan, tinjau pesan kesalahan, *device_data_endpoint_address*, *string tanda* tangan, dan nilai header lainnya.

Simpan permintaan ini di Postman untuk digunakan di bagian berikutnya.

Langkah 6: Lihat pesan di klien pengujian MQTT

Pada langkah sebelumnya, Anda mengirim pesan perangkat simulasi ke AWS IoT dengan menggunakan Postman. Tanggapan yang berhasil menunjukkan bahwa otorisasi khusus Anda mengizinkan koneksi ke AWS IoT dan bahwa pesan pengujian dikirimkan ke broker di AWS IoT Core. Di bagian ini, Anda akan menggunakan klien pengujian MQTT di AWS IoT konsol untuk melihat konten pesan dari pesan tersebut seperti perangkat dan layanan lain.

Untuk melihat pesan pengujian yang diotorisasi oleh otorisasi kustom

1. Di AWS IoT konsol, buka klien uji [MQTT](#).

2. Di tab Berlangganan topik, di Filter topik, masukkan **test/cust-auth/topic**, yang merupakan topik pesan yang digunakan dalam contoh Tukang Pos dari bagian sebelumnya.
3. Pilih Langganan.

Biarkan jendela ini terlihat untuk langkah berikutnya.

4. Di Postman, dalam permintaan yang Anda buat untuk bagian sebelumnya, pilih Kirim.

Tinjau tanggapan untuk memastikan itu berhasil. Jika tidak, pecahkan masalah kesalahan seperti yang dijelaskan bagian sebelumnya.

5. Di klien pengujian MQTT, Anda akan melihat entri baru yang menunjukkan topik pesan dan, jika diperluas, muatan pesan dari permintaan yang Anda kirim dari Tukang Pos.

Jika Anda tidak melihat pesan Anda di klien pengujian MQTT, berikut adalah beberapa hal yang perlu diperiksa:

- Pastikan permintaan Tukang Pos Anda berhasil dikembalikan. Jika AWS IoT menolak koneksi dan mengembalikan kesalahan, pesan dalam permintaan tidak diteruskan ke broker pesan.
- Pastikan Akun AWS dan yang Wilayah AWS digunakan untuk membuka AWS IoT konsol sama dengan yang Anda gunakan di URL Postman.
- Pastikan Anda telah memasukkan topik dengan benar di klien pengujian MQTT. Filter topik peka huruf besar/kecil. Jika ragu, Anda juga dapat berlangganan # topik, yang berlangganan semua pesan MQTT yang melewati broker pesan Akun AWS dan Wilayah AWS digunakan untuk membuka konsol. AWS IoT

Langkah 7: Tinjau hasil dan langkah selanjutnya

Dalam tutorial ini:

- Anda membuat fungsi Lambda untuk menjadi penanganan otorisasi khusus
- Anda membuat otorisasi khusus dengan penandatanganan token diaktifkan
- Anda menguji otorisasi kustom Anda menggunakan perintah `test-invoke-authorizer`
- Anda menerbitkan topik MQTT dengan menggunakan [Postman](#) dan memvalidasi permintaan dengan otorisasi khusus Anda
- Anda menggunakan klien pengujian MQTT untuk melihat pesan yang dikirim dari tes Postman Anda

Langkah selanjutnya

Setelah Anda mengirim beberapa pesan dari Postman untuk memverifikasi bahwa otorisasi kustom berfungsi, coba bereksperimen untuk melihat bagaimana mengubah berbagai aspek tutorial ini memengaruhi hasil. Berikut adalah beberapa contoh untuk membantu Anda memulai.

- Ubah string tanda tangan sehingga tidak lagi valid untuk melihat bagaimana upaya koneksi yang tidak sah ditangani. Anda akan mendapatkan respons kesalahan, seperti ini, dan pesan tidak akan muncul di klien pengujian MQTT.

```
{
  "message": "Forbidden",
  "traceId": "15969756-a4a4-917c-b47a-5433e25b1356"
}
```

- Untuk mempelajari lebih lanjut tentang cara menemukan kesalahan yang mungkin terjadi saat Anda mengembangkan dan menggunakan AWS IoT aturan, lihat [Pemantauan AWS IoT](#).

Langkah 8: Membersihkan

Jika Anda ingin mengulangi tutorial ini, Anda mungkin perlu menghapus beberapa otorisasi khusus Anda. Anda hanya Akun AWS dapat memiliki sejumlah otorisasi khusus yang dikonfigurasi pada satu waktu dan Anda bisa mendapatkan `LimitExceededException` ketika Anda mencoba menambahkan yang baru tanpa menghapus otorisasi kustom yang ada.

Untuk menghapus otorisasi kustom (konsol)

1. Buka [halaman otorisasi khusus AWS IoT konsol](#), dan dalam daftar otorisasi khusus, temukan otorisasi khusus untuk dihapus.
2. Buka halaman detail otorisasi khusus dan, dari menu Tindakan, pilih Edit.
3. Hapus centang pada Activate authorizer, lalu pilih Update.

Anda tidak dapat menghapus otorisasi khusus saat sedang aktif.

4. Dari halaman Detail otorisasi khusus, buka menu Tindakan, dan pilih Hapus.

Untuk menghapus otorisasi kustom () AWS CLI

1. Buat daftar otorisasi khusus yang telah Anda instal dan temukan nama otorisasi khusus yang ingin Anda hapus.

```
aws iot list-authorizers
```

2. Atur otorisasi khusus `inactive` dengan menjalankan perintah ini setelah mengganti `Custom_Auth_Name` dengan otorisasi khusus untuk dihapus. `authorizerName`

```
aws iot update-authorizer --status INACTIVE --authorizer-name Custom_Auth_Name
```

3. Hapus otorisasi khusus dengan menjalankan perintah ini setelah mengganti `Custom_Auth_Name` dengan otorisasi khusus untuk dihapus. `authorizerName`

```
aws iot delete-authorizer --authorizer-name Custom_Auth_Name
```

Tutorial: Memantau kelembaban tanah dengan AWS IoT dan Raspberry Pi

Tutorial ini menunjukkan kepada Anda cara menggunakan [Raspberry Pi](#), sensor kelembaban, dan AWS IoT untuk memantau tingkat kelembaban tanah untuk tanaman rumah atau taman. Raspberry Pi menjalankan kode yang membaca tingkat kelembaban dan suhu dari sensor dan kemudian mengirimkan data ke AWS IoT. Anda membuat aturan AWS IoT yang mengirim email ke alamat yang berlangganan topik Amazon SNS saat tingkat kelembaban turun di bawah ambang batas.

Note

Tutorial ini mungkin tidak up to date. Beberapa referensi mungkin telah digantikan karena topik ini awalnya diterbitkan.

Daftar Isi

- [Prasyarat](#)
- [Menyiapkan AWS IoT](#)
 - [Langkah 1: Buat AWS IoT kebijakan](#)
 - [Langkah 2: Buat AWS IoT benda, sertifikat, dan kunci pribadi](#)
 - [Langkah 3: Buat topik dan langganan Amazon SNS](#)
 - [Langkah 4: Buat AWS IoT aturan untuk mengirim email](#)
- [Menyiapkan Raspberry Pi dan sensor kelembaban](#)

Prasyarat

Untuk menyelesaikan tutorial ini, Anda memerlukan:

- Sebuah Akun AWS.
- Pengguna IAM dengan izin administrator.
- [Komputer pengembangan yang menjalankan Windows, macOS, Linux, atau Unix untuk mengakses konsol. AWS IoT](#)
- [Raspberry Pi 3B atau 4B](#) yang menjalankan OS [Raspbian](#) terbaru. Untuk petunjuk penginstalan, lihat [Menginstal gambar sistem operasi](#) di situs web Raspberry Pi.
- Monitor, keyboard, mouse, dan jaringan Wi-Fi atau koneksi Ethernet untuk Raspberry Pi Anda.
- Sensor kelembaban yang kompatibel dengan Raspberry Pi. Sensor yang digunakan dalam tutorial ini adalah [Adafruit STEMMA I2C Capacitive Moisture Sensor](#) dengan header kabel soket 4-pin [JST ke wanita](#).

Menyiapkan AWS IoT

Untuk menyelesaikan tutorial ini, Anda perlu membuat sumber daya berikut. Untuk menghubungkan perangkat AWS IoT, Anda membuat IoT, sertifikat perangkat, dan kebijakan. AWS IoT

- AWS IoT Sesuatu.

Sesuatu mewakili perangkat fisik (dalam hal ini, Raspberry Pi Anda) dan berisi metadata statis tentang perangkat.

- Sertifikat perangkat.

Semua perangkat harus memiliki sertifikat perangkat untuk terhubung dan mengautentikasi. AWS IoT

- Sebuah AWS IoT kebijakan.

Setiap sertifikat perangkat memiliki satu atau lebih AWS IoT kebijakan yang terkait dengannya. Kebijakan ini menentukan AWS IoT sumber daya mana yang dapat diakses perangkat.

- Sertifikat CA AWS IoT root.

Perangkat dan klien lain menggunakan sertifikat CA AWS IoT root untuk mengautentikasi AWS IoT server yang dengannya mereka berkomunikasi. Untuk informasi selengkapnya, lihat [Otentikasi server](#).

- Sebuah AWS IoT aturan.

Aturan berisi kueri dan satu atau beberapa tindakan aturan. Kueri mengekstrak data dari pesan perangkat untuk menentukan apakah data pesan harus diproses. Tindakan aturan menentukan apa yang harus dilakukan jika data cocok dengan query.

- Langganan topik dan topik Amazon SNS.

Aturan mendengarkan data kelembaban dari Raspberry Pi Anda. Jika nilainya di bawah ambang batas, ia mengirim pesan ke topik Amazon SNS. Amazon SNS mengirimkan pesan itu ke semua alamat email yang berlangganan topik tersebut.

Langkah 1: Buat AWS IoT kebijakan

Buat AWS IoT kebijakan yang memungkinkan Raspberry Pi Anda terhubung dan mengirim pesan AWS IoT.

1. Di [AWS IoT konsol](#), jika tombol Mulai muncul, pilih. Jika tidak, di panel navigasi, perluas Keamanan, lalu pilih Kebijakan.
2. Jika kotak dialog Anda belum memiliki kebijakan apa pun muncul, pilih Buat kebijakan. Jika tidak, pilih Buat.
3. Masukkan nama untuk AWS IoT kebijakan (misalnya, **MoistureSensorPolicy**).
4. Di bagian Tambah pernyataan, ganti kebijakan yang ada dengan JSON berikut. Ganti *wilayah* dan *akun* dengan Akun AWS nomor Wilayah AWS dan Anda.

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "iot:Connect",
    "Resource": "arn:aws:iot:region:account:client/RaspberryPi"
  },
  {
    "Effect": "Allow",
    "Action": "iot:Publish",
    "Resource": [
      "arn:aws:iot:region:account:topic/$aws/things/RaspberryPi/shadow/
update",
```

```
    "arn:aws:iot:region:account:topic/$aws/things/RaspberryPi/shadow/
delete",
    "arn:aws:iot:region:account:topic/$aws/things/RaspberryPi/shadow/get"
  ]
},
{
  "Effect": "Allow",
  "Action": "iot:Receive",
  "Resource": [
    "arn:aws:iot:region:account:topic/$aws/things/RaspberryPi/shadow/
update/accepted",
    "arn:aws:iot:region:account:topic/$aws/things/RaspberryPi/shadow/
delete/accepted",
    "arn:aws:iot:region:account:topic/$aws/things/RaspberryPi/shadow/get/
accepted",
    "arn:aws:iot:region:account:topic/$aws/things/RaspberryPi/shadow/
update/rejected",
    "arn:aws:iot:region:account:topic/$aws/things/RaspberryPi/shadow/
delete/rejected"
  ]
},
{
  "Effect": "Allow",
  "Action": "iot:Subscribe",
  "Resource": [
    "arn:aws:iot:region:account:topicfilter/$aws/things/RaspberryPi/shadow/
update/accepted",
    "arn:aws:iot:region:account:topicfilter/$aws/things/RaspberryPi/shadow/
delete/accepted",
    "arn:aws:iot:region:account:topicfilter/$aws/things/RaspberryPi/shadow/
get/accepted",
    "arn:aws:iot:region:account:topicfilter/$aws/things/RaspberryPi/shadow/
update/rejected",
    "arn:aws:iot:region:account:topicfilter/$aws/things/RaspberryPi/shadow/
delete/rejected"
  ]
},
{
  "Effect": "Allow",
  "Action": [
    "iot:GetThingShadow",
    "iot:UpdateThingShadow",
    "iot>DeleteThingShadow"
  ]
},
```

```
    "Resource": "arn:aws:iot:region:account:thing/RaspberryPi"
  }
]
}
```

5. Pilih Buat.

Langkah 2: Buat AWS IoT benda, sertifikat, dan kunci pribadi

Buat sesuatu di AWS IoT registri untuk mewakili Raspberry Pi Anda.

1. Di [AWS IoT konsol](#), di panel navigasi, pilih Kelola, lalu pilih Things.
2. Jika kotak dialog Anda belum memiliki hal apa pun ditampilkan, pilih Daftarkan sesuatu. Jika tidak, pilih Buat.
3. Pada halaman Creating AWS IoT things, pilih Create a single.
4. Pada halaman Tambahkan perangkat Anda ke registri perangkat, masukkan nama untuk hal IoT Anda (misalnya, **RaspberryPi**), lalu pilih Berikutnya. Anda tidak dapat mengubah nama sesuatu setelah Anda membuatnya. Untuk mengubah nama hal, Anda harus membuat hal baru, memberikan nama baru, dan kemudian menghapus hal lama.
5. Pada halaman Tambahkan sertifikat untuk hal Anda, pilih Buat sertifikat.
6. Pilih tautan Unduh untuk mengunduh sertifikat, kunci pribadi, dan sertifikat root CA.

Important

Ini adalah satu-satunya waktu Anda dapat mengunduh sertifikat dan kunci pribadi Anda.


7. Untuk mengaktifkan sertifikat, pilih Aktifkan. Sertifikat harus aktif agar perangkat dapat terhubung AWS IoT.
8. Pilih Lampirkan kebijakan.
9. Untuk Tambahkan kebijakan untuk barang Anda, pilih MoistureSensorPolicy, lalu pilih Register Thing.

Langkah 3: Buat topik dan langganan Amazon SNS

Buat topik dan langganan Amazon SNS.

1. Dari [konsol AWS SNS](#), di panel navigasi, pilih Topik, lalu pilih Buat topik.

2. Pilih ketik sebagai Standar dan masukkan nama untuk topik (misalnya, **MoistureSensorTopic**).
3. Masukkan nama tampilan untuk topik (misalnya, **Moisture Sensor Topic**). Ini adalah nama yang ditampilkan untuk topik Anda di konsol Amazon SNS.
4. Pilih Buat topik.
5. Di halaman detail topik Amazon SNS, pilih Buat langganan.
6. Untuk Protokol, pilih Email.
7. Untuk Titik Akhir, masukkan alamat email Anda.
8. Pilih Buat langganan.
9. Buka klien email Anda dan cari pesan dengan subjek **MoistureSensorTopic**. Buka email dan klik tautan Konfirmasi berlangganan.

 Important

Anda tidak akan menerima peringatan email apa pun dari topik Amazon SNS ini sampai Anda mengonfirmasi langganan.

Anda harus menerima pesan email dengan teks yang Anda ketik.

Langkah 4: Buat AWS IoT aturan untuk mengirim email

AWS IoT Aturan mendefinisikan kueri dan satu atau beberapa tindakan yang harus diambil saat pesan diterima dari perangkat. Mesin AWS IoT aturan mendengarkan pesan yang dikirim oleh perangkat dan menggunakan data dalam pesan untuk menentukan apakah beberapa tindakan harus diambil. Untuk informasi selengkapnya, lihat [Aturan untuk AWS IoT](#).

Dalam tutorial ini, Raspberry Pi Anda menerbitkan pesan di `diaws/things/RaspberryPi/shadow/update`. Ini adalah topik MQTT internal yang digunakan oleh perangkat dan layanan Thing Shadow. Raspberry Pi menerbitkan pesan yang memiliki bentuk berikut:

```
{
  "reported": {
    "moisture" : moisture-reading,
    "temp" : temperature-reading
  }
}
```

Anda membuat kueri yang mengekstrak data kelembaban dan suhu dari pesan yang masuk. Anda juga membuat tindakan Amazon SNS yang mengambil data dan mengirimkannya ke pelanggan topik Amazon SNS jika pembacaan kelembaban di bawah nilai ambang batas.

Buat aturan Amazon SNS

1. Di [AWS IoT konsol](#), pilih Perutean pesan, lalu pilih Aturan. Jika kotak dialog Anda belum memiliki aturan apa pun muncul, pilih Buat aturan. Jika tidak, pilih Buat aturan.
2. Di halaman properti Aturan, masukkan nama Aturan seperti **MoistureSensorRule**, dan berikan deskripsi Aturan singkat seperti **Sends an alert when soil moisture level readings are too low**.
3. Pilih Berikutnya dan konfigurasi pernyataan SQL Anda. Pilih versi SQL sebagai 2016-03-23, dan masukkan pernyataan query SQL berikut: AWS IoT

```
SELECT * FROM '$aws/things/RaspberryPi/shadow/update/accepted' WHERE
state.reported.moisture < 400
```

Pernyataan ini memicu tindakan aturan ketika `moisture` pembacaan kurang dari 400.

Note

Anda mungkin harus menggunakan nilai yang berbeda. Setelah kode berjalan pada Raspberry Pi Anda, Anda dapat melihat nilai yang Anda dapatkan dari sensor Anda dengan menyentuh sensor, menemukannya di air, atau menemukannya di penanam.

4. Pilih Berikutnya dan lampirkan tindakan aturan. Untuk Tindakan 1, pilih Layanan Pemberitahuan Sederhana. Deskripsi untuk tindakan aturan ini adalah Kirim pesan sebagai pemberitahuan push SNS.
5. Untuk topik SNS, pilih topik yang Anda buat [Langkah 3: Buat topik dan langganan Amazon SNS](#) **MoistureSensorTopic**, dan biarkan format Pesan sebagai RAW. Untuk Peran IAM, pilih Buat Peran Baru. Masukkan nama untuk peran, misalnya **LowMoistureTopicRole**, lalu pilih Buat peran.
6. Pilih Berikutnya untuk meninjau dan kemudian pilih Buat untuk membuat aturan.

Menyiapkan Raspberry Pi dan sensor kelembaban

Masukkan kartu microSD Anda ke Raspberry Pi, sambungkan monitor, keyboard, mouse, dan, jika Anda tidak menggunakan Wi-Fi, kabel Ethernet. Jangan sambungkan kabel daya dulu.

Hubungkan kabel jumper JST ke sensor kelembaban. Sisi lain dari jumper memiliki empat kabel:

- Hijau: I2C SCL
- Putih: I2C SDA
- Merah: daya (3,5 V)
- Hitam: tanah

Pegang Raspberry Pi dengan jack Ethernet di sebelah kanan. Dalam orientasi ini, ada dua baris pin GPIO di bagian atas. Hubungkan kabel dari sensor kelembaban ke baris bawah pin dengan urutan berikut. Mulai dari pin paling kiri, sambungkan merah (power), putih (SDA), dan hijau (SCL). Lewati satu pin, lalu hubungkan kabel hitam (ground). Untuk informasi selengkapnya, lihat [Pengkabelan Komputer Python](#).

Pasang kabel daya ke Raspberry Pi dan colokkan ujung lainnya ke stopkontak untuk menyalakannya.

Konfigurasi Raspberry Pi Anda

1. Pada Selamat Datang di Raspberry Pi, pilih Berikutnya.
2. Pilih negara, bahasa, zona waktu, dan tata letak keyboard Anda. Pilih Berikutnya.
3. Masukkan kata sandi untuk Raspberry Pi Anda, lalu pilih Berikutnya.
4. Pilih jaringan Wi-Fi Anda, lalu pilih Berikutnya. Jika Anda tidak menggunakan jaringan Wi-Fi, pilih Lewati.
5. Pilih Berikutnya untuk memeriksa pembaruan perangkat lunak. Ketika pembaruan selesai, pilih Restart untuk memulai ulang Raspberry Pi Anda.

Setelah Raspberry Pi Anda dimulai, aktifkan antarmuka I2C.

1. Di sudut kiri atas desktop Raspbian, klik ikon Raspberry, pilih Preferences, dan kemudian pilih Raspberry Pi Configuration.
2. Pada tab Antarmuka, untuk I2C, pilih Aktifkan.
3. Pilih OK.

Perpustakaan untuk sensor kelembaban Adafruit STEMMA ditulis untuk. CircuitPython Untuk menjalankannya di Raspberry Pi, Anda perlu menginstal versi terbaru Python 3.

1. Jalankan perintah berikut dari prompt perintah untuk memperbarui perangkat lunak Raspberry Pi Anda:

```
sudo apt-get update
```

```
sudo apt-get upgrade
```

2. Jalankan perintah berikut untuk memperbarui instalasi Python 3 Anda:

```
sudo pip3 install --upgrade setuptools
```

3. Jalankan perintah berikut untuk menginstal perpustakaan Raspberry Pi GPIO:

```
pip3 install RPI.GPIO
```

4. Jalankan perintah berikut untuk menginstal perpustakaan Adafruit Blinka:

```
pip3 install adafruit-blinka
```

Untuk informasi selengkapnya, lihat [Menginstal CircuitPython Pustaka di Raspberry Pi](#).

5. Jalankan perintah berikut untuk menginstal perpustakaan Adafruit Seesaw:

```
sudo pip3 install adafruit-circuitpython-seesaw
```

6. Jalankan perintah berikut untuk menginstal AWS IoT Device SDK untuk Python:

```
pip3 install AWSIoTPythonSDK
```

Raspberry Pi Anda sekarang memiliki semua pustaka yang diperlukan. Buat file bernama **moistureSensor.py** dan salin kode Python berikut ke dalam file:

```
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)
```

Simpan file ke tempat Anda dapat menemukannya. Jalankan `moistureSensor.py` dari baris perintah dengan parameter berikut:

titik akhir

AWS IoT Titik akhir kustom Anda. Untuk informasi selengkapnya, lihat [Device Shadow REST API](#).

rootCA

Jalur lengkap ke sertifikat CA AWS IoT root Anda.

sertifikat

Jalur lengkap ke sertifikat AWS IoT perangkat Anda.

kunci

Jalur lengkap ke kunci pribadi sertifikat AWS IoT perangkat Anda.

thingName

Nama barang Anda (dalam hal ini,RaspberryPi).

clientId

ID klien MQTT. Gunakan RaspberryPi.

Baris perintah akan terlihat seperti ini:

```
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
```


Coba sentuh sensor, letakkan di penanam, atau masukkan ke dalam segelas air untuk melihat bagaimana sensor merespons berbagai tingkat kelembapan. Jika diperlukan, Anda dapat mengubah nilai ambang batas di `MoistureSensorRule`. Saat pembacaan sensor kelembapan berada di bawah nilai yang ditentukan dalam pernyataan kueri SQL aturan Anda, AWS IoT menerbitkan pesan ke topik Amazon SNS. Anda harus menerima pesan email yang berisi data kelembapan dan suhu.

Setelah Anda memverifikasi penerimaan pesan email dari Amazon SNS, tekan CTRL+C untuk menghentikan program Python. Tidak mungkin program Python akan mengirim pesan yang cukup untuk dikenakan biaya, tetapi ini adalah praktik terbaik untuk menghentikan program ketika Anda selesai.

Mengelola perangkat dengan AWS IoT

AWS IoT menyediakan registri yang membantu Anda mengelola berbagai hal. Sesuatu adalah representasi dari perangkat tertentu atau entitas logis. Ini bisa berupa perangkat fisik atau sensor (misalnya, bola lampu atau sakelar di dinding). Ini juga bisa menjadi entitas logis seperti contoh aplikasi atau entitas fisik yang tidak terhubung AWS IoT tetapi terkait dengan perangkat lain yang melakukannya (misalnya, mobil yang memiliki sensor mesin atau panel kontrol).

Informasi tentang sesuatu disimpan dalam registri sebagai data JSON. Berikut adalah contoh hal:

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

Hal-hal diidentifikasi dengan nama. Hal-hal juga dapat memiliki atribut, yang merupakan pasangan nama-nilai yang dapat Anda gunakan untuk menyimpan informasi tentang benda tersebut, seperti nomor seri atau pabrikannya.

Kasus penggunaan perangkat yang khas melibatkan penggunaan nama benda sebagai ID klien MQTT default. Meskipun kami tidak menerapkan pemetaan antara nama registri sesuatu dan penggunaan ID klien MQTT, sertifikat, atau status bayangan, kami sarankan Anda memilih nama sesuatu dan menggunakannya sebagai ID klien MQTT untuk registri dan layanan Device Shadow. Ini memberikan organisasi dan kenyamanan bagi armada IoT Anda tanpa menghilangkan fleksibilitas model atau bayangan sertifikat perangkat yang mendasarinya.

Anda tidak perlu membuat sesuatu di registri untuk menghubungkan perangkat AWS IoT. Menambahkan hal-hal ke registri memungkinkan Anda mengelola dan mencari perangkat dengan lebih mudah.

Cara mengelola sesuatu dengan registri

Anda menggunakan AWS IoT konsol, AWS IoT API, atau AWS CLI untuk berinteraksi dengan registri. Bagian berikut menunjukkan cara menggunakan CLI untuk bekerja dengan registri.

Saat menamai benda benda Anda:

- Jangan gunakan informasi yang dapat diidentifikasi secara pribadi dalam nama barang Anda. Nama benda dapat muncul dalam komunikasi dan laporan yang tidak terenkripsi.

Buat sesuatu

Perintah berikut menunjukkan cara menggunakan AWS IoT CreateThing perintah dari CLI untuk membuat sesuatu. Anda tidak dapat mengubah nama sesuatu setelah Anda membuatnya. Untuk mengubah nama sesuatu, buat hal baru, berikan nama baru, lalu hapus yang lama.

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

CreateThingPerintah menampilkan nama dan Amazon Resource Name (ARN) dari hal baru Anda:

```
{
  "thingArn": "arn:aws:iot:us-east-1:123456789012:thing/MyLightBulb",
  "thingName": "MyLightBulb",
  "thingId": "12345678abcdefghijklmnop12345678"
}
```

Note

Kami tidak menyarankan menggunakan informasi identitas pribadi dalam nama barang Anda.

Untuk informasi selengkapnya, lihat [create-thing](#) dari Command Reference. AWS CLI

Daftar hal-hal

Anda dapat menggunakan ListThings perintah untuk membuat daftar semua hal di akun Anda:

```
$ aws iot list-things
```

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

Anda dapat menggunakan ListThings perintah untuk mencari semua hal dari jenis hal tertentu:

```
$ 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"
    }
  ]
}
```

Anda dapat menggunakan ListThings perintah untuk mencari semua hal yang memiliki atribut dengan nilai tertentu. Perintah ini mencari hingga tiga atribut.

```
$ 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"
    }
  ]
}
```

Untuk informasi selengkapnya, lihat [daftar hal-hal dari Referensi](#) AWS CLI Perintah.

Jelaskan hal-hal

Anda dapat menggunakan DescribeThing perintah untuk menampilkan informasi lebih rinci tentang suatu hal:

```
$ 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"
  }
}
```

Untuk informasi selengkapnya, lihat [menjelaskan hal dari Referensi](#) Perintah. AWS CLI

Perbarui sesuatu

Anda dapat menggunakan UpdateThing perintah untuk memperbarui sesuatu. Perintah ini hanya memperbarui atribut benda itu. Anda tidak dapat mengubah nama. Untuk mengubah nama sesuatu, buat hal baru, berikan nama baru, lalu hapus yang lama.

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

UpdateThingPerintah tidak menghasilkan output. Anda dapat menggunakan DescribeThing perintah untuk melihat hasilnya:

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

```
"version": 2,  
"thingName": "MyLightBulb"  
}
```

Untuk informasi selengkapnya, lihat [update-thing](#) dari Command Reference. AWS CLI

Hapus sesuatu

Anda dapat menggunakan DeleteThing perintah untuk menghapus sesuatu:

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

Perintah ini berhasil kembali tanpa kesalahan jika penghapusan berhasil atau Anda menentukan hal yang tidak ada.

Untuk informasi selengkapnya, lihat [Hapus-hal dari Referensi](#) AWS CLI Perintah.

Lampirkan kepala sekolah pada suatu hal

Perangkat fisik harus memiliki sertifikat X.509 untuk berkomunikasi. AWS IoT Anda dapat mengaitkan sertifikat di perangkat Anda dengan benda di registri yang mewakili perangkat Anda. Untuk melampirkan sertifikat ke barang Anda, gunakan AttachThingPrincipal perintah:

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

AttachThingPrincipalPerintah tidak menghasilkan output apa pun.

Untuk informasi selengkapnya, lihat [attach-thing-principal](#) dari Command Reference. AWS CLI

Lepaskan kepala sekolah dari suatu hal

Anda dapat menggunakan DetachThingPrincipal perintah untuk melepaskan sertifikat dari suatu hal:

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

DetachThingPrincipalPerintah tidak menghasilkan output apa pun.

Untuk informasi selengkapnya, lihat [detach-thing-principal](#) dari Command Reference. AWS CLI

Jenis benda

Tipe hal memungkinkan Anda untuk menyimpan deskripsi dan informasi konfigurasi yang umum untuk semua hal yang berhubungan dengan tipe hal yang sama. Ini menyederhanakan pengelolaan hal-hal dalam registri. Misalnya, Anda dapat menentukan tipe LightBulb benda. Semua hal yang terkait dengan tipe LightBulb benda berbagi satu set atribut: nomor seri, pabrikan, dan watt. Saat Anda membuat sesuatu bertipe LightBulb (atau mengubah jenis benda yang ada menjadi LightBulb), Anda dapat menentukan nilai untuk setiap atribut yang ditentukan dalam tipe LightBulb benda.

Meskipun jenis benda bersifat opsional, penggunaannya membuatnya lebih mudah untuk menemukan sesuatu.

- Hal-hal dengan tipe hal dapat memiliki hingga 50 atribut.
- Hal-hal tanpa tipe hal dapat memiliki hingga tiga atribut.
- Suatu hal dapat dikaitkan dengan hanya satu tipe hal.
- Tidak ada batasan jumlah jenis barang yang dapat Anda buat di akun Anda.

Jenis benda tidak dapat diubah. Anda tidak dapat mengubah nama tipe benda setelah dibuat. Anda dapat menghentikan jenis barang kapan saja untuk mencegah hal-hal baru dikaitkan dengannya. Anda juga dapat menghapus jenis benda yang tidak memiliki hal-hal yang terkait dengannya.

Buat tipe benda

Anda dapat menggunakan CreateThingType perintah untuk membuat tipe sesuatu:

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

CreateThingTypePerintah mengembalikan respons yang berisi tipe benda dan ARN-nya:

```
{  
  "thingTypeName": "LightBulb",  
  "thingTypeId": "df9c2d8c-894d-46a9-8192-9068d01b2886",  
  "thingTypeArn": "arn:aws:iot:us-west-2:123456789012:thingtype/LightBulb"
```



```
}
```

Daftar jenis hal

Anda dapat menggunakan `ListThingTypes` perintah untuk membuat daftar jenis hal:

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

`ListThingTypesPerintah` mengembalikan daftar jenis hal yang didefinisikan dalam Akun AWS:

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

Jelaskan tipe benda

Anda dapat menggunakan `DescribeThingType` perintah untuk mendapatkan informasi tentang jenis benda:

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

`DescribeThingTypePerintah` mengembalikan informasi tentang jenis yang ditentukan:

```
{
  "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
}
}
```

Kaitkan tipe benda dengan sesuatu

Anda dapat menggunakan `CreateThing` perintah untuk menentukan jenis benda saat Anda membuat sesuatu:

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

Anda dapat menggunakan `UpdateThing` perintah kapan saja untuk mengubah jenis hal yang terkait dengan suatu hal:

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

Anda juga dapat menggunakan `UpdateThing` perintah untuk memisahkan sesuatu dari tipe benda.

Menghentikan tipe benda

Jenis benda tidak dapat diubah. Mereka tidak dapat diubah setelah didefinisikan. Namun, Anda dapat menghentikan tipe benda untuk mencegah pengguna mengaitkan hal-hal baru dengannya. Semua hal yang ada terkait dengan jenis benda tidak berubah.

Untuk menghentikan tipe benda, gunakan perintah: `DeprecateThingType`

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

Anda dapat menggunakan DescribeThingType perintah untuk melihat hasilnya:

```
$ 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
  }
}
```

Menghilangkan tipe benda adalah operasi yang dapat dibalik. Anda dapat membatalkan penghentian dengan menggunakan bendera dengan perintah --undo-deprecate CLI: DeprecateThingType

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

Anda dapat menggunakan perintah DescribeThingType CLI untuk melihat hasilnya:

```
$ 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,  
  }  
}
```

Hapus tipe benda

Anda dapat menghapus jenis benda hanya setelah mereka tidak digunakan lagi. Untuk menghapus tipe benda, gunakan DeleteThingType perintah:

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

Note

Sebelum Anda dapat menghapus jenis sesuatu, tunggu selama lima menit setelah Anda menghentikannya.

Kelompok benda statis

Kelompok benda statis memungkinkan Anda mengelola beberapa hal sekaligus dengan mengkategorikannya ke dalam kelompok. Grup benda statis berisi sekelompok hal yang dikelola dengan menggunakan konsol, CLI, atau API. [Grup benda dinamis](#), di sisi lain, berisi hal-hal yang cocok dengan kueri tertentu. Grup benda statis juga dapat berisi grup benda statis lainnya - Anda dapat membangun hierarki grup. Anda dapat melampirkan kebijakan ke grup induk dan itu diwarisi oleh grup turunannya, dan oleh semua hal dalam grup dan di grup anaknya. Ini membuat kontrol izin mudah untuk sejumlah besar hal.

Note

Kebijakan grup hal tidak mengizinkan akses ke operasi pesawat AWS IoT Greengrass data. Untuk mengizinkan akses sesuatu ke operasi pesawat AWS IoT Greengrass data, tambahkan izin ke AWS IoT kebijakan yang Anda lampirkan ke sertifikat benda tersebut. Untuk informasi selengkapnya, lihat [Otentikasi dan otorisasi perangkat](#) di panduan AWS IoT Greengrass pengembang.

Berikut adalah hal-hal yang dapat Anda lakukan dengan grup benda statis:

- Buat, jelaskan, atau hapus grup.
- Tambahkan sesuatu ke grup, atau ke lebih dari satu grup.
- Hapus sesuatu dari grup.
- Buat daftar grup yang telah Anda buat.
- Buat daftar semua kelompok anak dari suatu kelompok (keturunan langsung dan tidak langsungnya.)
- Buat daftar hal-hal dalam kelompok, termasuk semua hal dalam kelompok anaknya.
- Buat daftar semua kelompok leluhur dari suatu kelompok (orang tua langsung dan tidak langsungnya.)
- Tambahkan, hapus, atau perbarui atribut grup. (Atribut adalah pasangan nama-nilai yang dapat Anda gunakan untuk menyimpan informasi tentang grup.)
- Lampirkan atau lepaskan kebijakan ke atau dari grup.
- Buat daftar kebijakan yang dilampirkan ke grup.
- Buat daftar kebijakan yang diwarisi oleh suatu hal (berdasarkan kebijakan yang dilampirkan pada grupnya, atau salah satu kelompok induknya.)
- Konfigurasi opsi logging untuk hal-hal dalam grup. Lihat [Konfigurasi AWS IoT logging](#).
- Buat pekerjaan yang dikirim ke dan dieksekusi pada setiap hal dalam kelompok dan kelompok anaknya. Lihat [Tugas](#).

Note

Ketika sesuatu dilampirkan ke grup benda statis tempat AWS IoT Core kebijakan dilampirkan, nama benda harus cocok dengan ID klien.

Berikut adalah beberapa batasan kelompok benda statis:

- Sebuah kelompok dapat memiliki paling banyak satu orang tua langsung.
- Jika grup adalah anak dari grup lain, tentukan ini pada saat itu dibuat.
- Anda tidak dapat mengubah induk grup nanti, jadi pastikan untuk merencanakan hierarki grup Anda dan membuat grup induk sebelum Anda membuat grup anak yang dikandungnya.
-

Jumlah kelompok yang dapat dimiliki suatu benda [terbatas](#).

- Anda tidak dapat menambahkan sesuatu ke lebih dari satu grup dalam hierarki yang sama. (Dengan kata lain, Anda tidak dapat menambahkan apa pun ke dua grup yang memiliki induk yang sama.)
- Anda tidak dapat mengganti nama grup.
- Nama grup benda tidak dapat berisi karakter internasional, seperti û, é dan ñ.
- Jangan gunakan informasi yang dapat diidentifikasi secara pribadi dalam nama grup barang Anda. Nama grup benda dapat muncul dalam komunikasi dan laporan yang tidak terenkripsi.

Melampirkan dan melepaskan kebijakan ke grup dapat meningkatkan keamanan AWS IoT operasi Anda dalam beberapa cara yang signifikan. Metode per-perangkat untuk melampirkan kebijakan ke sertifikat, yang kemudian dilampirkan pada suatu hal, memakan waktu dan menyulitkan untuk memperbarui atau mengubah kebijakan dengan cepat di seluruh armada perangkat. Memiliki kebijakan yang dilampirkan pada grup benda menyimpan langkah-langkah ketika tiba saatnya untuk memutar sertifikat pada suatu hal. Dan kebijakan diterapkan secara dinamis pada berbagai hal ketika mereka mengubah keanggotaan grup, sehingga Anda tidak perlu membuat ulang kumpulan izin yang rumit setiap kali perangkat mengubah keanggotaan dalam grup.

Buat grup benda statis

Gunakan `CreateThingGroup` perintah untuk membuat grup benda statis:

```
$ aws iot create-thing-group --thing-group-name LightBulbs
```

`CreateThingGroup` perintah mengembalikan respon yang berisi nama grup benda statis, ID, dan ARN:

```
{
  "thingGroupName": "LightBulbs",
  "thingGroupId": "abcdefgh12345678ijklmnop12345678qrstuvwxyz",
  "thingGroupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/LightBulbs"
}
```

Note

Kami tidak menyarankan menggunakan informasi identitas pribadi dalam nama grup barang Anda.

Berikut adalah contoh yang menentukan induk dari kelompok hal statis ketika dibuat:

```
$ aws iot create-thing-group --thing-group-name RedLights --parent-group-name LightBulbs
```

Seperti sebelumnya, CreateThingGroup perintah mengembalikan respons yang berisi nama grup benda statis, ID, dan ARN:

```
{
  "thingGroupName": "RedLights",
  "thingGroupId": "abcdefgh12345678ijklmnop12345678qrstuvwxyz",
  "thingGroupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RedLights",
}
```

Important

Ingatlah batas-batas berikut saat membuat hierarki grup benda:

- Kelompok benda hanya dapat memiliki satu orang tua langsung.
- Jumlah kelompok anak langsung yang dapat dimiliki oleh suatu kelompok [terbatas](#).
- Kedalaman maksimum hierarki grup [terbatas](#).
- Jumlah atribut yang dapat dimiliki suatu kelompok [terbatas](#). (Atribut adalah pasangan nama-nilai yang dapat Anda gunakan untuk menyimpan informasi tentang grup.) Panjang setiap nama atribut dan setiap nilai juga [terbatas](#).

Jelaskan kelompok hal

Anda dapat menggunakan DescribeThingGroup perintah untuk mendapatkan informasi tentang grup hal:

```
$ aws iot describe-thing-group --thing-group-name RedLights
```

DescribeThingGroupPerintah mengembalikan informasi tentang kelompok tertentu:

```
{
  "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"
},
}

```

Tambahkan sesuatu ke grup benda statis

Anda dapat menggunakan `AddThingToThingGroup` perintah untuk menambahkan sesuatu ke grup benda statis:

```
$ aws iot add-thing-to-thing-group --thing-name MyLightBulb --thing-group-name
RedLights
```

`AddThingToThingGroup`Perintah tidak menghasilkan output apa pun.

Important

Anda dapat menambahkan sesuatu ke maksimal 10 grup. Tetapi Anda tidak dapat menambahkan sesuatu ke lebih dari satu grup dalam hierarki yang sama. (Dengan kata lain, Anda tidak dapat menambahkan sesuatu ke dua grup yang berbagi induk yang sama.)

Jika sesuatu milik sebanyak mungkin grup benda, dan satu atau lebih grup tersebut adalah grup benda dinamis, Anda dapat menggunakan [overrideDynamicGroups](#) bendera untuk membuat grup statis diprioritaskan daripada grup dinamis.

Hapus sesuatu dari grup benda statis

Anda dapat menggunakan `RemoveThingFromThingGroup` perintah untuk menghapus sesuatu dari grup:

```
$ aws iot remove-thing-from-thing-group --thing-name MyLightBulb --thing-group-name RedLights
```

`RemoveThingFromThingGroup`Perintah tidak menghasilkan output apa pun.

Daftar hal-hal dalam kelompok hal

Anda dapat menggunakan `ListThingsInThingGroup` perintah untuk membuat daftar hal-hal yang termasuk dalam grup:

```
$ aws iot list-things-in-thing-group --thing-group-name LightBulbs
```

`ListThingsInThingGroup`Perintah mengembalikan daftar hal-hal dalam kelompok yang diberikan:

```
{
  "things": [
    "TestThingA"
  ]
}
```

Dengan `--recursive` parameter, Anda dapat membuat daftar hal-hal milik grup dan yang ada di salah satu grup turunannya:

```
$ aws iot list-things-in-thing-group --thing-group-name LightBulbs --recursive
```

```
{
  "things": [
    "TestThingA",

```

```
    "MyLightBulb"  
  ]  
}
```

Note

Operasi ini [pada akhirnya konsisten](#). Dengan kata lain, perubahan pada kelompok benda mungkin tidak tercermin sekaligus.

Daftar kelompok hal

Anda dapat menggunakan `ListThingGroups` perintah untuk membuat daftar grup hal akun Anda:

```
$ aws iot list-thing-groups
```

`ListThingGroupsPerintah` mengembalikan daftar grup hal di Akun 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"  
    }  
  ]  
}
```

```
    }  
  ]  
}
```

Gunakan filter opsional untuk mencantumkan grup yang memiliki grup tertentu sebagai parent (`--parent-group`) atau grup yang namanya dimulai dengan awalan (`--name-prefix-filter`). `--recursive` parameter ini memungkinkan Anda untuk membuat daftar semua grup anak, bukan hanya grup anak langsung dari grup hal:

```
$ aws iot list-thing-groups --parent-group LightBulbs
```

Dalam hal ini, `ListThingGroups` perintah mengembalikan daftar grup anak langsung dari grup hal yang didefinisikan dalam Akun AWS:

```
{  
  "childGroups": [  
    {  
      "groupName": "RedLights",  
      "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RedLights"  
    }  
  ]  
}
```

Gunakan `--recursive` parameter dengan `ListThingGroups` perintah untuk mencantumkan semua grup anak dari grup benda, bukan hanya mengarahkan anak-anak:

```
$ aws iot list-thing-groups --parent-group LightBulbs --recursive
```

`ListThingGroups` perintah mengembalikan daftar semua kelompok anak dari grup hal:

```
{  
  "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

Operasi ini [pada akhirnya konsisten](#). Dengan kata lain, perubahan pada kelompok benda mungkin tidak tercermin sekaligus.

Daftar grup untuk suatu hal

Anda dapat menggunakan `ListThingGroupsForThing` perintah untuk membuat daftar grup langsung yang dimiliki suatu benda:

```
$ aws iot list-thing-groups-for-thing --thing-name MyLightBulb
```

`ListThingGroupsForThing` perintah mengembalikan daftar grup hal langsung yang menjadi milik benda ini:

```
{
  "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"
    }
  ]
}
```

```
}
```

Perbarui grup benda statis

Anda dapat menggunakan UpdateThingGroup perintah untuk memperbarui atribut grup benda statis:

```
$ aws iot update-thing-group --thing-group-name "LightBulbs" --thing-group-properties  
"thingGroupDescription=\"this is a test group\", attributePayload=\"{\n\"attributes  
\"={\n\"owner\"=\"150\", \"modelNames\"=\"456\"}\"}\"
```

UpdateThingGroupPerintah mengembalikan respons yang berisi nomor versi grup setelah pembaruan:

```
{  
  "version": 4  
}
```

Note

Jumlah atribut yang dapat dimiliki suatu benda [terbatas](#).

Hapus grup benda

Untuk menghapus grup sesuatu, gunakan DeleteThingGroup perintah:

```
$ aws iot delete-thing-group --thing-group-name "RedLights"
```

DeleteThingGroupPerintah tidak menghasilkan output apa pun.

Important

Jika Anda mencoba menghapus grup benda yang memiliki grup anak, Anda menerima kesalahan:

```
A client error (InvalidRequestException) occurred when calling the  
DeleteThingGroup  
operation: Cannot delete thing group : RedLights when there are still child  
groups attached to it.
```

Sebelum Anda menghapus grup, hapus grup anak apa pun terlebih dahulu.

Anda dapat menghapus grup yang memiliki item turunan, tetapi izin apa pun yang diberikan untuk hal-hal tersebut oleh keanggotaan dalam grup tidak lagi berlaku. Sebelum menghapus grup yang memiliki kebijakan terlampir, periksa dengan seksama bahwa menghapus izin tersebut tidak akan menghentikan hal-hal dalam grup agar tidak dapat berfungsi dengan baik. Selain itu, perintah yang menunjukkan grup mana yang dimiliki (misalnya, `ListGroupsForThing`) mungkin terus menampilkan grup saat catatan di cloud sedang diperbarui.

Lampirkan kebijakan ke grup benda statis

Anda dapat menggunakan `AttachPolicy` perintah untuk melampirkan kebijakan ke grup benda statis dan karenanya, dengan ekstensi, ke semua hal dalam grup itu dan hal-hal di salah satu grup turunannya:

```
$ aws iot attach-policy \  
  --target "arn:aws:iot:us-west-2:123456789012:thinggroup/LightBulbs" \  
  --policy-name "myLightBulbPolicy"
```

`AttachPolicy`Perintah tidak menghasilkan output apa pun

Important

Anda dapat melampirkan jumlah maksimum dua kebijakan ke grup.

Note

Kami tidak menyarankan menggunakan informasi identitas pribadi dalam nama kebijakan Anda.

--targetParameternya bisa berupa grup ARN (seperti di atas), sertifikat ARN, atau Identitas Amazon Cognito. Untuk informasi selengkapnya tentang kebijakan, sertifikat, dan otentikasi, lihat [Autentikasi](#).

Untuk informasi selengkapnya, lihat [AWS IoT Core kebijakan](#).

Lepaskan kebijakan dari grup benda statis

Anda dapat menggunakan `DetachPolicy` perintah untuk melepaskan kebijakan dari grup dan dengan demikian, dengan ekstensi, ke semua hal dalam grup itu dan hal-hal di salah satu grup turunannya:

```
$ aws iot detach-policy --target "arn:aws:iot:us-west-2:123456789012:thinggroup/LightBulbs" --policy-name "myLightBulbPolicy"
```

`DetachPolicy`Perintah tidak menghasilkan output apa pun.

Buat daftar kebijakan yang dilampirkan ke grup benda statis

Anda dapat menggunakan `ListAttachedPolicies` perintah untuk membuat daftar kebijakan yang dilampirkan ke grup hal statis:

```
$ aws iot list-attached-policies --target "arn:aws:iot:us-west-2:123456789012:thinggroup/RedLights"
```

`--target`Parameternya bisa berupa ARN grup benda (seperti di atas), sertifikat ARN, atau identitas Amazon Cognito.

Tambahkan `--recursive` parameter opsional untuk menyertakan semua kebijakan yang dilampirkan ke grup induk grup.

`ListAttachedPolicies`Perintah mengembalikan daftar kebijakan:

```
{
  "policies": [
    "MyLightBulbPolicy"
    ...
  ]
}
```

Buat daftar grup untuk kebijakan

Anda dapat menggunakan `ListTargetsForPolicy` perintah untuk mencantumkan target, termasuk grup apa pun, yang dilampirkan kebijakan:

```
$ aws iot list-targets-for-policy --policy-name "MyLightBulbPolicy"
```

Tambahkan `--page-size` *number* parameter opsional untuk menentukan jumlah maksimum hasil yang akan dikembalikan untuk setiap kueri, dan `--marker` *string* parameter pada panggilan berikutnya untuk mengambil set hasil berikutnya, jika ada.

ListTargetsForPolicyPerintah mengembalikan daftar target dan token yang akan digunakan untuk mengambil lebih banyak hasil:

```
{
  "nextMarker": "string",
  "targets": [ "string" ... ]
}
```

Dapatkan kebijakan yang efektif untuk suatu hal

Anda dapat menggunakan GetEffectivePolicies perintah untuk membuat daftar kebijakan yang berlaku untuk suatu hal, termasuk kebijakan yang dilampirkan ke grup mana pun yang menjadi miliknya (apakah grup tersebut adalah induk langsung atau leluhur tidak langsung):

```
$ aws iot get-effective-policies \
  --thing-name "MyLightBulb" \
  --principal "arn:aws:iot:us-east-1:123456789012:cert/
a0c01f5835079de0a7514643d68ef8414ab739a1e94ee4162977b02b12842847"
```

Gunakan `--principal` parameter untuk menentukan ARN dari sertifikat yang dilampirkan pada benda itu. Jika Anda menggunakan otentikasi identitas Amazon Cognito, gunakan `--cognito-identity-pool-id` parameter dan, secara opsional, tambahkan `--principal` parameter untuk menentukan identitas Amazon Cognito. Jika Anda hanya menentukan `--cognito-identity-pool-id`, kebijakan yang terkait dengan peran kumpulan identitas tersebut untuk pengguna yang tidak diautentikasi akan dikembalikan. Jika Anda menggunakan keduanya, kebijakan yang terkait dengan peran kumpulan identitas tersebut untuk pengguna yang diautentikasi akan dikembalikan.

`--thing-name` parameter opsional dan dapat digunakan sebagai pengganti `--principal` parameter. Saat digunakan, kebijakan yang dilampirkan ke grup mana pun yang menjadi miliknya, dan kebijakan yang dilampirkan ke grup induk mana pun dari grup ini (hingga grup root dalam hierarki) dikembalikan.

GetEffectivePoliciesPerintah mengembalikan daftar kebijakan:

```
{
```



```
"effectivePolicies": [  
  {  
    "policyArn": "string",  
    "policyDocument": "string",  
    "policyName": "string"  
  }  
  ...  
]  
}
```

Otorisasi uji untuk tindakan MQTT

Anda dapat menggunakan `TestAuthorization` perintah untuk menguji apakah tindakan [MQTT](#) (`Publish`, `Subscribe`) diizinkan untuk suatu hal:

```
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\"]}"
```

Gunakan `--principal` parameter untuk menentukan ARN dari sertifikat yang dilampirkan pada benda itu. Jika menggunakan autentikasi Identitas Amazon Cognito, tentukan Identitas Cognito sebagai `--principal` atau gunakan parameter, atau keduanya. `--cognito-identity-pool-id` (Jika Anda hanya menentukan kebijakan yang `--cognito-identity-pool-id` terkait dengan peran kumpulan identitas tersebut untuk pengguna yang tidak diautentikasi akan dipertimbangkan. Jika Anda menggunakan keduanya, kebijakan yang terkait dengan peran kumpulan identitas tersebut untuk pengguna yang diautentikasi akan dipertimbangkan.

Tentukan satu atau beberapa tindakan MQTT yang ingin Anda uji dengan mencantumkan kumpulan sumber daya dan jenis tindakan yang mengikuti parameter. `--auth-infos actionTypeBidang` harus berisi "PUBLISH", "SUBSCRIBE", "RECEIVE", atau "CONNECT". `resourcesBidang` harus berisi daftar ARN sumber daya. Untuk informasi selengkapnya, lihat [AWS IoT Core kebijakan](#).

Anda dapat menguji efek penambahan kebijakan dengan menentukannya dengan `--policy-names-to-add` parameter. Atau Anda dapat menguji efek penghapusan kebijakan oleh mereka dengan `--policy-names-to-skip` parameter.

Anda dapat menggunakan `--client-id` parameter opsional untuk lebih menyempurnakan hasil Anda.

TestAuthorizationPerintah mengembalikan detail tentang tindakan yang diizinkan atau ditolak untuk setiap kumpulan --auth-infos kueri yang Anda tentukan:

```
{
  "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" ]
    }
  ]
}
```

Kelompok hal yang dinamis

Grup benda dinamis dibuat dari kueri penelusuran tertentu di registri. Parameter kueri penelusuran seperti konektivitas perangkat, pembuatan bayangan perangkat, dan data AWS IoT Device Defender pelanggaran mendukung hal ini. Grup benda dinamis memerlukan pengindeksan armada yang diaktifkan untuk mengindeks, mencari, dan mengumpulkan data perangkat Anda. Anda dapat melihat pratinjau hal-hal dalam grup hal dinamis menggunakan kueri penelusuran pengindeksan armada sebelum membuatnya. Untuk informasi selengkapnya, lihat [Pengindeksan armada](#) dan [Sintaks kueri](#).

Note

Operasi grup benda dinamis diukur di bawah operasi registri. Untuk informasi selengkapnya, lihat [detail pengukuran AWS IoT Core tambahan](#).

Kelompok benda dinamis berbeda dari kelompok benda statis dengan cara berikut:

- Keanggotaan hal tidak didefinisikan secara eksplisit. Untuk membuat grup hal dinamis, tentukan [string kueri penelusuran](#) untuk menentukan keanggotaan grup.
- Kelompok benda dinamis tidak dapat menjadi bagian dari hierarki.
- Grup benda dinamis tidak dapat menerapkan kebijakan pada mereka.
- Anda menggunakan serangkaian perintah yang berbeda untuk membuat, memperbarui, dan menghapus grup hal dinamis. Untuk semua operasi lainnya, Anda menggunakan perintah yang sama untuk kedua jenis grup hal.
- Jumlah grup dinamis per Akun AWS [terbatas](#).
- Jangan gunakan informasi yang dapat diidentifikasi secara pribadi dalam nama grup barang Anda. Nama grup benda dapat muncul dalam komunikasi dan laporan yang tidak terenkripsi.

Untuk informasi selengkapnya tentang grup benda statis, lihat [Kelompok benda statis](#).

Gunakan kasus kelompok benda dinamis

Anda dapat menggunakan grup benda dinamis untuk kasus penggunaan berikut:

Tentukan grup benda dinamis sebagai target pekerjaan

Membuat pekerjaan berkelanjutan dengan kelompok hal dinamis sebagai target memungkinkan Anda untuk secara otomatis menargetkan perangkat ketika mereka memenuhi kriteria yang diinginkan. Kriteria dapat berupa status konektivitas atau kriteria apa pun yang disimpan dalam registri atau bayangan seperti versi perangkat lunak atau model. Jika sesuatu tidak muncul di grup hal dinamis, itu tidak akan menerima dokumen pekerjaan dari pekerjaan.

Misalnya, jika armada perangkat Anda memerlukan pembaruan firmware untuk meminimalkan risiko gangguan selama proses pembaruan, dan Anda hanya ingin memperbarui firmware pada perangkat dengan masa pakai baterai lebih dari 80%. Anda dapat membuat grup benda dinamis bernama `80PercentBatteryLife` yang hanya menyertakan perangkat dengan masa pakai baterai di atas 80% dan menggunakannya sebagai target untuk pekerjaan Anda. Hanya perangkat yang memenuhi kriteria masa pakai baterai Anda yang akan menerima pembaruan firmware. Saat perangkat mencapai kriteria masa pakai baterai 80%, mereka secara otomatis ditambahkan ke grup benda dinamis dan akan menerima pembaruan firmware.

Anda mungkin juga memiliki beberapa model perangkat dengan firmware atau sistem operasi yang berbeda, yang memerlukan versi pembaruan perangkat lunak baru yang berbeda. Ini adalah kasus penggunaan paling umum untuk grup dinamis dengan pekerjaan berkelanjutan, di mana Anda dapat membuat grup dinamis untuk setiap model perangkat, firmware, dan kombinasi OS. Anda kemudian dapat mengatur pekerjaan berkelanjutan untuk masing-masing grup dinamis ini untuk mendorong pembaruan perangkat lunak karena perangkat secara otomatis menjadi anggota grup ini berdasarkan kriteria yang ditentukan.

Untuk informasi selengkapnya tentang menentukan kelompok benda sebagai target pekerjaan, lihat [CreateJob](#).

Gunakan perubahan keanggotaan grup dinamis untuk melakukan tindakan yang diinginkan

Setiap kali perangkat ditambahkan atau dihapus dari grup hal dinamis, pemberitahuan dikirim ke topik MQTT sebagai bagian dari pembaruan acara [registri](#). Anda dapat mengonfigurasi [AWS IoT Core aturan](#) untuk berinteraksi dengan AWS layanan berdasarkan pembaruan keanggotaan grup dinamis dan mengambil tindakan yang diinginkan. Contoh tindakan termasuk menulis ke Amazon DynamoDB, memanggil fungsi Lambda, atau mengirim pemberitahuan ke Amazon SNS.

Tambahkan perangkat ke grup benda dinamis untuk deteksi pelanggaran otomatis

AWS IoT Device Defender Deteksi pelanggaran dapat menentukan [profil keamanan](#) pada grup hal dinamis. Perangkat grup benda dinamis secara otomatis terdeteksi untuk pelanggaran oleh profil keamanan yang ditentukan pada grup.

Tetapkan level log pada grup benda dinamis untuk mengamati perangkat dengan logging berbutir halus

Anda dapat menentukan tingkat log pada grup hal dinamis. Ini berguna jika Anda hanya ingin menyesuaikan tingkat logging dan detail untuk perangkat yang memenuhi kriteria tertentu. Misalnya, jika Anda mencurigai perangkat dengan versi firmware tertentu menyebabkan kesalahan pada topik yang dipublikasikan aturan tertentu, Anda mungkin ingin menyetel pencatatan terperinci untuk men-debug masalah ini. Dalam hal ini, Anda dapat membuat grup dinamis untuk semua perangkat yang memiliki versi firmware ini, yang kami asumsikan disimpan sebagai atribut registri atau dalam bayangan perangkat. Anda kemudian dapat mengatur tingkat debug, dengan target logging didefinisikan sebagai grup hal dinamis ini. Untuk informasi selengkapnya tentang logging berbutir halus, lihat [Memantau AWS IoT menggunakan Log. CloudWatch](#) Untuk informasi selengkapnya tentang cara menentukan tingkat logging untuk grup hal tertentu, lihat [Mengonfigurasi login khusus sumber daya](#). AWS IoT

Buat grup hal yang dinamis

Gunakan `CreateDynamicThingGroup` perintah untuk membuat grup benda dinamis. Untuk membuat grup benda dinamis untuk `PercentBatteryLife` skenario 80, gunakan perintah `create-dynamic-thing-group` CLI:

```
$ aws iot create-dynamic-thing-group --thing-group-name "80PercentBatteryLife" --query-string "attributes.batterylife80"
```

Note

Jangan gunakan informasi yang dapat diidentifikasi secara pribadi dalam nama grup benda dinamis Anda.

`CreateDynamicThingGroup`Perintah mengembalikan respons. Respons berisi nama indeks, string kueri, versi kueri, nama grup benda, ID grup benda, dan Nama Sumber Daya Amazon (ARN) dari grup hal Anda:

```
{
  "indexName": "AWS_Things",
  "queryVersion": "2017-09-30",
  "thingGroupName": "80PercentBatteryLife",
  "thingGroupArn": "arn:aws:iot:us-
west-2:123456789012:thinggroup/80PercentBatteryLife",
  "queryString": "attributes.batteryLife80\n",
  "thingGroupId": "abcdefgh12345678ijklmnop12345678qrstuvwxyz"
}
```

Penciptaan kelompok benda dinamis tidak terjadi sekaligus. Isi ulang grup hal dinamis membutuhkan waktu untuk diselesaikan. Saat Anda membuat grup benda dinamis, status grup diatur keBUILDING. Ketika isi ulang selesai, status berubah menjadiACTIVE. Untuk memeriksa status grup benda dinamis Anda, gunakan perintah [DescribeThingGroup](#).

Jelaskan kelompok hal yang dinamis

Gunakan DescribeThingGroup perintah untuk mendapatkan informasi tentang grup hal dinamis:

```
$ aws iot describe-thing-group --thing-group-name "80PercentBatteryLife"
```

DescribeThingGroupPerintah mengembalikan informasi tentang kelompok tertentu:

```
{
  "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"
}
```

Berjalan `DescribeThingGroup` pada grup hal dinamis mengembalikan atribut yang khusus untuk grup hal dinamis. Contoh atribut kembali adalah `QueryString` dan `status`.

Status grup benda dinamis dapat mengambil nilai-nilai berikut:

ACTIVE

Kelompok benda dinamis siap digunakan.

BUILDING

Grup hal dinamis sedang dibuat, dan keanggotaan sedang diproses.

REBUILDING

Keanggotaan grup hal dinamis sedang diperbarui, mengikuti penyesuaian kueri penelusuran grup.

Note

Setelah Anda membuat grup benda dinamis, gunakan itu terlepas dari statusnya. Hanya grup benda dinamis dengan `ACTIVE` status yang menyertakan semua hal yang cocok dengan kueri penelusuran untuk grup hal dinamis itu. Grup benda dinamis dengan `BUILDING` dan `REBUILDING` status mungkin tidak menyertakan semua hal yang cocok dengan kueri penelusuran.

Perbarui grup hal dinamis

Gunakan `UpdateDynamicThingGroup` perintah untuk memperbarui atribut grup benda dinamis, termasuk kueri penelusuran grup. Perintah berikut memperbarui dua atribut. Salah satunya adalah deskripsi grup benda, dan yang lainnya adalah string kueri yang mengubah kriteria keanggotaan menjadi masa pakai baterai > 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"
```

`UpdateDynamicThingGroup`Perintah mengembalikan respons yang berisi nomor versi grup setelah pembaruan:

```
{
```

```
"version": 2
}
```

Pembaruan grup hal dinamis tidak terjadi sekaligus. Isi ulang grup hal dinamis membutuhkan waktu untuk diselesaikan. Saat Anda memperbarui grup hal dinamis, status grup berubah menjadi REBUILDING saat grup memperbarui keanggotaannya. Ketika isi ulang selesai, status berubah menjadi ACTIVE. Untuk memeriksa status grup benda dinamis Anda, gunakan perintah [DescribeThingGroup](#).

Hapus grup benda dinamis

Gunakan `DeleteDynamicThingGroup` perintah untuk menghapus grup hal dinamis:

```
$ aws iot delete-dynamic-thing-group --thing-group-name "80PercentBatteryLife"
```

`DeleteDynamicThingGroup`Perintah tidak menghasilkan output apa pun.

Perintah yang menunjukkan grup mana yang dimiliki (misalnya, `ListGroupsForThing`) mungkin terus menampilkan grup saat catatan di cloud sedang diperbarui.

Batasan Grup Hal Dinamis dan Statis

Grup benda dinamis dan grup benda statis berbagi batasan berikut:

- Jumlah atribut yang dapat dimiliki kelompok benda [terbatas](#).
- Jumlah kelompok yang dapat dimiliki suatu benda [terbatas](#).
- Anda tidak dapat mengganti nama grup benda.
- Nama grup benda tidak dapat berisi karakter internasional, seperti û, é, dan ñ.

Batasan Grup Hal Dinamis

Kelompok benda dinamis memiliki keterbatasan sebagai berikut:

Pengindeksan armada

Dengan mengaktifkan layanan pengindeksan armada, Anda dapat melakukan kueri penelusuran di armada perangkat Anda. Anda dapat membuat dan mengelola grup benda dinamis setelah pengisian ulang pengindeksan armada selesai. Waktu penyelesaian untuk proses pengurukan langsung dipengaruhi oleh ukuran armada perangkat Anda yang terdaftar di AWS Cloud Setelah

Anda mengaktifkan layanan pengindeksan armada untuk grup hal dinamis, Anda tidak dapat menonaktifkannya sampai Anda menghapus semua grup benda dinamis Anda.

Note

Jika Anda memiliki izin untuk menanyakan indeks armada, Anda dapat mengakses data berbagai hal di seluruh armada.

Jumlah kelompok benda dinamis terbatas

Jumlah kelompok benda dinamis [terbatas](#).

Perintah yang berhasil dapat mencatat kesalahan

Saat Anda membuat atau memperbarui grup hal dinamis, ada kemungkinan beberapa hal memenuhi syarat untuk dimasukkan dalam grup hal dinamis, tetapi tidak ditambahkan ke dalamnya. [Skenario itu akan menyebabkan perintah buat atau perbarui yang berhasil saat mencatat kesalahan dan menghasilkan metrik. `AddThingToDynamicThingGroupsFailed`](#) Sebuah metrik tunggal dapat mewakili beberapa entri log.

[Entri log kesalahan](#) di CloudWatch log dibuat ketika hal berikut terjadi:

- Hal yang memenuhi syarat tidak dapat ditambahkan ke grup hal dinamis.
- Sesuatu dihapus dari grup benda dinamis untuk menambahnya ke grup lain.

Ketika sesuatu memenuhi syarat untuk ditambahkan ke grup hal dinamis, pertimbangkan hal berikut:

- Apakah hal itu sudah ada dalam kelompok sebanyak mungkin? (Lihat [batas](#))
 - TIDAK: Benda itu ditambahkan ke grup hal dinamis.
 - YA: Apakah benda itu anggota dari kelompok benda dinamis?
 - TIDAK: Masalahnya tidak dapat ditambahkan ke grup hal dinamis, kesalahan dicatat, dan [AddThingToDynamicThingGroupsFailedmetrik](#) dihasilkan.
 - YA: Apakah grup hal dinamis untuk bergabung lebih tua dari grup benda dinamis mana pun yang sudah menjadi anggotanya?
 - TIDAK: Masalahnya tidak dapat ditambahkan ke grup hal dinamis, kesalahan dicatat, dan [AddThingToDynamicThingGroupsFailedmetrik](#) dihasilkan.

- YA: Hapus benda itu dari grup hal dinamis terbaru, catat kesalahan, dan tambahkan benda itu ke grup benda dinamis. Ini menghasilkan kesalahan dan [AddThingToDynamicThingGroupsFailedmetrik](#) untuk grup hal dinamis dari mana benda itu dihapus.

Ketika sesuatu dalam grup hal dinamis tidak lagi memenuhi kueri penelusuran, benda itu dihapus dari grup hal dinamis. Demikian juga, ketika sesuatu diperbarui untuk memenuhi permintaan pencarian grup hal dinamis, hal itu kemudian ditambahkan ke grup seperti yang dijelaskan sebelumnya. Penambahan dan penghapusan ini normal dan tidak menghasilkan entri log kesalahan.

Dengan **overrideDynamicGroups** diaktifkan, grup statis lebih diprioritaskan daripada grup dinamis

Jumlah kelompok yang dapat dimiliki suatu benda [terbatas](#). Saat Anda menggunakan perintah [AddThingToThingGrup](#) atau [UpdateThingGroupsForBenda](#) untuk memperbarui keanggotaan sesuatu, menambahkan `--overrideDynamicGroups` parameter memberikan prioritas grup benda statis daripada grup hal dinamis.

Saat Anda menambahkan sesuatu ke grup benda statis, pertimbangkan hal berikut:

- Apakah benda itu sudah termasuk dalam jumlah kelompok maksimum?
 - TIDAK: Benda itu ditambahkan ke grup benda statis.
 - YA: Apakah ada dalam kelompok dinamis?
 - TIDAK: Benda itu tidak dapat ditambahkan ke grup benda. Perintah tersebut menimbulkan pengecualian.
 - YA: `--overrideDynamicGroups` Diaktifkan?
 - TIDAK: Benda itu tidak dapat ditambahkan ke grup benda. Perintah tersebut menimbulkan pengecualian.
 - YA: Masalahnya dihapus dari grup hal dinamis yang terbaru dibuat, kesalahan dicatat, dan [AddThingToDynamicThingGroupsFailedmetrik](#) dihasilkan untuk grup hal dinamis dari mana benda itu dihapus. Kemudian, benda itu ditambahkan ke grup benda statis.

Kelompok hal dinamis yang lebih lama diprioritaskan daripada yang lebih baru

Jumlah kelompok yang dapat dimiliki suatu benda [terbatas](#). Ketika operasi buat atau perbarui membuat kelayakan grup tambahan untuk suatu hal dan benda tersebut telah mencapai batas

grupnya, penghapusan dari grup hal dinamis lain dapat terjadi untuk mengaktifkan penambahan ini. Untuk informasi lebih lanjut tentang bagaimana hal ini terjadi, lihat [Perintah yang berhasil dapat mencatat kesalahan](#) dan [Dengan `overrideDynamicGroups` diaktifkan, grup statis lebih diprioritaskan daripada grup dinamis](#) untuk contoh.

Ketika sesuatu dihapus dari grup hal dinamis, kesalahan dicatat dan sebuah peristiwa dimunculkan.

Anda tidak dapat menerapkan kebijakan ke grup benda dinamis

Mencoba menerapkan kebijakan ke grup hal dinamis menghasilkan pengecualian.

Keanggotaan grup hal dinamis pada akhirnya konsisten

Hanya keadaan akhir dari suatu hal yang dievaluasi untuk registri. Status perantara dapat dilewati jika status diperbarui dengan cepat. Hindari mengaitkan aturan atau pekerjaan dengan kelompok hal dinamis yang keanggotaannya bergantung pada keadaan perantara.

Menandai sumber daya Anda AWS IoT

Untuk membantu Anda mengelola dan mengatur grup hal, jenis hal, aturan topik, pekerjaan, audit terjadwal, dan profil keamanan, Anda dapat secara opsional menetapkan metadata Anda sendiri ke masing-masing sumber daya ini dalam bentuk tag. Bagian ini menjelaskan tag dan menunjukkan cara membuatnya.

Untuk membantu Anda mengelola biaya yang terkait dengan hal-hal, Anda dapat membuat [grup penagihan](#) yang berisi hal-hal. Anda kemudian dapat menetapkan tag yang berisi metadata Anda ke masing-masing grup penagihan ini. Bagian ini juga membahas grup penagihan dan perintah yang tersedia untuk membuat dan mengelolanya.

Dasar-dasar tanda

Anda dapat menggunakan tag untuk mengkategorikan AWS IoT sumber daya Anda dengan cara yang berbeda (misalnya, berdasarkan tujuan, pemilik, atau lingkungan). Ini berguna ketika Anda memiliki banyak sumber daya dari jenis yang sama - Anda dapat dengan cepat mengidentifikasi sumber daya berdasarkan tag yang telah Anda tetapkan padanya. Setiap tanda terdiri dari kunci dan nilai opsional, yang keduanya Anda tentukan. Misalnya, Anda dapat menentukan satu set tag untuk jenis barang Anda yang membantu Anda melacak perangkat berdasarkan jenis. Kami menyarankan agar Anda merancang serangkaian kunci tanda yang memenuhi kebutuhan Anda untuk setiap jenis sumber daya. Penggunaan serangkaian kunci tanda akan mempermudah Anda dalam mengelola sumber daya Anda.

Anda dapat mencari dan memfilter sumber daya berdasarkan tag yang Anda tambahkan atau terapkan. Anda juga dapat menggunakan tag grup penagihan untuk mengkategorikan dan melacak biaya Anda. Anda juga dapat menggunakan tag untuk mengontrol akses ke sumber daya Anda seperti yang dijelaskan dalam [Menggunakan tanda dengan kebijakan IAM](#).

Untuk kemudahan penggunaan, Editor Tag di AWS Management Console menyediakan cara terpusat dan terpadu untuk membuat dan mengelola tag Anda. Untuk informasi selengkapnya, lihat [Bekerja dengan Editor Tag](#) di [Bekerja dengan Konsol AWS Manajemen](#).

Anda juga dapat bekerja dengan tag menggunakan AWS CLI dan AWS IoT API. Anda dapat mengaitkan tag dengan grup hal, jenis hal, aturan topik, pekerjaan, profil keamanan, kebijakan, grup penagihan, dan paket serta versi yang terkait dengan hal-hal saat Anda membuatnya dengan menggunakan Tags bidang dalam perintah berikut:

- [CreateBillingGroup](#)
- [CreateDestination](#)
- [CreateDeviceProfile](#)
- [CreateDynamicThingGroup](#)
- [CreateJob](#)
- [createotaupdate](#)
- [CreatePolicy](#)
- [CreateScheduledAudit](#)
- [CreateSecurityProfile](#)
- [CreateServiceProfile](#)
- [CreateStream](#)
- [CreateThingGroup](#)
- [CreateThingType](#)
- [CreateTopicRule](#)
- [CreateWirelessGateway](#)
- [CreateWirelessDevice](#)

Anda juga dapat menambahkan, mengubah, atau menghapus tanda untuk sumber daya yang sudah ada yang mendukung penandaan dengan menggunakan perintah berikut:

- [TagResource](#)
- [ListTagsForResource](#)
- [UntagResource](#)

Anda dapat mengedit kunci dan nilai tanda, dan dapat menghapus tanda dari sumber daya kapan saja. Anda dapat mengatur nilai tanda ke string kosong, tetapi tidak dapat mengatur nilai tanda ke null. Jika Anda menambahkan tanda yang memiliki kunci yang sama dengan tanda yang ada pada sumber daya tersebut, nilai yang baru akan menimpa nilai yang lama. Jika Anda menghapus sebuah sumber daya, semua tanda yang terkait dengan sumber daya tersebut juga dihapus.

Pembatasan dan batasan tanda

Batasan dasar berikut berlaku untuk tanda:

- Jumlah maksimum tag per sumber daya - 50
- Panjang kunci maksimum - 127 karakter Unicode di UTF-8
- Panjang nilai maksimum - 255 karakter Unicode di UTF-8
- Kunci dan nilai tag peka huruf besar-kecil.
- Jangan gunakan `aws:` awalan dalam nama atau nilai tag Anda. Ini dicadangkan untuk AWS digunakan. Anda tidak dapat mengedit atau menghapus nama atau nilai tanda dengan awalan ini. Tag dengan awalan ini tidak dihitung terhadap tag Anda per batas sumber daya.
- Jika skema penandaan Anda digunakan di beberapa layanan dan sumber daya, harap perhatikan bahwa layanan lain mungkin memiliki pembatasan pada karakter yang diizinkan. Karakter yang diizinkan termasuk huruf, spasi, dan angka yang dapat direpresentasikan dalam UTF-8, dan karakter khusus berikut: `+ - =. _:/@`.

Menggunakan tanda dengan kebijakan IAM

Anda dapat menerapkan izin tingkat sumber daya berbasis tag dalam kebijakan IAM yang Anda gunakan untuk tindakan API. AWS IoT Hal ini memberi Anda kontrol yang lebih baik atas sumber daya yang dapat dibuat, dimodifikasi, atau digunakan oleh pengguna. Anda menggunakan elemen `Condition` (juga disebut blok `Condition`) dengan kunci konteks syarat berikut dan nilai-nilai dalam kebijakan IAM untuk mengontrol akses pengguna (izin) berdasarkan tanda sumber daya:

- Gunakan `aws:ResourceTag/tag-key: tag-value` untuk mengizinkan atau menolak tindakan pengguna pada sumber daya dengan tag tertentu.
- Gunakan `aws:RequestTag/tag-key: tag-value` untuk mengharuskan tag tertentu digunakan (atau tidak digunakan) saat membuat permintaan API untuk membuat atau memodifikasi sumber daya yang memungkinkan tag.
- Gunakan `aws:TagKeys: [tag-key, ...]` untuk mengharuskan sekumpulan kunci tag tertentu digunakan (atau tidak digunakan) saat membuat permintaan API untuk membuat atau memodifikasi sumber daya yang memungkinkan tag.

Note

Kunci konteks kondisi dan nilai dalam kebijakan IAM hanya berlaku untuk AWS IoT tindakan tersebut di mana pengidentifikasi untuk sumber daya yang mampu diberi tag adalah parameter wajib. Misalnya, penggunaan tidak [DescribeEndpoint](#) diizinkan atau ditolak berdasarkan kunci dan nilai konteks kondisi karena tidak ada sumber daya yang dapat

diberi tag (grup benda, tipe benda, aturan topik, pekerjaan, atau profil keamanan) yang direferensikan dalam permintaan ini. Untuk informasi selengkapnya tentang AWS IoT sumber daya yang dapat diberi tag dan kunci kondisi yang didukungnya, baca [kunci Tindakan, sumber daya, dan kondisi](#). AWS IoT

Untuk informasi selengkapnya tentang penggunaan tag, lihat [Mengontrol Akses Menggunakan Tag](#) di Panduan AWS Identity and Access Management Pengguna. Bagian [Referensi Kebijakan IAM JSON](#) dari panduan itu memiliki sintaks terperinci, deskripsi, dan contoh elemen, variabel, dan logika evaluasi kebijakan JSON di IAM.

Kebijakan contoh berikut menerapkan dua batasan berbasis tag untuk ThingGroup tindakan tersebut. Pengguna IAM dibatasi oleh kebijakan ini:

- Tidak dapat membuat sesuatu mengelompokkan tag “env=prod” (dalam contoh, lihat baris).
"aws:RequestTag/env" : "prod"
- Tidak dapat memodifikasi atau mengakses grup benda yang memiliki tag “env=prod” yang ada (dalam contoh, lihat baris). "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": "*"
  }
]
```

Anda juga dapat menentukan beberapa nilai tag untuk kunci tag tertentu dengan melampirkannya dalam daftar, seperti ini:

```
    "StringEquals" : {
      "aws:ResourceTag/env" : ["dev", "test"]
    }
```

Note

Jika Anda mengizinkan atau menolak akses para pengguna ke sumber daya berdasarkan tanda, maka Anda harus mempertimbangkan untuk menolak secara eksplisit memberikan kemampuan kepada pengguna untuk menambahkan atau menghapus tanda tersebut dari sumber daya yang sama. Jika tidak, pengguna dapat mengakali pembatasan Anda dan mendapatkan akses atas sumber daya dengan melakukan modifikasi pada tanda dari sumber daya tersebut.

Grup penagihan

AWS IoT tidak memungkinkan Anda untuk secara langsung menerapkan tag ke hal-hal individual, tetapi itu memungkinkan Anda untuk menempatkan hal-hal dalam grup penagihan dan menerapkan tag untuk ini. Untuk AWS IoT, alokasi data biaya dan penggunaan berdasarkan tag terbatas pada grup penagihan.

AWS IoT Core untuk sumber daya LoRa WAN, seperti perangkat nirkabel dan gateway, tidak dapat ditambahkan ke grup penagihan. Namun, mereka dapat dikaitkan dengan AWS IoT hal-hal, yang dapat ditambahkan ke grup penagihan.

Perintah berikut tersedia:

- [AddThingToBillingGroup](#) menambahkan sesuatu ke grup penagihan.
- [CreateBillingGroup](#) membuat grup penagihan.
- [DeleteBillingGroup](#) menghapus grup penagihan.
- [DescribeBillingGroup](#) mengembalikan informasi tentang grup penagihan.
- [ListBillingGroups](#) daftar grup penagihan yang telah Anda buat.
- [ListThingsInBillingGroup](#) daftar hal-hal yang telah Anda tambahkan ke grup penagihan yang diberikan.
- [RemoveThingFromBillingGroup](#) menghapus hal yang diberikan dari grup penagihan.
- [UpdateBillingGroup](#) memperbarui informasi tentang grup penagihan.
- [CreateThing](#) memungkinkan Anda menentukan grup penagihan untuk hal tersebut saat Anda membuatnya.
- [DescribeThing](#) mengembalikan deskripsi sesuatu termasuk grup penagihan milik benda itu, jika ada.

AWS IoT Wireless API menyediakan tindakan ini untuk mengaitkan perangkat nirkabel dan gateway dengan AWS IoT berbagai hal.

- [AssociateWirelessDeviceWithThing](#)
- [AssociateWirelessGatewayWithThing](#)

Melihat alokasi biaya dan data penggunaan

Anda dapat menggunakan tag grup penagihan untuk mengkategorikan dan melacak biaya Anda. Saat Anda menerapkan tag ke grup penagihan (dan sebagainya untuk hal-hal yang disertakan), buat AWS laporan alokasi biaya sebagai file nilai dipisahkan koma (CSV) dengan penggunaan dan biaya yang dikumpulkan oleh tag Anda. Anda dapat menerapkan tanda yang mewakili kategori bisnis (seperti pusat biaya, nama aplikasi, atau pemilik) untuk mengatur biaya Anda di berbagai layanan. Untuk informasi selengkapnya tentang penggunaan tag untuk alokasi biaya, lihat [Menggunakan Tag Alokasi Biaya di Panduan Pengguna AWS Billing and Cost Management](#).

Note

Untuk mengaitkan data penggunaan dan biaya secara akurat dengan hal-hal yang telah Anda tempatkan dalam grup penagihan, setiap perangkat atau aplikasi harus:

- Terdaftar sebagai sesuatu di AWS IoT. Untuk informasi selengkapnya, lihat [Mengelola perangkat dengan AWS IoT](#).
- Connect ke broker AWS IoT pesan melalui MQTT hanya menggunakan nama benda sebagai ID klien. Untuk informasi selengkapnya, lihat [the section called “Protokol komunikasi perangkat”](#).
- Otentikasi menggunakan sertifikat klien yang terkait dengan hal tersebut.

Dimensi harga berikut tersedia untuk grup penagihan (berdasarkan aktivitas hal-hal yang terkait dengan grup penagihan):

- Konektivitas (berdasarkan nama benda yang digunakan sebagai ID klien untuk terhubung).
- Pesan (berdasarkan pesan masuk dari, dan keluar ke, sesuatu; hanya MQTT).
- Operasi bayangan (berdasarkan hal yang pesannya memicu pembaruan bayangan).
- Aturan dipicu (berdasarkan hal yang pesan masuknya memicu aturan; tidak berlaku untuk aturan yang dipicu oleh peristiwa siklus hidup MQTT).
- Pembaruan indeks hal (berdasarkan hal yang ditambahkan ke indeks).
- Tindakan jarak jauh (berdasarkan hal yang diperbarui).
- [AWS IoT Device Defender mendeteksi](#) laporan (berdasarkan hal yang aktivitasnya dilaporkan).

Data biaya dan penggunaan berdasarkan tag (dan dilaporkan untuk grup penagihan) tidak mencerminkan aktivitas berikut:

- Operasi registri perangkat (termasuk pembaruan untuk hal-hal, grup hal, dan jenis benda). Untuk informasi selengkapnya, lihat [Mengelola perangkat dengan AWS IoT](#).
- Pembaruan indeks grup benda (saat menambahkan grup sesuatu).
- Kueri pencarian indeks.
- [Penyediaan perangkat](#).
- AWS IoT Device Defender laporan [audit](#).

Keamanan di AWS IoT

Keamanan cloud di AWS adalah prioritas tertinggi. Sebagai AWS pelanggan, Anda mendapat manfaat dari pusat data dan arsitektur jaringan yang dibangun untuk memenuhi persyaratan organisasi yang paling sensitif terhadap keamanan.

Keamanan adalah tanggung jawab bersama antara Anda AWS dan Anda. [Model tanggung jawab bersama](#) menjelaskan hal ini sebagai keamanan cloud dan keamanan dalam cloud:

- Keamanan cloud — AWS bertanggung jawab untuk melindungi infrastruktur yang menjalankan AWS layanan di AWS Cloud. AWS juga memberi Anda layanan yang dapat Anda gunakan dengan aman. Auditor pihak ketiga secara berkala menguji dan memverifikasi efektivitas keamanan kami sebagai bagian dari [Program kepatuhan AWS](#). Untuk mempelajari tentang program kepatuhan yang berlaku AWS IoT, lihat [AWS Layanan dalam Lingkup berdasarkan Program Kepatuhan](#).
- Keamanan di cloud — Tanggung jawab Anda ditentukan oleh AWS layanan yang Anda gunakan. Anda juga bertanggung jawab atas faktor lain, termasuk sensitivitas data Anda, persyaratan perusahaan Anda, serta undang-undang dan peraturan yang berlaku.

Dokumentasi ini membantu Anda memahami cara menerapkan model tanggung jawab bersama saat menggunakan AWS IoT. Topik berikut menunjukkan cara mengonfigurasi AWS IoT untuk memenuhi tujuan keamanan dan kepatuhan Anda. Anda juga belajar cara menggunakan AWS layanan lain yang membantu Anda memantau dan mengamankan AWS IoT sumber daya Anda.

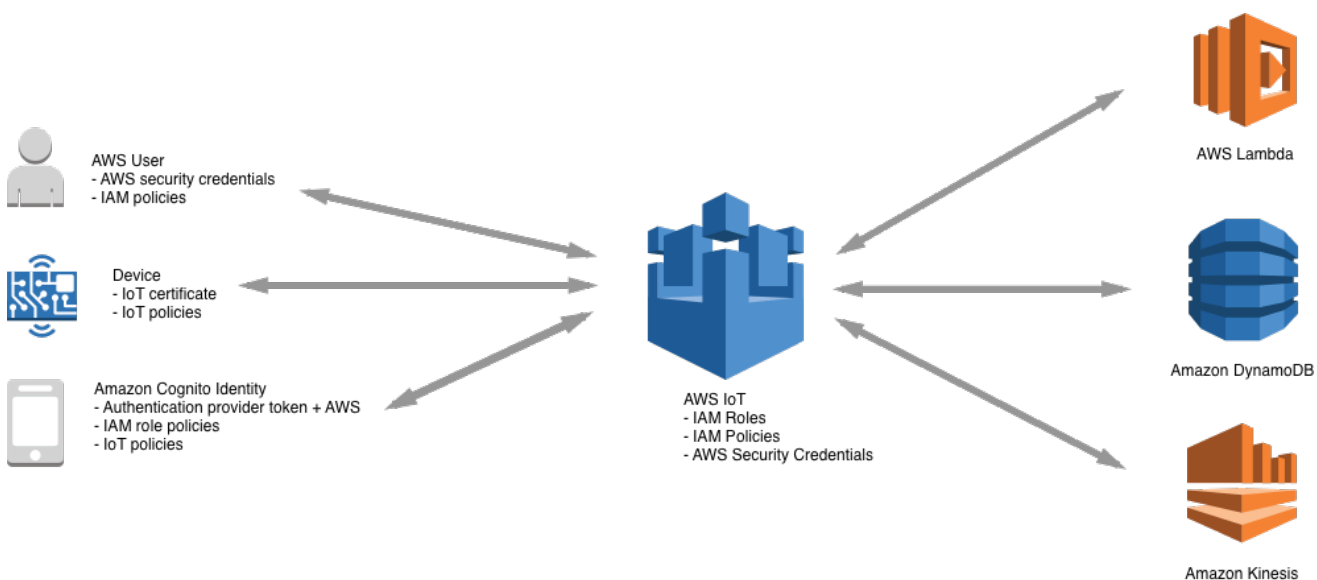
Topik

- [AWS IoT keamanan](#)
- [Autentikasi](#)
- [Otorisasi](#)
- [Perlindungan data di AWS IoT Core](#)
- [Identitas dan manajemen akses untuk AWS IoT](#)
- [Pembuatan Log dan Pemantauan](#)
- [Validasi kepatuhan untuk Core AWS IoT](#)
- [Ketahanan dalam IoT Core AWS](#)
- [Menggunakan AWS IoT Core dengan antarmuka VPC endpoint](#)
- [Keamanan infrastruktur di AWS IoT](#)

- [Pemantauan keamanan armada produksi atau perangkat dengan Core AWS IoT](#)
- [Praktik terbaik keamanan di AWS IoT Core](#)
- [AWS pelatihan dan sertifikasi](#)

AWS IoT keamanan

Setiap perangkat atau klien yang terhubung harus memiliki kredensi untuk berinteraksi AWS IoT. Semua lalu lintas ke dan dari AWS IoT dikirim dengan aman melalui Transport Layer Security (TLS). AWS Mekanisme keamanan cloud melindungi data saat bergerak antara AWS IoT dan AWS layanan lainnya.



- Anda bertanggung jawab untuk mengelola kredensial perangkat (sertifikat X.509, kredensial AWS, identitas Amazon Cognito, identitas federasi, atau token otentikasi kustom) dan kebijakan di AWS IoT. Untuk informasi selengkapnya, lihat [Manajemen kunci di AWS IoT](#). Anda bertanggung jawab untuk menetapkan identitas unik untuk setiap perangkat dan mengelola izin untuk setiap perangkat atau grup perangkat.
- Perangkat Anda terhubung AWS IoT menggunakan sertifikat X.509 atau identitas Amazon Cognito melalui koneksi TLS yang aman. Selama penelitian dan pengembangan, dan untuk beberapa aplikasi yang melakukan panggilan atau penggunaan API WebSockets, Anda juga dapat mengautentikasi menggunakan pengguna dan grup IAM atau token otentikasi khusus. Untuk informasi selengkapnya, lihat [Pengguna IAM, grup IAM, dan IAM role](#).

- Saat menggunakan AWS IoT otentikasi, broker pesan bertanggung jawab untuk mengautentikasi perangkat Anda, menyerap data perangkat dengan aman, dan memberikan atau menolak izin akses yang Anda tentukan untuk perangkat Anda menggunakan kebijakan. AWS IoT
- Saat menggunakan autentikasi kustom, otorisasi khusus bertanggung jawab untuk mengautentikasi perangkat Anda dan memberikan atau menolak izin akses yang Anda tentukan untuk perangkat yang Anda gunakan atau kebijakan IAM. AWS IoT
- Mesin AWS IoT aturan meneruskan data perangkat ke perangkat lain atau AWS layanan lain sesuai dengan aturan yang Anda tetapkan. Ini digunakan AWS Identity and Access Management untuk mentransfer data dengan aman ke tujuan akhirnya. Untuk informasi selengkapnya, lihat [Identitas dan manajemen akses untuk AWS IoT](#).

Autentikasi

Otentikasi adalah mekanisme di mana Anda memverifikasi identitas klien atau server. Otentikasi server adalah proses di mana perangkat atau klien lain memastikan mereka berkomunikasi dengan titik akhir yang sebenarnya AWS IoT . Otentikasi klien adalah proses di mana perangkat atau klien lain mengautentikasi diri mereka sendiri. AWS IoT

AWS pelatihan dan sertifikasi

Ikuti kursus berikut untuk mempelajari tentang otentikasi di AWS IoT: [Menyelam Jauh ke dalam AWS IoT Otentikasi dan Otorisasi](#).

Ikhtisar Sertifikat X.509

Sertifikat X.509 adalah sertifikat digital yang menggunakan [standar infrastruktur kunci publik X.509 untuk mengaitkan kunci](#) publik dengan identitas yang terkandung dalam sertifikat. Sertifikat X.509 dikeluarkan oleh entitas tepercaya yang disebut otoritas sertifikasi (CA). CA mempertahankan satu atau lebih sertifikat khusus yang disebut sertifikat CA yang digunakannya untuk mengeluarkan sertifikat X.509. Hanya otoritas sertifikasi yang memiliki akses ke sertifikat CA. Rantai sertifikat X.509 digunakan baik untuk otentikasi server oleh klien dan otentikasi klien oleh server.

Otentikasi server

Ketika perangkat Anda atau klien lain mencoba untuk terhubung AWS IoT Core, AWS IoT Core server akan mengirimkan sertifikat X.509 yang digunakan perangkat Anda untuk mengautentikasi server. Otentikasi berlangsung di lapisan TLS melalui validasi rantai sertifikat [X.509](#). Ini adalah

metode yang sama yang digunakan oleh browser Anda ketika Anda mengunjungi URL HTTPS. Jika Anda ingin menggunakan sertifikat dari otoritas sertifikat Anda sendiri, lihat [Kelola sertifikat CA Anda](#).

Saat perangkat Anda atau klien lain membuat koneksi TLS ke AWS IoT Core titik akhir, AWS IoT Core tunjukkan rantai sertifikat yang digunakan perangkat untuk memverifikasi bahwa mereka berkomunikasi AWS IoT Core dan bukan meniru server lain. AWS IoT Core Rantai yang disajikan tergantung pada kombinasi dari jenis titik akhir yang terhubung ke perangkat dan [cipher suite](#) yang klien dan AWS IoT Core dinegosiasikan selama jabat tangan TLS.

Jenis titik akhir

AWS IoT Core mendukung dua jenis titik akhir data yang berbeda, `iot:Data` dan `iot:Data-ATS`. `iot:Data` endpoint menyajikan sertifikat yang ditandatangani oleh sertifikat [CA root VeriSign Class 3 Public Primary G5](#). `iot:Data-ATS` endpoint menyajikan sertifikat server yang ditandatangani oleh [Amazon Trust Services](#) CA.

Sertifikat yang diberikan oleh titik akhir ATS ditandatangani silang oleh Starfield. Beberapa implementasi klien TLS memerlukan validasi akar kepercayaan dan mengharuskan sertifikat Starfield CA dipasang di toko kepercayaan klien.

Warning

Menggunakan metode penyematan sertifikat yang melakukan hash seluruh sertifikat (termasuk nama penerbit, dan sebagainya) tidak disarankan karena ini akan menyebabkan verifikasi sertifikat gagal karena sertifikat ATS yang kami berikan ditandatangani silang oleh Starfield dan memiliki nama penerbit yang berbeda.

Important

Gunakan `iot:Data-ATS` titik akhir kecuali perangkat Anda memerlukan sertifikat Symantec atau Verisign CA. Sertifikat Symantec dan Verisign telah usang dan tidak lagi didukung oleh sebagian besar browser web.

Anda dapat menggunakan `describe-endpoint` perintah untuk membuat titik akhir ATS Anda.

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

`describe-endpoint` Perintah mengembalikan endpoint dalam format berikut.

```
account-specific-prefix.iot.your-region.amazonaws.com
```

Note

Pertama kali `describe-endpoint` dipanggil, titik akhir dibuat. Semua panggilan berikutnya untuk `describe-endpoint` mengembalikan titik akhir yang sama.

Untuk kompatibilitas mundur, AWS IoT Core masih mendukung titik akhir Symantec. Untuk informasi selengkapnya, lihat [Bagaimana AWS IoT Core Membantu Pelanggan Menavigasi Ketidakpercayaan Mendatang terhadap Otoritas Sertifikat Symantec](#). Perangkat yang beroperasi pada titik akhir ATS sepenuhnya dapat dioperasikan dengan perangkat yang beroperasi pada titik akhir Symantec di akun yang sama dan tidak memerlukan pendaftaran ulang apa pun.

Note

Untuk melihat `iot:Data-ATS` titik akhir Anda di AWS IoT Core konsol, pilih Pengaturan. Konsol hanya menampilkan `iot:Data-ATS` titik akhir. Secara default, `describe-endpoint` perintah menampilkan `iot:Data` titik akhir untuk kompatibilitas mundur. Untuk melihat `iot:Data-ATS` titik akhir, tentukan `--endpointType` parameternya, seperti pada contoh sebelumnya.

Membuat **`IotDataPlaneClient`** dengan AWS SDK for Java

Secara default, [AWS SDK for Java - Versi 2](#) membuat `IotDataPlaneClient` dengan menggunakan `endpointiot:Data`. Untuk membuat klien yang menggunakan `iot:Data-ATS` endpoint, Anda harus melakukan hal berikut.

- Buat `iot:Data-ATS` titik akhir dengan menggunakan [DescribeEndpoint](#) API.
- Tentukan titik akhir itu saat Anda membuat `IotDataPlaneClient`

Contoh berikut melakukan kedua operasi ini.

```
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();
}
```

Sertifikat CA untuk otentikasi server

Bergantung pada jenis titik akhir data yang Anda gunakan dan rangkaian sandi mana yang telah Anda negosiasikan, sertifikat otentikasi AWS IoT Core server ditandatangani oleh salah satu sertifikat CA root berikut:

Titik Akhir Amazon Trust Services (lebih disukai)

Note

Anda mungkin perlu mengklik kanan tautan ini dan memilih Simpan tautan sebagai... untuk menyimpan sertifikat ini sebagai file.

- Kunci RSA 2048 bit: [Amazon Root CA 1](#)
- Kunci RSA 4096 bit: Amazon Root CA 2 Terpesan untuk digunakan di masa mendatang.
- Kunci ECC 256 bit: [Amazon Root CA 3](#).
- Kunci ECC 384 bit: Amazon Root CA 4 Terpesan untuk digunakan di masa mendatang.

Semua sertifikat ini ditandatangani silang oleh [Starfield Root CA](#) Certificate. Semua AWS IoT Core wilayah baru, dimulai dengan peluncuran 9 Mei 2018 AWS IoT Core di Wilayah Asia Pasifik (Mumbai), hanya melayani sertifikat ATS.

VeriSign Titik akhir (warisan)

- Kunci RSA 2048 bit: Sertifikat CA root [G5 Primer Publik VeriSign Kelas 3](#)

Pedoman otentikasi server

Ada banyak variabel yang dapat mempengaruhi kemampuan perangkat untuk memvalidasi sertifikat otentikasi AWS IoT Core server. Misalnya, perangkat mungkin terlalu dibatasi memori untuk menyimpan semua sertifikat CA root yang mungkin, atau perangkat dapat menerapkan metode validasi sertifikat non-standar. Untuk alasan ini kami sarankan mengikuti pedoman ini:

- Kami menyarankan Anda menggunakan titik akhir ATS dan menginstal semua Amazon Root CA sertifikat yang didukung.
- Jika Anda tidak dapat menyimpan semua sertifikat ini di perangkat Anda dan jika perangkat Anda tidak menggunakan validasi berbasis ECC, Anda dapat menghilangkan sertifikat dan ECC. [Amazon Root CA 3](#)[Amazon Root CA 4](#) Jika perangkat Anda tidak menerapkan validasi sertifikat berbasis RSA, Anda dapat menghilangkan sertifikat dan RSA. [Amazon Root CA 1](#)[Amazon Root CA 2](#) Anda mungkin perlu mengklik kanan tautan ini dan memilih Simpan tautan sebagai... untuk menyimpan sertifikat ini sebagai file.
- Jika Anda mengalami masalah validasi sertifikat server saat menyambung ke titik akhir ATS, coba tambahkan sertifikat Amazon Root CA bertanda tangan silang yang relevan ke toko kepercayaan Anda. Anda mungkin perlu mengklik kanan tautan ini dan memilih Simpan tautan sebagai... untuk menyimpan sertifikat ini sebagai file.
 - [Tanda tangan silang Amazon Root CA 1](#)
 - [Tanda tangan silang Amazon Root CA 2](#) - Dicadangkan untuk penggunaan di masa mendatang.
 - [Tanda tangan silang Amazon Root CA 3](#)
 - [Tanda tangan silang Amazon Root CA 4 - Dicadangkan untuk penggunaan di masa mendatang.](#)
- Jika Anda mengalami masalah validasi sertifikat server, perangkat Anda mungkin perlu mempercayai root CA secara eksplisit. Coba tambahkan [Starfield Root CA Certificate](#) ke toko kepercayaan Anda.
- Jika Anda masih mengalami masalah setelah menjalankan langkah-langkah di atas, silakan hubungi [Dukungan AWS Pengembang](#).

Note

Sertifikat CA memiliki tanggal kedaluwarsa setelah itu mereka tidak dapat digunakan untuk memvalidasi sertifikat server. Sertifikat CA mungkin harus diganti sebelum tanggal kedaluwarsa. Pastikan Anda dapat memperbarui sertifikat root CA di semua perangkat atau

klien Anda untuk membantu memastikan konektivitas yang berkelanjutan dan mengikuti perkembangan praktik terbaik keamanan.

Note

Saat menghubungkan ke AWS IoT Core dalam kode perangkat Anda, teruskan sertifikat ke API yang Anda gunakan untuk menyambungkan. API yang Anda gunakan akan bervariasi menurut SDK. Untuk informasi selengkapnya, lihat [SDK AWS IoT Core Perangkat](#).

Autentikasi Klien

AWS IoT mendukung tiga jenis prinsip identitas untuk otentikasi perangkat atau klien:

- [Sertifikat klien X.509](#)
- [Pengguna IAM, grup IAM, dan IAM role](#)
- [Identitas Amazon Cognito](#)

Identitas ini dapat digunakan dengan perangkat, seluler, web, atau aplikasi desktop. Mereka bahkan dapat digunakan oleh perintah user typing AWS IoT command line interface (CLI). Biasanya, AWS IoT perangkat menggunakan sertifikat X.509, sedangkan aplikasi seluler menggunakan identitas Amazon Cognito. Aplikasi web dan desktop menggunakan IAM atau identitas federasi. AWS CLI perintah menggunakan IAM. Untuk informasi lebih lanjut tentang identitas IAM, lihat [Identitas dan manajemen akses untuk AWS IoT](#)

Sertifikat klien X.509

Sertifikat X.509 menyediakan AWS IoT kemampuan untuk mengotentikasi koneksi klien dan perangkat. Sertifikat klien harus terdaftar AWS IoT sebelum klien dapat berkomunikasi dengan AWS IoT. Sertifikat klien dapat didaftarkan dalam beberapa Akun AWS detik yang sama Wilayah AWS untuk memfasilitasi pemindahan perangkat antara Akun AWS s Anda di wilayah yang sama. Untuk informasi selengkapnya, lihat [Menggunakan sertifikat klien X.509 dalam beberapa Akun AWS detik dengan pendaftaran multi-akun](#).

Kami menyarankan agar setiap perangkat atau klien diberikan sertifikat unik untuk mengaktifkan tindakan manajemen klien yang berbutir halus, termasuk pencabutan sertifikat. Perangkat dan klien

juga harus mendukung rotasi dan penggantian sertifikat untuk membantu memastikan kelancaran pengoperasian saat sertifikat kedaluwarsa.

Untuk informasi tentang penggunaan sertifikat X.509 untuk mendukung lebih dari beberapa perangkat, lihat [Penyediaan perangkat](#) untuk meninjau berbagai opsi manajemen dan penyediaan sertifikat yang mendukung. AWS IoT

AWS IoT mendukung jenis sertifikat klien X.509 ini:

- Sertifikat X.509 yang dihasilkan oleh AWS IoT
- Sertifikat X.509 ditandatangani oleh CA yang terdaftar dengan. AWS IoT
- Sertifikat X.509 ditandatangani oleh CA yang tidak terdaftar. AWS IoT

Bagian ini menjelaskan cara mengelola sertifikat X.509 di. AWS IoT Anda dapat menggunakan AWS IoT konsol atau AWS CLI untuk melakukan operasi sertifikat ini:

- [Buat sertifikat AWS IoT klien](#)
- [Buat sertifikat klien Anda sendiri](#)
- [Daftarkan sertifikat klien](#)
- [Mengaktifkan atau menonaktifkan sertifikat klien](#)
- [Mencabut sertifikat klien](#)

Untuk informasi selengkapnya tentang AWS CLI perintah yang melakukan operasi ini, lihat [AWS IoT Referensi CLI](#).

Menggunakan sertifikat klien X.509

Sertifikat X.509 mengotentikasi koneksi klien dan perangkat ke. AWS IoT Sertifikat X.509 memberikan beberapa manfaat dibandingkan mekanisme identifikasi dan otentikasi lainnya. Sertifikat X.509 memungkinkan kunci asimetris untuk digunakan dengan perangkat. Misalnya, Anda dapat membakar kunci pribadi ke dalam penyimpanan aman pada perangkat sehingga materi kriptografi sensitif tidak pernah meninggalkan perangkat. Sertifikat X.509 memberikan otentikasi klien yang lebih kuat atas skema lain, seperti nama pengguna dan kata sandi atau token pembawa, karena kunci pribadi tidak pernah meninggalkan perangkat.

AWS IoT mengotentikasi sertifikat klien menggunakan mode otentikasi klien protokol TLS. Dukungan TLS tersedia dalam banyak bahasa pemrograman dan sistem operasi dan umumnya digunakan

untuk mengenkripsi data. Dalam otentikasi klien TLS, AWS IoT meminta sertifikat klien X.509 dan memvalidasi status sertifikat dan Akun AWS terhadap registri sertifikat. Ini kemudian menantang klien untuk bukti kepemilikan kunci pribadi yang sesuai dengan kunci publik yang terkandung dalam sertifikat. AWS IoT mengharuskan klien untuk mengirim [ekstensi Server Name Indication \(SNI\) ke protokol](#) Transport Layer Security (TLS). Untuk informasi selengkapnya tentang mengonfigurasi ekstensi SNI, lihat. [Keamanan transportasi di AWS IoT Core](#)

Untuk memfasilitasi koneksi klien yang aman dan konsisten ke AWS IoT inti, sertifikat klien X.509 harus memiliki yang berikut:

- Terdaftar di AWS IoT Core. Untuk informasi selengkapnya, lihat [Daftarkan sertifikat klien](#).
- Memiliki status statusACTIVE. Untuk informasi selengkapnya, lihat [Mengaktifkan atau menonaktifkan sertifikat klien](#).
- Belum mencapai tanggal kedaluwarsa sertifikat.

Anda dapat membuat sertifikat klien yang menggunakan Amazon Root CA dan Anda dapat menggunakan sertifikat klien Anda sendiri yang ditandatangani oleh otoritas sertifikat (CA) lain. Untuk informasi selengkapnya tentang penggunaan AWS IoT konsol untuk membuat sertifikat yang menggunakan Amazon Root CA, lihat [Buat sertifikat AWS IoT klien](#). Untuk informasi selengkapnya tentang menggunakan sertifikat X.509 Anda sendiri, lihat. [Buat sertifikat klien Anda sendiri](#)

Tanggal dan waktu ketika sertifikat yang ditandatangani oleh sertifikat CA kedaluwarsa ditetapkan saat sertifikat dibuat. Sertifikat X.509 yang dihasilkan dengan AWS IoT kedaluwarsa pada tengah malam UTC pada tanggal 31 Desember 2049 (2049-12-31T 23:59:59 Z).

AWS IoT Device Defender dapat melakukan audit pada perangkat Anda Akun AWS dan perangkat yang mendukung praktik terbaik keamanan IoT umum. Ini termasuk mengelola tanggal kedaluwarsa sertifikat X.509 yang ditandatangani oleh CA Anda atau Amazon Root CA. Untuk informasi selengkapnya tentang mengelola tanggal kedaluwarsa sertifikat, lihat Sertifikat [perangkat kedaluwarsa dan sertifikat CA kedaluwarsa](#).

Di AWS IoT blog resmi, penyelaman lebih dalam tentang pengelolaan rotasi sertifikat perangkat dan praktik terbaik keamanan dieksplorasi di [Cara mengelola rotasi sertifikat perangkat IoT](#) menggunakan. AWS IoT

Menggunakan sertifikat klien X.509 dalam beberapa Akun AWS detik dengan pendaftaran multi-akun

Pendaftaran multi-akun memungkinkan untuk memindahkan perangkat antara Akun AWS s Anda di Wilayah yang sama atau di Wilayah yang berbeda. Anda dapat mendaftar, menguji, dan

mengonfigurasi perangkat di akun pra-produksi, lalu mendaftar dan menggunakan perangkat dan sertifikat perangkat yang sama di akun produksi. Anda juga dapat mendaftarkan sertifikat klien pada perangkat atau sertifikat perangkat tanpa CA yang terdaftar AWS IoT. Untuk informasi selengkapnya, lihat [Mendaftarkan sertifikat klien yang ditandatangani oleh CA \(CLI\) yang tidak terdaftar](#).

Note

Sertifikat yang digunakan untuk pendaftaran multi-akun didukung pada jenis `iot:Data-ATS`, `iot:Data` (warisan), `iot:Jobs`, dan titik akhir `iot:CredentialProvider`. Untuk informasi selengkapnya tentang titik akhir AWS IoT perangkat, lihat [AWS IoT data perangkat dan titik akhir layanan](#).

Perangkat yang menggunakan registrasi multi-akun harus mengirimkan [ekstensi Server Name Indication \(SNI\)](#) ke protokol Transport Layer Security (TLS) dan memberikan alamat endpoint lengkap di `host_name` lapangan, ketika mereka terhubung ke AWS IoT. AWS IoT menggunakan alamat titik akhir `host_name` untuk merutekan koneksi ke AWS IoT akun yang benar. Perangkat yang ada yang tidak mengirim alamat titik akhir yang valid `host_name` akan terus berfungsi, tetapi mereka tidak akan dapat menggunakan fitur yang memerlukan informasi ini. Untuk informasi lebih lanjut tentang ekstensi SNI dan untuk mempelajari cara mengidentifikasi alamat titik akhir untuk `host_name` bidang tersebut, lihat [Keamanan transportasi di AWS IoT Core](#)

Untuk menggunakan pendaftaran multi-akun

1. Anda dapat mendaftarkan sertifikat perangkat dengan CA. Anda dapat mendaftarkan CA penandatanganan dalam beberapa akun dalam `SNI_ONLY` mode dan menggunakan CA tersebut untuk mendaftarkan sertifikat klien yang sama ke beberapa akun. Untuk informasi selengkapnya, lihat [Daftarkan sertifikat CA dalam mode SNI_ONLY \(CLI\) - Direkomendasikan](#).
2. Anda dapat mendaftarkan sertifikat perangkat tanpa CA. Lihat [Daftarkan sertifikat klien yang ditandatangani oleh CA \(CLI\) yang tidak terdaftar](#). Mendaftarkan CA adalah opsional. Anda tidak diharuskan mendaftarkan CA yang menandatangani sertifikat perangkat AWS IoT.


Algoritma penandatanganan sertifikat didukung oleh AWS IoT

AWS IoT mendukung algoritma penandatanganan sertifikat berikut:

- SHA256WITHRSA
- SHA384WITHRSA

- SHA512WITHRSA
- SHA256WITHRSAANDMGF1 (RSASSA-PSS)
- SHA384WITHRSAANDMGF1 (RSASSA-PSS)
- SHA512WITHRSAANDMGF1 (RSASSA-PSS)
- DSA_DENGAN_SHA256
- ECDSA-DENGAN-SHA256
- ECDSA-DENGAN-SHA384
- ECDSA-DENGAN-SHA512

Untuk informasi selengkapnya tentang otentikasi dan keamanan sertifikat, lihat [Kualitas kunci sertifikat perangkat](#).

 Note

Permintaan penandatanganan sertifikat (CSR) harus menyertakan kunci publik. Kunci dapat berupa kunci RSA dengan panjang setidaknya 2.048 bit atau kunci ECC dari kurva NIST P-256, NIST P-384, atau NIST P-521. Untuk informasi selengkapnya, lihat [CreateCertificateFromCsrdi](#) Panduan Referensi AWS IoT API.

Algoritma kunci yang didukung oleh AWS IoT

Tabel di bawah ini menunjukkan bagaimana algoritma kunci didukung:

Algoritma kunci	Algoritma penandatanganan sertifikat	Versi TLS	Didukung? Ya atau Tidak
RSA dengan ukuran kunci minimal 2048 bit	Semua	TLS 1.2 TLS 1.3	Ya
ECC NIST P-256/P-384/P-521	Semua	TLS 1.2 TLS 1.3	Ya
RSA-PSS dengan ukuran kunci minimal 2048 bit	Semua	TLS 1.2	Tidak

Algoritma kunci	Algoritma penandatanganan sertifikat	Versi TLS	Didukung? Ya atau Tidak
RSA-PSS dengan ukuran kunci minimal 2048 bit	Semua	TLS 1.3	Ya

Untuk membuat sertifikat menggunakan [CreateCertificateFromCSR](#), Anda dapat menggunakan algoritma kunci yang didukung untuk menghasilkan kunci publik untuk CSR Anda. Untuk mendaftarkan sertifikat Anda sendiri menggunakan [RegisterCertificate](#) atau [RegisterCertificate tanpa CA](#), Anda dapat menggunakan algoritme kunci yang didukung untuk menghasilkan kunci publik untuk sertifikat.

Untuk informasi selengkapnya, lihat [Kebijakan keamanan](#).

Buat sertifikat AWS IoT klien

AWS IoT menyediakan sertifikat klien yang ditandatangani oleh otoritas sertifikat Amazon Root (CA).

Topik ini menjelaskan cara membuat sertifikat klien yang ditandatangani oleh otoritas sertifikat Amazon Root dan mengunduh file sertifikat. Setelah Anda membuat file sertifikat klien, Anda harus menginstalnya pada klien.

Note

Setiap sertifikat klien X.509 yang disediakan oleh AWS IoT memiliki atribut penerbit dan subjek yang Anda tetapkan pada saat pembuatan sertifikat. Atribut sertifikat tidak dapat diubah hanya setelah sertifikat dibuat.

Anda dapat menggunakan AWS IoT konsol atau AWS CLI untuk membuat AWS IoT sertifikat yang ditandatangani oleh otoritas sertifikat Amazon Root.

Buat AWS IoT sertifikat (konsol)

Untuk membuat AWS IoT sertifikat menggunakan AWS IoT konsol

1. Masuk ke AWS Management Console dan buka [AWS IoT konsol](#).

2. Di panel navigasi, pilih Keamanan, lalu pilih Sertifikat, lalu pilih Buat.
3. Pilih Pembuatan sertifikat sekali klik (disarankan) - Buat sertifikat.
4. Dari halaman yang dibuat Sertifikat, unduh file sertifikat klien untuk benda tersebut, kunci publik, dan kunci pribadi ke lokasi yang aman. Sertifikat yang dihasilkan oleh ini hanya AWS IoT tersedia untuk digunakan dengan AWS IoT layanan.

Jika Anda juga memerlukan file sertifikat Amazon Root CA, halaman ini juga memiliki tautan ke halaman tempat Anda dapat mengunduhnya.

5. Sertifikat klien sekarang telah dibuat dan didaftarkan AWS IoT. Anda harus mengaktifkan sertifikat sebelum menggunakannya di klien.

Untuk mengaktifkan sertifikat klien sekarang, pilih Aktifkan. Jika Anda tidak ingin mengaktifkan sertifikat sekarang, lihat [Aktifkan sertifikat klien \(konsol\)](#) untuk mempelajari cara mengaktifkan sertifikat nanti.

6. Jika Anda ingin melampirkan kebijakan ke sertifikat, pilih Lampirkan kebijakan.

Jika Anda tidak ingin melampirkan kebijakan sekarang, pilih Selesai untuk menyelesaikan. Anda dapat melampirkan kebijakan nanti.

Setelah Anda menyelesaikan prosedur, instal file sertifikat pada klien.

Buat AWS IoT sertifikat (CLI)

AWS CLI Ini menyediakan [create-keys-and-certificate](#) perintah untuk membuat sertifikat klien yang ditandatangani oleh otoritas sertifikat Amazon Root. Perintah ini, bagaimanapun, tidak mengunduh file sertifikat Amazon Root CA. Anda dapat mengunduh file sertifikat Amazon Root CA dari file [Sertifikat CA untuk otentikasi server](#).

Perintah ini membuat kunci pribadi, kunci publik, dan file sertifikat X.509 dan mendaftarkan dan mengaktifkan sertifikat dengan AWS IoT

```
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
```

Jika Anda tidak ingin mengaktifkan sertifikat saat Anda membuat dan mendaftarkannya, perintah ini membuat kunci pribadi, kunci publik, dan file sertifikat X.509 dan mendaftarkan sertifikat, tetapi tidak mengaktifkannya. [Aktifkan sertifikat klien \(CLI\)](#) menjelaskan cara mengaktifkan sertifikat nanti.

```
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
```

Instal file sertifikat pada klien.

Buat sertifikat klien Anda sendiri

AWS IoT mendukung sertifikat klien yang ditandatangani oleh otoritas sertifikat root atau menengah (CA). AWS IoT menggunakan sertifikat CA untuk memverifikasi kepemilikan sertifikat. Untuk menggunakan sertifikat perangkat yang ditandatangani oleh CA yang bukan CA Amazon, sertifikat CA harus terdaftar AWS IoT sehingga kami dapat memverifikasi kepemilikan sertifikat perangkat.

AWS IoT mendukung berbagai cara untuk membawa sertifikat Anda sendiri (BYOC):

- Pertama, daftarkan CA yang digunakan untuk menandatangani sertifikat klien dan kemudian mendaftarkan sertifikat klien individu. Jika Anda ingin mendaftarkan perangkat atau klien ke sertifikat kliennya saat pertama kali terhubung ke AWS IoT (juga dikenal sebagai [Penyediaan Just-in-Time](#)), Anda harus mendaftarkan CA penandatanganan dan mengaktifkan pendaftaran otomatis. AWS IoT
- Jika Anda tidak dapat mendaftarkan CA penandatanganan, Anda dapat memilih untuk mendaftarkan sertifikat klien tanpa CA. Untuk perangkat yang terdaftar tanpa CA, Anda harus menunjukkan [Server Name Indication \(SNI\)](#) saat Anda AWS IoT menghubungkannya.

Note

Untuk mendaftarkan sertifikat klien menggunakan CA, Anda harus mendaftarkan CA penandatanganan dengan AWS IoT, bukan CA lain dalam hierarki.

Note

Sertifikat CA dapat didaftarkan dalam DEFAULT mode hanya dengan satu akun di Wilayah.
Sertifikat CA dapat didaftarkan dalam SNI_ONLY mode oleh beberapa akun di Wilayah.

Untuk informasi selengkapnya tentang penggunaan sertifikat X.509 untuk mendukung lebih dari beberapa perangkat, lihat [Penyediaan perangkat](#) untuk meninjau berbagai opsi manajemen dan penyediaan sertifikat yang mendukung. AWS IoT

Topik

- [Kelola sertifikat CA Anda](#)
- [Buat sertifikat klien menggunakan sertifikat CA Anda](#)

Kelola sertifikat CA Anda

Bagian ini menjelaskan tugas-tugas umum untuk mengelola sertifikat otoritas sertifikat (CA) Anda sendiri.

Anda dapat mendaftarkan otoritas sertifikat (CA) AWS IoT jika Anda menggunakan sertifikat klien yang ditandatangani oleh CA yang AWS IoT tidak mengenali.

Jika Anda ingin klien mendaftarkan sertifikat klien mereka secara otomatis AWS IoT ketika mereka pertama kali terhubung, CA yang menandatangani sertifikat klien harus terdaftar AWS IoT. Jika tidak, Anda tidak perlu mendaftarkan sertifikat CA yang menandatangani sertifikat klien.

Note

Sertifikat CA dapat didaftarkan dalam DEFAULT mode hanya dengan satu akun di Wilayah.
Sertifikat CA dapat didaftarkan dalam SNI_ONLY mode oleh beberapa akun di Wilayah.

Topik:

- [Buat sertifikat CA](#)
- [Daftarkan sertifikat CA Anda](#)
- [Nonaktifkan sertifikat CA](#)

Buat sertifikat CA

Jika Anda tidak memiliki sertifikat CA, Anda dapat menggunakan alat [OpenSSL v1.1.1i](#) untuk membuatnya.

Note

Anda tidak dapat melakukan prosedur ini di AWS IoT konsol.

Untuk membuat sertifikat CA menggunakan alat [OpenSSL v1.1.1i](#)

1. Hasilkan key pair.

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

2. Gunakan kunci pribadi dari key pair untuk menghasilkan sertifikat CA.

```
openssl req -x509 -new -nodes \  
-key root_CA_key_filename.key \  
-sha256 -days 1024 \  
-out root_CA_cert_filename.pem
```

Daftarkan sertifikat CA Anda

Prosedur ini menjelaskan cara mendaftarkan sertifikat dari otoritas sertifikat (CA) yang bukan CA Amazon. AWS IoT Core menggunakan sertifikat CA untuk memverifikasi kepemilikan sertifikat. Untuk menggunakan sertifikat perangkat yang ditandatangani oleh CA yang bukan CA Amazon, Anda harus mendaftarkan sertifikat CA AWS IoT Core sehingga dapat memverifikasi kepemilikan sertifikat perangkat.

Daftarkan sertifikat CA (konsol)

Note

Untuk mendaftarkan sertifikat CA di konsol, mulai di konsol di [Daftar sertifikat CA](#). Anda dapat mendaftarkan CA Anda dalam mode Multi-akun dan tanpa perlu memberikan sertifikat verifikasi atau akses ke kunci pribadi. CA dapat didaftarkan dalam mode Multi-akun dengan beberapa akun Akun AWS yang sama Wilayah AWS. Anda dapat mendaftarkan CA Anda

dalam mode Single-account dengan memberikan sertifikat verifikasi dan bukti kepemilikan kunci pribadi CA.

Daftarkan sertifikat CA (CLI)

Anda dapat mendaftarkan sertifikat CA dalam DEFAULT mode atau SNI_ONLY mode. CA dapat didaftarkan dalam DEFAULT mode per Akun AWS satu Wilayah AWS. CA dapat didaftarkan dalam SNI_ONLY mode dengan beberapa Akun AWS dalam mode yang sama Wilayah AWS. Untuk informasi selengkapnya tentang mode sertifikat CA, lihat [CertificateMode](#).

Note

Kami menyarankan Anda mendaftarkan CA dalam SNI_ONLY mode. Anda tidak perlu memberikan sertifikat verifikasi atau akses ke kunci pribadi, dan Anda dapat mendaftarkan CA dengan beberapa Akun AWS di yang sama Wilayah AWS.

Daftarkan sertifikat CA dalam mode SNI_ONLY (CLI) - Direkomendasikan

Prasyarat

Pastikan Anda memiliki yang berikut ini tersedia di komputer Anda sebelum melanjutkan:

- File sertifikat root CA (direferensikan dalam contoh berikut sebagai *root_CA_cert_filename.pem*)
- [OpenSSL v1.1.1i](#) atau yang lebih baru

Untuk mendaftarkan sertifikat CA dalam **SNI_ONLY** mode menggunakan AWS CLI

1. Daftarkan sertifikat CA dengan AWS IoT. Menggunakan `register-ca-certificate` perintah, masukkan nama file sertifikat CA. Untuk informasi selengkapnya, lihat [register-ca-certificate](#) di Command Reference.AWS CLI

```
aws iot register-ca-certificate \  
  --ca-certificate file://root_CA_cert_filename.pem \  
  --certificate-mode SNI_ONLY
```

Jika berhasil, perintah ini mengembalikan *CertificateId*.

2. Pada titik ini, sertifikat CA telah terdaftar AWS IoT tetapi tidak aktif. Sertifikat CA harus aktif sebelum Anda dapat mendaftarkan sertifikat klien apa pun yang telah ditandatangani.

Langkah ini mengaktifkan sertifikat CA.

Untuk mengaktifkan sertifikat CA, gunakan `update-certificate` perintah sebagai berikut. Untuk informasi selengkapnya, lihat [sertifikat pembaruan di Referensi Perintah.AWS CLI](#)

```
aws iot update-ca-certificate \  
  --certificate-id certificateId \  
  --new-status ACTIVE
```

Untuk melihat status sertifikat CA, gunakan `describe-ca-certificate` perintah. Untuk informasi selengkapnya, lihat [describe-ca-certificate](#) di Command Reference.AWS CLI

Daftarkan sertifikat CA dalam **DEFAULT** mode (CLI)

Prasyarat

Pastikan Anda memiliki yang berikut ini tersedia di komputer Anda sebelum melanjutkan:

- File sertifikat root CA (direferensikan dalam contoh berikut sebagai *root_CA_cert_filename.pem*)
- File kunci pribadi sertifikat CA root (direferensikan dalam contoh berikut sebagai *root_CA_key_filename.key*)
- [OpenSSL v1.1.1i](#) atau yang lebih baru

Untuk mendaftarkan sertifikat CA dalam **DEFAULT** mode menggunakan AWS CLI

1. Untuk mendapatkan kode registrasi dari AWS IoT, gunakan `get-registration-code`. Simpan yang dikembalikan `registrationCode` untuk digunakan sebagai sertifikat verifikasi kunci pribadi. Common Name Untuk informasi selengkapnya, lihat [get-registration-code](#) di Command Reference.AWS CLI

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

2. Buat key pair untuk sertifikat verifikasi kunci pribadi:

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

3. Buat permintaan penandatanganan sertifikat (CSR) untuk sertifikat verifikasi kunci pribadi. Atur Common Name bidang sertifikat ke yang registrationCode dikembalikan oleh get-registration-code.

```
openssl req -new \  
-key verification_cert_key_filename.key \  
-out verification_cert_csr_filename.csr
```

Anda diminta untuk beberapa informasi, termasuk Common Name untuk sertifikat.

```
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. Gunakan CSR untuk membuat sertifikat verifikasi kunci pribadi:

```
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. Daftarkan sertifikat CA dengan AWS IoT. Masukkan nama file sertifikat CA dan nama file sertifikat verifikasi kunci pribadi ke register-ca-certificate perintah, sebagai berikut. Untuk informasi selengkapnya, lihat [register-ca-certificate](#) di Command Reference.AWS CLI

```
aws iot register-ca-certificate \  
  --ca-certificate file://root_CA_cert_filename.pem \  
  --verification-cert file://verification_cert_filename.pem
```

Perintah ini mengembalikan *CertificateId*, jika berhasil.

6. Pada titik ini, sertifikat CA telah terdaftar AWS IoT tetapi tidak aktif. Sertifikat CA harus aktif sebelum Anda dapat mendaftarkan sertifikat klien yang telah ditandatangani.

Langkah ini mengaktifkan sertifikat CA.

Untuk mengaktifkan sertifikat CA, gunakan update-certificate perintah sebagai berikut. Untuk informasi selengkapnya, lihat [sertifikat pembaruan di Referensi Perintah.AWS CLI](#)

```
aws iot update-ca-certificate \  
  --certificate-id certificateId \  
  --new-status ACTIVE
```

Untuk melihat status sertifikat CA, gunakan describe-ca-certificate perintah. Untuk informasi selengkapnya, lihat [describe-ca-certificate](#) di Command Reference.AWS CLI

Membuat sertifikat verifikasi CA untuk mendaftarkan sertifikat CA di konsol

Note

Prosedur ini hanya untuk digunakan jika Anda mendaftarkan sertifikat CA dari AWS IoT konsol.

Jika Anda tidak datang ke prosedur ini dari AWS IoT konsol, mulai proses pendaftaran sertifikat CA di konsol di [Daftar sertifikat CA](#).

Pastikan Anda memiliki yang berikut ini tersedia di komputer yang sama sebelum melanjutkan:

- File sertifikat root CA (direferensikan dalam contoh berikut sebagai `root_CA_cert_filename.pem`)
- File kunci pribadi sertifikat CA root (direferensikan dalam contoh berikut sebagai `root_CA_key_filename.key`)
- [OpenSSL v1.1.1i](#) atau yang lebih baru

Untuk menggunakan antarmuka baris perintah untuk membuat sertifikat verifikasi CA untuk mendaftarkan sertifikat CA Anda di konsol

1. Ganti `verification_cert_key_filename.key` dengan nama file kunci sertifikat verifikasi yang ingin Anda buat (misalnya, `verification_cert.key`). Kemudian jalankan perintah ini untuk menghasilkan key pair untuk sertifikat verifikasi kunci pribadi:

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

2. Ganti `verification_cert_key_filename.key` dengan nama file kunci yang Anda buat di langkah 1.

Ganti `verification_cert_csr_filename.csr` dengan nama file permintaan penandatanganan sertifikat (CSR) yang ingin Anda buat. Misalnya, `verification_cert.csr`.

Jalankan perintah ini untuk membuat file CSR.

```
openssl req -new \  
-key verification_cert_key_filename.key \  
-out verification_cert_csr_filename.csr
```

Perintah meminta Anda untuk informasi tambahan yang dijelaskan nanti.

3. Di AWS IoT konsol, dalam wadah sertifikat Verifikasi, salin kode registrasi.
4. Informasi yang diminta openssl perintah Anda ditampilkan dalam contoh berikut. Kecuali untuk Common Name bidang, Anda dapat memasukkan nilai Anda sendiri atau mengosongkannya.

Di Common Name bidang, tempel kode registrasi yang Anda salin di langkah sebelumnya.

```
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 []:

Setelah Anda selesai, perintah membuat file CSR.

5. Ganti *verification_cert_csr_filename.csr* dengan yang *verification_cert_csr_filename.csr* Anda gunakan pada langkah sebelumnya.

Ganti *root_CA_cert_filename.pem* dengan nama file sertifikat CA yang ingin Anda daftarkan.

Ganti *root_CA_key_filename.key* dengan nama file file kunci pribadi sertifikat CA.

Ganti *verification_cert_filename.pem* dengan nama file sertifikat verifikasi yang ingin Anda buat. Misalnya, **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. Setelah perintah OpenSSL selesai, Anda harus memiliki file-file ini siap digunakan ketika Anda kembali ke konsol.
 - File sertifikat CA Anda (*root_CA_cert_filename.pem*) digunakan dalam perintah sebelumnya)

- Sertifikat verifikasi yang Anda buat pada langkah sebelumnya (*verification_cert_filename.pem* digunakan dalam perintah sebelumnya)

Nonaktifkan sertifikat CA

Ketika sertifikat otoritas sertifikat (CA) diaktifkan untuk pendaftaran sertifikat klien otomatis, AWS IoT memeriksa sertifikat CA untuk memastikan CA tersebut **ACTIVE**. Jika sertifikat CA adalah **INACTIVE**, AWS IoT tidak mengizinkan sertifikat klien untuk didaftarkan.

Dengan menyetel sertifikat CA **INACTIVE**, Anda mencegah sertifikat klien baru yang dikeluarkan oleh CA agar tidak terdaftar secara otomatis.

Note

Setiap sertifikat klien terdaftar yang ditandatangani oleh sertifikat CA yang dikompromikan terus berfungsi sampai Anda secara eksplisit mencabut masing-masing sertifikat tersebut.

Nonaktifkan sertifikat CA (konsol)

Untuk menonaktifkan sertifikat CA menggunakan konsol AWS IoT

1. Masuk ke AWS Management Console dan buka [AWS IoT konsol](#).
2. Di panel navigasi kiri, pilih Aman, pilih CA.
3. Dalam daftar otoritas sertifikat, temukan yang ingin Anda nonaktifkan, dan pilih ikon elipsis untuk membuka menu opsi.
4. Pada menu opsi, pilih Nonaktifkan.

Otoritas sertifikat harus ditampilkan sebagai Tidak Aktif dalam daftar.

Note

AWS IoT Konsol tidak menyediakan cara untuk mencantumkan sertifikat yang ditandatangani oleh CA yang Anda nonaktifkan. Untuk AWS CLI opsi untuk mencantumkan sertifikat tersebut, lihat [Nonaktifkan sertifikat CA \(CLI\)](#).

Nonaktifkan sertifikat CA (CLI)

AWS CLI Ini menyediakan [update-ca-certificate](#) perintah untuk menonaktifkan sertifikat CA.

```
aws iot update-ca-certificate \  
  --certificate-id certificateId \  
  --new-status INACTIVE
```

Gunakan [list-certificates-by-ca](#) perintah untuk mendapatkan daftar semua sertifikat klien terdaftar yang ditandatangani oleh CA yang ditentukan. Untuk setiap sertifikat klien yang ditandatangani oleh sertifikat CA yang ditentukan, gunakan [update-certificate](#) perintah untuk mencabut sertifikat klien agar tidak digunakan.

Gunakan [describe-ca-certificate](#) perintah untuk melihat status sertifikat CA.

Buat sertifikat klien menggunakan sertifikat CA Anda

Anda dapat menggunakan otoritas sertifikat (CA) Anda sendiri untuk membuat sertifikat klien. Sertifikat klien harus terdaftar AWS IoT sebelum digunakan. Untuk informasi tentang opsi pendaftaran untuk sertifikat klien Anda, lihat [Daftarkan sertifikat klien](#).

Buat sertifikat klien (CLI)

Note

Anda tidak dapat melakukan prosedur ini di AWS IoT konsol.

Untuk membuat sertifikat klien menggunakan AWS CLI

1. Hasilkan key pair.

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

2. Buat CSR untuk sertifikat klien.

```
openssl req -new \  
  -key device_cert_key_filename.key \  
  -out device_cert_csr_filename.csr
```

Anda diminta untuk beberapa informasi, seperti yang ditunjukkan di sini:

```
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. Buat sertifikat klien dari 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
```

Pada titik ini, sertifikat klien telah dibuat, tetapi belum terdaftar AWS IoT. Untuk informasi tentang bagaimana dan kapan mendaftarkan sertifikat klien, lihat [Daftarkan sertifikat klien](#).

Daftarkan sertifikat klien

Sertifikat klien harus terdaftar AWS IoT untuk memungkinkan komunikasi antara klien dan AWS IoT. Anda dapat mendaftarkan setiap sertifikat klien secara manual, atau Anda dapat mengonfigurasi sertifikat klien untuk mendaftar secara otomatis ketika klien terhubung AWS IoT untuk pertama kalinya.

Jika Anda ingin klien dan perangkat Anda mendaftarkan sertifikat klien mereka ketika mereka pertama kali terhubung, Anda harus [Daftarkan sertifikat CA Anda](#) menggunakan untuk menandatangani sertifikat klien dengan AWS IoT di Wilayah di mana Anda ingin menggunakannya. Amazon Root CA secara otomatis terdaftar AWS IoT.

Sertifikat klien dapat dibagikan oleh Akun AWS dan Wilayah. Prosedur dalam topik ini harus dilakukan di setiap akun dan Wilayah di mana Anda ingin menggunakan sertifikat klien. Pendaftaran sertifikat klien di satu akun atau Wilayah tidak secara otomatis dikenali oleh yang lain.

Note

Klien yang menggunakan protokol Transport Layer Security (TLS) untuk terhubung AWS IoT harus mendukung [ekstensi Server Name Indication \(SNI\)](#) ke TLS. Untuk informasi selengkapnya, lihat [Keamanan transportasi di AWS IoT Core](#).

Topik

- [Daftarkan sertifikat klien secara manual](#)
- [Daftarkan sertifikat klien saat klien terhubung ke AWS IoT just-in-time registrasi \(JITR\)](#)

Daftarkan sertifikat klien secara manual

Anda dapat mendaftarkan sertifikat klien secara manual dengan menggunakan AWS IoT konsol dan AWS CLI.

Prosedur pendaftaran yang digunakan tergantung pada apakah sertifikat akan dibagikan oleh Akun AWS s dan Wilayah. Pendaftaran sertifikat klien di satu akun atau Wilayah tidak secara otomatis dikenali oleh yang lain.

Prosedur dalam topik ini harus dilakukan di setiap akun dan Wilayah di mana Anda ingin menggunakan sertifikat klien. Sertifikat klien dapat dibagikan oleh Akun AWS s dan Wilayah.

Daftarkan sertifikat klien yang ditandatangani oleh CA (konsol) terdaftar

Note


Sebelum Anda melakukan prosedur ini, pastikan bahwa Anda memiliki file.pem sertifikat klien dan bahwa sertifikat klien ditandatangani oleh CA yang telah Anda [daftarkan](#). AWS IoT

Untuk mendaftarkan sertifikat yang ada dengan AWS IoT menggunakan konsol

1. Masuk ke Konsol AWS Manajemen dan buka [AWS IoT konsol](#).
2. Di panel navigasi, di bawah bagian Kelola, pilih Keamanan, lalu pilih Sertifikat.
3. Pada halaman Sertifikat di kotak dialog Sertifikat, pilih Tambahkan sertifikat, lalu pilih Daftar sertifikat.
4. Pada halaman Daftar sertifikat di kotak dialog Sertifikat untuk diunggah, lakukan hal berikut:
 - Pilih CA terdaftar AWS IoT.
 - Dari Pilih sertifikat CA, pilih otoritas Sertifikasi Anda.
 - Pilih Daftarkan CA baru untuk mendaftarkan otoritas Sertifikasi baru yang tidak terdaftar AWS IoT.
 - Biarkan Pilih sertifikat CA kosong jika otoritas sertifikat Amazon Root adalah otoritas sertifikasi Anda.
 - Pilih hingga 10 sertifikat untuk diunggah dan didaftarkan AWS IoT.
 - Gunakan file sertifikat yang Anda buat [Buat sertifikat AWS IoT klien](#) dan [Buat sertifikat klien menggunakan sertifikat CA Anda](#).
 - Pilih Aktifkan atau Nonaktifkan. Jika Anda memilih Deaktif, [Mengaktifkan atau menonaktifkan sertifikat klien](#) jelaskan cara mengaktifkan sertifikat Anda setelah pendaftaran sertifikat.
 - Pilih Pendaftaran.

Pada halaman Sertifikat di kotak dialog Sertifikat, sertifikat terdaftar Anda sekarang akan muncul.

Daftarkan sertifikat klien yang ditandatangani oleh CA (konsol) yang tidak terdaftar

 Note

Sebelum Anda melakukan prosedur ini, pastikan bahwa Anda memiliki file .pem sertifikat klien.

Untuk mendaftarkan sertifikat yang ada dengan AWS IoT menggunakan konsol

1. Masuk ke Konsol AWS Manajemen dan buka [AWS IoT konsol](#).
2. Di panel navigasi kiri, pilih Aman, pilih Sertifikat, lalu pilih Buat.
3. Pada Buat sertifikat, cari entri Gunakan sertifikat saya, dan pilih Memulai.

4. Pada Pilih CA, pilih Berikutnya.
5. Pada Daftarkan sertifikat perangkat yang ada, pilih Pilih sertifikat, dan pilih hingga 10 file sertifikat untuk didaftarkan.
6. Setelah menutup kotak dialog file, pilih apakah Anda ingin mengaktifkan atau mencabut sertifikat klien saat Anda mendaftarkannya.

Jika Anda tidak mengaktifkan sertifikat saat terdaftar, [Aktifkan sertifikat klien \(konsol\)](#) jelaskan cara mengaktifkannya nanti.

Jika sertifikat dicabut saat terdaftar, sertifikat tersebut tidak dapat diaktifkan nanti.

Setelah Anda memilih file sertifikat untuk didaftarkan, dan pilih tindakan yang akan diambil setelah pendaftaran, pilih Daftarkan sertifikat.

Sertifikat klien yang terdaftar berhasil muncul dalam daftar sertifikat.

Daftarkan sertifikat klien yang ditandatangani oleh CA terdaftar (CLI)

Note

Sebelum Anda melakukan prosedur ini, pastikan bahwa Anda memiliki sertifikat otoritas (CA) .pem dan berkas .pem sertifikat klien. Sertifikat klien harus ditandatangani oleh otoritas sertifikat (CA) yang telah Anda [daftarkan AWS IoT](#).

Gunakan [register-certificate](#) perintah untuk mendaftar, tetapi tidak mengaktifkan, sertifikat klien.

```
aws iot register-certificate \  
  --certificate-pem file://device_cert_filename.pem \  
  --ca-certificate-pem file://ca_cert_filename.pem
```

Sertifikat klien terdaftar AWS IoT, tetapi belum aktif. Lihat [Aktifkan sertifikat klien \(CLI\)](#) untuk informasi tentang cara mengaktifkannya nanti.

Anda juga dapat mengaktifkan sertifikat klien ketika Anda mendaftarkannya dengan menggunakan perintah ini.

```
aws iot register-certificate \  
  --set-as-active \  
  --certificate-pem file://device_cert_filename.pem \  
  --ca-certificate-pem file://ca_cert_filename.pem
```



```
--certificate-pem file://device_cert_filename.pem \  
--ca-certificate-pem file://ca_cert_filename.pem
```

Untuk informasi selengkapnya tentang mengaktifkan sertifikat sehingga dapat digunakan untuk terhubung AWS IoT, lihat [Mengaktifkan atau menonaktifkan sertifikat klien](#)

Daftarkan sertifikat klien yang ditandatangani oleh CA (CLI) yang tidak terdaftar

Note

Sebelum Anda melakukan prosedur ini, pastikan Anda memiliki file.pem sertifikat.

Gunakan [register-certificate-without-ca](#) perintah untuk mendaftar, tetapi tidak mengaktifkan, sertifikat klien.

```
aws iot register-certificate-without-ca \  
--certificate-pem file://device_cert_filename.pem
```

Sertifikat klien terdaftar AWS IoT, tetapi belum aktif. Lihat [Aktifkan sertifikat klien \(CLI\)](#) untuk informasi tentang cara mengaktifkannya nanti.

Anda juga dapat mengaktifkan sertifikat klien ketika Anda mendaftarkannya dengan menggunakan perintah ini.

```
aws iot register-certificate-without-ca \  
--status ACTIVE \  
--certificate-pem file://device_cert_filename.pem
```

Untuk informasi selengkapnya tentang mengaktifkan sertifikat sehingga dapat digunakan untuk terhubung AWS IoT, lihat [Mengaktifkan atau menonaktifkan sertifikat klien](#).

Daftarkan sertifikat klien saat klien terhubung ke AWS IoT just-in-time registrasi (JITR)

Anda dapat mengonfigurasi sertifikat CA untuk mengaktifkan sertifikat klien yang telah ditandatangani untuk mendaftar AWS IoT secara otomatis saat pertama kali klien terhubung AWS IoT.

Untuk mendaftarkan sertifikat klien saat klien terhubung AWS IoT untuk pertama kalinya, Anda harus mengaktifkan sertifikat CA untuk pendaftaran otomatis dan mengonfigurasi koneksi pertama oleh klien untuk memberikan sertifikat yang diperlukan.

Konfigurasi sertifikat CA untuk mendukung pendaftaran otomatis (konsol)

Untuk mengonfigurasi sertifikat CA untuk mendukung pendaftaran sertifikat klien otomatis menggunakan AWS IoT konsol

1. Masuk ke Konsol AWS Manajemen dan buka [AWS IoT konsol](#).
2. Di panel navigasi kiri, pilih Aman, pilih CA.
3. Dalam daftar otoritas sertifikat, temukan yang ingin Anda aktifkan pendaftaran otomatis, dan buka menu opsi dengan menggunakan ikon elipsis.
4. Pada menu opsi, pilih Aktifkan pendaftaran otomatis.

Note

Status registrasi otomatis tidak ditampilkan dalam daftar otoritas sertifikat. Untuk melihat status registrasi otomatis otoritas sertifikat, Anda harus membuka halaman Detail otoritas sertifikat.

Konfigurasi sertifikat CA untuk mendukung pendaftaran otomatis (CLI)

Jika Anda telah mendaftarkan sertifikat CA Anda AWS IoT, gunakan [update-ca-certificate](#) perintah untuk mengatur `autoRegistrationStatus` sertifikat CA ke `ENABLE`.

```
aws iot update-ca-certificate \  
--certificate-id caCertificateId \  
--new-auto-registration-status ENABLE
```

Jika Anda ingin mengaktifkan `autoRegistrationStatus` ketika Anda mendaftarkan sertifikat CA, gunakan [register-ca-certificate](#) perintah.

```
aws iot register-ca-certificate \  
--allow-auto-registration \  
--ca-certificate file://root_CA_cert_filename.pem \  
--verification-cert file://verification_cert_filename.pem
```

Gunakan [describe-ca-certificate](#) perintah untuk melihat status sertifikat CA.

Konfigurasi koneksi pertama oleh klien untuk pendaftaran otomatis

Ketika klien mencoba untuk terhubung AWS IoT untuk pertama kalinya, sertifikat klien yang ditandatangani oleh sertifikat CA Anda harus ada pada klien selama jabat tangan Transport Layer Security (TLS).

Saat klien terhubung AWS IoT, gunakan sertifikat klien yang Anda buat di [Buat sertifikat AWS IoT klien](#) atau [Buat sertifikat klien Anda sendiri](#). AWS IoT mengakui sertifikat CA sebagai sertifikat CA terdaftar, mendaftarkan sertifikat klien, dan menetapkan statusnya. `PENDING_ACTIVATION` Ini berarti bahwa sertifikat klien terdaftar secara otomatis dan sedang menunggu aktivasi. Status sertifikat klien harus `ACTIVE` sebelum dapat digunakan untuk terhubung AWS IoT. Lihat [Mengaktifkan atau menonaktifkan sertifikat klien](#) untuk informasi tentang mengaktifkan sertifikat klien.

Note

Anda dapat menyediakan perangkat menggunakan fitur AWS IoT Core just-in-time registration (JITR) tanpa harus mengirim seluruh rantai kepercayaan pada koneksi pertama perangkat ke AWS IoT Core. Menyajikan sertifikat CA adalah opsional tetapi perangkat diperlukan untuk mengirim ekstensi [Server Name Indication \(SNI\)](#) ketika mereka terhubung.

Ketika AWS IoT secara otomatis mendaftarkan sertifikat atau ketika klien menyajikan sertifikat dalam `PENDING_ACTIVATION` status, AWS IoT menerbitkan pesan ke topik MQTT berikut:

```
$aws/events/certificates/registered/caCertificateId
```

caCertificateId Dimana ID sertifikat CA yang mengeluarkan sertifikat klien.

Pesan yang dipublikasikan untuk topik ini memiliki struktur sebagai berikut:

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

Anda dapat membuat aturan yang mendengarkan topik ini dan melakukan beberapa tindakan. Sebaiknya Anda membuat aturan Lambda yang memverifikasi sertifikat klien tidak ada dalam

daftar pencabutan sertifikat (CRL), mengaktifkan sertifikat, dan membuat serta melampirkan kebijakan ke sertifikat. Kebijakan menentukan sumber daya mana yang dapat diakses klien. Untuk informasi selengkapnya tentang cara membuat aturan Lambda yang mendengarkan `$aws/events/certificates/registered/caCertificateID` topik dan melakukan tindakan ini, lihat [just-in-time pendaftaran Sertifikat Klien di](#). AWS IoT

Jika terjadi kesalahan atau pengecualian selama registrasi otomatis sertifikat klien, AWS IoT kirimkan peristiwa atau pesan ke log Anda di CloudWatch Log. Untuk informasi selengkapnya tentang menyiapkan log untuk akun Anda, lihat [CloudWatch dokumentasi Amazon](#).

Mengaktifkan atau menonaktifkan sertifikat klien

AWS IoT memverifikasi bahwa sertifikat klien aktif saat mengautentikasi koneksi.

Anda dapat membuat dan mendaftarkan sertifikat klien tanpa mengaktifkannya sehingga tidak dapat digunakan sampai Anda ingin menggunakannya. Anda juga dapat menonaktifkan sertifikat klien aktif untuk menonaktifkannya sementara. Terakhir, Anda dapat mencabut sertifikat klien untuk mencegahnya dari penggunaan di masa mendatang.

Aktifkan sertifikat klien (konsol)

Untuk mengaktifkan sertifikat klien menggunakan AWS IoT konsol

1. Masuk ke Konsol AWS Manajemen dan buka [AWS IoT konsol](#).
2. Di panel navigasi kiri, pilih Aman, pilih Sertifikat.
3. Dalam daftar sertifikat, cari sertifikat yang ingin Anda aktifkan, dan buka menu opsi dengan menggunakan ikon elipsis.
4. Di menu opsi, pilih Aktifkan.

Sertifikat harus ditampilkan sebagai Aktif dalam daftar sertifikat.

Nonaktifkan sertifikat klien (konsol)

Untuk menonaktifkan sertifikat klien menggunakan konsol AWS IoT

1. Masuk ke Konsol AWS Manajemen dan buka [AWS IoT konsol](#).
2. Di panel navigasi kiri, pilih Aman, pilih Sertifikat.
3. Dalam daftar sertifikat, cari sertifikat yang ingin Anda nonaktifkan, dan buka menu opsi dengan menggunakan ikon elipsis.

4. Di menu opsi, pilih Nonaktifkan.

Sertifikat harus ditampilkan sebagai Tidak Aktif dalam daftar sertifikat.

Aktifkan sertifikat klien (CLI)

AWS CLI Ini menyediakan [update-certificate](#) perintah untuk mengaktifkan sertifikat.

```
aws iot update-certificate \  
  --certificate-id certificateId \  
  --new-status ACTIVE
```

Jika perintah berhasil, status sertifikat akan menjadi ACTIVE. Jalankan [describe-certificate](#) untuk melihat status sertifikat.

```
aws iot describe-certificate \  
  --certificate-id certificateId
```

Nonaktifkan sertifikat klien (CLI)

AWS CLI Ini menyediakan [update-certificate](#) perintah untuk menonaktifkan sertifikat.

```
aws iot update-certificate \  
  --certificate-id certificateId \  
  --new-status INACTIVE
```

Jika perintah berhasil, status sertifikat akan menjadi INACTIVE. Jalankan [describe-certificate](#) untuk melihat status sertifikat.

```
aws iot describe-certificate \  
  --certificate-id certificateId
```

Lampirkan sesuatu atau kebijakan ke sertifikat klien

Ketika Anda membuat dan mendaftarkan sertifikat terpisah dari AWS IoT sesuatu, itu tidak akan memiliki kebijakan yang mengotorisasi AWS IoT operasi apa pun, juga tidak akan dikaitkan dengan objek apa AWS IoT pun. Bagian ini menjelaskan cara menambahkan hubungan ini ke sertifikat terdaftar.

⚠ Important

Untuk menyelesaikan prosedur ini, Anda harus sudah membuat hal atau kebijakan yang ingin Anda lampirkan ke sertifikat.

Sertifikat mengautentikasi perangkat AWS IoT sehingga dapat terhubung. Melampirkan sertifikat ke sumber daya sesuatu menetapkan hubungan antara perangkat (melalui sertifikat) dan sumber daya benda. Untuk mengizinkan perangkat melakukan AWS IoT tindakan, seperti mengizinkan perangkat terhubung dan mempublikasikan pesan, kebijakan yang sesuai harus dilampirkan ke sertifikat perangkat.

Lampirkan sesuatu ke sertifikat klien (konsol)

Anda akan memerlukan nama objek benda untuk menyelesaikan prosedur ini.

Untuk melampirkan objek benda ke sertifikat terdaftar

1. Masuk ke Konsol AWS Manajemen dan buka [AWS IoT konsol](#).
2. Di panel navigasi kiri, pilih Aman, pilih Sertifikat.
3. Dalam daftar sertifikat, cari sertifikat yang ingin Anda lampirkan kebijakan, buka menu opsi sertifikat dengan memilih ikon elipsis, dan pilih Lampirkan hal.
4. Di pop-up, cari nama benda yang ingin Anda lampirkan ke sertifikat, pilih kotak centang, dan pilih Lampirkan.

Objek benda sekarang harus muncul dalam daftar hal-hal di halaman detail sertifikat.

Lampirkan kebijakan ke sertifikat klien (konsol)

Anda akan memerlukan nama objek kebijakan untuk menyelesaikan prosedur ini.

Untuk melampirkan objek kebijakan ke sertifikat terdaftar

1. Masuk ke Konsol AWS Manajemen dan buka [AWS IoT konsol](#).
2. Di panel navigasi kiri, pilih Aman, pilih Sertifikat.
3. Dalam daftar sertifikat, cari sertifikat yang ingin Anda lampirkan kebijakan, buka menu opsi sertifikat dengan memilih ikon elipsis, dan pilih Lampirkan kebijakan.

4. Di pop-up, cari nama kebijakan yang ingin dilampirkan ke sertifikat, pilih kotak centang, dan pilih Lampirkan.

Objek kebijakan sekarang akan muncul dalam daftar kebijakan di halaman detail sertifikat.

Lampirkan sesuatu ke sertifikat klien (CLI)

AWS CLI Memberikan [attach-thing-principal](#) perintah untuk melampirkan benda benda ke sertifikat.

```
aws iot attach-thing-principal \  
  --principal certificateArn \  
  --thing-name thingName
```

Lampirkan kebijakan ke sertifikat klien (CLI)

AWS CLI Memberikan [attach-policy](#) perintah untuk melampirkan objek kebijakan ke sertifikat.

```
aws iot attach-policy \  
  --target certificateArn \  
  --policy-name policyName
```

Mencabut sertifikat klien

Jika Anda mendeteksi aktivitas mencurigakan pada sertifikat klien terdaftar, Anda dapat mencabutnya sehingga tidak dapat digunakan lagi.

Note

Setelah sertifikat dicabut, statusnya tidak dapat diubah. Artinya, status sertifikat tidak dapat diubah menjadi Active atau status lainnya.

Mencabut sertifikat klien (konsol)

Untuk mencabut sertifikat klien menggunakan konsol AWS IoT

1. Masuk ke Konsol AWS Manajemen dan buka [AWS IoT konsol](#).
2. Di panel navigasi kiri, pilih Aman, pilih Sertifikat.
3. Dalam daftar sertifikat, cari sertifikat yang ingin Anda cabut, dan buka menu opsi dengan menggunakan ikon elipsis.

4. Di menu opsi, pilih Cabut.

Jika sertifikat berhasil dicabut, itu akan ditampilkan sebagai Dicabut dalam daftar sertifikat.

Mencabut sertifikat klien (CLI)

AWS CLI Memberikan [update-certificate](#) perintah untuk mencabut sertifikat.

```
aws iot update-certificate \  
  --certificate-id certificateId \  
  --new-status REVOKED
```

Jika perintah berhasil, status sertifikat akan menjadi REVOKED. Jalankan [describe-certificate](#) untuk melihat status sertifikat.

```
aws iot describe-certificate \  
  --certificate-id certificateId
```

Transfer sertifikat ke akun lain

Sertifikat X.509 milik satu Akun AWS dapat ditransfer ke yang lain. Akun AWS

Untuk mentransfer sertifikat X.509 dari satu ke yang lain Akun AWS

1. [the section called “Mulai transfer sertifikat”](#)

Sertifikat harus dinonaktifkan dan terlepas dari semua kebijakan dan hal-hal sebelum memulai transfer.

2. [the section called “Menerima atau menolak transfer sertifikat”](#)

Akun penerima harus secara eksplisit menerima atau menolak sertifikat yang ditransfer. Setelah akun penerima menerima sertifikat, sertifikat harus diaktifkan sebelum digunakan.

3. [the section called “Batalkan transfer sertifikat”](#)

Akun asal dapat membatalkan transfer, jika sertifikat belum diterima.

Mulai transfer sertifikat

Anda dapat mulai mentransfer sertifikat ke yang lain Akun AWS dengan menggunakan [AWS IoT konsol](#) atau AWS CLI.

Mulai transfer sertifikat (konsol)

Untuk menyelesaikan prosedur ini, Anda memerlukan ID sertifikat yang ingin Anda transfer.

Lakukan prosedur ini dari akun dengan sertifikat untuk ditransfer.

Untuk mulai mentransfer sertifikat ke yang lain Akun AWS

1. Masuk ke Konsol AWS Manajemen dan buka [AWS IoT konsol](#).
2. Di panel navigasi kiri, pilih Aman, pilih Sertifikat.

Pilih sertifikat dengan status Aktif atau Tidak Aktif yang ingin Anda transfer dan buka halaman detailnya.

3. Pada halaman Detail sertifikat, di menu Tindakan, jika opsi Nonaktifkan tersedia, pilih opsi Nonaktifkan untuk menonaktifkan sertifikat.
4. Pada halaman Detail sertifikat, di menu sebelah kiri, pilih Kebijakan.
5. Pada halaman Kebijakan sertifikat, jika ada kebijakan yang dilampirkan pada sertifikat, lepaskan masing-masing dengan membuka menu opsi kebijakan dan memilih Lepaskan.

Sertifikat tidak boleh memiliki kebijakan terlampir sebelum Anda melanjutkan.

6. Pada halaman Kebijakan sertifikat, di menu sebelah kiri, pilih Things.
7. Pada halaman Things sertifikat, jika ada hal yang dilampirkan pada sertifikat, lepaskan masing-masing dengan membuka menu opsi benda dan memilih Lepaskan.

Sertifikat tidak boleh memiliki hal-hal terlampir sebelum Anda melanjutkan.

8. Pada halaman Things sertifikat, di menu sebelah kiri, pilih Detail.
9. Pada halaman Detail sertifikat, di menu Tindakan, pilih Mulai transfer untuk membuka kotak dialog Mulai transfer.
10. Dalam kotak dialog Mulai transfer, masukkan Akun AWS nomor akun untuk menerima sertifikat dan pesan singkat opsional.
11. Pilih Mulai transfer untuk mentransfer sertifikat.

Konsol harus menampilkan pesan yang menunjukkan keberhasilan atau kegagalan transfer. Jika transfer dimulai, status sertifikat diperbarui ke Transfer.

Mulai transfer sertifikat (CLI)

Untuk menyelesaikan prosedur ini, Anda memerlukan `CertificateID` dan `CertificateArn` dari sertifikat yang ingin Anda transfer.

Lakukan prosedur ini dari akun dengan sertifikat untuk ditransfer.

Untuk mulai mentransfer sertifikat ke AWS akun lain

1. Gunakan [update-certificate](#) perintah untuk menonaktifkan sertifikat.

```
aws iot update-certificate --certificate-id certificateId --new-status INACTIVE
```

2. Lepaskan semua kebijakan.

1. Gunakan [list-attached-policies](#) perintah untuk membuat daftar kebijakan yang dilampirkan pada sertifikat.

```
aws iot list-attached-policies --target certificateArn
```

2. Untuk setiap kebijakan terlampir, gunakan [detach-policy](#) perintah untuk melepaskan kebijakan.

```
aws iot detach-policy --target certificateArn --policy-name policy-name
```

3. Lepaskan semua hal.

1. Gunakan [list-principal-things](#) perintah untuk membuat daftar hal-hal yang dilampirkan pada sertifikat.

```
aws iot list-principal-things --principal certificateArn
```

2. Untuk setiap hal yang terlampir, gunakan [detach-thing-principal](#) perintah untuk melepaskan benda itu.

```
aws iot detach-thing-principal --principal certificateArn --thing-name thing-name
```

4. Gunakan [transfer-certificate](#) perintah untuk memulai transfer sertifikat.

```
aws iot transfer-certificate --certificate-id certificateId --target-aws-account account-id
```

Menerima atau menolak transfer sertifikat

Anda dapat menerima atau menolak sertifikat yang ditransfer kepada Anda Akun AWS dari orang lain Akun AWS dengan menggunakan [AWS IoT konsol](#) atau AWS CLI

Menerima atau menolak transfer sertifikat (konsol)

Untuk menyelesaikan prosedur ini, Anda memerlukan ID sertifikat yang ditransfer ke akun Anda.

Lakukan prosedur ini dari akun yang menerima sertifikat yang ditransfer.

Untuk menerima atau menolak sertifikat yang ditransfer ke Akun AWS

1. Masuk ke Konsol AWS Manajemen dan buka [AWS IoT konsol](#).
2. Di panel navigasi kiri, pilih Aman, pilih Sertifikat.

Pilih sertifikat dengan status Transfer tertunda yang ingin Anda terima atau tolak dan buka halaman detailnya.

3. Pada halaman Detail sertifikat, di menu Tindakan,
 - Untuk menerima sertifikat, pilih Terima transfer.
 - Untuk tidak menerima sertifikat, pilih Tolak transfer.

Menerima atau menolak transfer sertifikat (CLI)

Untuk menyelesaikan prosedur ini, Anda memerlukan *CertificateID* dari transfer sertifikat yang ingin Anda terima atau tolak.

Lakukan prosedur ini dari akun yang menerima sertifikat yang ditransfer.

Untuk menerima atau menolak sertifikat yang ditransfer ke Akun AWS

1. Gunakan [accept-certificate-transfer](#) perintah untuk menerima sertifikat.

```
aws iot accept-certificate-transfer --certificate-id certificateId
```

2. Gunakan [reject-certificate-transfer](#) perintah untuk menolak sertifikat.

```
aws iot reject-certificate-transfer --certificate-id certificateId
```

Batalkan transfer sertifikat

Anda dapat membatalkan transfer sertifikat sebelum diterima dengan menggunakan [AWS IoT konsol](#) atau AWS CLI.

Membatalkan transfer sertifikat (konsol)

Untuk menyelesaikan prosedur ini, Anda memerlukan ID transfer sertifikat yang ingin Anda batalkan.

Lakukan prosedur ini dari akun yang memulai transfer sertifikat.

Untuk membatalkan transfer sertifikat

1. Masuk ke Konsol AWS Manajemen dan buka [AWS IoT konsol](#).
2. Di panel navigasi kiri, pilih Aman, pilih Sertifikat.

Pilih sertifikat dengan status Transfer yang transfernya ingin Anda batalkan dan buka menu opsinya.

3. Pada menu opsi sertifikat, pilih opsi Batalkan transfer untuk membatalkan transfer sertifikat.

Important

Berhati-hatilah untuk tidak salah mengira opsi Batalkan transfer dengan opsi Cabut. Opsi Batalkan transfer membatalkan transfer sertifikat, sedangkan opsi Cabut membuat sertifikat tidak dapat digunakan secara permanen oleh AWS IoT

Membatalkan transfer sertifikat (CLI)

Untuk menyelesaikan prosedur ini, Anda memerlukan *CertificateID* dari transfer sertifikat yang ingin Anda batalkan.

Lakukan prosedur ini dari akun yang memulai transfer sertifikat.

Gunakan [cancel-certificate-transfer](#) perintah untuk membatalkan transfer sertifikat.

```
aws iot cancel-certificate-transfer --certificate-id certificateId
```

Pengguna IAM, grup IAM, dan IAM role

Pengguna, grup, dan peran IAM adalah mekanisme standar untuk mengelola identitas dan otentikasi di AWS. Anda dapat menggunakannya untuk terhubung ke antarmuka AWS IoT HTTP menggunakan AWS SDK dan AWS CLI.

Peran IAM juga memungkinkan AWS IoT untuk mengakses AWS sumber daya lain di akun Anda atas nama Anda. Misalnya, jika Anda ingin perangkat mempublikasikan statusnya ke tabel DynamoDB, peran IAM AWS IoT memungkinkan untuk berinteraksi dengan Amazon DynamoDB. Untuk informasi lebih lanjut, lihat [Peran IAM](#).

Untuk koneksi broker pesan melalui HTTP, AWS IoT mengotentikasi pengguna, grup, dan peran menggunakan proses penandatanganan Versi Tanda Tangan 4. Untuk selengkapnya, lihat [Menandatangani Permintaan AWS API](#).

Saat menggunakan AWS Signature Version 4 dengan AWS IoT, klien harus mendukung yang berikut dalam implementasi TLS mereka:

- TLS 1.2
- Validasi tanda tangan sertifikat SHA-256 RSA
- Salah satu cipher suite dari bagian dukungan cipher suite TLS

Untuk informasi, lihat [Identitas dan manajemen akses untuk AWS IoT](#).

Identitas Amazon Cognito

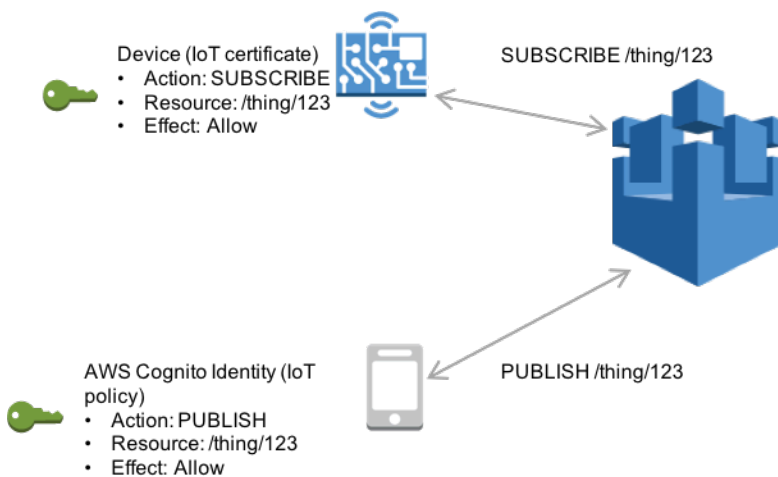
Amazon Cognito Identity memungkinkan Anda membuat AWS kredensial hak istimewa sementara dan terbatas untuk digunakan dalam aplikasi seluler dan web. Saat Anda menggunakan Identitas Amazon Cognito, buat kumpulan identitas yang membuat identitas unik untuk pengguna Anda dan autentikasi dengan penyedia identitas seperti Login with Amazon, Facebook, dan Google. Anda juga dapat menggunakan identitas Amazon Cognito dengan identitas otentikasi pengembang Anda sendiri. Untuk informasi selengkapnya, lihat [Identitas Amazon Cognito](#).

Untuk menggunakan Identitas Amazon Cognito, tentukan kumpulan identitas Amazon Cognito yang dikaitkan dengan peran IAM. Peran IAM dikaitkan dengan kebijakan IAM yang memberikan izin identitas dari kumpulan identitas Anda untuk mengakses AWS sumber daya seperti layanan panggilan. AWS

Identitas Amazon Cognito menciptakan identitas yang tidak diautentikasi dan diautentikasi. Identitas yang tidak diautentikasi digunakan untuk pengguna tamu di aplikasi seluler atau web yang ingin menggunakan aplikasi tanpa masuk. Pengguna yang tidak diautentikasi hanya diberikan izin yang ditentukan dalam kebijakan IAM yang terkait dengan kumpulan identitas.

Saat Anda menggunakan identitas yang diautentikasi, selain kebijakan IAM yang dilampirkan pada kumpulan identitas, Anda harus melampirkan AWS IoT kebijakan ke Identitas Amazon Cognito. Untuk melampirkan AWS IoT kebijakan, gunakan [AttachPolicy](#) API dan berikan izin kepada pengguna individual AWS IoT aplikasi Anda. Anda dapat menggunakan AWS IoT kebijakan ini untuk menetapkan izin berbutir halus bagi pelanggan tertentu dan perangkat mereka.

Pengguna yang diautentikasi dan tidak diautentikasi adalah jenis identitas yang berbeda. Jika Anda tidak melampirkan AWS IoT kebijakan ke Identitas Amazon Cognito, pengguna yang diautentikasi akan gagal melakukan otorisasi AWS IoT dan tidak memiliki akses ke AWS IoT sumber daya dan tindakan. Untuk informasi selengkapnya tentang membuat kebijakan untuk identitas Amazon Cognito, lihat dan [Contoh kebijakan Publikasi/Berlangganan Otorisasi dengan identitas Amazon Cognito](#)



Otentikasi dan otorisasi khusus

AWS IoT Core memungkinkan Anda menentukan otorisasi khusus sehingga Anda dapat mengelola otentikasi dan otorisasi klien Anda sendiri. Ini berguna ketika Anda perlu menggunakan mekanisme otentikasi selain yang mendukung AWS IoT Core secara asli. (Untuk informasi lebih lanjut tentang mekanisme yang didukung secara asli, lihat [the section called “Autentikasi Klien”](#)).

Misalnya, jika Anda memigrasikan perangkat yang ada di bidang ke AWS IoT Core dan perangkat ini menggunakan token pembawa kustom atau nama pengguna dan kata sandi MQTT untuk mengautentikasi, Anda dapat memigrasikannya AWS IoT Core tanpa harus memberikan identitas baru untuknya. Anda dapat menggunakan otentikasi khusus dengan salah satu protokol komunikasi

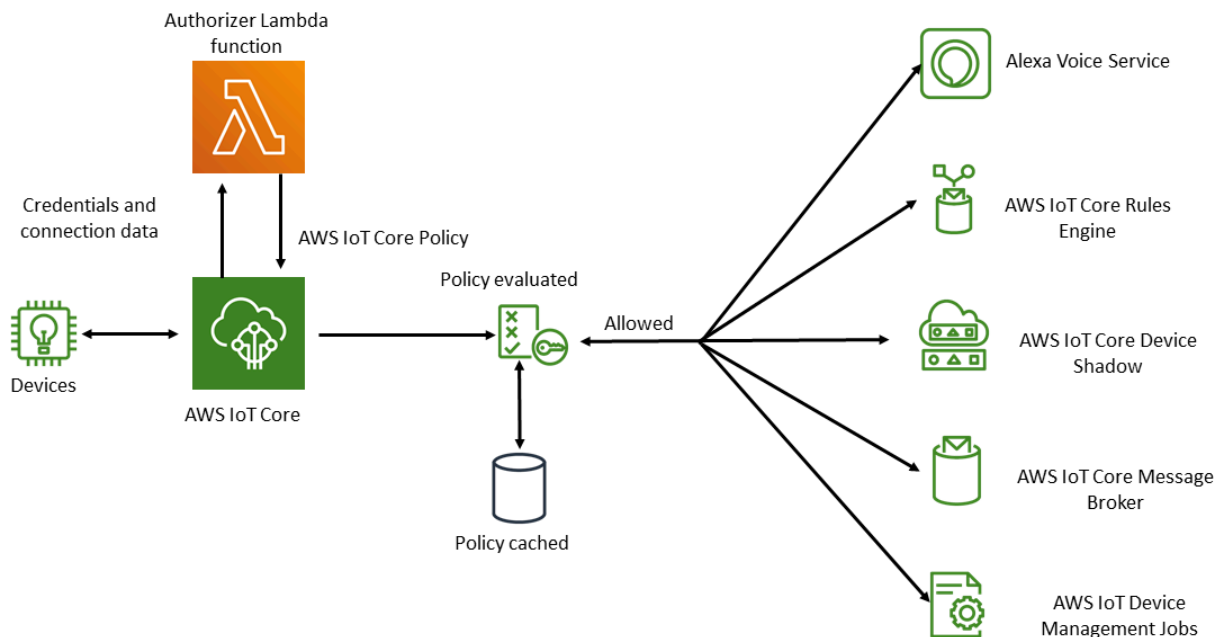
yang mendukung. AWS IoT Core Untuk informasi selengkapnya tentang protokol yang AWS IoT Core mendukung, lihat [the section called “Protokol komunikasi perangkat”](#)

Topik

- [Memahami alur kerja otentikasi kustom](#)
- [Membuat dan mengelola otorisasi kustom](#)
- [Menghubungkan ke AWS IoT Core dengan menggunakan otentikasi kustom](#)
- [Memecahkan masalah otorisasi Anda](#)

Memahami alur kerja otentikasi kustom

[Otentikasi khusus memungkinkan Anda menentukan cara mengautentikasi dan mengotorisasi klien dengan menggunakan sumber daya otorisasi.](#) Setiap authorizer berisi referensi ke fungsi Lambda yang dikelola pelanggan, kunci publik opsional untuk memvalidasi kredensi perangkat, dan informasi konfigurasi tambahan. Diagram berikut menggambarkan alur kerja otorisasi untuk otentikasi kustom di AWS IoT Core



AWS IoT Core otentikasi kustom dan alur kerja otorisasi

Daftar berikut menjelaskan setiap langkah dalam alur kerja otentikasi dan otorisasi kustom.

1. Perangkat terhubung ke titik akhir AWS IoT Core data pelanggan dengan menggunakan salah satu yang didukung [the section called “Protokol komunikasi perangkat”](#). Perangkat meneruskan kredensial baik di bidang header permintaan atau parameter kueri (untuk HTTP Publish atau MQTT melalui WebSockets protokol), atau di bidang nama pengguna dan kata sandi pesan MQTT CONNECT (untuk MQTT dan MQTT melalui protokol). WebSockets
2. AWS IoT Core memeriksa salah satu dari dua kondisi:
 - Permintaan yang masuk menentukan otorisasi.
 - Titik akhir AWS IoT Core data yang menerima permintaan memiliki otorisasi default yang dikonfigurasi untuknya.

Jika AWS IoT Core menemukan otorisasi dengan salah satu cara ini, AWS IoT Core memicu fungsi Lambda yang terkait dengan otorisasi.

3. (Opsional) Jika Anda telah mengaktifkan penandatanganan token, AWS IoT Core validasi tanda tangan permintaan dengan menggunakan kunci publik yang disimpan di otorisasi sebelum memicu fungsi Lambda. Jika validasi gagal, AWS IoT Core hentikan permintaan tanpa menjalankan fungsi Lambda.
4. Fungsi Lambda menerima kredensial dan metadata koneksi dalam permintaan dan membuat keputusan otentikasi.
5. Fungsi Lambda mengembalikan hasil keputusan otentikasi dan dokumen AWS IoT Core kebijakan yang menentukan tindakan apa yang diizinkan dalam koneksi. Fungsi Lambda juga mengembalikan informasi yang menentukan seberapa sering AWS IoT Core memvalidasi ulang kredensial dalam permintaan dengan menjalankan fungsi Lambda.
6. AWS IoT Core mengevaluasi aktivitas pada koneksi terhadap kebijakan yang diterimanya dari fungsi Lambda.
7. Setelah koneksi dibuat dan Lambda otorisasi kustom Anda awalnya dipanggil, pemanggilan berikutnya dapat ditunda hingga 5 menit pada koneksi idle tanpa operasi MQTT apa pun. Setelah itu, pemanggilan berikutnya akan mengikuti interval penyegaran di Lambda otorisasi khusus Anda. Pendekatan ini dapat mencegah pemanggilan berlebihan yang dapat melebihi batas konkurensi Lambda Anda. Akun AWS

Pertimbangan penskalaan

Karena fungsi Lambda menangani otentikasi dan otorisasi untuk otorisasi Anda, fungsi tersebut tunduk pada harga Lambda dan batas layanan, seperti tingkat eksekusi bersamaan. [Untuk informasi selengkapnya tentang harga Lambda, lihat Harga Lambda](#). Anda dapat mengelola

beban pada fungsi Lambda Anda dengan menyesuaikan `refreshAfterInSeconds` dan `disconnectAfterInSeconds` parameter dalam respons fungsi Lambda Anda. Untuk informasi selengkapnya tentang isi respons fungsi Lambda Anda, lihat [the section called “Mendefinisikan fungsi Lambda Anda”](#)

Note

Jika Anda membiarkan penandatanganan diaktifkan, Anda dapat mencegah pemacu Lambda yang berlebihan oleh klien yang tidak dikenal. Pertimbangkan ini sebelum Anda menonaktifkan penandatanganan di otorisasi Anda.

Note

Batas waktu tunggu fungsi Lambda untuk otorisasi khusus adalah 5 detik.

Membuat dan mengelola otorisasi kustom

AWS IoT Core [mengimplementasikan otentikasi kustom dan skema otorisasi dengan menggunakan sumber daya otorisasi](#). Setiap authorizer terdiri dari komponen-komponen berikut:

- Nama: String unik yang ditentukan pengguna yang mengidentifikasi otorisasi.
- Fungsi Lambda ARN: Nama Sumber Daya Amazon (ARN) dari fungsi Lambda yang mengimplementasikan logika otorisasi dan otentikasi.
- Nama kunci token: Nama kunci yang digunakan untuk mengekstrak token dari header HTTP, parameter kueri, atau nama pengguna MQTT CONNECT untuk melakukan validasi tanda tangan. Nilai ini diperlukan jika penandatanganan diaktifkan di otorisasi Anda.
- Penandatanganan flag dinonaktifkan (opsional): Nilai Boolean yang menentukan apakah akan menonaktifkan persyaratan penandatanganan pada kredensial. Ini berguna untuk skenario di mana penandatanganan kredensial tidak masuk akal, seperti skema otentikasi yang menggunakan nama pengguna dan kata sandi MQTT. Nilai defaultnya adalah `false`, jadi penandatanganan diaktifkan secara default.
- Kunci publik penandatanganan token: Kunci publik yang AWS IoT Core digunakan untuk memvalidasi tanda tangan token. Panjang minimumnya adalah 2.048 bit. Nilai ini diperlukan jika penandatanganan diaktifkan di otorisasi Anda.

Lambda menagih Anda untuk berapa kali fungsi Lambda Anda berjalan dan untuk jumlah waktu yang diperlukan untuk kode dalam fungsi Anda untuk mengeksekusi. [Untuk informasi selengkapnya tentang harga Lambda, lihat Harga Lambda.](#) Untuk informasi selengkapnya tentang membuat fungsi Lambda, lihat Panduan Pengembang [Lambda](#).

Note

Jika Anda membiarkan penandatanganan diaktifkan, Anda dapat mencegah pemicu Lambda yang berlebihan oleh klien yang tidak dikenal. Pertimbangkan ini sebelum Anda menonaktifkan penandatanganan di otorisasi Anda.

Note

Batas waktu tunggu fungsi Lambda untuk otorisasi khusus adalah 5 detik.

Mendefinisikan fungsi Lambda Anda

Saat AWS IoT Core memanggil otorisasi Anda, itu memicu Lambda terkait yang terkait dengan otorisasi dengan peristiwa yang berisi objek JSON berikut. Contoh objek JSON berisi semua bidang yang mungkin. Bidang apa pun yang tidak relevan dengan permintaan koneksi tidak disertakan.

```
{
  "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.
  },
}
```

Fungsi Lambda harus menggunakan informasi ini untuk mengautentikasi koneksi yang masuk dan memutuskan tindakan apa yang diizinkan dalam koneksi. Fungsi harus mengirim respons yang berisi nilai-nilai berikut.

- `isAuthenticated`: Nilai Boolean yang menunjukkan apakah permintaan diautentikasi.
- `principalId`: String alfanumerik yang bertindak sebagai pengidentifikasi untuk token yang dikirim oleh permintaan otorisasi kustom. Nilai harus berupa string alfanumerik dengan setidaknya satu, dan tidak lebih dari 128, karakter dan cocok dengan pola ekspresi reguler (regex) ini: `([a-zA-Z0-9]){1,128}` Karakter khusus yang tidak alfanumerik tidak diizinkan untuk digunakan dengan `principalId` AWS IoT Core Lihat dokumentasi untuk AWS layanan lain jika karakter khusus non-alfanumerik diizinkan untuk `principalId`
- `policyDocuments`: Daftar dokumen AWS IoT Core kebijakan berformat JSON Untuk informasi selengkapnya tentang membuat AWS IoT Core kebijakan, lihat [the section called “AWS IoT Core kebijakan”](#) Jumlah maksimum dokumen kebijakan adalah 10 dokumen kebijakan. Setiap dokumen kebijakan dapat berisi maksimal 2.048 karakter.
- `disconnectAfterInSeconds`: Sebuah integer yang menentukan durasi maksimum (dalam detik) dari koneksi ke gateway. AWS IoT Core Nilai minimum adalah 300 detik, dan nilai maksimumnya adalah 86.400 detik. Nilai defaultnya adalah 86.400.

Note

Nilai `disconnectAfterInSeconds` (dikembalikan oleh fungsi Lambda) diatur saat koneksi dibuat. Nilai ini tidak dapat diubah selama pemanggilan Lambda refresh kebijakan berikutnya.

- `refreshAfterInSeconds`: Sebuah integer yang menentukan interval antara kebijakan refresh. Ketika interval ini berlalu, AWS IoT Core memanggil fungsi Lambda untuk mengizinkan refresh kebijakan. Nilai minimum adalah 300 detik, dan nilai maksimumnya adalah 86.400 detik.

Objek JSON berikut berisi contoh respons yang dapat dikirim oleh fungsi Lambda Anda.

```
{
  "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"
        }
      ]
    }
  ]
}
```

`policyDocument` Nilai harus berisi dokumen AWS IoT Core kebijakan yang valid. Untuk informasi selengkapnya tentang AWS IoT Core kebijakan, lihat [the section called “AWS IoT Core kebijakan”](#). Di MQTT melalui TLS dan MQTT melalui WebSockets koneksi, AWS IoT Core cache kebijakan ini untuk interval yang ditentukan dalam nilai bidang. `refreshAfterInSeconds` Dalam kasus koneksi HTTP, fungsi Lambda dipanggil untuk setiap permintaan otorisasi kecuali perangkat Anda menggunakan koneksi persisten HTTP (juga disebut HTTP keep-alive atau penggunaan kembali koneksi HTTP) Anda dapat memilih untuk mengaktifkan caching saat mengonfigurasi otorisasi. Selama interval ini, AWS IoT Core mengotorisasi tindakan dalam koneksi yang dibuat terhadap kebijakan cache ini tanpa memicu fungsi Lambda Anda lagi. Jika kegagalan terjadi selama otentikasi kustom, AWS IoT Core menghentikan koneksi. AWS IoT Core juga mengakhiri koneksi jika telah terbuka lebih lama dari nilai yang ditentukan dalam `disconnectAfterInSeconds` parameter.

Berikut ini JavaScript berisi contoh fungsi Lambda Node.js yang mencari kata sandi dalam pesan MQTT Connect dengan nilai `test` dan mengembalikan kebijakan yang memberikan izin untuk

terhubung dengan klien `myClientName` bernama dan memublikasikan AWS IoT Core ke topik yang berisi nama klien yang sama. Jika tidak menemukan kata sandi yang diharapkan, ia mengembalikan kebijakan yang menyangkal kedua tindakan tersebut.

```
// 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;
}
```

Fungsi Lambda sebelumnya mengembalikan JSON berikut ketika menerima kata sandi yang diharapkan dari dalam test pesan MQTT Connect. Nilai password dan principalId properti akan menjadi nilai dari pesan 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
}
```

Membuat Authorizer

Anda dapat membuat authorizer dengan menggunakan [CreateAuthorizerAPI](#). Contoh berikut menjelaskan perintah.

```
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]
```

Anda dapat menggunakan `signing-disabled` parameter untuk memilih keluar dari validasi tanda tangan untuk setiap pemanggilan otorisasi Anda. Kami sangat menyarankan agar Anda tidak menonaktifkan penandatanganan kecuali Anda harus melakukannya. Validasi tanda tangan melindungi Anda dari pemanggilan berlebihan fungsi Lambda Anda dari perangkat yang tidak dikenal. Anda tidak dapat memperbarui `signing-disabled` status otorisasi setelah Anda membuatnya. Untuk mengubah perilaku ini, Anda harus membuat otorisasi khusus lain dengan nilai `signing-disabled` parameter yang berbeda.

Nilai untuk `tokenSigningPublicKeys` parameter `tokenKeyName` dan bersifat opsional jika Anda menonaktifkan penandatanganan. Mereka adalah nilai yang diperlukan jika penandatanganan diaktifkan.

Setelah Anda membuat fungsi Lambda dan otorisasi kustom, Anda harus secara eksplisit memberikan izin AWS IoT Core layanan untuk memanggil fungsi atas nama Anda. Anda dapat melakukan ini dengan perintah berikut.

```
aws lambda add-permission --function-name <lambda_function_name> --
principal iot.amazonaws.com --source-arn <authorizer_arn> --statement-id
Id-123 --action "lambda:InvokeFunction"
```

Menguji otorisasi Anda

Anda dapat menggunakan [TestInvokeAuthorizer](#) API untuk menguji pemanggilan dan mengembalikan nilai otorisasi Anda. API ini memungkinkan Anda menentukan metadata protokol dan menguji validasi tanda tangan di otorisasi Anda.

Tab berikut menunjukkan cara menggunakan AWS CLI untuk menguji otorisasi Anda.

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
```

Nilai token-signature parameter adalah token yang ditandatangani. Untuk mempelajari cara mendapatkan nilai ini, lihat [the section called “Menandatangani token”](#).

Jika otorisasi Anda mengambil nama pengguna dan kata sandi, Anda dapat meneruskan informasi ini dengan menggunakan `--mqtt-context` parameter. Tab berikut menunjukkan cara menggunakan `TestInvokeAuthorizer` API untuk mengirim objek JSON yang berisi nama pengguna, kata sandi, dan nama klien ke otorisasi kustom Anda.

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"}'
```

Kata sandi harus dikodekan base64. Contoh berikut menunjukkan cara menyandikan kata sandi di lingkungan mirip Unix.

```
echo -n PASSWORD | base64
```

Mengelola otorisasi khusus

Anda dapat mengelola otorisasi Anda dengan menggunakan API berikut.

- [ListAuthorizers](#): Tampilkan semua otorisasi di akun Anda.
- [DescribeAuthorizer](#): Menampilkan properti dari authorizer yang ditentukan. Nilai-nilai ini termasuk tanggal pembuatan, tanggal modifikasi terakhir, dan atribut lainnya.
- [SetDefaultAuthorizer](#): Menentukan authorizer default untuk endpoint data Anda AWS IoT Core . AWS IoT Core menggunakan otorisasi ini jika perangkat tidak meneruskan AWS IoT Core kredensial dan tidak menentukan otorisasi. Untuk informasi selengkapnya tentang menggunakan AWS IoT Core kredensial, lihat [the section called “Autentikasi Klien”](#)
- [UpdateAuthorizer](#): Mengubah status, nama kunci token, atau kunci publik untuk otorisasi yang ditentukan.
- [DeleteAuthorizer](#): Menghapus otorisasi yang ditentukan.

Note

Anda tidak dapat memperbarui persyaratan penandatanganan otorisasi. Ini berarti Anda tidak dapat menonaktifkan penandatanganan di otorisasi yang sudah ada yang memerlukannya. Anda juga tidak dapat meminta masuk ke otorisasi yang sudah ada yang tidak memerlukannya.

Menghubungkan ke AWS IoT Core dengan menggunakan otentikasi kustom

Perangkat dapat terhubung AWS IoT Core dengan menggunakan otentikasi khusus dengan protokol apa pun yang AWS IoT Core mendukung pesan perangkat. Untuk informasi selengkapnya tentang protokol komunikasi yang didukung, lihat [the section called “Protokol komunikasi perangkat”](#). Data koneksi yang Anda berikan ke fungsi Lambda otorisasi Anda tergantung pada protokol yang Anda gunakan. Untuk informasi selengkapnya tentang membuat fungsi Lambda otorisasi Anda, lihat [the section called “Mendefinisikan fungsi Lambda Anda”](#). Bagian berikut menjelaskan cara menghubungkan ke otentikasi dengan menggunakan setiap protokol yang didukung.

HTTPS

Perangkat yang mengirim data AWS IoT Core dengan menggunakan [HTTP Publish API](#) dapat meneruskan kredensial baik melalui header permintaan atau parameter kueri dalam permintaan HTTP POST mereka. Perangkat dapat menentukan otorisasi untuk dipanggil dengan menggunakan `x-amz-customauthorizer-name` header atau parameter kueri. Jika Anda mengaktifkan penandatanganan token di otorisasi, Anda harus meneruskan `token-key-name` dan `x-amz-customauthorizer-signature` dalam header permintaan atau parameter kueri. Perhatikan bahwa `token-signature` nilainya harus dikodekan URL saat menggunakan JavaScript dari dalam browser.

Note

Otorisasi pelanggan untuk protokol HTTPS hanya mendukung operasi publikasi. Untuk informasi selengkapnya tentang protokol HTTPS, lihat [the section called “Protokol komunikasi perangkat”](#).

Contoh permintaan berikut menunjukkan bagaimana Anda meneruskan parameter ini di header permintaan dan parameter kueri.

```
//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

Perangkat yang terhubung AWS IoT Core dengan menggunakan koneksi MQTT dapat meneruskan kredensial melalui username dan password bidang pesan MQTT. usernameNilai juga dapat secara opsional berisi string kueri yang meneruskan nilai tambahan (termasuk token, tanda tangan, dan nama otorisasi) ke otorisasi Anda. Anda dapat menggunakan string kueri ini jika Anda ingin menggunakan skema otentikasi berbasis token, bukan dan nilai. username password

Note

Data di bidang kata sandi dikodekan oleh base64. AWS IoT Core Fungsi Lambda Anda harus memecahkan kode itu.

Contoh berikut berisi username string yang berisi parameter tambahan yang menentukan token dan tanda tangan.

```
username?x-amz-customauthorizer-name=authorizer-name&x-amz-customauthorizer-signature=token-signature&token-key-name=token-value
```

Untuk memanggil otorisasi, perangkat yang terhubung AWS IoT Core dengan menggunakan MQTT dan otentikasi khusus harus terhubung pada port 443. Mereka juga harus melewati ekstensi Application Layer Protocol Negotiation (ALPN) TLS dengan nilai mqtt dan ekstensi Server Name Indication (SNI) dengan nama host dari endpoint data mereka. AWS IoT Core Untuk menghindari potensi kesalahan, nilai untuk x-amz-customauthorizer-signature harus dikodekan URL. Kami juga sangat merekomendasikan bahwa nilai-nilai x-amz-customauthorizer-name dan URL token-key-name dikodekan. Untuk informasi selengkapnya tentang nilai-nilai ini, lihat [the section called "Protokol komunikasi perangkat"](#). V2 [AWS IoT SDK Perangkat](#), [SDK Seluler](#), dan [AWS IoT Klien Perangkat](#) dapat mengkonfigurasi kedua ekstensi ini.

MQTT lebih WebSockets

Perangkat yang terhubung AWS IoT Core dengan menggunakan MQTT over WebSockets dapat meneruskan kredensial dengan salah satu dari dua cara berikut.

- Melalui header permintaan atau parameter kueri dalam permintaan HTTP UPGRADE untuk membuat WebSockets koneksi.

- Melalui `username` dan `password` bidang dalam pesan MQTT CONNECT.

Jika Anda meneruskan kredensial melalui pesan koneksi MQTT, ekstensi ALPN dan SNI TLS diperlukan. Untuk informasi selengkapnya tentang ekstensi ini, lihat [the section called "MQTT"](#). Contoh berikut menunjukkan bagaimana untuk meneruskan kredensial melalui permintaan HTTP Upgrade.

```
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
```

Menandatangani token

Anda harus menandatangani token dengan kunci pribadi dari public-private key pair yang Anda gunakan dalam panggilan. `create-authorizer` Contoh berikut menunjukkan cara membuat tanda tangan token dengan menggunakan perintah Unix-like dan JavaScript Mereka menggunakan algoritma hash SHA-256 untuk menandatangani tanda tangan.

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')
```

Memecahkan masalah otorisasi Anda

Topik ini membahas masalah umum yang dapat menyebabkan masalah dalam alur kerja otentikasi khusus dan langkah-langkah untuk menyelesaikannya. Untuk memecahkan masalah dengan paling efektif, aktifkan CloudWatch log untuk AWS IoT Core dan atur level log ke DEBUG. Anda dapat mengaktifkan CloudWatch log di AWS IoT Core konsol (<https://console.aws.amazon.com/iot/>). Untuk informasi selengkapnya tentang mengaktifkan dan mengonfigurasi log AWS IoT Core, lihat [the section called “Konfigurasi AWS IoT logging”](#)

Note

Jika Anda meninggalkan tingkat log di DEBUG untuk jangka waktu yang lama, CloudWatch mungkin menyimpan data logging dalam jumlah besar. Hal ini dapat meningkatkan CloudWatch biaya Anda. Pertimbangkan untuk menggunakan logging berbasis sumber daya untuk meningkatkan verbositas hanya untuk perangkat dalam grup hal tertentu. Untuk informasi selengkapnya tentang logging berbasis sumber daya, lihat [the section called “Konfigurasi AWS IoT logging”](#). Selain itu, setelah Anda selesai memecahkan masalah, kurangi level log ke level yang kurang bertele-tele.

Sebelum Anda memulai pemecahan masalah, tinjau [the section called “Memahami alur kerja otentikasi kustom”](#) tampilan tingkat tinggi dari proses otentikasi kustom. Ini membantu Anda memahami di mana harus mencari sumber masalah.

Topik ini membahas dua bidang berikut untuk Anda selidiki.

- Masalah yang terkait dengan fungsi Lambda otorisasi Anda.
- Masalah yang terkait dengan perangkat Anda.

Periksa masalah dalam fungsi Lambda otorisasi Anda

Lakukan langkah-langkah berikut untuk memastikan bahwa upaya koneksi perangkat Anda menjalankan fungsi Lambda Anda.

1. Verifikasi fungsi Lambda mana yang terkait dengan otorisasi Anda.

Anda dapat melakukan ini dengan memanggil [DescribeAuthorizer](#) API atau dengan mengklik otorisasi yang diinginkan di bagian Aman AWS IoT Core konsol.

2. Periksa metrik pemanggilan untuk fungsi Lambda. Lakukan langkah-langkah berikut untuk melakukan ini.
 - a. Buka AWS Lambda konsol (<https://console.aws.amazon.com/lambda/>) dan pilih fungsi yang terkait dengan otorisasi Anda.
 - b. Pilih tab Monitor dan lihat metrik untuk kerangka waktu yang relevan dengan masalah Anda.
3. Jika Anda tidak melihat pemanggilan, verifikasi bahwa AWS IoT Core memiliki izin untuk menjalankan fungsi Lambda Anda. Jika Anda melihat pemanggilan, lewati ke langkah berikutnya. Lakukan langkah-langkah berikut untuk memverifikasi bahwa fungsi Lambda Anda memiliki izin yang diperlukan.
 - a. Pilih tab Izin untuk fungsi Anda di AWS Lambda konsol.
 - b. Temukan bagian Kebijakan Berbasis Sumber Daya di bagian bawah halaman. Jika fungsi Lambda Anda memiliki izin yang diperlukan, kebijakan akan terlihat seperti contoh berikut.

```
{
  "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. Kebijakan ini memberikan `InvokeFunction` izin atas fungsi Anda kepada AWS IoT Core kepala sekolah. Jika Anda tidak melihatnya, Anda harus menambahkannya dengan menggunakan [AddPermission](#) API. Contoh berikut menunjukkan kepada Anda bagaimana melakukan ini dengan menggunakan AWS CLI.

```
aws lambda add-permission --function-name FunctionName --principal  
iot.amazonaws.com --source-arn AuthorizerARN --statement-id Id-123 --action  
"lambda:InvokeFunction"
```

4. Jika Anda melihat pemanggilan, verifikasi bahwa tidak ada kesalahan. Kesalahan mungkin menunjukkan bahwa fungsi Lambda tidak menangani peristiwa koneksi yang AWS IoT Core mengirimkannya dengan benar.

Untuk informasi tentang penanganan acara di fungsi Lambda Anda, lihat [the section called "Mendefinisikan fungsi Lambda Anda"](#) Anda dapat menggunakan fitur pengujian di AWS Lambda konsol (<https://console.aws.amazon.com/lambda/>) ke nilai pengujian kode keras dalam fungsi untuk memastikan bahwa fungsi tersebut menangani peristiwa dengan benar.

5. Jika Anda melihat pemanggilan tanpa kesalahan, tetapi perangkat Anda tidak dapat terhubung (atau memublikasikan, berlangganan, dan menerima pesan), masalahnya mungkin kebijakan yang ditampilkan oleh fungsi Lambda tidak memberikan izin untuk tindakan yang coba dilakukan perangkat Anda. Lakukan langkah-langkah berikut untuk menentukan apakah ada yang salah dengan kebijakan yang dikembalikan fungsi.
 - a. Gunakan kueri Amazon CloudWatch Logs Insights untuk memindai log dalam waktu singkat untuk memeriksa kegagalan. Contoh kueri berikut mengurutkan peristiwa berdasarkan stempel waktu dan mencari kegagalan.

```
display clientId, eventType, status, @timestamp | sort @timestamp desc | filter  
status = "Failure"
```

- b. Perbarui fungsi Lambda Anda untuk mencatat data yang dikembalikan AWS IoT Core dan peristiwa yang memicu fungsi tersebut. Anda dapat menggunakan log ini untuk memeriksa kebijakan yang dibuat oleh fungsi tersebut.

6. Jika Anda melihat pemanggilan tanpa kesalahan, tetapi perangkat Anda tidak dapat terhubung (atau mempublikasikan, berlangganan, dan menerima pesan), alasan lain adalah fungsi Lambda Anda melebihi batas waktu. Batas waktu tunggu fungsi Lambda untuk otorisasi khusus adalah 5 detik. Anda dapat memeriksa durasi fungsi di CloudWatch log atau metrik.

Menyelidiki masalah perangkat

Jika Anda tidak menemukan masalah dengan menjalankan fungsi Lambda Anda atau dengan kebijakan yang ditampilkan fungsi tersebut, cari masalah dengan upaya koneksi perangkat Anda. Permintaan koneksi yang salah dapat menyebabkan AWS IoT Core tidak memicu otorisasi Anda. Masalah koneksi dapat terjadi pada lapisan TLS dan aplikasi.

Kemungkinan masalah lapisan TLS:

- Pelanggan harus meneruskan header nama host (HTTP, MQTT over WebSockets) atau ekstensi TLS Indikasi Nama Server (HTTP, MQTT over, MQTT) di WebSockets semua permintaan otentikasi kustom. Dalam kedua kasus tersebut, nilai yang diteruskan harus cocok dengan salah satu titik akhir AWS IoT Core data akun Anda. Ini adalah titik akhir yang dikembalikan ketika Anda melakukan perintah CLI berikut.
 - `aws iot describe-endpoint --endpoint-type iot:Data-ATS`
 - `aws iot describe-endpoint --endpoint-type iot:Data`(untuk titik VeriSign akhir warisan)
- Perangkat yang menggunakan otentikasi khusus untuk koneksi MQTT juga harus melewati ekstensi TLS Application Layer Protocol Negotiation (ALPN) dengan nilai `mqtt`
- Otentikasi khusus saat ini hanya tersedia di port 443.

Kemungkinan masalah lapisan aplikasi:

- Jika penandatanganan diaktifkan (`signingDisabled`bidangnya salah di otorisasi Anda), cari masalah tanda tangan berikut.
 - Pastikan Anda meneruskan tanda tangan token di `x-amz-customauthorizer-signature` header atau dalam parameter string kueri.
 - Pastikan bahwa layanan tidak menandatangani nilai selain token.
 - Pastikan Anda meneruskan token di header atau parameter kueri yang Anda tentukan di `token-key-name` bidang di otorisasi Anda.

- Pastikan bahwa nama otorisasi yang Anda berikan di parameter `x-amz-customauthorizer-name` header atau string kueri valid atau Anda memiliki otorisasi default yang ditentukan untuk akun Anda.

Otorisasi

Otorisasi adalah proses pemberian izin untuk identitas yang diautentikasi. Anda memberikan izin dalam AWS IoT Core menggunakan AWS IoT Core dan kebijakan IAM. Topik ini mencakup AWS IoT Core kebijakan. Untuk informasi selengkapnya tentang kebijakan IAM, lihat [Identitas dan manajemen akses untuk AWS IoT](#) dan [Bagaimana AWS IoT bekerja dengan IAM](#).

AWS IoT Core kebijakan menentukan apa yang dapat dilakukan identitas yang diautentikasi. Identitas yang diautentikasi digunakan oleh perangkat, aplikasi seluler, aplikasi web, dan aplikasi desktop. Identitas yang diautentikasi bahkan bisa menjadi pengguna yang mengetik perintah AWS IoT Core CLI. Identitas dapat menjalankan AWS IoT Core operasi hanya jika memiliki kebijakan yang memberinya izin untuk operasi tersebut.

Baik AWS IoT Core kebijakan dan kebijakan IAM digunakan AWS IoT Core untuk mengontrol operasi yang dapat dilakukan oleh identitas (juga disebut prinsipal). Jenis kebijakan yang Anda gunakan bergantung pada jenis identitas yang Anda gunakan untuk mengautentikasi. AWS IoT Core

AWS IoT Core operasi dibagi menjadi dua kelompok:

- Control plane API memungkinkan Anda untuk melakukan tugas-tugas administratif seperti membuat atau memperbarui sertifikat, hal-hal, aturan, dan sebagainya.
- Data plane API memungkinkan Anda mengirim data ke dan menerima data dari AWS IoT Core.

Jenis kebijakan yang Anda gunakan bergantung pada apakah Anda menggunakan bidang kontrol atau API bidang data.

Tabel berikut menunjukkan tipe identitas, protokol yang mereka gunakan, dan jenis kebijakan yang dapat digunakan untuk otorisasi.

AWS IoT Core API pesawat data dan tipe kebijakan

Protokol dan mekanisme otentikasi	SDK	Jenis identitas	Jenis kebijakan		
MQTT melalui TLS/TCP, otentikasi timbal balik TLS (port 8883 atau 443) †)	AWS IoT Perangkat SDK	Sertifikat X.509	AWS IoT Core kebijakan		
MQTT melalui HTTPS/, otentikasi AWS SiGv4 (WebSocket port 443)	AWS SDK Seluler	Identitas Amazon Cognito yang diautentikasi	IAM dan kebijakan AWS IoT Core		
		Identitas Amazon Cognito yang tidak diautentikasi	Kebijakan IAM		
		IAM, atau identitas federasi	Kebijakan IAM		
HTTPS, otentikasi AWS Tanda Tangan Versi 4 (port 443)	AWS CLI	Amazon Cognito, IAM, atau identitas federasi	Kebijakan IAM		
HTTPS, otentikasi timbal balik	Tidak ada dukungan SDK	Sertifikat X.509	AWS IoT Core kebijakan		

Protokol dan mekanisme otentikasi	SDK	Jenis identitas	Jenis kebijakan		
TLS (port 8443)					
HTTPS melalui otentikasi khusus (Port 443)	AWS IoT Perangkat SDK	Authorizer kustom	Kebijakan otorisasi khusus		

AWS IoT Core API bidang kontrol dan jenis kebijakan

Protokol dan mekanisme otentikasi	SDK	Jenis identitas	Jenis kebijakan		
Otentikasi HTTPS AWS Signature Versi 4 (port 443)	AWS CLI	Identitas Amazon Cognito IAM, atau identitas federasi	Kebijakan IAM Kebijakan IAM		

AWS IoT Core kebijakan dilampirkan ke sertifikat X.509, identitas Amazon Cognito, atau grup hal. Kebijakan IAM dilampirkan ke pengguna, grup, atau peran IAM. Jika Anda menggunakan AWS IoT konsol atau AWS IoT Core CLI untuk melampirkan kebijakan (ke sertifikat, Identitas Amazon Cognito, atau grup hal), Anda menggunakan kebijakan. AWS IoT Core Jika tidak, Anda menggunakan kebijakan IAM. AWS IoT Core kebijakan yang dilampirkan pada grup sesuatu berlaku untuk apa pun dalam kelompok benda itu. Agar AWS IoT Core kebijakan berlaku, nama `clientId` dan benda harus cocok.

Otorisasi berbasis kebijakan adalah alat yang ampuh. Ini memberi Anda kontrol penuh atas apa yang dapat dilakukan perangkat, pengguna, atau aplikasi AWS IoT Core. Misalnya, pertimbangkan

perangkat yang terhubung AWS IoT Core dengan sertifikat. Anda dapat mengizinkan perangkat mengakses semua topik MQTT, atau Anda dapat membatasi aksesnya ke satu topik. Dalam contoh lain, pertimbangkan pengguna mengetik perintah CLI di baris perintah. Dengan menggunakan kebijakan, Anda dapat mengizinkan atau menolak akses ke perintah atau AWS IoT Core sumber daya apa pun untuk pengguna. Anda juga dapat mengontrol akses aplikasi ke AWS IoT Core sumber daya.

Perubahan yang dilakukan pada kebijakan dapat memakan waktu beberapa menit untuk menjadi efektif karena cara menyimpan dokumen kebijakan dalam AWS IoT cache. Artinya, mungkin perlu beberapa menit untuk mengakses sumber daya yang baru-baru ini diberikan akses, dan sumber daya dapat diakses selama beberapa menit setelah aksesnya dicabut.

AWS pelatihan dan sertifikasi

Untuk informasi tentang otorisasi AWS IoT Core, ikuti kursus [Deep Dive into AWS IoT Core Authentication and Authorization](#) di situs web AWS Pelatihan dan Sertifikasi.

AWS IoT Core kebijakan

AWS IoT Core Kebijakan adalah dokumen JSON. Mereka mengikuti konvensi yang sama dengan kebijakan IAM. AWS IoT Core mendukung kebijakan bernama sehingga banyak identitas dapat merferensikan dokumen kebijakan yang sama. Kebijakan bernama diberi versi sehingga dapat dengan mudah diputar kembali.

AWS IoT Core kebijakan memungkinkan Anda mengontrol akses ke bidang AWS IoT Core data. Bidang AWS IoT Core data terdiri dari operasi yang memungkinkan Anda terhubung ke broker AWS IoT Core pesan, mengirim dan menerima pesan MQTT, dan mendapatkan atau memperbarui Device Shadow sesuatu.

AWS IoT Core Kebijakan adalah dokumen JSON yang berisi satu atau beberapa pernyataan kebijakan. Setiap pernyataan berisi:

- **Effect**, yang menentukan apakah tindakan tersebut diizinkan atau ditolak.
- **Action**, yang menentukan tindakan yang diizinkan atau ditolak oleh kebijakan.
- **Resource**, yang menentukan sumber daya atau sumber daya tempat tindakan diizinkan atau ditolak.

Perubahan yang dilakukan pada kebijakan dapat memakan waktu antara 6 dan 8 menit untuk menjadi efektif karena cara AWS IoT menyimpan dokumen kebijakan. Artinya, mungkin perlu

beberapa menit untuk mengakses sumber daya yang baru-baru ini diberikan akses, dan sumber daya dapat diakses selama beberapa menit setelah aksesnya dicabut.

AWS IoT Core kebijakan dapat dilampirkan ke sertifikat X.509, identitas Amazon Cognito, dan grup hal. Kebijakan yang dilampirkan pada grup sesuatu berlaku untuk apa pun dalam grup itu. Agar kebijakan berlaku, nama `clientId` dan benda harus cocok. AWS IoT Core kebijakan mengikuti logika evaluasi kebijakan yang sama dengan kebijakan IAM. Secara default, semua kebijakan ditolak secara implisit. Izin eksplisit dalam kebijakan berbasis identitas atau berbasis sumber daya akan mengesampingkan perilaku default. Penolakan secara tegas dalam kebijakan apa pun akan mengesampingkan izin apa pun. Untuk informasi selengkapnya, lihat [Logika evaluasi kebijakan](#) di Panduan AWS Identity and Access Management Pengguna.

Topik

- [AWS IoT Core tindakan kebijakan](#)
- [AWS IoT Core sumber daya aksi](#)
- [AWS IoT Core variabel kebijakan](#)
- [Pencegahan confused deputy lintas layanan](#)
- [AWS IoT Core contoh kebijakan](#)
- [Otorisasi dengan identitas Amazon Cognito](#)

AWS IoT Core tindakan kebijakan

Tindakan kebijakan berikut ditentukan oleh AWS IoT Core:

Tindakan Kebijakan MQTT

`iot:Connect`

Merupakan izin untuk terhubung ke broker AWS IoT Core pesan. `iot:Connect` izin diperiksa setiap kali `CONNECT` permintaan dikirim ke broker. Broker pesan tidak mengizinkan dua klien dengan ID klien yang sama untuk tetap terhubung pada saat yang sama. Setelah klien kedua terhubung, broker menutup koneksi yang ada. Gunakan `iot:Connect` izin untuk memastikan hanya klien yang berwenang menggunakan ID klien tertentu yang dapat terhubung.

`iot:GetRetainedMessage`

Merupakan izin untuk mendapatkan isi dari satu pesan yang dipertahankan. Pesan yang disimpan adalah pesan yang diterbitkan dengan tanda `RETAIN` yang disetel dan disimpan

oleh AWS IoT Core. Untuk izin mendapatkan daftar semua pesan yang disimpan akun, lihat [iot:ListRetainedMessages](#).

iot:ListRetainedMessages

Merupakan izin untuk mengambil informasi ringkasan tentang pesan yang disimpan akun, tetapi bukan isi pesan. Pesan yang disimpan adalah pesan yang diterbitkan dengan tanda RETAIN yang disetel dan disimpan oleh AWS IoT Core. Sumber daya ARN yang ditentukan untuk tindakan ini harus. * Untuk izin mendapatkan isi dari satu pesan yang disimpan, lihat [iot:GetRetainedMessage](#).

iot:Publish

Merupakan izin untuk menerbitkan topik MQTT. Izin ini diperiksa setiap kali permintaan PUBLISH dikirim ke broker. Anda dapat menggunakan ini untuk memungkinkan klien mempublikasikan ke pola topik tertentu.

Note

Untuk memberikan `iot:Publish` izin, Anda juga harus memberikan `iot:Connect` izin.

iot:Receive

Merupakan izin untuk menerima pesan dari AWS IoT Core. `iot:Receive` izin dikonfirmasi setiap kali pesan dikirimkan ke klien. Karena izin ini diperiksa pada setiap pengiriman, Anda dapat menggunakannya untuk mencabut izin ke klien yang saat ini berlangganan topik.

iot:RetainPublish

Merupakan izin untuk mempublikasikan pesan MQTT dengan set flag RETAIN.

Note

Untuk memberikan `iot:RetainPublish` izin, Anda juga harus memberikan `iot:Publish` izin.

iot:Subscribe

Merupakan izin untuk berlangganan filter topik. Izin ini diperiksa setiap kali permintaan SUBSCRIBE dikirim ke broker. Gunakan untuk memungkinkan klien berlangganan topik yang sesuai dengan pola topik tertentu.

Note

Untuk memberikan `iot:Subscribe` izin, Anda juga harus memberikan `iot:Connect` izin.

Tindakan Kebijakan Device Shadow

iot:DeleteThingShadow

Merupakan izin untuk menghapus Device Shadow sesuatu. `iot:DeleteThingShadow` izin dicentang setiap kali permintaan dibuat untuk menghapus konten Device Shadow sesuatu.

iot:GetThingShadow

Merupakan izin untuk mengambil Device Shadow sesuatu. `iot:GetThingShadow` izin diperiksa setiap kali permintaan dibuat untuk mengambil konten Device Shadow sesuatu.

iot:ListNamedShadowsForThing

Merupakan izin untuk membuat daftar sesuatu bernama Shadows.

`iot:ListNamedShadowsForThing` izin diperiksa setiap kali permintaan dibuat untuk mencantumkan sesuatu yang bernama Shadows.

iot:UpdateThingShadow

Merupakan izin untuk memperbarui bayangan perangkat. `iot:UpdateThingShadow` izin diperiksa setiap kali permintaan dibuat untuk memperbarui konten Device Shadow sesuatu.

Note

Tindakan kebijakan eksekusi pekerjaan hanya berlaku untuk titik akhir HTTP TLS. Jika Anda menggunakan titik akhir MQTT, Anda harus menggunakan tindakan kebijakan MQTT yang ditentukan dalam topik ini.

Untuk contoh kebijakan pelaksanaan pekerjaan yang menunjukkan hal ini, lihat [the section called “Contoh kebijakan pekerjaan dasar”](#) yang berfungsi dengan protokol MQTT.

Tindakan AWS IoT Core Kebijakan Eksekusi Job

`iot:DescribeJobExecution`

Merupakan izin untuk mengambil eksekusi pekerjaan untuk hal tertentu.

`iot:DescribeJobExecution` izin diperiksa setiap kali permintaan dibuat untuk mendapatkan eksekusi pekerjaan.

`iot:GetPendingJobExecutions`

Merupakan izin untuk mengambil daftar pekerjaan yang tidak dalam status terminal untuk suatu hal. `iot:GetPendingJobExecutions` izin diperiksa setiap kali permintaan dibuat untuk mengambil daftar.

`iot:UpdateJobExecution`

Merupakan izin untuk memperbarui eksekusi pekerjaan. `iot:UpdateJobExecution` izin diperiksa setiap kali permintaan dibuat untuk memperbarui status eksekusi pekerjaan.

`iot:StartNextPendingJobExecution`

Merupakan izin untuk mendapatkan dan memulai eksekusi pekerjaan tertunda berikutnya untuk suatu hal. (Yaitu, untuk memperbarui eksekusi pekerjaan dengan status QUEUED ke IN_PROGRESS.) `iot:StartNextPendingJobExecution` izin diperiksa setiap kali permintaan dibuat untuk memulai eksekusi pekerjaan tertunda berikutnya.

AWS IoT Core Tindakan Kebijakan Penyedia Kredensyal

`iot:AssumeRoleWithCertificate`

Merupakan izin untuk memanggil penyedia AWS IoT Core kredensi untuk mengambil peran IAM dengan otentikasi berbasis sertifikat. `iot:AssumeRoleWithCertificate` izin diperiksa setiap kali permintaan dibuat ke penyedia AWS IoT Core kredensi untuk mengambil peran.

AWS IoT Core sumber daya aksi

Untuk menentukan sumber daya untuk tindakan AWS IoT Core kebijakan, gunakan Nama Sumber Daya Amazon (ARN) sumber daya. Semua ARN sumber daya mengikuti format berikut:

```
arn:partition:iot:region:AWS-account-ID:Resource-type/Resource-name
```

Tabel berikut menunjukkan sumber daya untuk menentukan untuk setiap jenis tindakan. Contoh ARN adalah untuk ID akun123456789012, di partisaws, dan khusus untuk wilayah tersebut. us-east-1 Untuk informasi selengkapnya tentang format ARN, lihat [Amazon Resource Names \(ARN\)](#) dari AWS Identity and Access Management Panduan Pengguna.

Tindakan	Tipe sumber daya	Nama sumber daya	Contoh ARN
iot:Connect	client	ID klien	arn:aws:iot:us-east-1:123456789012:client/myClientId
iot:DeleteThingShadow	thing	Nama benda, dan nama bayangan, jika berlaku	arn:aws:iot:us-east-1:123456789012:thing/thingOne arn:aws:iot:us-east-1:123456789012:thing/thingOne/shadowOne
iot:DescribeJobExecution	thing	Nama benda itu	arn:aws:iot:us-east-1:123456789012:thing/thingOne
iot:GetPendingJobExecutions	thing	Nama benda itu	arn:aws:iot:us-east-1:123456789012:thing/thingOne
iot:GetRetainedMessage	topic	Topik pesan yang dipertahankan	arn:aws:iot:us-east-1:123456789012:topic/myTopicName

Tindakan	Tipe sumber daya	Nama sumber daya	Contoh ARN
<code>iot:GetThingShadow</code>	<code>thing</code>	Nama benda, dan nama bayangan, jika berlaku	<code>arn:aws:iot:us-east-1:123456789012:thing/thingOne</code> <code>arn:aws:iot:us-east-1:123456789012:thing/thingOne/shadowOne</code>
<code>iot:ListNamedShadowsForThing</code>	Semua	Semua	*
<code>iot:ListRetainedMessages</code>	Semua	Semua	*
<code>iot:Publish</code>	<code>topic</code>	String topik	<code>arn:aws:iot:us-east-1:123456789012:topic/myTopicName</code>
<code>iot:Receive</code>	<code>topic</code>	String topik	<code>arn:aws:iot:us-east-1:123456789012:topic/myTopicName</code>
<code>iot:RetainPublish</code>	<code>topic</code>	Topik untuk dipublikasikan dengan set bendera RESTAIN	<code>arn:aws:iot:us-east-1:123456789012:topic/myTopicName</code>
<code>iot:StartNextPendingJobExecution</code>	<code>thing</code>	Nama benda itu	<code>arn:aws:iot:us-east-1:123456789012:thing/thingOne</code>

Tindakan	Tipe sumber daya	Nama sumber daya	Contoh ARN
<code>iot:Subscribe</code>	<code>topicfilter</code>	String filter topik	<code>arn:aws:iot:us-east-1:123456789012:topicfilter/myTopicFilter</code>
<code>iot:UpdateJobExecution</code>	<code>thing</code>	Nama benda itu	<code>arn:aws:iot:us-east-1:123456789012:thing/thingOne</code>
<code>iot:UpdateThingShadow</code>	<code>thing</code>	Nama benda, dan nama bayangan, jika berlaku	<code>arn:aws:iot:us-east-1:123456789012:thing/thingOne</code> <code>arn:aws:iot:us-east-1:123456789012:thing/thingOne/shadowOne</code>
<code>iot:AssumeRoleWithCertificate</code>	<code>rolealias</code>	Alias peran yang menunjuk ke peran ARN	<code>arn:aws:iot:us-east-1:123456789012:rolealias/CredentialProviderRole_alias</code>

AWS IoT Core variabel kebijakan

AWS IoT Core mendefinisikan variabel kebijakan yang dapat digunakan dalam AWS IoT Core kebijakan di `Resource` atau `Condition` blok. Ketika kebijakan dievaluasi, variabel kebijakan diganti dengan nilai aktual. Misalnya, jika perangkat terhubung ke broker AWS IoT Core pesan dengan ID klien 100-234-3456, variabel `iot:ClientId` kebijakan diganti dalam dokumen kebijakan dengan 100-234-3456.

AWS IoT Core kebijakan dapat menggunakan karakter wildcard dan mengikuti konvensi serupa dengan kebijakan IAM. Memasukkan `*` (asterik) dalam string dapat diperlakukan sebagai wildcard, cocok dengan karakter apa pun. Misalnya, Anda dapat menggunakan `*` untuk mendeskripsikan beberapa nama topik MQTT dalam `Resource` atribut kebijakan. Karakter `+` dan `#` diperlakukan

sebagai string literal dalam kebijakan. Untuk contoh kebijakan yang menunjukkan cara menggunakan wildcard, lihat [Menggunakan karakter wildcard di MQTT dan kebijakan AWS IoT Core](#).

Anda juga dapat menggunakan variabel kebijakan yang telah ditentukan dengan nilai tetap untuk mewakili karakter yang memiliki arti khusus. Karakter khusus ini termasuk `$ (*)`, `$ (?)`, dan `$ ($)`. Untuk informasi selengkapnya tentang variabel kebijakan dan karakter khusus, lihat [Elemen Kebijakan IAM: Variabel dan tag](#) dan [Membuat kondisi dengan beberapa kunci atau nilai](#).

Topik

- [Variabel AWS IoT Core kebijakan dasar](#)
- [Variabel kebijakan hal](#)
- [Variabel kebijakan Sertifikat X.509 AWS IoT Core](#)

Variabel AWS IoT Core kebijakan dasar

AWS IoT Core mendefinisikan variabel kebijakan dasar berikut:

- `iot:ClientId`: ID klien yang digunakan untuk terhubung ke broker AWS IoT Core pesan.
- `aws:SourceIp`: Alamat IP klien yang terhubung ke broker AWS IoT Core pesan.

AWS IoT Core Kebijakan berikut menunjukkan kebijakan yang menggunakan variabel kebijakan. `aws:SourceIp` dapat digunakan dalam elemen Kondisi kebijakan Anda untuk mengizinkan prinsipal membuat permintaan API hanya dalam rentang alamat tertentu. Sebagai contoh, lihat [Mengotorisasi pengguna dan layanan cloud untuk menggunakan AWS IoT Jobs](#).

```
{
  "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"
  }
}
}
]
}

```

Dalam contoh `${iot:ClientId}` ini, diganti dengan ID klien yang terhubung ke broker AWS IoT Core pesan ketika kebijakan dievaluasi. Bila Anda menggunakan variabel kebijakan seperti `${iot:ClientId}`, Anda dapat secara tidak sengaja membuka akses ke topik yang tidak diinginkan. Misalnya, jika Anda menggunakan kebijakan yang digunakan `${iot:ClientId}` untuk menentukan filter topik:

```

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

```

Klien dapat terhubung menggunakan `+` sebagai ID klien. Ini akan memungkinkan pengguna untuk berlangganan topik apa pun yang cocok dengan filter topik `my/+/topic`. Untuk melindungi dari kesenjangan keamanan tersebut, gunakan tindakan `iot:Connect` kebijakan untuk mengontrol ID klien mana yang dapat terhubung. Misalnya, kebijakan ini hanya mengizinkan klien yang ID kliennya `clientid1` terhubung:

```

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

```

```
    "Resource": [  
      "arn:aws:iot:us-east-1:123456789012:client/clientid1"  
    ]  
  }  
]
```

Note

Menggunakan variabel kebijakan `${iot:ClientId}` dengan tidak Connect disarankan. Tidak ada pemeriksaan pada nilai `ClientId`, sehingga lampiran dengan ID klien yang berbeda dapat lulus validasi tetapi menyebabkan pemutusan. Karena `ClientId` ada yang diizinkan, menyetel ID klien acak dapat melewati kebijakan grup hal.

Variabel kebijakan hal

Variabel kebijakan hal memungkinkan Anda menulis AWS IoT Core kebijakan yang memberikan atau menolak izin berdasarkan properti benda seperti nama benda, tipe benda, dan nilai atribut benda. Anda dapat menggunakan variabel kebijakan hal untuk menerapkan kebijakan yang sama untuk mengontrol banyak AWS IoT Core perangkat. Untuk informasi selengkapnya tentang penyediaan perangkat, lihat Penyediaan [Perangkat](#). Nama benda diperoleh dari ID klien dalam Connect pesan MQTT yang dikirim ketika sesuatu terhubung ke AWS IoT Core

Ingatlah hal berikut saat menggunakan variabel kebijakan hal dalam AWS IoT Core kebijakan.

- Gunakan [AttachThingPrincipal](#) API untuk melampirkan sertifikat atau prinsipal (identitas Amazon Cognito yang diautentikasi) ke suatu benda.
- Saat Anda mengganti nama benda dengan variabel kebijakan hal, nilai `clientId` dalam pesan koneksi MQTT atau koneksi TLS harus sama persis dengan nama benda.

Variabel kebijakan hal berikut tersedia:

- `iot:Connection.Thing.ThingName`

Ini menyelesaikan nama benda dalam AWS IoT Core registri tempat kebijakan sedang dievaluasi. AWS IoT Core menggunakan sertifikat yang disajikan perangkat saat mengautentikasi untuk menentukan hal mana yang akan digunakan untuk memverifikasi koneksi. Variabel kebijakan ini hanya tersedia jika perangkat terhubung melalui MQTT atau MQTT melalui protokol WebSocket

- `iot:Connection.Thing.ThingTypeName`

Ini menyelesaikan jenis hal yang terkait dengan hal yang kebijakan sedang dievaluasi. ID klien WebSocket MQTT/koneksi harus sama dengan nama benda. Variabel kebijakan ini hanya tersedia saat menghubungkan melalui MQTT atau MQTT melalui protokol. WebSocket

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

Ini menyelesaikan nilai atribut tertentu yang terkait dengan hal yang kebijakan sedang dievaluasi. Sesuatu dapat memiliki hingga 50 atribut. Setiap atribut tersedia sebagai variabel kebijakan:

`iot:Connection.Thing.Attributes[attributeName]` di mana *AttributeName* adalah nama atribut. ID klien WebSocket MQTT/koneksi harus sama dengan nama benda. Variabel kebijakan ini hanya tersedia saat menghubungkan melalui MQTT atau MQTT melalui protokol. WebSocket

- `iot:Connection.Thing.IsAttached`

`iot:Connection.Thing.IsAttached: ["true"]` Menegaskan bahwa hanya perangkat yang terdaftar AWS IoT dan dilampirkan pada prinsipal yang dapat mengakses izin di dalam kebijakan. [Anda dapat menggunakan variabel ini untuk mencegah perangkat terhubung AWS IoT Core jika menunjukkan sertifikat yang tidak dilampirkan ke benda IoT di AWS IoT Core registri. Variabel ini memiliki nilai `true` atau `false` menunjukkan bahwa hal yang menghubungkan dilampirkan ke sertifikat atau identitas Amazon Cognito di registri menggunakan `Principal API. AttachThing` Nama benda diambil sebagai ID klien.](#)

Variabel kebijakan Sertifikat X.509 AWS IoT Core

Variabel kebijakan sertifikat X.509 membantu dengan kebijakan penulisan. AWS IoT Core Kebijakan ini memberikan izin berdasarkan atribut sertifikat X.509. Bagian berikut menjelaskan cara menggunakan variabel kebijakan sertifikat ini.

Important

Jika sertifikat X.509 Anda tidak menyertakan atribut sertifikat tertentu tetapi variabel kebijakan sertifikat terkait digunakan dalam dokumen kebijakan Anda, evaluasi kebijakan dapat menyebabkan perilaku yang tidak terduga.

CertificateId

Di [RegisterCertificate](#) API, `certificateId` muncul di badan respons. Untuk mendapatkan informasi tentang sertifikat Anda, gunakan `certificateId` in [DescribeCertificate](#).

Atribut penerbit

Variabel AWS IoT Core kebijakan berikut mendukung mengizinkan atau menolak izin, berdasarkan atribut sertifikat yang ditetapkan oleh penerbit sertifikat.

- `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`

Atribut subjek

Variabel AWS IoT Core kebijakan berikut mendukung pemberian atau penolakan izin, berdasarkan atribut subjek sertifikat yang ditetapkan oleh penerbit sertifikat.

- `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`

Sertifikat X.509 menyediakan atribut ini dengan opsi untuk memuat satu atau lebih nilai. Secara default, variabel kebijakan untuk setiap atribut multi-nilai mengembalikan nilai pertama. Misalnya, `Certificate.Subject.Country` atribut mungkin berisi daftar nama negara, tetapi ketika dievaluasi dalam kebijakan, `iot:Certificate.Subject.Country` akan diganti dengan nama negara pertama.

Anda dapat meminta nilai atribut tertentu selain nilai pertama dengan menggunakan indeks berbasis satu. Misalnya, `iot:Certificate.Subject.Country.1` diganti dengan nama negara kedua di `Certificate.Subject.Country` atribut. Jika Anda menentukan nilai indeks yang tidak ada (misalnya, jika Anda meminta nilai ketiga ketika hanya ada dua nilai yang ditetapkan ke atribut), tidak ada substitusi yang dibuat dan otorisasi gagal. Anda dapat menggunakan `.List` akhiran pada nama variabel kebijakan untuk menentukan semua nilai atribut.

Atribut nama alternatif penerbit

Variabel AWS IoT Core kebijakan berikut mendukung pemberian atau penolakan izin, berdasarkan atribut nama alternatif penerbit yang ditetapkan oleh penerbit sertifikat.

- `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`

Atribut nama alternatif subjek

Variabel AWS IoT Core kebijakan berikut mendukung pemberian atau penolakan izin, berdasarkan atribut nama alternatif subjek yang ditetapkan oleh penerbit sertifikat.

- `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`

Atribut lainnya

Anda dapat menggunakan `iot:Certificate.SerialNumber` untuk mengizinkan atau menolak akses ke AWS IoT Core sumber daya, berdasarkan nomor seri sertifikat. Variabel `iot:Certificate.AvailableKeys` kebijakan berisi nama semua variabel kebijakan sertifikat yang berisi nilai.

Menggunakan variabel kebijakan sertifikat X.509

Topik ini memberikan rincian tentang cara menggunakan variabel kebijakan sertifikat. Variabel kebijakan sertifikat X.509 sangat penting saat Anda membuat AWS IoT Core kebijakan yang memberikan izin berdasarkan atribut sertifikat X.509. Jika sertifikat X.509 Anda tidak menyertakan atribut sertifikat tertentu tetapi variabel kebijakan sertifikat terkait digunakan dalam dokumen kebijakan Anda, evaluasi kebijakan dapat menyebabkan perilaku yang tidak terduga. Ini karena variabel kebijakan yang hilang tidak dievaluasi dalam pernyataan kebijakan.

Dalam topik ini:

- [Contoh sertifikat X.509](#)
- [Menggunakan atribut penerbit sertifikat sebagai variabel kebijakan sertifikat](#)
- [Menggunakan atribut subjek sertifikat sebagai variabel kebijakan sertifikat](#)
- [Menggunakan atribut nama alternatif Penerbit sertifikat sebagai variabel kebijakan sertifikat](#)
- [Menggunakan atribut nama alternatif subjek sertifikat sebagai variabel kebijakan sertifikat](#)
- [Menggunakan atribut sertifikat lain sebagai variabel kebijakan sertifikat](#)
- [X.509 Keterbatasan variabel kebijakan sertifikat](#)
- [Contoh kebijakan menggunakan variabel kebijakan sertifikat](#)

Contoh sertifikat X.509

Sertifikat X.509 yang khas mungkin muncul sebagai berikut. Sertifikat contoh ini mencakup atribut sertifikat. Selama evaluasi AWS IoT Core kebijakan, atribut sertifikat berikut akan diisi sebagai

variabel kebijakan sertifikat:Serial Number,,Issuer, SubjectX509v3 Issuer Alternative Name, danX509v3 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 >>

Menggunakan atribut penerbit sertifikat sebagai variabel kebijakan sertifikat

Tabel berikut memberikan rincian tentang bagaimana atribut penerbit sertifikat akan diisi dalam kebijakan. AWS IoT Core

Atribut penerbit yang akan diisi dalam kebijakan

Atribut penerbit sertifikat	Variabel kebijakan sertifikat
<ul style="list-style-type: none"> • C = KAMI • O = Perangkat IoT • OU = SmartHome • ST=WA • CN = Perangkat IoT CA Utama • GN = CA1 primer • inisiasi = XY • dnQualifier=Contoh corp • SN= SmartHome • judul=CA1 • pseudonym=Primer_CA • generationQualifier=2 • serialNumber=987 	<ul style="list-style-type: none"> • <code>iot:Certificate.Issuer.Country = US</code> • <code>iot:Certificate.Issuer.Organization = IoT Devices</code> • <code>iot:Certificate.Issuer.OrganizationalUnit = SmartHome</code> • <code>iot:Certificate.Issuer.State = WA</code> • <code>iot:Certificate.Issuer.CommonName = IoT Devices Primary CA</code> • <code>iot:Certificate.Issuer.GivenName = Primary CA1</code> • <code>iot:Certificate.Issuer.initials = XY</code> • <code>iot:Certificate.Issuer.Distinguished NameQualifier = Example corp</code> • <code>iot:Certificate.Issuer.Surname = SmartHome</code> • <code>iot:Certificate.Issuer.Title = CA1</code> • <code>iot:Certificate.Issuer.Pseudonym = Primary_CA</code> • <code>iot:Certificate.Issuer.GenerationQualifier = 2</code> • <code>iot:Certificate.Issuer.SerialNumber = 987</code>

Menggunakan atribut subjek sertifikat sebagai variabel kebijakan sertifikat

Tabel berikut memberikan rincian tentang bagaimana atribut subjek sertifikat akan diisi dalam AWS IoT Core kebijakan.

Atribut subjek yang akan diisi dalam kebijakan

Atribut subjek sertifikat	Variabel kebijakan sertifikat
<ul style="list-style-type: none"> • C = KAMI • O = Perangkat IoT • ST=NY • CN = LightBulb Sertifikat Perangkat • GN = bohlam • inisial=Zz • dnQualifier=Bulb001 • sn=Multi Warna • judul=RGB • Pseudonym=Perangkat RGB • generationQualifier=4 • serialNumber=123 	<ul style="list-style-type: none"> • <code>iot:Certificate.Subject.Country = US</code> • <code>iot:Certificate.Subject.Organization = IoT Devices</code> • <code>iot:Certificate.Subject.State = NY</code> • <code>iot:Certificate.Subject.CommonName = LightBulb Device Cert</code> • <code>iot:Certificate.Subject.GivenName = Bulb</code> • <code>iot:Certificate.Subject.initials = ZZ</code> • <code>iot:Certificate.Subject.DistinguishedNameQualifier = Bulb001</code> • <code>iot:Certificate.Subject.Surname = Multi Color</code> • <code>iot:Certificate.Subject.Title = RGB</code> • <code>iot:Certificate.Subject.Pseudonym = RGB Device</code> • <code>iot:Certificate.Subject.GenerationQualifier = 4</code> • <code>iot:Certificate.Subject.SerialNumber = 123</code>

Menggunakan atribut nama alternatif Penerbit sertifikat sebagai variabel kebijakan sertifikat

Tabel berikut memberikan rincian tentang bagaimana atribut nama alternatif penerbit sertifikat akan diisi dalam kebijakan. AWS IoT Core

Atribut nama alternatif penerbit yang akan diisi dalam kebijakan

Nama Alternatif Penerbit X509v3	Atribut dalam kebijakan
<ul style="list-style-type: none"> • DNS:issuer.com • Alamat IP:5.6.7.8 	<ul style="list-style-type: none"> • <code>iot:Certificate.Issuer.AlternativeName.DNSName = issuer.com</code>

Nama Alternatif Penerbit X509v3	Atribut dalam kebijakan
<ul style="list-style-type: none"> JENIS: PrimarySigner CA E-mail: primary@issuer.com DirName: /c=us/o=issuer/ou=Perangkat IoT/CN=Penerbit Utama CA 	<ul style="list-style-type: none"> 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

Menggunakan atribut nama alternatif subjek sertifikat sebagai variabel kebijakan sertifikat

Tabel berikut memberikan rincian tentang bagaimana atribut nama alternatif subjek sertifikat akan diisi dalam AWS IoT Core kebijakan.

Atribut nama alternatif subjek yang akan diisi dalam kebijakan

X509v3 Nama Alternatif Subjek	Atribut dalam kebijakan
<ul style="list-style-type: none"> DNS:example.com Alamat IP:1.2.3.4 JENIS: ResourceIdentifier001 E-mail: device1@example.com DirName: /c = us/o = IoT/OU = /CN = Sertifikat SmartHome LightBulb 	<ul style="list-style-type: none"> 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

Menggunakan atribut sertifikat lain sebagai variabel kebijakan sertifikat

Tabel berikut memberikan rincian tentang bagaimana atribut sertifikat lainnya akan diisi dalam AWS IoT Core kebijakan.

Atribut lain yang akan diisi dalam kebijakan

Atribut sertifikat lainnya	Variabel kebijakan sertifikat
Serial Number: 92:12:85:cb:b7:a5: e0:86	<code>iot:Certificate.SerialNumber = 105256223 89124227206</code>

X.509 Keterbatasan variabel kebijakan sertifikat

Batasan berikut berlaku untuk variabel kebijakan sertifikat X.509:

Variabel kebijakan yang hilang

Jika sertifikat X.509 Anda tidak menyertakan atribut sertifikat tertentu tetapi variabel kebijakan sertifikat terkait digunakan dalam dokumen kebijakan Anda, evaluasi kebijakan dapat menyebabkan perilaku yang tidak terduga. Ini karena variabel kebijakan yang hilang tidak dievaluasi dalam pernyataan kebijakan.

SerialNumber Format sertifikat

AWS IoT Core memperlakukan nomor seri sertifikat sebagai representasi string dari bilangan bulat desimal. Misalnya, jika kebijakan hanya mengizinkan koneksi dengan ID Klien yang cocok dengan nomor seri sertifikat, ID klien harus berupa nomor seri dalam format desimal.

Wildcard

Jika karakter wildcard hadir dalam atribut sertifikat, variabel kebijakan tidak diganti dengan nilai atribut sertifikat. Ini akan meninggalkan `${policy-variable}` teks dalam dokumen kebijakan. Hal ini dapat menyebabkan kegagalan otorisasi. Karakter wildcard berikut dapat digunakan: *, \$, +?, dan #.

Bidang array

Atribut sertifikat yang berisi array dibatasi hingga lima item. Item tambahan diabaikan.

Panjang tali

Semua nilai string dibatasi hingga 1024 karakter. Jika atribut sertifikat berisi string yang lebih panjang dari 1024 karakter, variabel kebijakan tidak diganti dengan nilai atribut sertifikat. Ini akan meninggalkan `${policy-variable}` dokumen kebijakan. Hal ini dapat menyebabkan kegagalan otorisasi.

Karakter khusus

Setiap karakter khusus, seperti `,`, `"`, `\`, `+`, `=`, `<`, `>` dan `;` harus diawali dengan garis miring terbalik (`\`) bila digunakan dalam variabel kebijakan. Misalnya, `Amazon Web Services O=Amazon.com Inc. L=Seattle ST=Washington C=US` menjadi `Amazon Web Service O\=Amazon.com Inc. L\=Seattle ST\=Washington C\=US`.

Contoh kebijakan menggunakan variabel kebijakan sertifikat

Dokumen kebijakan berikut memungkinkan koneksi dengan ID klien yang cocok dengan nomor seri sertifikat dan penerbitan ke topik yang cocok dengan pola: `${iot:Certificate.Subject.Organization}/device-stats/${iot:ClientId}/*`.

Important

Jika sertifikat X.509 Anda tidak menyertakan atribut sertifikat tertentu tetapi variabel kebijakan sertifikat terkait digunakan dalam dokumen kebijakan Anda, evaluasi kebijakan dapat menyebabkan perilaku yang tidak terduga. Ini karena variabel kebijakan yang hilang tidak dievaluasi dalam pernyataan kebijakan. Misalnya, jika Anda melampirkan dokumen kebijakan berikut ke sertifikat yang tidak berisi `iot:Certificate.Subject.Organization` atribut, variabel kebijakan `iot:Certificate.Subject.Organization` sertifikat tidak akan diisi selama evaluasi kebijakan.

```
{
  "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}/*"
    ]
  }
]
}

```

Anda juga dapat menggunakan [operator kondisi Null](#) untuk memastikan bahwa variabel kebijakan sertifikat yang digunakan dalam kebijakan diisi selama evaluasi kebijakan. Dokumen kebijakan berikut mengizinkan `iot:Connect` sertifikat hanya jika terdapat atribut Nama umum Sertifikat Sertifikat dan Subjek Sertifikat.

Semua variabel kebijakan sertifikat memiliki nilai String, sehingga semua [operator kondisi String](#) didukung.

```

{
  "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"
        }
      }
    }
  ]
}

```

```
]
}
```

Pencegahan confused deputy lintas layanan

Masalah confused deputy adalah masalah keamanan saat entitas yang tidak memiliki izin untuk melakukan suatu tindakan dapat memaksa entitas yang lebih berhak untuk melakukan tindakan tersebut. Pada tahun AWS, peniruan lintas layanan dapat mengakibatkan masalah wakil yang membingungkan. Peniruan identitas lintas layanan dapat terjadi ketika satu layanan (layanan yang dipanggil) memanggil layanan lain (layanan yang dipanggil). Layanan panggilan dapat dimanipulasi untuk menggunakan izinnya untuk bertindak atas sumber daya pelanggan lain dengan cara yang seharusnya tidak memiliki izin untuk mengakses. Untuk mencegah hal ini, AWS menyediakan alat yang membantu Anda melindungi data untuk semua layanan dengan pengguna utama layanan yang telah diberi akses ke sumber daya di akun Anda.

Untuk membatasi izin yang AWS IoT memberikan layanan lain ke sumber daya, sebaiknya gunakan kunci konteks kondisi `aws:SourceAccount` global `aws:SourceArn` dan global dalam kebijakan sumber daya. Jika Anda menggunakan kedua kunci konteks kondisi global, `aws:SourceAccount` nilai dan akun dalam `aws:SourceArn` nilai harus menggunakan ID akun yang sama saat digunakan dalam pernyataan kebijakan yang sama.

Cara paling efektif untuk melindungi dari masalah wakil yang membingungkan adalah dengan menggunakan kunci konteks kondisi `aws:SourceArn` global dengan Nama Sumber Daya Amazon (ARN) lengkap dari sumber daya. Untuk AWS IoT, Anda `aws:SourceArn` harus mematuhi format: `arn:aws:iot:region:account-id:*`. Pastikan *wilayah tersebut cocok dengan AWS IoT Wilayah* Anda dan id *akun sesuai dengan ID* akun pelanggan Anda.

Contoh berikut menunjukkan bagaimana mencegah masalah wakil yang bingung dengan menggunakan kunci konteks kondisi `aws:SourceAccount` global `aws:SourceArn` dan dalam kebijakan kepercayaan AWS IoT peran.

```
{
  "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:*"
  }
}
]
```

AWS IoT Core contoh kebijakan

Contoh kebijakan di bagian ini menggambarkan dokumen kebijakan yang digunakan untuk menyelesaikan tugas-tugas umum di AWS IoT Core bagian ini. Anda dapat menggunakannya sebagai contoh untuk memulai saat membuat kebijakan untuk solusi Anda.

Contoh di bagian ini menggunakan elemen kebijakan ini:

- [the section called “AWS IoT Core tindakan kebijakan”](#)
- [the section called “AWS IoT Core sumber daya aksi”](#)
- [the section called “Contoh kebijakan berbasis identitas”](#)
- [the section called “Variabel AWS IoT Core kebijakan dasar”](#)
- [the section called “Variabel kebijakan Sertifikat X.509 AWS IoT Core”](#)

Contoh kebijakan di bagian ini:

- [Contoh kebijakan Connect](#)
- [Contoh kebijakan Publikasi/Berlangganan](#)
- [Connect dan publikasikan contoh kebijakan](#)
- [Contoh kebijakan pesan yang dipertahankan](#)
- [Contoh kebijakan sertifikat](#)
- [Contoh kebijakan hal](#)
- [Contoh kebijakan pekerjaan dasar](#)

Contoh kebijakan Connect

Kebijakan berikut menolak izin ke ID klien `client1` dan `client2` untuk terhubung AWS IoT Core, sementara memungkinkan perangkat untuk terhubung menggunakan ID klien. ID klien cocok dengan nama benda yang terdaftar di AWS IoT Core registri dan dilampirkan ke prinsipal yang digunakan untuk koneksi:

Note

Untuk perangkat terdaftar, kami menyarankan Anda menggunakan [variabel kebijakan hal](#) untuk Connect tindakan dan melampirkan benda ke prinsipal yang digunakan untuk koneksi.

```
{
  "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"
        }
      }
    }
  ]
}
```

```
}
```

Kebijakan berikut memberikan izin untuk terhubung AWS IoT Core dengan ID `client1` klien. Contoh kebijakan ini untuk perangkat yang tidak terdaftar.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect"
      ],
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:client/client1"
      ]
    }
  ]
}
```

Contoh kebijakan sesi persisten MQTT

`connectAttributes` memungkinkan Anda untuk menentukan atribut apa yang ingin Anda gunakan dalam pesan `connect` Anda dalam kebijakan IAM Anda seperti `PersistentConnect` dan `LastWill`. Untuk informasi selengkapnya, lihat [Menggunakan ConnectAttributes](#).

Kebijakan berikut memungkinkan terhubung dengan `PersistentConnect` fitur:

```
{
  "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"
          ]
        }
      }
    }
  ]
}
```

```
}  
}  
]  
}
```

Kebijakan berikut ini melarang `PersistentConnect`, fitur lain diperbolehkan:

```
{  
  "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"  
          ]  
        }  
      }  
    }  
  ]  
}
```

Kebijakan di atas juga dapat dinyatakan menggunakan `StringEquals`, fitur lain termasuk fitur baru diperbolehkan:

```
{  
  "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": "arn:aws:iot:us-east-1:123456789012:client/client1",  
    }  
  ]  
}
```

```

    "iot:Connect"
  ],
  "Resource": "*",
  "Condition": {
    "ForAnyValue:StringEquals": {
      "iot:ConnectAttributes": [
        "PersistentConnect"
      ]
    }
  }
}
]
}

```

Kebijakan berikut memungkinkan koneksi oleh keduanya `PersistentConnect` dan `LastWill`, fitur baru lainnya tidak diperbolehkan:

```

{
  "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"
          ]
        }
      }
    }
  ]
}

```

Kebijakan berikut memungkinkan koneksi bersih oleh klien dengan atau tanpa `LastWill`, tidak ada fitur lain yang diizinkan:

```

{

```

```

"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": [
          "LastWill"
        ]
      }
    }
  }
]
}

```

Kebijakan berikut hanya mengizinkan koneksi menggunakan fitur default:

```

{
  "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": []
        }
      }
    }
  ]
}

```

Kebijakan berikut hanya mengizinkan koneksi dengan `PersistentConnect`, fitur baru apa pun diizinkan selama koneksi menggunakan `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"
        ]
      }
    }
  }
]
}

```

Kebijakan berikut menyatakan koneksi harus memiliki keduanya PersistentConnect dan LastWill penggunaan, tidak ada fitur baru yang diizinkan:

```

{
  "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": []
    }
  }
}
]
}

```

Kebijakan berikut tidak boleh dimiliki PersistentConnect tetapi dapat dimiliki LastWill, fitur baru lainnya tidak diperbolehkan:

```

{
  "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"
          ]
        }
      }
    }
  ]
}

```

Kebijakan berikut memungkinkan terhubung hanya oleh klien yang memiliki topik LastWill dengan "my/lastwill/topicName", fitur apa pun diizinkan selama menggunakan LastWill topik:

```

{
  "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:region:account-id:topic/my/
lastwill/topicName"
      }
    }
  ]
}

```

Kebijakan berikut hanya mengizinkan koneksi bersih menggunakan fitur tertentu `LastWillTopic`, fitur apa pun diizinkan selama menggunakan `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:region:account-id:topic/my/
lastwill/topicName"
        }
      }
    },
    {
      "Effect": "Deny",
      "Action": [
        "iot:Connect"
      ],
      "Resource": "*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "iot:ConnectAttributes": [

```

```
    "PersistentConnect"  
  ]  
} ]  
}
```

Contoh kebijakan Publikasi/Berlangganan

Kebijakan yang Anda gunakan bergantung pada cara Anda terhubung AWS IoT Core. Anda dapat terhubung AWS IoT Core dengan menggunakan klien MQTT, HTTP, atau WebSocket. Saat Anda terhubung dengan klien MQTT, Anda mengautentikasi dengan sertifikat X.509. Saat Anda terhubung melalui HTTP atau WebSocket protokol, Anda mengautentikasi dengan Signature Version 4 dan Amazon Cognito.

Note

Untuk perangkat terdaftar, kami menyarankan Anda menggunakan [variabel kebijakan hal](#) untuk Connect tindakan dan melampirkan benda ke prinsipal yang digunakan untuk koneksi.

Di bagian ini:

- [Menggunakan karakter wildcard di MQTT dan kebijakan AWS IoT Core](#)
- [Kebijakan untuk mempublikasikan, berlangganan, dan menerima pesan ke/dari topik tertentu](#)
- [Kebijakan untuk mempublikasikan, berlangganan, dan menerima pesan ke/dari topik dengan awalan tertentu](#)
- [Kebijakan untuk mempublikasikan, berlangganan, dan menerima pesan ke/dari topik khusus untuk setiap perangkat](#)
- [Kebijakan untuk mempublikasikan, berlangganan, dan menerima pesan ke/dari topik dengan atribut benda dalam nama topik](#)
- [Kebijakan untuk menolak mempublikasikan pesan ke subtopik nama topik](#)
- [Kebijakan untuk menolak menerima pesan dari subtopik nama topik](#)
- [Kebijakan untuk berlangganan topik menggunakan karakter wildcard MQTT](#)
- [Kebijakan untuk HTTP dan WebSocket klien](#)

Menggunakan karakter wildcard di MQTT dan kebijakan AWS IoT Core

MQTT dan AWS IoT Core kebijakan memiliki karakter wildcard yang berbeda dan Anda harus memilihnya setelah mempertimbangkan dengan cermat. Di MQTT, karakter wildcard + dan # digunakan dalam [filter topik MQTT untuk berlangganan beberapa nama topik](#). AWS IoT Core kebijakan menggunakan * dan ? sebagai karakter wildcard dan mengikuti konvensi kebijakan [IAM](#). Dalam dokumen kebijakan, * mewakili kombinasi karakter dan tanda tanya ? mewakili karakter tunggal apa pun. Dalam dokumen kebijakan, karakter wildcard MQTT, + dan # diperlakukan sebagai karakter tanpa arti khusus. Untuk menjelaskan beberapa nama topik dan filter topik dalam resource atribut kebijakan, gunakan karakter * dan ? wildcard sebagai pengganti karakter wildcard MQTT.

Saat Anda memilih karakter wildcard yang akan digunakan dalam dokumen kebijakan, pertimbangkan bahwa * karakter tersebut tidak terbatas pada satu tingkat topik. +Karakter terbatas pada tingkat topik tunggal dalam filter topik MQTT. Untuk membantu membatasi spesifikasi wildcard ke satu tingkat filter topik MQTT, pertimbangkan untuk menggunakan beberapa karakter. ? Untuk informasi selengkapnya tentang penggunaan karakter wildcard dalam sumber daya kebijakan dan contoh lainnya tentang apa yang cocok, lihat [Menggunakan wildcard di ARN sumber daya](#).

Tabel di bawah ini menunjukkan karakter wildcard berbeda yang digunakan dalam MQTT dan AWS IoT Core kebijakan untuk klien MQTT.

Karakter wildcard	Apakah karakter wildcard MQTT	Contoh di MQTT	Apakah karakter wildcard AWS IoT Core kebijakan	Contoh dalam AWS IoT Core kebijakan untuk klien MQTT
#	Ya	some/#	Tidak	N/A
+	Ya	some/+/topic	Tidak	N/A
*	Tidak	N/A	Ya	topicfilter/some/*/topic topicfilter/some/sensor*/topic
?	Tidak	N/A	Ya	topic/some/?????/topic

Karakter wildcard	Apakah karakter wildcard MQTT	Contoh di MQTT	Apakah karakter wildcard AWS IoT Core kebijakan	Contoh dalam AWS IoT Core kebijakan untuk klien MQTT
				topicfilter/some/sensor??#/topic

Kebijakan untuk mempublikasikan, berlangganan, dan menerima pesan ke/dari topik tertentu

Berikut ini menunjukkan contoh untuk perangkat terdaftar dan tidak terdaftar untuk menerbitkan, berlangganan dan menerima pesan ke/dari topik bernama "some_specific_topic". Contoh juga menyoroti itu Publish dan Receive menggunakan "topik" sebagai sumber daya, dan Subscribe menggunakan "topicfilter" sebagai sumber daya.

Registered devices

Untuk perangkat yang terdaftar di AWS IoT Core registri, kebijakan berikut memungkinkan perangkat terhubung dengan ClientID yang cocok dengan nama sesuatu di registri. Ini juga menyediakan Publish, Subscribe dan Receive izin untuk topik bernama "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

Untuk perangkat yang tidak terdaftar dalam AWS IoT Core registri, kebijakan berikut memungkinkan perangkat untuk terhubung menggunakan ClientID1, ClientID2 atau ClientID3. Ini juga menyediakan Publish, Subscribe dan Receive izin untuk topik bernama "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"
    ]
  }
]
```

Kebijakan untuk mempublikasikan, berlangganan, dan menerima pesan ke/dari topik dengan awalan tertentu

Berikut ini menunjukkan contoh untuk perangkat terdaftar dan tidak terdaftar untuk menerbitkan, berlangganan dan menerima pesan ke/dari topik yang diawali dengan "topic_prefix".

Note

Perhatikan penggunaan karakter wildcard * dalam contoh ini. Meskipun * berguna untuk memberikan izin untuk beberapa nama topik dalam satu pernyataan, ini dapat menyebabkan konsekuensi yang tidak diinginkan dengan memberikan lebih banyak hak istimewa ke perangkat daripada yang diperlukan. Jadi kami menyarankan Anda hanya menggunakan karakter wildcard * setelah mempertimbangkan dengan cermat.

Registered devices

Untuk perangkat yang terdaftar di AWS IoT Core registri, kebijakan berikut memungkinkan perangkat terhubung dengan ClientID yang cocok dengan nama sesuatu di registri. Ini juga menyediakan Publish, Subscribe dan Receive izin untuk topik yang diawali dengan "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

Untuk perangkat yang tidak terdaftar dalam AWS IoT Core registri, kebijakan berikut memungkinkan perangkat untuk terhubung menggunakan ClientID1, ClientID2 atau ClientID3. Ini juga menyediakan Publish, Subscribe dan Receive izin untuk topik yang diawali dengan "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*"
    ]
  }
]
}

```

Kebijakan untuk mempublikasikan, berlangganan, dan menerima pesan ke/dari topik khusus untuk setiap perangkat

Berikut ini menunjukkan contoh untuk perangkat terdaftar dan tidak terdaftar untuk menerbitkan, berlangganan dan menerima pesan ke/dari topik yang khusus untuk perangkat yang diberikan.

Registered devices

Untuk perangkat yang terdaftar di AWS IoT Core registri, kebijakan berikut memungkinkan perangkat terhubung dengan ClientID yang cocok dengan nama sesuatu di registri. Ini memberikan izin untuk mempublikasikan ke topik spesifik (`sensor/device/${iot:Connection.Thing.ThingName}`) dan juga berlangganan dan menerima dari topik khusus hal (`command/device/${iot:Connection.Thing.ThingName}`). Jika nama benda dalam registri adalah `thing1`, perangkat akan dapat mempublikasikan ke topik `sensor/perangkat/halang1`. Perangkat juga akan dapat berlangganan dan menerima dari topik `command/device/thing1`.

```

{
  "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

Untuk perangkat yang tidak terdaftar dalam AWS IoT Core registri, kebijakan berikut memungkinkan perangkat untuk terhubung menggunakan ClientID1, ClientId2 atau ClientID3. Ini memberikan izin untuk mempublikasikan ke topik khusus klien (`sensor/device/${iot:ClientId}`), dan juga berlangganan dan menerima dari topik khusus klien (`command/device/${iot:ClientId}`). Jika perangkat terhubung dengan ClientID sebagai ClientId1, itu akan dapat mempublikasikan ke topik "Sensor/perangkat/clientid1". Perangkat juga akan dapat berlangganan dan menerima dari topik `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}"
    ]
  }
]
}

```

Kebijakan untuk mempublikasikan, berlangganan, dan menerima pesan ke/dari topik dengan atribut benda dalam nama topik

Berikut ini menunjukkan contoh untuk perangkat terdaftar untuk menerbitkan, berlangganan dan menerima pesan ke/dari topik yang namanya termasuk atribut hal.

Note

Atribut benda hanya ada untuk perangkat yang terdaftar di AWS IoT Core Registry. Tidak ada contoh yang sesuai untuk perangkat yang tidak terdaftar.

Registered devices

Untuk perangkat yang terdaftar di AWS IoT Core registri, kebijakan berikut memungkinkan perangkat terhubung dengan ClientID yang cocok dengan nama sesuatu di registri. Ini memberikan izin untuk mempublikasikan ke topik (`sensor/${iot:Connection.Thing.Attributes[version]}`), dan berlangganan dan menerima dari topik (`command/${iot:Connection.Thing.Attributes[location]}`) di mana nama topik mencakup atribut hal. Jika nama benda dalam registri memiliki `version=v1` dan `location=Seattle`, perangkat akan dapat mempublikasikan ke topik "sensor/v1", dan berlangganan dan terima dari topik "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

Karena atribut thing hanya ada untuk perangkat yang terdaftar di AWS IoT Core registri, tidak ada contoh yang sesuai untuk hal-hal yang tidak terdaftar.

Kebijakan untuk menolak mempublikasikan pesan ke subtopik nama topik

Berikut ini menunjukkan contoh untuk perangkat terdaftar dan tidak terdaftar untuk mempublikasikan pesan ke semua topik kecuali subtopik tertentu.

Registered devices

Untuk perangkat yang terdaftar di AWS IoT Core registri, kebijakan berikut memungkinkan perangkat terhubung dengan ClientID yang cocok dengan nama sesuatu di registri. Ini memberikan izin untuk mempublikasikan ke semua topik yang diawali dengan "departemen/" tetapi tidak ke subtopik "departemen/admin".

```

{
  "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

Untuk perangkat yang tidak terdaftar dalam AWS IoT Core registri, kebijakan berikut memungkinkan perangkat untuk terhubung menggunakan ClientID1, ClientId2 atau ClientID3. Ini memberikan izin untuk mempublikasikan ke semua topik yang diawali dengan “departemen/” tetapi tidak ke subtopik “departemen/admin”.

```

{
  "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"
    ]
  }
]
}

```

Kebijakan untuk menolak menerima pesan dari subtopik nama topik

Berikut ini menunjukkan contoh untuk perangkat terdaftar dan tidak terdaftar untuk berlangganan dan menerima pesan dari topik dengan awalan tertentu kecuali subtopik tertentu.

Registered devices

Untuk perangkat yang terdaftar di AWS IoT Core registri, kebijakan berikut memungkinkan perangkat terhubung dengan ClientID yang cocok dengan nama sesuatu di registri. Kebijakan ini memungkinkan perangkat untuk berlangganan topik apa pun yang diawali dengan “topic_prefix”. Dengan menggunakan NotResource dalam pernyataan untuk `iot:Receive`, kami mengizinkan perangkat menerima pesan dari semua topik yang telah dilangganani perangkat, kecuali topik yang diawali dengan “topic_prefix/restricted”. Misalnya, dengan kebijakan ini, perangkat dapat berlangganan “topic_prefix/topic1” dan bahkan “topic_prefix/restricted”, namun, mereka hanya akan menerima pesan dari topik “topic_prefix/topic1” dan tidak ada pesan dari topik “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

Untuk perangkat yang tidak terdaftar dalam AWS IoT Core registri, kebijakan berikut memungkinkan perangkat untuk terhubung menggunakan ClientID1, ClientID2 atau ClientID3. Kebijakan ini memungkinkan perangkat untuk berlangganan topik apa pun yang diawali dengan "topic_prefix". Dengan menggunakan NotResource dalam pernyataan untuk `iot:Receive`, kami mengizinkan perangkat menerima pesan dari semua topik yang telah dilanggan perangkat, kecuali topik yang diawali dengan "topic_prefix/restricted". Misalnya, dengan kebijakan ini, perangkat dapat berlangganan "topic_prefix/topic1" dan bahkan "topic_prefix/restricted". Namun, mereka hanya akan menerima pesan dari topik "topic_prefix/topic1" dan tidak ada pesan dari topik "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/*"
  }
]
}

```

Kebijakan untuk berlangganan topik menggunakan karakter wildcard MQTT

Karakter wildcard MQTT + dan # diperlakukan sebagai string literal, tetapi mereka tidak diperlakukan sebagai wildcard saat digunakan dalam kebijakan. AWS IoT Core Di MQTT, + dan # diperlakukan sebagai wildcard hanya saat berlangganan filter topik tetapi sebagai string literal di semua konteks lainnya. Kami menyarankan Anda hanya menggunakan wildcard MQTT ini sebagai bagian dari AWS IoT Core kebijakan setelah mempertimbangkan dengan cermat.

Berikut ini menunjukkan contoh untuk hal-hal terdaftar dan tidak terdaftar menggunakan wildcard MQTT dalam kebijakan. AWS IoT Core Wildcard ini diperlakukan sebagai string literal.

Registered devices

Untuk perangkat yang terdaftar di AWS IoT Core registri, kebijakan berikut memungkinkan perangkat terhubung dengan ClientID yang cocok dengan nama sesuatu di registri. Kebijakan ini memungkinkan perangkat untuk berlangganan topik “departemen+/karyawan” dan “lokasi/

#". Karena + dan # diperlakukan sebagai string literal dalam AWS IoT Core kebijakan, perangkat dapat berlangganan topik "departemen+/karyawan" tetapi tidak ke topik "departemen/teknik/karyawan". Demikian pula, perangkat dapat berlangganan topik "lokasi/#" tetapi tidak ke topik "Lokasi/Seattle". Namun, setelah perangkat berlangganan topik "departemen+/karyawan", kebijakan akan memungkinkan mereka untuk menerima pesan dari topik "departemen/teknik/karyawan". Demikian pula, setelah perangkat berlangganan topik "lokasi/#", mereka akan menerima pesan dari topik "Lokasi/Seattle" juga.

```
{
  "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

Untuk perangkat yang tidak terdaftar dalam AWS IoT Core registri, kebijakan berikut memungkinkan perangkat untuk terhubung menggunakan ClientID1, ClientId2 atau ClientID3. Kebijakan ini memungkinkan perangkat untuk berlangganan topik “departemen+/karyawan” dan “lokasi/#”. Karena + dan # diperlakukan sebagai string literal dalam AWS IoT Core kebijakan, perangkat dapat berlangganan topik “departemen+/karyawan” tetapi tidak ke topik “departemen/teknik/karyawan”. Demikian pula, perangkat dapat berlangganan topik “lokasi/#” tetapi tidak “lokasi/Seattle”. Namun, setelah perangkat berlangganan topik “departemen+/karyawan”, kebijakan akan memungkinkan mereka untuk menerima pesan dari topik “departemen/teknik/karyawan”. Demikian pula, setelah perangkat berlangganan topik “lokasi/#”, mereka akan menerima pesan dari topik “Lokasi/Seattle” juga.

```
{
  "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/*"
    }
]
}
```

Kebijakan untuk HTTP dan WebSocket klien

Saat Anda terhubung melalui HTTP atau WebSocket protokol, Anda mengautentikasi dengan Signature Version 4 dan Amazon Cognito. Identitas Amazon Cognito dapat diautentikasi atau tidak diautentikasi. Identitas terautentikasi adalah milik pengguna yang diautentikasi oleh penyedia identitas yang didukung. Identitas yang tidak diautentikasi biasanya milik pengguna tamu yang tidak mengautentikasi dengan penyedia identitas. Amazon Cognito menyediakan pengenal dan AWS kredensial unik untuk mendukung identitas yang tidak diautentikasi. Untuk informasi selengkapnya, lihat [the section called “Otorisasi dengan identitas Amazon Cognito”](#).

Untuk operasi berikut, AWS IoT Core gunakan AWS IoT Core kebijakan yang dilampirkan pada identitas Amazon Cognito melalui API. AttachPolicy Ini mencakup izin yang dilampirkan ke kumpulan Identitas Amazon Cognito dengan identitas yang diautentikasi.

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

Itu berarti Identitas Amazon Cognito memerlukan izin dari kebijakan peran IAM dan kebijakan tersebut. AWS IoT Core Anda melampirkan kebijakan peran IAM ke kumpulan dan AWS IoT Core kebijakan ke Identitas Amazon Cognito melalui AWS IoT Core AttachPolicy API.

Pengguna yang diautentikasi dan tidak diautentikasi adalah jenis identitas yang berbeda. Jika Anda tidak melampirkan AWS IoT kebijakan ke Identitas Amazon Cognito, pengguna yang diautentikasi akan gagal melakukan otorisasi AWS IoT dan tidak memiliki akses ke AWS IoT sumber daya dan tindakan.

Note

Untuk AWS IoT Core operasi lain atau untuk identitas yang tidak diautentikasi, AWS IoT Core tidak mencakup izin yang dilampirkan ke peran kumpulan identitas Amazon Cognito. Untuk identitas yang diautentikasi dan tidak diautentikasi, ini adalah kebijakan paling permisif yang kami sarankan Anda lampirkan ke peran kumpulan Amazon Cognito.

HTTP

Untuk mengizinkan identitas Amazon Cognito yang tidak diautentikasi memublikasikan pesan melalui HTTP pada topik khusus Identitas Amazon Cognito, lampirkan kebijakan IAM berikut ke peran kumpulan Identitas 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}"]
    }
  ]
}
```

Untuk mengizinkan pengguna yang diautentikasi, lampirkan kebijakan sebelumnya ke peran kumpulan Identitas Amazon Cognito dan ke Identitas Amazon Cognito menggunakan API. AWS IoT Core [AttachPolicy](#)

Note

Saat mengotorisasi identitas Amazon Cognito AWS IoT Core, pertimbangkan kebijakan dan berikan hak istimewa paling sedikit yang ditentukan. Tindakan hanya diperbolehkan jika kedua kebijakan mengizinkan tindakan yang diminta. Jika salah satu kebijakan melarang tindakan, tindakan itu tidak sah.

MQTT

Untuk mengizinkan identitas Amazon Cognito yang tidak diautentikasi memublikasikan pesan WebSocket MQTT tentang topik khusus Identitas Amazon Cognito di akun Anda, lampirkan kebijakan IAM berikut ke peran kumpulan Identitas Amazon Cognito:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish"
      ],
      "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/${cognito-identity.amazonaws.com:sub}"]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect"
      ],
      "Resource": ["arn:aws:iot:us-east-1:123456789012:client/${cognito-identity.amazonaws.com:sub}"]
    }
  ]
}
```

Untuk mengizinkan pengguna yang diautentikasi, lampirkan kebijakan sebelumnya ke peran kumpulan Identitas Amazon Cognito dan ke Identitas Amazon Cognito menggunakan API. AWS IoT Core [AttachPolicy](#)

Note

Saat mengotorisasi identitas Amazon Cognito AWS IoT Core , pertimbangkan keduanya dan berikan hak istimewa paling sedikit yang ditentukan. Tindakan hanya diperbolehkan jika kedua kebijakan mengizinkan tindakan yang diminta. Jika salah satu kebijakan melarang tindakan, tindakan itu tidak sah.

Connect dan publikasikan contoh kebijakan

Untuk perangkat yang terdaftar sebagai benda dalam AWS IoT Core registri, kebijakan berikut memberikan izin untuk terhubung dengan ID klien yang cocok AWS IoT Core dengan nama benda dan membatasi perangkat untuk dipublikasikan pada ID Klien atau topik MQTT khusus nama benda. Agar koneksi berhasil, nama benda harus terdaftar di AWS IoT Core registri dan diautentikasi menggunakan identitas atau prinsipal yang dilampirkan pada benda tersebut:

```
{
  "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}"]
    }
  ]
}
```

Untuk perangkat yang tidak terdaftar sebagai benda dalam AWS IoT Core registri, kebijakan berikut memberikan izin untuk terhubung AWS IoT Core dengan ID klien `client1` dan membatasi perangkat untuk dipublikasikan pada topik MQTT khusus Klien:

```
{
  "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"]
    }
  ]
}
```

```
    }  
  ]  
}
```

Contoh kebijakan pesan yang dipertahankan

Menggunakan [pesan yang disimpan](#) memerlukan kebijakan khusus. Pesan yang disimpan adalah pesan MQTT yang diterbitkan dengan tanda RETAIN disetel dan disimpan oleh. AWS IoT Core Bagian ini menyajikan contoh kebijakan yang memungkinkan penggunaan umum pesan yang disimpan.

Di bagian ini:

- [Kebijakan untuk menghubungkan dan memublikasikan pesan yang disimpan](#)
- [Kebijakan untuk menghubungkan dan memublikasikan pesan Will yang disimpan](#)
- [Kebijakan untuk membuat daftar dan mendapatkan pesan yang disimpan](#)

Kebijakan untuk menghubungkan dan memublikasikan pesan yang disimpan

Agar perangkat dapat memublikasikan pesan yang disimpan, perangkat harus dapat terhubung, menerbitkan (pesan MQTT apa pun), dan memublikasikan pesan yang disimpan MQTT. Kebijakan berikut memberikan izin ini untuk topik: `device/sample/configuration` kepada klien. **device1** Untuk contoh lain yang memberikan izin untuk terhubung, lihat [the section called "Connect dan publikasikan contoh kebijakan"](#).

```
{  
  "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"
  ]
}
]
}

```

Kebijakan untuk menghubungkan dan mempublikasikan pesan Will yang disimpan

Klien dapat mengonfigurasi pesan yang AWS IoT Core akan dipublikasikan ketika klien terputus secara tak terduga. [MQTT menyebut pesan seperti itu sebagai pesan Will](#). Klien harus memiliki kondisi tambahan yang ditambahkan ke izin sambungannya untuk memasukkannya.

Dokumen kebijakan berikut memberikan izin kepada semua klien untuk menghubungkan dan menerbitkan pesan Will, yang diidentifikasi berdasarkan topiknyawill, yang juga AWS IoT Core akan disimpan.

```

{
  "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"
  ]
}
]
}

```

Kebijakan untuk membuat daftar dan mendapatkan pesan yang disimpan

Layanan dan aplikasi dapat mengakses pesan yang disimpan tanpa perlu mendukung klien MQTT dengan menelepon dan. [ListRetainedMessagesGetRetainedMessage](#) Layanan dan aplikasi yang memanggil tindakan ini harus diotorisasi dengan menggunakan kebijakan seperti contoh berikut.

```

{
  "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"
      ]
    }
  ]
}

```

Contoh kebijakan sertifikat

Untuk perangkat yang terdaftar di AWS IoT Core registri, kebijakan berikut memberikan izin untuk terhubung AWS IoT Core dengan ID klien yang cocok dengan nama benda, dan memublikasikan ke

topik yang namanya sama dengan sertifikat yang digunakan perangkat untuk mengautentikasi dirinya sendiri: `certificateId`

```
{
  "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}"]
    }
  ]
}
```

Untuk perangkat yang tidak terdaftar dalam AWS IoT Core registri, kebijakan berikut memberikan izin untuk terhubung AWS IoT Core dengan ID klien, `client1`, `client2`, `client3` dan untuk mempublikasikan ke topik yang namanya sama dengan sertifikat yang digunakan perangkat untuk mengautentikasi dirinya sendiri: `certificateId`

```
{
  "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"
    ]
  }
]
}

```

Untuk perangkat yang terdaftar di AWS IoT Core registri, kebijakan berikut memberikan izin untuk terhubung AWS IoT Core dengan ID klien yang cocok dengan nama benda, dan memublikasikan ke topik yang namanya sama dengan CommonName bidang subjek sertifikat yang digunakan perangkat untuk mengautentikasi dirinya sendiri:

```

{
  "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

Dalam contoh ini, nama umum subjek sertifikat digunakan sebagai pengidentifikasi topik, dengan asumsi bahwa nama umum subjek unik untuk setiap sertifikat terdaftar. Jika sertifikat dibagikan di beberapa perangkat, nama umum subjek sama untuk semua perangkat yang berbagi sertifikat ini, sehingga memungkinkan hak istimewa publikasi ke topik yang sama dari beberapa perangkat (tidak disarankan).

Untuk perangkat yang tidak terdaftar dalam AWS IoT Core registri, kebijakan berikut memberikan izin untuk terhubung AWS IoT Core dengan ID klien, `client1`, `client2`, `client3` dan untuk mempublikasikan ke topik yang namanya sama dengan `CommonName` bidang subjek sertifikat yang digunakan perangkat untuk mengautentikasi dirinya sendiri:

```
{
  "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

Dalam contoh ini, nama umum subjek sertifikat digunakan sebagai pengidentifikasi topik, dengan asumsi bahwa nama umum subjek unik untuk setiap sertifikat terdaftar. Jika sertifikat dibagikan di beberapa perangkat, nama umum subjek sama untuk semua perangkat yang berbagi sertifikat ini, sehingga memungkinkan hak istimewa publikasi ke topik yang sama dari beberapa perangkat (tidak disarankan).

Untuk perangkat yang terdaftar di AWS IoT Core registri, kebijakan berikut memberikan izin untuk terhubung dengan ID klien yang cocok AWS IoT Core dengan nama benda, dan memublikasikan ke topik yang namanya diawali admin/ saat sertifikat yang digunakan untuk mengautentikasi perangkat memiliki `Subject.CommonName.2` bidang yang disetel ke: 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"
        }
      }
    }
  ]
}
```

Untuk perangkat yang tidak terdaftar di AWS IoT Core registri, kebijakan berikut memberikan izin untuk terhubung AWS IoT Core dengan ID klien `client1`, `client2`, `client3` dan untuk memublikasikan ke topik yang namanya diawali `admin/` ketika sertifikat yang digunakan untuk mengautentikasi perangkat memiliki `Subject.CommonName.2` bidangnya disetel ke: `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": {
        "StringEquals": {
          "iot:Certificate.Subject.CommonName.2": "Administrator"
        }
      }
    }
  ]
}
```

Untuk perangkat yang terdaftar di AWS IoT Core registri, kebijakan berikut memungkinkan perangkat menggunakan nama benda untuk dipublikasikan pada topik tertentu yang terdiri dari `admin/` diikuti oleh `ThingName` ketika sertifikat yang digunakan untuk mengautentikasi perangkat memiliki salah satu `Subject.CommonName` bidangnya yang disetel ke `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"
          }
        }
      }
    ]
  }
}

```

Untuk perangkat yang tidak terdaftar di AWS IoT Core registri, kebijakan berikut memberikan izin untuk terhubung AWS IoT Core dengan ID klien `client1client2`, `client3` dan memublikasikan ke topik `admin` ketika sertifikat yang digunakan untuk mengautentikasi perangkat memiliki salah satu `Subject.CommonName` bidangnya yang disetel ke: `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"
        }
      }
    }
  ]
}

```

Contoh kebijakan hal

Kebijakan berikut memungkinkan perangkat untuk terhubung jika sertifikat yang digunakan untuk mengautentikasi AWS IoT Core dilampirkan pada hal yang sedang dievaluasi kebijakan:

```

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

```

Kebijakan berikut memungkinkan perangkat untuk mempublikasikan jika sertifikat dilampirkan ke sesuatu dengan jenis hal tertentu dan jika benda tersebut memiliki atribut `attributeName` dengan nilai `attributeValue`. Untuk informasi selengkapnya tentang variabel kebijakan hal, lihat [Variabel kebijakan Thing](#).

```
{
  "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"
        }
      }
    }
  ]
}
```

Kebijakan berikut memungkinkan perangkat untuk memublikasikan ke topik yang dimulai dengan atribut benda tersebut. Jika sertifikat perangkat tidak terkait dengan hal tersebut, variabel ini tidak akan diselesaikan dan akan mengakibatkan kesalahan akses ditolak. Untuk informasi selengkapnya tentang variabel kebijakan hal, lihat [Variabel kebijakan Thing](#).

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

Contoh kebijakan pekerjaan dasar

Contoh ini menunjukkan statment kebijakan yang diperlukan untuk target pekerjaan yang merupakan perangkat tunggal untuk menerima permintaan pekerjaan dan mengkomunikasikan status pelaksanaan pekerjaan dengan AWS IoT.

Ganti *us-west-2:57EXAMPLE833* dengan karakter titik dua (:), dan Akun AWS nomor 12 digit Anda, lalu ganti *unik ThingName* dengan nama sumber daya benda yang mewakili perangkat. Wilayah AWS IoT

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect"
      ],
      "Resource": [
        "arn:aws:iot:us-west-2:57EXAMPLE833:client/uniqueThingName"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish"
      ],
      "Resource": [
        "arn:aws:iot:us-west-2:57EXAMPLE833:topic/test/dc/pubtopic",
        "arn:aws:iot:us-west-2:57EXAMPLE833:topic/$aws/events/job/*",
        "arn:aws:iot:us-west-2:57EXAMPLE833:topic/$aws/events/jobExecution/*",
        "arn:aws:iot:us-west-2:57EXAMPLE833:topic/$aws/things/uniqueThingName/jobs/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Subscribe"
      ],
      "Resource": [
        "arn:aws:iot:us-west-2:57EXAMPLE833:topicfilter/test/dc/subtopic",
        "arn:aws:iot:us-west-2:57EXAMPLE833:topicfilter/$aws/events/jobExecution/*",

```

```

    "arn:aws:iot:us-west-2:57EXAMPLE833:topicfilter/$aws/things/uniqueThingName/
jobs/*"
  ],
  {
    "Effect": "Allow",
    "Action": [
      "iot:Receive"
    ],
    "Resource": [
      "arn:aws:iot:us-west-2:57EXAMPLE833:topic/test/dc/subtopic",
      "arn:aws:iot:us-west-2:57EXAMPLE833:topic/$aws/things/uniqueThingName/jobs/*"
    ]
  },
  {
    "Effect": "Allow",
    "Action": [
      "iot:DescribeJobExecution",
      "iot:GetPendingJobExecutions",
      "iot:StartNextPendingJobExecution",
      "iot:UpdateJobExecution"
    ],
    "Resource": [
      "arn:aws:iot:us-west-2:57EXAMPLE833:topic/$aws/things/uniqueThingName"
    ]
  }
]
}

```

Otorisasi dengan identitas Amazon Cognito

Ada dua jenis identitas Amazon Cognito: tidak diautentikasi dan diautentikasi. Jika aplikasi Anda mendukung identitas Amazon Cognito yang tidak diautentikasi, tidak ada autentikasi yang dilakukan, sehingga Anda tidak tahu siapa pengguna tersebut.

Identitas Tidak Diautentikasi: Untuk identitas Amazon Cognito yang tidak diautentikasi, Anda memberikan izin dengan melampirkan peran IAM ke kumpulan identitas yang tidak diautentikasi. Kami menyarankan Anda hanya memberikan akses ke sumber daya yang Anda inginkan tersedia untuk pengguna yang tidak dikenal.

⚠ Important

Untuk pengguna Amazon Cognito yang tidak diautentikasi yang terhubung AWS IoT Core ke, kami menyarankan Anda memberikan akses ke sumber daya yang sangat terbatas dalam kebijakan IAM.

Identitas Terautentikasi: Untuk identitas Amazon Cognito yang diautentikasi, Anda perlu menentukan izin di dua tempat:

- Lampirkan kebijakan IAM ke kumpulan Identitas Amazon Cognito yang diautentikasi dan
- Lampirkan AWS IoT Core kebijakan ke Identitas Amazon Cognito (pengguna yang diautentikasi).

Contoh kebijakan untuk pengguna Amazon Cognito yang tidak diautentikasi dan diautentikasi yang terhubung AWS IoT Core

Contoh berikut menunjukkan izin dalam kebijakan IAM dan kebijakan IoT dari identitas Amazon Cognito. Pengguna yang diautentikasi ingin mempublikasikan ke topik khusus perangkat (misalnya 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"
      ]
    }
  ]
}
```

```

    ]
  }
]
}

```

Contoh berikut menunjukkan izin dalam kebijakan IAM dari peran Amazon Cognito yang tidak diautentikasi. Pengguna yang tidak diautentikasi ingin mempublikasikan ke topik khusus non-perangkat yang tidak memerlukan otentikasi.

```

{
  "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 contoh

Contoh aplikasi web berikut GitHub menunjukkan cara menggabungkan lampiran kebijakan ke pengguna yang diautentikasi ke dalam proses pendaftaran dan otentikasi pengguna.

- [MQTT menerbitkan/berlangganan aplikasi web React menggunakan dan AWS Amplify AWS IoT Device SDK for JavaScript](#)
- [MQTT menerbitkan/berlangganan aplikasi web React menggunakan, fungsi, dan AWS Amplify Lambda AWS IoT Device SDK for JavaScript](#)

Amplify adalah seperangkat alat dan layanan yang membantu Anda membangun aplikasi web dan seluler yang terintegrasi dengan AWS layanan. Untuk informasi selengkapnya tentang Amplify, lihat Amplify Framework [Documentation](#),.

Kedua contoh melakukan langkah-langkah berikut.

1. Saat pengguna mendaftar untuk akun, aplikasi akan membuat kumpulan pengguna dan identitas Amazon Cognito.
2. Ketika pengguna mengautentikasi, aplikasi membuat dan melampirkan kebijakan ke identitas. Ini memberi pengguna izin mempublikasikan dan berlangganan.
3. Pengguna dapat menggunakan aplikasi untuk mempublikasikan dan berlangganan topik MQTT.

Contoh pertama menggunakan operasi `AttachPolicy` API langsung di dalam operasi otentikasi. Contoh berikut menunjukkan bagaimana menerapkan panggilan API ini di dalam aplikasi web React yang menggunakan Amplify dan AWS IoT Device SDK for 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);
    }
  });
}
```

Kode ini muncul di [AuthDisplayfile.js](#).

Contoh kedua mengimplementasikan operasi `AttachPolicy` API dalam fungsi Lambda. Contoh berikut menunjukkan bagaimana Lambda menggunakan panggilan API ini.

```
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});
  }
});
```

Kode ini muncul di dalam `iot.GetPolicy` fungsi dalam file [app.js](#).

Note

Saat Anda memanggil fungsi dengan AWS kredensial yang Anda peroleh melalui kumpulan Identitas Amazon Cognito, objek konteks dalam fungsi Lambda Anda berisi nilai untuk `context.cognito_identity_id` Untuk informasi selengkapnya, lihat hal berikut.

- [AWS Lambda objek konteks di Node.js](#)
- [AWS Lambda objek konteks dengan Python](#)
- [AWS Lambda objek konteks di Ruby](#)
- [AWS Lambda objek konteks di Jawa](#)
- [AWS Lambda objek konteks di Go](#)
- [AWS Lambda objek konteks dalam C #](#)
- [AWS Lambda objek konteks di PowerShell](#)

Mengotorisasi panggilan langsung ke AWS layanan menggunakan penyedia AWS IoT Core kredensyal

Perangkat dapat menggunakan sertifikat X.509 untuk terhubung AWS IoT Core menggunakan protokol otentikasi timbal balik TLS. AWS [Layanan lain tidak mendukung otentikasi berbasis sertifikat, tetapi mereka dapat dipanggil menggunakan AWS kredensyal dalam format Signature Version 4.](#) [AWS Algoritma Signature Version 4](#) biasanya mengharuskan penelepon untuk memiliki ID kunci akses dan kunci akses rahasia. AWS IoT Core memiliki penyedia kredensyal yang memungkinkan Anda menggunakan [sertifikat X.509](#) bawaan sebagai identitas perangkat unik untuk mengautentikasi permintaan. AWS Ini menghilangkan kebutuhan untuk menyimpan ID kunci akses dan kunci akses rahasia di perangkat Anda.

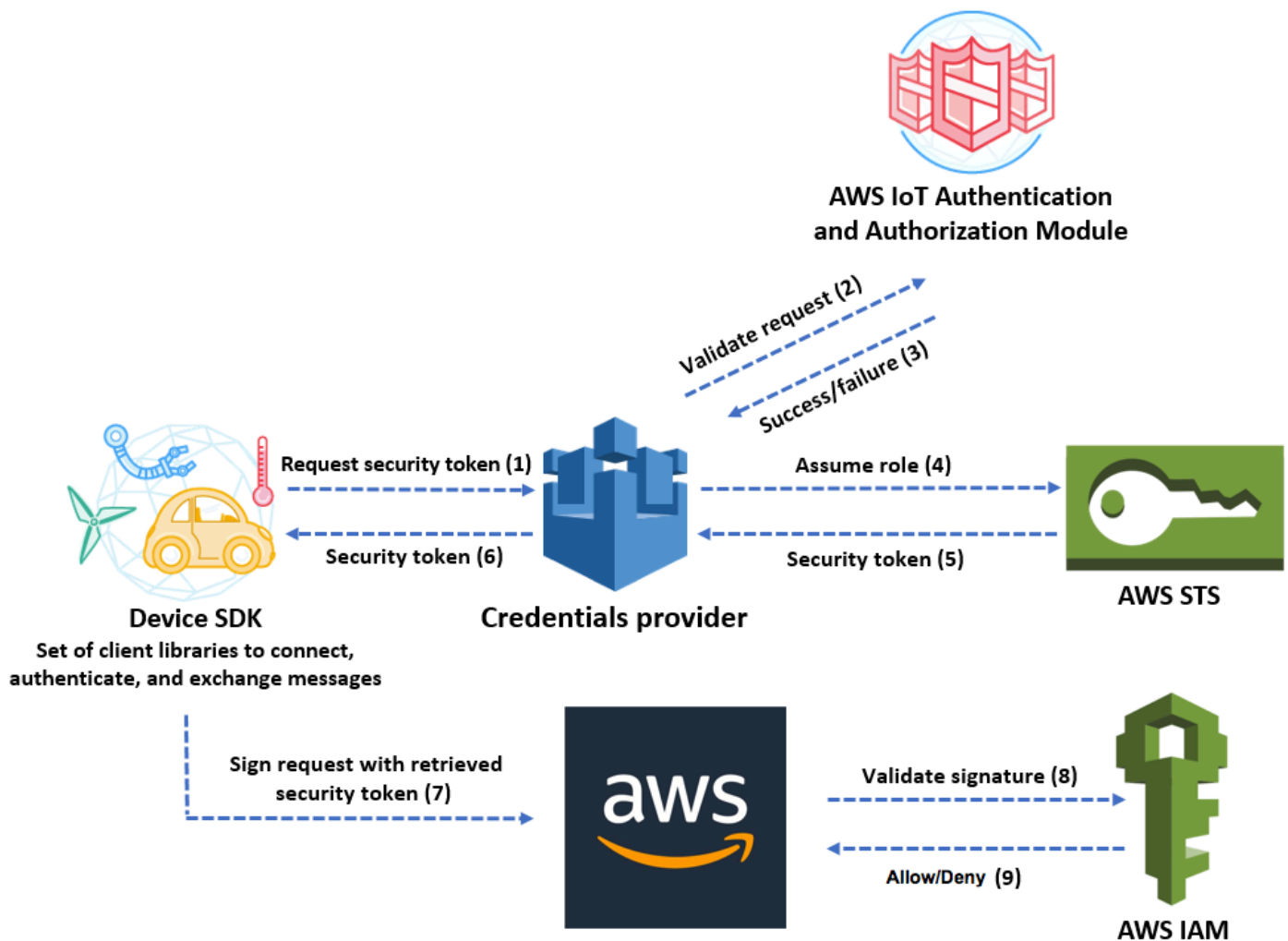
Penyedia kredensyal mengautentikasi pemanggil menggunakan sertifikat X.509 dan mengeluarkan token keamanan hak istimewa sementara yang terbatas. Token dapat digunakan untuk menandatangani dan mengotentikasi AWS permintaan apa pun. Cara mengautentikasi AWS permintaan ini mengharuskan Anda untuk membuat dan mengonfigurasi [peran AWS Identity and Access Management \(IAM\)](#) dan melampirkan kebijakan IAM yang sesuai ke peran tersebut sehingga penyedia kredensyal dapat mengambil peran atas nama Anda. Untuk informasi lebih lanjut tentang AWS IoT Core dan IAM, lihat [Identitas dan manajemen akses untuk AWS IoT](#).

AWS IoT memerlukan perangkat untuk mengirim [ekstensi Server Name Indication \(SNI\)](#) ke protokol Transport Layer Security (TLS) dan memberikan alamat endpoint lengkap di lapangan. `host_name` `host_name` Bidang harus berisi titik akhir yang Anda panggil, dan itu harus:

- Yang `endpointAddress` dikembalikan oleh `aws iot describe-endpoint --endpoint-type iot:CredentialProvider`.

Koneksi yang dicoba oleh perangkat tanpa `host_name` nilai yang benar akan gagal.

Diagram berikut mengilustrasikan alur kerja penyedia kredensyal.



1. AWS IoT Core Perangkat membuat permintaan HTTPS ke penyedia kredensial untuk token keamanan. Permintaan tersebut mencakup sertifikat perangkat X.509 untuk otentikasi.
2. Penyedia kredensial meneruskan permintaan ke modul AWS IoT Core otentikasi dan otorisasi untuk memvalidasi sertifikat dan memverifikasi bahwa perangkat memiliki izin untuk meminta token keamanan.
3. Jika sertifikat valid dan memiliki izin untuk meminta token keamanan, modul AWS IoT Core otentikasi dan otorisasi mengembalikan keberhasilan. Jika tidak, ia mengirimkan pengecualian ke perangkat.
4. Setelah berhasil memvalidasi sertifikat, penyedia kredensial memanggil [AWS Security Token Service \(AWS STS\)](#) untuk mengambil peran IAM yang Anda buat untuknya.
5. AWS STS mengembalikan token keamanan hak istimewa terbatas sementara ke penyedia kredensial.
6. Penyedia kredensial mengembalikan token keamanan ke perangkat.

7. Perangkat menggunakan token keamanan untuk menandatangani AWS permintaan dengan AWS Signature Version 4.
8. Layanan yang diminta memanggil IAM untuk memvalidasi tanda tangan dan mengotorisasi permintaan terhadap kebijakan akses yang dilampirkan pada peran IAM yang Anda buat untuk penyedia kredensial.
9. Jika IAM berhasil memvalidasi tanda tangan dan mengotorisasi permintaan, permintaan berhasil. Jika tidak, IAM mengirimkan pengecualian.

Bagian berikut menjelaskan cara menggunakan sertifikat untuk mendapatkan token keamanan. Itu ditulis dengan asumsi bahwa Anda telah [mendaftarkan perangkat dan membuat serta mengaktifkan sertifikat Anda sendiri](#) untuk itu.

Cara menggunakan sertifikat untuk mendapatkan token keamanan

1. Konfigurasi peran IAM yang diasumsikan oleh penyedia kredensial atas nama perangkat Anda. Lampirkan kebijakan kepercayaan berikut ke peran tersebut.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Principal": {"Service": "credentials.iot.amazonaws.com"},
    "Action": "sts:AssumeRole"
  }
}
```

Untuk setiap AWS layanan yang ingin Anda panggil, lampirkan kebijakan akses ke peran tersebut. Penyedia kredensial mendukung variabel kebijakan berikut:

- `credentials-iot:ThingName`
- `credentials-iot:ThingTypeName`
- `credentials-iot:AwsCertificateId`

Ketika perangkat memberikan nama benda dalam permintaannya ke AWS layanan, penyedia kredensial menambahkan `credentials-iot:ThingName` dan `credentials-iot:ThingTypeName` sebagai variabel konteks ke token keamanan. Penyedia kredensial menyediakan `credentials-iot:AwsCertificateId` sebagai variabel konteks meskipun

perangkat tidak memberikan nama benda dalam permintaan. Anda meneruskan nama benda sebagai nilai header permintaan `x-amzn-iot-thingname` HTTP.

Ketiga variabel ini hanya berfungsi untuk kebijakan IAM, bukan AWS IoT Core kebijakan.

2. Pastikan bahwa pengguna yang melakukan langkah berikutnya (membuat alias peran) memiliki izin untuk meneruskan peran yang baru dibuat. AWS IoT Core Kebijakan berikut memberikan keduanya `iam:GetRole` dan `iam:PassRole` izin kepada AWS pengguna. `iam:GetRole` izin memungkinkan pengguna untuk mendapatkan informasi tentang peran yang baru saja Anda buat. `iam:PassRole` izin memungkinkan pengguna untuk meneruskan peran ke AWS layanan lain.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "iam:GetRole",
      "iam:PassRole"
    ],
    "Resource": "arn:aws:iam::your Akun AWS id:role/your role name"
  }
}
```

3. Buat alias AWS IoT Core peran. Perangkat yang akan melakukan panggilan langsung ke AWS layanan harus tahu peran ARN mana yang akan digunakan saat menghubungkan. AWS IoT Core Hard-coding peran ARN bukanlah solusi yang baik karena mengharuskan Anda untuk memperbarui perangkat setiap kali peran ARN berubah. Solusi yang lebih baik adalah dengan menggunakan `CreateRoleAlias` API untuk membuat alias peran yang menunjuk ke peran ARN. Jika peran ARN berubah, Anda cukup memperbarui alias peran. Tidak ada perubahan yang diperlukan pada perangkat. API ini mengambil parameter berikut:

`roleAlias`

Wajib. String arbitrer yang mengidentifikasi alias peran. Ini berfungsi sebagai kunci utama dalam model data alias peran. Ini berisi 1-128 karakter dan harus menyertakan hanya karakter alfanumerik dan simbol `=`, `@`, dan `-`. Karakter alfabet huruf besar dan kecil diperbolehkan.

`roleArn`

Wajib. ARN dari peran yang dirujuk alias peran.

credentialDurationSeconds

Tidak wajib. Berapa lama (dalam hitungan detik) kredensialnya valid. Nilai minimum adalah 900 detik (15 menit). Nilai maksimumnya adalah 43.200 detik (12 jam). Nilai default adalah 3.600 detik (1 jam).

Important

Penyedia AWS IoT Core Kredensial dapat mengeluarkan kredensi dengan masa pakai maksimum adalah 43.200 detik (12 jam). Memiliki kredensi yang valid hingga 12 jam dapat membantu mengurangi jumlah panggilan ke penyedia kredensi dengan menyimpan kredensi lebih lama.

`credentialDurationSeconds` Nilai harus kurang dari atau sama dengan durasi sesi maksimum peran IAM yang dirujuk alias peran. Untuk informasi selengkapnya, lihat [Memodifikasi durasi sesi maksimum peran \(AWS API\)](#) dari Panduan Pengguna AWS Identity and Access Management.

Untuk informasi selengkapnya tentang API ini, lihat [CreateRoleAlias](#).

4. Lampirkan kebijakan ke sertifikat perangkat. Kebijakan yang dilampirkan pada sertifikat perangkat harus memberikan izin perangkat untuk mengambil peran. Anda melakukan ini dengan memberikan izin untuk `iot:AssumeRoleWithCertificate` tindakan ke alias peran, seperti pada contoh berikut.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:AssumeRoleWithCertificate",
      "Resource": "arn:aws:iot:your_region:your_aws_account_id:rolealias/your_role_alias"
    }
  ]
}
```

5. Buat permintaan HTTPS ke penyedia kredensi untuk mendapatkan token keamanan. Berikan informasi berikut:

- **Sertifikat:** Karena ini adalah permintaan HTTP melalui otentikasi timbal balik TLS, Anda harus memberikan sertifikat dan kunci pribadi kepada klien Anda saat membuat permintaan. Gunakan sertifikat dan kunci pribadi yang sama dengan yang Anda gunakan saat mendaftarkan sertifikat Anda AWS IoT Core.

Untuk memastikan perangkat Anda berkomunikasi dengan AWS IoT Core (dan bukan layanan yang meniru itu), lihat [Otentikasi Server](#), ikuti tautan untuk mengunduh sertifikat CA yang sesuai, lalu salin ke perangkat Anda.

- **RoleAlias:** Nama alias peran yang Anda buat untuk penyedia kredensial.
- **ThingName:** Nama benda yang Anda buat ketika Anda mendaftarkan AWS IoT Core barang Anda. Ini diteruskan sebagai nilai header `x-amzn-iot-thingname` HTTP. Nilai ini diperlukan hanya jika Anda menggunakan atribut benda sebagai variabel kebijakan dalam AWS IoT Core atau kebijakan IAM.

Note

ThingName yang Anda berikan `x-amzn-iot-thingname` harus cocok dengan nama sumber daya AWS IoT Thing yang ditetapkan ke sertifikat. Jika tidak cocok, kesalahan 403 dikembalikan.

Jalankan perintah berikut di AWS CLI untuk mendapatkan titik akhir penyedia kredensial untuk Anda. Untuk informasi selengkapnya tentang API ini, lihat [DescribeEndpoint](#).

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

Objek JSON berikut adalah contoh output dari `describe-endpoint` perintah. Ini berisi `endpointAddress` yang Anda gunakan untuk meminta token keamanan.

```
{
  "endpointAddress": "your_aws_account_specific_prefix.credentials.iot.your
region.amazonaws.com"
}
```

Gunakan titik akhir untuk membuat permintaan HTTPS ke penyedia kredensial untuk mengembalikan token keamanan. Contoh perintah berikut menggunakan `curl`, tetapi Anda dapat menggunakan klien HTTP apa pun.

```
curl --cert your certificate --key your device certificate key pair -H "x-amzn-iot-thingname: your thing name" --cacert AmazonRootCA1.pem https://your endpoint /role-aliases/your role alias/credentials
```

Perintah ini mengembalikan objek token keamanan yang berisi `accessKeyId`, `a`, `secretAccessKey`, `secretAccessKey`, `secretAccessKey`, dan `secretAccessKey`. Objek JSON berikut adalah contoh output dari `curl` perintah.

```
{"credentials":{"accessKeyId":"access key","secretAccessKey":"secret access key","sessionToken":"session token","expiration":"2018-01-18T09:18:06Z"}}
```

Anda kemudian dapat menggunakan `accessKeyId`, `secretAccessKey`, dan `secretAccessKey` nilai untuk menandatangani permintaan ke AWS layanan. Untuk end-to-end demonstrasi, lihat [Cara Menghilangkan Kebutuhan AWS Kredensial Hard-Coded di Perangkat dengan Menggunakan posting blog Penyedia AWS IoT Kredensial di Blog Keamanan.AWS](#)

Akses lintas akun dengan IAM

AWS IoT Core memungkinkan Anda untuk mengaktifkan kepala sekolah untuk menerbitkan atau berlangganan topik yang didefinisikan dalam Akun AWS tidak dimiliki oleh kepala sekolah. Anda mengonfigurasi akses lintas akun dengan membuat kebijakan IAM dan peran IAM, lalu melampirkan kebijakan ke peran tersebut.

Pertama, buat kebijakan IAM yang dikelola pelanggan seperti yang dijelaskan dalam [Membuat Kebijakan IAM](#), seperti yang Anda lakukan untuk pengguna dan sertifikat lain di Anda. Akun AWS

Untuk perangkat yang terdaftar di AWS IoT Core registri, kebijakan berikut memberikan izin untuk perangkat terhubung AWS IoT Core menggunakan ID klien yang cocok dengan nama perangkat dan memublikasikan ke `my/topic/thing-name` tempat *nama benda adalah nama* perangkat:

```
{
  "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}"],
  }
]
}

```

Untuk perangkat yang tidak terdaftar di AWS IoT Core registri, kebijakan berikut memberikan izin kepada perangkat untuk menggunakan nama benda yang `client1` terdaftar di AWS IoT Core registri akun Anda (123456789012) untuk terhubung ke AWS IoT Core dan mempublikasikan ke topik khusus ID klien yang namanya diawali dengan: `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}"
      ]
    }
  ]
}

```

```
}
```

Selanjutnya, ikuti langkah-langkah dalam [Membuat peran untuk mendelegasikan izin ke pengguna IAM](#). Masukkan ID akun Akun AWS yang ingin Anda bagikan akses. Kemudian, pada langkah terakhir, lampirkan kebijakan yang baru saja Anda buat ke peran tersebut. Jika, di lain waktu, Anda perlu mengubah ID AWS akun yang Anda berikan akses, Anda dapat menggunakan format kebijakan kepercayaan berikut untuk melakukannya:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam:us-east-1:567890123456:user/MyUser"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Perlindungan data di AWS IoT Core

[Model tanggung jawab AWS bersama model](#) berlaku untuk perlindungan data di AWS IoT Core. Seperti yang dijelaskan dalam model AWS ini, bertanggung jawab untuk melindungi infrastruktur global yang menjalankan semua AWS Cloud. Anda bertanggung jawab untuk mempertahankan kendali atas konten yang di-host pada infrastruktur ini. Anda juga bertanggung jawab atas tugas-tugas konfigurasi dan manajemen keamanan untuk Layanan AWS yang Anda gunakan. Lihat informasi yang lebih lengkap tentang privasi data dalam [Pertanyaan Umum Privasi Data](#). Lihat informasi tentang perlindungan data di Eropa di pos blog [Model Tanggung Jawab Bersama dan GDPR AWS](#) di Blog Keamanan AWS .

Untuk tujuan perlindungan data, kami menyarankan Anda melindungi Akun AWS kredensial dan mengatur pengguna individu dengan AWS IAM Identity Center atau AWS Identity and Access Management (IAM). Dengan cara itu, setiap pengguna hanya diberi izin yang diperlukan untuk memenuhi tanggung jawab tugasnya. Kami juga menyarankan supaya Anda mengamankan data dengan cara-cara berikut:

- Gunakan autentikasi multi-faktor (MFA) pada setiap akun.

- Gunakan SSL/TLS untuk berkomunikasi dengan sumber daya. AWS Kami mensyaratkan TLS 1.2 dan menganjurkan TLS 1.3.
- Siapkan API dan pencatatan aktivitas pengguna dengan AWS CloudTrail.
- Gunakan solusi AWS enkripsi, bersama dengan semua kontrol keamanan default di dalamnya Layanan AWS.
- Gunakan layanan keamanan terkelola lanjut seperti Amazon Macie, yang membantu menemukan dan mengamankan data sensitif yang disimpan di Amazon S3.
- Jika Anda memerlukan modul kriptografi tervalidasi FIPS 140-2 saat mengakses AWS melalui antarmuka baris perintah atau API, gunakan titik akhir FIPS. Lihat informasi yang lebih lengkap tentang titik akhir FIPS yang tersedia di [Standar Pemrosesan Informasi Federal \(FIPS\) 140-2](#).

Kami sangat merekomendasikan agar Anda tidak pernah memasukkan informasi identifikasi yang sensitif, seperti nomor rekening pelanggan Anda, ke dalam tanda atau bidang isian bebas seperti bidang Nama. Ini termasuk saat Anda bekerja dengan AWS IoT atau lainnya Layanan AWS menggunakan konsol, API AWS CLI, atau AWS SDK. Data apa pun yang Anda masukkan ke dalam tanda atau bidang isian bebas yang digunakan untuk nama dapat digunakan untuk log penagihan atau log diagnostik. Saat Anda memberikan URL ke server eksternal, kami sangat menganjurkan supaya Anda tidak menyertakan informasi kredensial di dalam URL untuk memvalidasi permintaan Anda ke server itu.

Untuk informasi selengkapnya tentang perlindungan data, lihat postingan blog [Model Tanggung Jawab Bersama AWS dan GDPR](#) di Blog Keamanan AWS .

AWS IoT perangkat mengumpulkan data, melakukan beberapa manipulasi pada data itu, dan kemudian mengirim data tersebut ke layanan web lain. Anda dapat memilih untuk menyimpan beberapa data di perangkat Anda untuk waktu yang singkat. Anda bertanggung jawab untuk memberikan perlindungan data apa pun pada data tersebut saat istirahat. Ketika perangkat Anda mengirim data ke AWS IoT, ia melakukannya melalui koneksi TLS seperti yang dibahas nanti di bagian ini. AWS IoT perangkat dapat mengirim data ke AWS layanan apa pun. Untuk informasi selengkapnya tentang keamanan data setiap layanan, lihat dokumentasi untuk layanan tersebut. AWS IoT dapat dikonfigurasi untuk menulis log ke CloudWatch Log dan log panggilan AWS IoT API ke AWS CloudTrail. Untuk informasi selengkapnya tentang keamanan data untuk layanan ini, lihat [Otentikasi dan Kontrol Akses untuk Amazon CloudWatch dan Mengenkripsi File CloudTrail Log dengan AWS Kunci yang Dikelola KMS](#).

Enkripsi data di AWS IoT

Secara default, semua AWS IoT data dalam perjalanan dan saat istirahat dienkripsi. [Data dalam perjalanan dienkripsi menggunakan TLS](#), dan data saat istirahat dienkripsi menggunakan kunci yang dimiliki. AWS IoT Saat ini tidak mendukung yang dikelola pelanggan AWS KMS keys (kunci KMS) dari AWS Key Management Service (AWS KMS); namun, Device Advisor dan AWS IoT Wireless hanya menggunakan Kunci milik AWS untuk mengenkripsi data pelanggan.

Keamanan transportasi di AWS IoT Core

TLS (Transport Layer Security) adalah protokol kriptografi yang dirancang untuk komunikasi yang aman melalui jaringan komputer. AWS IoT Core Device Gateway mengharuskan pelanggan untuk mengenkripsi semua komunikasi saat dalam perjalanan dengan menggunakan TLS untuk koneksi dari perangkat ke Gateway. TLS digunakan untuk mencapai kerahasiaan protokol aplikasi (MQTT, HTTP, dan) yang didukung oleh WebSocket AWS IoT Core Dukungan TLS tersedia dalam sejumlah bahasa pemrograman dan sistem operasi. Data di dalamnya AWS dienkripsi oleh layanan tertentu AWS . Untuk informasi selengkapnya tentang enkripsi data pada AWS layanan lain, lihat dokumentasi keamanan untuk layanan tersebut.

Daftar Isi

- [Protokol TLS](#)
- [Kebijakan Keamanan](#)
- [Catatan penting untuk keamanan transportasi di AWS IoT Core](#)
- [Keamanan transportasi untuk perangkat nirkabel LoRa WAN](#)

Protokol TLS

AWS IoT Core mendukung versi protokol TLS berikut:

- TLS 1.3
- TLS 1.2

Dengan AWS IoT Core, Anda dapat mengkonfigurasi pengaturan TLS (untuk [TLS 1.2](#) dan [TLS 1.3](#)) dalam konfigurasi domain. Untuk informasi selengkapnya, lihat [???](#).

Kebijakan Keamanan

Kebijakan keamanan adalah kombinasi protokol TLS dan cipher mereka yang menentukan protokol dan cipher mana yang didukung selama negosiasi TLS antara klien dan server. Anda dapat mengonfigurasi perangkat untuk menggunakan kebijakan keamanan yang telah ditentukan berdasarkan kebutuhan Anda. Perhatikan bahwa AWS IoT Core tidak mendukung kebijakan keamanan khusus.

Anda dapat memilih salah satu kebijakan keamanan yang telah ditetapkan untuk perangkat Anda saat AWS IoT Core menghubungkannya. Nama-nama kebijakan keamanan standar terbaru AWS IoT Core termasuk informasi versi berdasarkan tahun dan bulan mereka dirilis. Kebijakan keamanan standar default adalah `IoTSecurityPolicy_TLS13_1_2_2022_10`. Untuk menentukan kebijakan keamanan, Anda dapat menggunakan AWS IoT konsol atau AWS CLI. Untuk informasi selengkapnya, lihat [???](#).


Tabel berikut menjelaskan kebijakan keamanan standar terbaru yang AWS IoT Core mendukung. `IotSecurityPolicy_` telah dihapus dari nama kebijakan di baris judul agar sesuai.

Kebijakan keamanan	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*		
Pelabuhan TCP	443/8443/8883	443/8443/8883	443/8443/8883	443	8443/8883	443	8443/8883
Protokol TLS							
TLS 1.2		✓	✓	✓	✓	✓	✓
TLS 1.3	✓	✓					
Cipher TLS							
TLS_AES_128_GCM_SHA256	✓	✓					
TLS_AES_128_GCM_SHA384	✓	✓					

Kebijakan keamanan	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*	
TLS_CHACHA20_POLY1305_SHA256	✓	✓					
ECDHE-RSA-AES128-GCM-SHA256		✓	✓	✓	✓	✓	✓
ECDHE-RSA-AES128-SHA256		✓	✓	✓	✓	✓	✓
ECDHE-RSA-AES128-SHA		✓	✓	✓	✓	✓	✓
ECDHE-RSA-AES256-GCM-SHA384		✓	✓	✓	✓	✓	✓
ECDHE-RSA-AES256-SHA384		✓	✓	✓	✓	✓	✓

Kebijakan keamanan	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-AES256-SHA		✓	✓	✓	✓	✓	✓
AES128-GCM-SHA256		✓	✓	✓	✓	✓	✓
AES128-SHA256		✓	✓	✓		✓	✓
AES128-SHA		✓	✓	✓	✓	✓	✓
AES256-GCM-SHA384		✓	✓	✓	✓	✓	✓
AES256-SHA256		✓	✓	✓	✓	✓	✓
AES256-SHA		✓	✓	✓	✓	✓	✓
DHE-RSA-AES256-SHA						✓	✓
ECDHE-ECDSA-AES128-GCM-SHA256		✓	✓	✓	✓	✓	✓

Kebijakan keamanan	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-AES128-SHA256		✓	✓	✓	✓	✓	✓
ECDHE-ECDSA-AES128-SHA		✓	✓	✓	✓	✓	✓
ECDHE-ECDSA-AES256-GCM-SHA384		✓	✓	✓	✓	✓	✓
ECDHE-ECDSA-AES256-SHA384		✓	✓	✓	✓	✓	✓
ECDHE-ECDSA-AES256-SHA		✓	✓	✓	✓	✓	✓

 Note

TLS12_1_0_2016_01 hanya tersedia sebagai berikut Wilayah AWS: ap-east-1, ap-northeast-2, ap-south-1, ap-south-1, ap-southeast-2, ca-central-1, cn-north-1, cn-north-1, cn-northwest-1, eu-north-1, eu-north-1, eu-west-2 st-2, eu-west-3, me-south-1, sa-east-1, us-east-2, -1, -2, us-west-1. us-gov-west us-gov-west

TLS12_1_0_2015_01 hanya tersedia dalam hal berikut Wilayah AWS: ap-northeast-1, ap-southeast-1, eu-central-1, eu-central-1, eu-west-1, us-east-1, us-east-1, us-west-2.

Catatan penting untuk keamanan transportasi di AWS IoT Core

Untuk perangkat yang terhubung AWS IoT Core menggunakan [MQTT, TLS](#) mengenkripsi koneksi antara perangkat dan broker, dan AWS IoT Core menggunakan otentikasi klien TLS untuk mengidentifikasi perangkat. Untuk informasi selengkapnya, lihat [Autentikasi klien](#). Untuk perangkat yang terhubung AWS IoT Core menggunakan [HTTP](#), TLS mengenkripsi koneksi antara perangkat dan broker, dan otentikasi didelegasikan ke Signature Version 4. Untuk informasi selengkapnya, lihat [Menandatangani permintaan dengan Tanda Tangan Versi 4](#) di Referensi AWS Umum.

Saat Anda menghubungkan perangkat AWS IoT Core, mengirim [ekstensi Server Name Indication \(SNI\)](#) tidak diperlukan tetapi sangat disarankan. Untuk menggunakan fitur seperti [pendaftaran multi-akun](#), [domain khusus](#), [titik akhir VPC](#), dan [kebijakan TLS yang dikonfigurasi](#), Anda harus menggunakan ekstensi SNI dan memberikan alamat titik akhir lengkap di bidang `host_name`. Bidang `host_name` harus berisi titik akhir yang Anda panggil. Titik akhir itu harus salah satu dari yang berikut:

- Yang `endpointAddress` dikembalikan oleh `aws iot describe-endpoint --endpoint-type iot:Data-ATS`
- Yang `domainName` dikembalikan oleh `aws iot describe-domain-configuration --domain-configuration-name "domain_configuration_name"`

Koneksi yang dicoba oleh perangkat dengan `host_name` nilai yang salah atau tidak valid akan gagal. AWS IoT Core akan mencatat kegagalan CloudWatch untuk jenis otentikasi Otentikasi [Kustom](#).

AWS IoT Core tidak mendukung [ekstensi SessionTicket TLS](#).

Keamanan transportasi untuk perangkat nirkabel LoRa WAN

LoRaPerangkat WAN mengikuti praktik keamanan yang dijelaskan dalam [LoRaWAN™ SECURITY: Buku Putih yang Disiapkan untuk LoRa Aliansi™ oleh Gemalto, Actility, dan Semtech](#).

Untuk informasi selengkapnya tentang keamanan transportasi dengan perangkat LoRa WAN, lihat [data LoRa WAN dan keamanan transportasi](#).

Enkripsi data di AWS IoT

Perlindungan data mengacu pada perlindungan data saat dalam perjalanan (saat bepergian ke dan dari AWS IoT) dan saat istirahat (saat disimpan di perangkat atau oleh AWS layanan lain). Semua data yang dikirim ke AWS IoT dikirim melalui koneksi TLS menggunakan MQTT, HTTPS, dan WebSocket protokol, sehingga aman secara default saat dalam perjalanan. AWS IoT perangkat mengumpulkan data dan kemudian mengirimkannya ke AWS layanan lain untuk diproses lebih lanjut. Untuk informasi selengkapnya tentang enkripsi data di layanan AWS lainnya, lihat dokumentasi keamanan untuk layanan tersebut.

FreeRTOS menyediakan pustaka PKCS #11 yang mengabstraksi penyimpanan kunci, mengakses objek kriptografi, dan mengelola sesi. Anda bertanggung jawab untuk menggunakan pustaka ini untuk mengenkripsi data saat istirahat di perangkat Anda. Untuk informasi selengkapnya, lihat Perpustakaan [Standar Kriptografi Kunci Publik FreeRTOS \(PKCS\) #11](#).

Penasihat Perangkat

Enkripsi bergerak

Data yang dikirim ke dan dari Device Advisor dienkripsi dalam perjalanan. Semua data yang dikirim ke dan dari layanan saat menggunakan Device Advisor API dienkripsi menggunakan Signature Version 4. Untuk informasi selengkapnya tentang cara permintaan AWS API ditandatangani, lihat [Menandatangani permintaan AWS API](#). Semua data yang dikirim dari perangkat pengujian ke titik akhir pengujian Device Advisor dikirim melalui koneksi TLS sehingga aman secara default saat transit.

Manajemen kunci di AWS IoT

Semua koneksi dilakukan menggunakan TLS, jadi tidak ada kunci enkripsi sisi klien yang diperlukan untuk koneksi TLS awal. AWS IoT

Perangkat harus mengautentikasi menggunakan sertifikat X.509 atau Identitas Amazon Cognito. Anda dapat AWS IoT membuat sertifikat untuk Anda, dalam hal ini akan menghasilkan public/private key pair. Jika Anda menggunakan AWS IoT konsol, Anda akan diminta untuk mengunduh sertifikat dan kunci. Jika Anda menggunakan perintah [create-keys-and-certificate](#) CLI, sertifikat dan kunci dikembalikan oleh perintah CLI. Anda bertanggung jawab untuk menyalin sertifikat dan kunci pribadi ke perangkat Anda dan menjaganya tetap aman.

AWS IoT Saat ini tidak mendukung yang dikelola pelanggan AWS KMS keys (kunci KMS) dari AWS Key Management Service (AWS KMS); namun, Device Advisor dan AWS IoT Wireless hanya menggunakan Kunci milik AWS untuk mengenkripsi data pelanggan.

Penasihat Perangkat

Semua data yang dikirim ke Device Advisor saat menggunakan AWS API dienkripsi saat istirahat. Device Advisor mengenkripsi semua data Anda saat istirahat menggunakan kunci KMS yang disimpan dan dikelola. [AWS Key Management Service](#) Device Advisor mengenkripsi data Anda menggunakan. Kunci milik AWS Untuk informasi lebih lanjut tentang Kunci milik AWS, lihat [Kunci milik AWS](#).

Identitas dan manajemen akses untuk AWS IoT

AWS Identity and Access Management (IAM) adalah Layanan AWS yang membantu administrator mengontrol akses ke AWS sumber daya dengan aman. Administrator IAM mengontrol siapa yang dapat diautentikasi (masuk) dan diberi wewenang (memiliki izin) untuk menggunakan sumber daya. AWS IoT IAM adalah Layanan AWS yang dapat Anda gunakan tanpa biaya tambahan.

Topik

- [Audiens](#)
- [Autentikasi dengan identitas IAM](#)
- [Mengelola akses menggunakan kebijakan](#)
- [Bagaimana AWS IoT bekerja dengan IAM](#)
- [AWS IoT contoh kebijakan berbasis identitas](#)
- [AWS kebijakan terkelola untuk AWS IoT](#)
- [Memecahkan masalah AWS IoT identitas dan akses](#)

Audiens

Cara Anda menggunakan AWS Identity and Access Management (IAM) berbeda, tergantung pada pekerjaan yang Anda lakukan. AWS IoT

Pengguna layanan — Jika Anda menggunakan AWS IoT layanan untuk melakukan pekerjaan Anda, maka administrator Anda memberi Anda kredensial dan izin yang Anda butuhkan. Saat Anda menggunakan lebih banyak AWS IoT fitur untuk melakukan pekerjaan Anda, Anda mungkin memerlukan izin tambahan. Memahami cara mengelola akses dapat membantu Anda meminta

izin yang tepat dari administrator Anda. Jika Anda tidak dapat mengakses fitur di AWS IoT, lihat [Memecahkan masalah AWS IoT identitas dan akses](#).

Administrator layanan — Jika Anda bertanggung jawab atas AWS IoT sumber daya di perusahaan Anda, Anda mungkin memiliki akses penuh ke AWS IoT. Tugas Anda adalah menentukan AWS IoT fitur dan sumber daya mana yang harus diakses pengguna layanan Anda. Kemudian, Anda harus mengirimkan permintaan kepada administrator IAM untuk mengubah izin pengguna layanan Anda. Tinjau informasi di halaman ini untuk memahami konsep Basic IAM. Untuk mempelajari lebih lanjut tentang bagaimana perusahaan Anda dapat menggunakan IAM AWS IoT, lihat [Bagaimana AWS IoT bekerja dengan IAM](#).

Administrator IAM – Jika Anda adalah administrator IAM, Anda mungkin ingin belajar dengan lebih detail tentang cara Anda menulis kebijakan untuk mengelola akses ke AWS IoT. Untuk melihat contoh kebijakan AWS IoT berbasis identitas yang dapat Anda gunakan di IAM, lihat [AWS IoT contoh kebijakan berbasis identitas](#)

Autentikasi dengan identitas IAM

Dalam AWS IoT identitas dapat berupa sertifikat perangkat (X.509), identitas Amazon Cognito, atau pengguna atau grup IAM. Topik ini membahas identitas IAM saja. Untuk informasi lebih lanjut tentang identitas lain yang AWS IoT mendukung, lihat [Autentikasi Klien](#).

Otentikasi adalah cara Anda masuk AWS menggunakan kredensial identitas Anda. Anda harus diautentikasi (masuk ke AWS) sebagai Pengguna root akun AWS, sebagai pengguna IAM, atau dengan mengasumsikan peran IAM.

Anda dapat masuk AWS sebagai identitas federasi dengan menggunakan kredensial yang disediakan melalui sumber identitas. AWS IAM Identity Center Pengguna (IAM Identity Center), autentikasi masuk tunggal perusahaan Anda, dan kredensial Google atau Facebook Anda adalah contoh identitas federasi. Saat Anda masuk sebagai identitas terfederasi, administrator Anda sebelumnya menyiapkan federasi identitas menggunakan peran IAM. Ketika Anda mengakses AWS dengan menggunakan federasi, Anda secara tidak langsung mengambil peran.

Bergantung pada jenis pengguna Anda, Anda dapat masuk ke AWS Management Console atau portal AWS akses. Untuk informasi selengkapnya tentang masuk AWS, lihat [Cara masuk ke Panduan AWS Sign-In Pengguna Anda Akun AWS](#).

Jika Anda mengakses AWS secara terprogram, AWS sediakan kit pengembangan perangkat lunak (SDK) dan antarmuka baris perintah (CLI) untuk menandatangani permintaan Anda secara kriptografis dengan menggunakan kredensial Anda. Jika Anda tidak menggunakan AWS alat, Anda

harus menandatangani permintaan sendiri. Untuk informasi selengkapnya tentang penggunaan metode yang disarankan untuk menandatangani permintaan sendiri, lihat [Menandatangani permintaan AWS API](#) di Panduan Pengguna IAM.

Apa pun metode autentikasi yang digunakan, Anda mungkin diminta untuk menyediakan informasi keamanan tambahan. Misalnya, AWS merekomendasikan agar Anda menggunakan otentikasi multi-faktor (MFA) untuk meningkatkan keamanan akun Anda. Untuk mempelajari selengkapnya, lihat [Autentikasi multi-faktor](#) dalam Panduan Pengguna AWS IAM Identity Center dan [Menggunakan autentikasi multi-faktor \(MFA\) dalam AWS](#) dalam Panduan Pengguna IAM.

Akun AWS pengguna root

Saat Anda membuat Akun AWS, Anda mulai dengan satu identitas masuk yang memiliki akses lengkap ke semua Layanan AWS dan sumber daya di akun. Identitas ini disebut pengguna Akun AWS root dan diakses dengan masuk dengan alamat email dan kata sandi yang Anda gunakan untuk membuat akun. Kami sangat menyarankan agar Anda tidak menggunakan pengguna root untuk tugas sehari-hari. Lindungi kredensial pengguna root Anda dan gunakan kredensial tersebut untuk melakukan tugas yang hanya dapat dilakukan pengguna root. Untuk daftar lengkap tugas yang mengharuskan Anda masuk sebagai pengguna root, lihat [Tugas yang memerlukan kredensial pengguna root](#) dalam Panduan Pengguna IAM.

Pengguna dan grup IAM

[Pengguna IAM](#) adalah identitas dalam diri Anda Akun AWS yang memiliki izin khusus untuk satu orang atau aplikasi. Jika memungkinkan, kami merekomendasikan untuk mengandalkan kredensial sementara, bukan membuat pengguna IAM yang memiliki kredensial jangka panjang seperti kata sandi dan kunci akses. Namun, jika Anda memiliki kasus penggunaan tertentu yang memerlukan kredensial jangka panjang dengan pengguna IAM, kami merekomendasikan Anda merotasi kunci akses. Untuk informasi selengkapnya, lihat [Merotasi kunci akses secara teratur untuk kasus penggunaan yang memerlukan kredensial jangka panjang](#) dalam Panduan Pengguna IAM.

[Grup IAM](#) adalah identitas yang menentukan sekumpulan pengguna IAM. Anda tidak dapat masuk sebagai grup. Anda dapat menggunakan grup untuk menentukan izin bagi beberapa pengguna sekaligus. Grup mempermudah manajemen izin untuk sejumlah besar pengguna sekaligus. Misalnya, Anda dapat memiliki grup yang bernama IAMAdmins dan memberikan izin ke grup tersebut untuk mengelola sumber daya IAM.

Pengguna berbeda dari peran. Pengguna secara unik terkait dengan satu orang atau aplikasi, tetapi peran dimaksudkan untuk dapat digunakan oleh siapa pun yang membutuhkannya. Pengguna

memiliki kredensial jangka panjang permanen, tetapi peran memberikan kredensial sementara. Untuk mempelajari selengkapnya, lihat [Kapan harus membuat pengguna IAM \(bukan peran\)](#) dalam Panduan Pengguna IAM.

Peran IAM

[Peran IAM](#) adalah identitas dalam diri Anda Akun AWS yang memiliki izin khusus. Peran ini mirip dengan pengguna IAM, tetapi tidak terkait dengan orang tertentu. Anda dapat mengambil peran IAM untuk sementara AWS Management Console dengan [beralih peran](#). Anda dapat mengambil peran dengan memanggil operasi AWS CLI atau AWS API atau dengan menggunakan URL kustom. Untuk informasi selengkapnya tentang cara menggunakan peran, lihat [Menggunakan peran IAM](#) dalam Panduan Pengguna IAM.

Peran IAM dengan kredensial sementara berguna dalam situasi berikut:

- Akses pengguna terfederasi – Untuk menetapkan izin ke identitas terfederasi, Anda membuat peran dan menentukan izin untuk peran tersebut. Ketika identitas terfederasi mengotentikasi, identitas tersebut terhubung dengan peran dan diberi izin yang ditentukan oleh peran. Untuk informasi tentang peran untuk federasi, lihat [Membuat peran untuk Penyedia Identitas pihak ketiga](#) dalam Panduan Pengguna IAM. Jika menggunakan Pusat Identitas IAM, Anda harus mengonfigurasi set izin. Untuk mengontrol apa yang dapat diakses identitas Anda setelah identitas tersebut diautentikasi, Pusat Identitas IAM akan mengorelasikan set izin ke peran dalam IAM. Untuk informasi tentang set izin, lihat [Set izin](#) dalam Panduan Pengguna AWS IAM Identity Center .
- Izin pengguna IAM sementara – Pengguna atau peran IAM dapat mengambil peran IAM guna mendapatkan berbagai izin secara sementara untuk tugas tertentu.
- Akses lintas akun – Anda dapat menggunakan peran IAM untuk mengizinkan seseorang (prinsipal tepercaya) di akun lain untuk mengakses sumber daya di akun Anda. Peran adalah cara utama untuk memberikan akses lintas akun. Namun, dengan beberapa Layanan AWS, Anda dapat melampirkan kebijakan secara langsung ke sumber daya (alih-alih menggunakan peran sebagai proxy). Untuk mempelajari perbedaan antara peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat [Akses sumber daya lintas akun di IAM di Panduan Pengguna IAM](#).
- Akses lintas layanan — Beberapa Layanan AWS menggunakan fitur lain Layanan AWS. Sebagai contoh, ketika Anda memanggil suatu layanan, biasanya layanan tersebut menjalankan aplikasi di Amazon EC2 atau menyimpan objek di Amazon S3. Sebuah layanan mungkin melakukannya menggunakan izin prinsipal yang memanggil, menggunakan peran layanan, atau peran terkait layanan.

- Sesi akses teruskan (FAS) — Saat Anda menggunakan pengguna atau peran IAM untuk melakukan tindakan AWS, Anda dianggap sebagai prinsipal. Ketika Anda menggunakan beberapa layanan, Anda mungkin melakukan sebuah tindakan yang kemudian menginisiasi tindakan lain di layanan yang berbeda. FAS menggunakan izin dari pemanggilan utama Layanan AWS, dikombinasikan dengan permintaan Layanan AWS untuk membuat permintaan ke layanan hilir. Permintaan FAS hanya dibuat ketika layanan menerima permintaan yang memerlukan interaksi dengan orang lain Layanan AWS atau sumber daya untuk menyelesaikannya. Dalam hal ini, Anda harus memiliki izin untuk melakukan kedua tindakan tersebut. Untuk detail kebijakan ketika mengajukan permintaan FAS, lihat [Sesi akses maju](#).
- Peran layanan – Peran layanan adalah [peran IAM](#) yang dijalankan oleh layanan untuk melakukan tindakan atas nama Anda. Administrator IAM dapat membuat, mengubah, dan menghapus peran layanan dari dalam IAM. Untuk informasi selengkapnya, lihat [Membuat sebuah peran untuk mendelegasikan izin ke Layanan AWS](#) dalam Panduan pengguna IAM.
- Peran terkait layanan — Peran terkait layanan adalah jenis peran layanan yang ditautkan ke peran layanan. Layanan AWS Layanan tersebut dapat menjalankan peran untuk melakukan tindakan atas nama Anda. Peran terkait layanan muncul di Anda Akun AWS dan dimiliki oleh layanan. Administrator IAM dapat melihat, tetapi tidak dapat mengedit izin untuk peran terkait layanan.
- Aplikasi yang berjalan di Amazon EC2 — Anda dapat menggunakan peran IAM untuk mengelola kredensial sementara untuk aplikasi yang berjalan pada instans EC2 dan membuat atau permintaan API. AWS CLI AWS Cara ini lebih dianjurkan daripada menyimpan kunci akses dalam instans EC2. Untuk menetapkan AWS peran ke instans EC2 dan membuatnya tersedia untuk semua aplikasinya, Anda membuat profil instance yang dilampirkan ke instance. Profil instans berisi peran dan memungkinkan program yang berjalan di instans EC2 mendapatkan kredensial sementara. Untuk informasi selengkapnya, lihat [Menggunakan peran IAM untuk memberikan izin ke aplikasi yang berjalan dalam instans Amazon EC2](#) dalam Panduan Pengguna IAM.

Untuk mempelajari apakah kita harus menggunakan peran IAM atau pengguna IAM, lihat [Kapan harus membuat peran IAM \(bukan pengguna\)](#) dalam Panduan Pengguna IAM.

Mengelola akses menggunakan kebijakan

Anda mengontrol akses AWS dengan membuat kebijakan dan melampirkannya ke AWS identitas atau sumber daya. Kebijakan adalah objek AWS yang, ketika dikaitkan dengan identitas atau sumber daya, menentukan izinnya. AWS mengevaluasi kebijakan ini ketika prinsipal (pengguna, pengguna root, atau sesi peran) membuat permintaan. Izin dalam kebijakan menentukan apakah permintaan

diizinkan atau ditolak. Sebagian besar kebijakan disimpan AWS sebagai dokumen JSON. Untuk informasi selengkapnya tentang struktur dan isi dokumen kebijakan JSON, lihat [Gambaran umum kebijakan JSON](#) dalam Panduan Pengguna IAM.

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Artinya, prinsipal manakah yang dapat melakukan tindakan pada sumber daya apa, dan dengan kondisi apa.

Secara default, pengguna dan peran tidak memiliki izin. Untuk memberikan izin kepada pengguna untuk melakukan tindakan di sumber daya yang mereka perlukan, administrator IAM dapat membuat kebijakan IAM. Administrator kemudian dapat menambahkan kebijakan IAM ke peran, dan pengguna dapat mengambil peran.

Kebijakan IAM mendefinisikan izin untuk suatu tindakan terlepas dari metode yang Anda gunakan untuk melakukan operasinya. Misalnya, anggaplah Anda memiliki kebijakan yang mengizinkan tindakan `iam:GetRole`. Pengguna dengan kebijakan tersebut bisa mendapatkan informasi peran dari AWS Management Console, API AWS CLI, atau AWS API.

Kebijakan berbasis identitas

Kebijakan berbasis identitas adalah dokumen kebijakan izin JSON yang dapat Anda lampirkan ke sebuah identitas, seperti pengguna IAM, grup pengguna IAM, atau peran IAM. Kebijakan ini mengontrol jenis tindakan yang dapat dilakukan oleh pengguna dan peran, di sumber daya mana, dan berdasarkan kondisi seperti apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat [Membuat kebijakan IAM](#) dalam Panduan Pengguna IAM.

Kebijakan berbasis identitas dapat dikategorikan lebih lanjut sebagai kebijakan inline atau kebijakan yang dikelola. Kebijakan inline disematkan langsung ke satu pengguna, grup, atau peran. Kebijakan terkelola adalah kebijakan mandiri yang dapat Anda lampirkan ke beberapa pengguna, grup, dan peran dalam Akun AWS. Kebijakan AWS terkelola mencakup kebijakan terkelola dan kebijakan yang dikelola pelanggan. Untuk mempelajari cara memilih antara kebijakan yang dikelola atau kebijakan inline, lihat [Memilih antara kebijakan yang dikelola dan kebijakan inline](#) dalam Panduan Pengguna IAM.

Kebijakan berbasis sumber daya

Kebijakan berbasis sumber daya adalah dokumen kebijakan JSON yang Anda lampirkan ke sumber daya. Contoh kebijakan berbasis sumber daya adalah kebijakan kepercayaan peran IAM dan kebijakan bucket Amazon S3. Dalam layanan yang mendukung kebijakan berbasis sumber daya,

administrator layanan dapat menggunakannya untuk mengontrol akses ke sumber daya tertentu. Untuk sumber daya tempat kebijakan dilampirkan, kebijakan menentukan tindakan apa yang dapat dilakukan oleh prinsipal tertentu pada sumber daya tersebut dan dalam kondisi apa. Anda harus [menentukan prinsipal](#) dalam kebijakan berbasis sumber daya. Prinsipal dapat mencakup akun, pengguna, peran, pengguna federasi, atau. Layanan AWS

Kebijakan berbasis sumber daya merupakan kebijakan inline yang terletak di layanan tersebut. Anda tidak dapat menggunakan kebijakan AWS terkelola dari IAM dalam kebijakan berbasis sumber daya.

Daftar kontrol akses (ACL)

Daftar kontrol akses (ACL) mengendalikan prinsipal mana (anggota akun, pengguna, atau peran) yang memiliki izin untuk mengakses sumber daya. ACL serupa dengan kebijakan berbasis sumber daya, meskipun kebijakan tersebut tidak menggunakan format dokumen kebijakan JSON.

Amazon S3, AWS WAF, dan Amazon VPC adalah contoh layanan yang mendukung ACL. Untuk mempelajari ACL selengkapnya, lihat [Gambaran umum daftar kontrol akses \(ACL\)](#) dalam Panduan Developer Amazon Simple Storage Service.

Jenis-jenis kebijakan lain

AWS mendukung jenis kebijakan tambahan yang kurang umum. Jenis-jenis kebijakan ini dapat mengatur izin maksimum yang diberikan kepada Anda oleh jenis kebijakan yang lebih umum.

- Batasan izin – Batasan izin adalah fitur lanjutan tempat Anda mengatur izin maksimum yang dapat diberikan oleh kebijakan berbasis identitas ke entitas IAM (pengguna IAM atau peran IAM). Anda dapat menetapkan batasan izin untuk suatu entitas. Izin yang dihasilkan adalah perpotongan antara kebijakan berbasis identitas milik entitas dan batasan izinnya. Kebijakan berbasis sumber daya yang menentukan pengguna atau peran dalam bidang `Principal` tidak dibatasi oleh batasan izin. Penolakan eksplisit dalam salah satu kebijakan ini akan menggantikan pemberian izin. Untuk informasi selengkapnya tentang batasan izin, lihat [Batasan izin untuk entitas IAM](#) dalam Panduan Pengguna IAM.
- Kebijakan kontrol layanan (SCP) — SCP adalah kebijakan JSON yang menentukan izin maksimum untuk organisasi atau unit organisasi (OU) di AWS Organizations. AWS Organizations adalah layanan untuk mengelompokkan dan mengelola secara terpusat beberapa Akun AWS yang dimiliki bisnis Anda. Jika Anda mengaktifkan semua fitur di organisasi, Anda dapat menerapkan kebijakan kontrol layanan (SCP) ke salah satu atau semua akun Anda. SCP membatasi izin untuk entitas di akun anggota, termasuk masing-masing. Pengguna root akun AWS Untuk informasi selengkapnya tentang Organisasi dan SCP, lihat [Cara kerja SCP](#) dalam Panduan Pengguna AWS Organizations .

- Kebijakan sesi – Kebijakan sesi adalah kebijakan lanjutan yang Anda berikan sebagai parameter ketika Anda membuat sesi sementara secara programatis untuk peran atau pengguna terfederasi. Izin sesi yang dihasilkan adalah perpotongan antara kebijakan berbasis identitas pengguna atau peran dan kebijakan sesi. Izin juga bisa datang dari kebijakan berbasis sumber daya. Penolakan eksplisit dalam salah satu kebijakan ini akan menggantikan pemberian izin. Untuk informasi selengkapnya, lihat [Kebijakan sesi](#) dalam Panduan Pengguna IAM.

Berbagai jenis kebijakan

Ketika beberapa jenis kebijakan berlaku pada suatu permintaan, izin yang dihasilkan lebih rumit untuk dipahami. Untuk mempelajari cara AWS menentukan apakah akan mengizinkan permintaan saat beberapa jenis kebijakan terlibat, lihat [Logika evaluasi kebijakan](#) di Panduan Pengguna IAM.

Bagaimana AWS IoT bekerja dengan IAM

Sebelum Anda menggunakan IAM untuk mengelola akses AWS IoT, Anda harus memahami fitur IAM mana yang tersedia untuk digunakan. AWS IoT Untuk mendapatkan tampilan tingkat tinggi tentang cara AWS IoT dan AWS layanan lain bekerja dengan IAM, lihat [AWS Layanan yang Bekerja dengan IAM di Panduan Pengguna IAM](#).

Topik

- [AWS IoT kebijakan berbasis identitas](#)
- [AWS IoT Kebijakan berbasis sumber daya](#)
- [Otorisasi berdasarkan tanda AWS IoT](#)
- [AWS IoT Peran IAM](#)

AWS IoT kebijakan berbasis identitas

Dengan kebijakan berbasis identitas IAM, Anda dapat menentukan tindakan dan sumber daya yang diizinkan atau ditolak, serta kondisi di mana tindakan tersebut diperbolehkan atau ditolak. AWS IoT mendukung tindakan tertentu, sumber daya, dan kunci syarat. Untuk mempelajari semua elemen yang Anda gunakan dalam kebijakan JSON, lihat [Referensi elemen kebijakan IAM JSON](#) dalam Panduan Pengguna IAM.


Tindakan

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Artinya, prinsipal manakah yang dapat melakukan tindakan pada sumber daya apa, dan dengan kondisi apa.

Elemen `Action` dari kebijakan JSON menjelaskan tindakan yang dapat Anda gunakan untuk mengizinkan atau menolak akses dalam sebuah kebijakan. Tindakan kebijakan biasanya memiliki nama yang sama dengan operasi AWS API terkait. Ada beberapa pengecualian, misalnya tindakan hanya izin yang tidak memiliki operasi API yang cocok. Ada juga beberapa operasi yang memerlukan beberapa tindakan dalam suatu kebijakan. Tindakan tambahan ini disebut tindakan dependen.


Menyertakan tindakan dalam kebijakan untuk memberikan izin untuk melakukan operasi terkait.

Tabel berikut mencantumkan tindakan IAM IoT, API AWS IoT terkait, dan sumber daya yang dimanipulasi tindakan.

Tindakan kebijakan	AWS IoT API	Sumber daya
iot: TransferAcceptCertificate	AcceptCertificateTransfer	arn:aws:iot: <i>region:account-id</i> :cert/ <i>cert-id</i>
<div style="border: 1px solid #add8e6; border-radius: 10px; padding: 10px; background-color: #e6f2ff;"> <p> Note Yang Akun AWS ditentukan dalam ARN harus berupa akun tempat sertifikat ditransfer.</p> </div>		
iot: GrupAddThingToThing	AddThingToThingKelompok	arn:aws:iot: <i>region:account-id</i> :thinggroup/ <i>thing-group-name</i> arn:aws:iot: <i>region:account-id</i> :thing/ <i>thing-name</i>
IOT: AssociateTargets WithJob	AssociateTargetsWithJob	none
IOT: AttachPolicy	AttachPolicy	arn:aws:iot: <i>region:account-id</i> :thinggroup/ <i>thing-group-name</i>

Tindakan kebijakan	AWS IoT API	Sumber daya
		atau <code>arn:aws:iot: <i>region</i>:<i>account-id</i> :cert/<i>cert-id</i></code>
iot: Kebijakan AttachPrincipal	AttachPrincipalKebijakan	<code>arn:aws:iot: <i>region</i>:<i>account-id</i> :cert/<i>cert-id</i></code>
iot: Profil AttachSecurity	AttachSecurityProfil	<code>arn:aws:iot: <i>region</i>:<i>account-id</i> :security-profile/<i>security-profile-name</i></code> <code>arn:aws:iot: <i>region</i>:<i>account-id</i> :dimension/<i>dimension-name</i></code>
iot: AttachThing Prinsipal	AttachThingKepala Sekolah	<code>arn:aws:iot: <i>region</i>:<i>account-id</i> :cert/<i>cert-id</i></code>
iot: Transfer CancelCertificate	CancelCertificateTransfer	<code>arn:aws:iot: <i>region</i>:<i>account-id</i> :cert/<i>cert-id</i></code>
		<div style="border: 1px solid #add8e6; border-radius: 10px; padding: 10px; background-color: #e6f2ff;"> <p> Note Yang Akun AWS ditentukan dalam ARN harus berupa akun tempat sertifikat ditransfer.</p> </div>
IOT: CancelJob	CancelJob	<code>arn:aws:iot: <i>region</i>:<i>account-id</i> :job/<i>job-id</i></code>
iot: Eksekusi CancelJob	CancelJob Eksekusi	<code>arn:aws:iot: <i>region</i>:<i>account-id</i> :job/<i>job-id</i></code> <code>arn:aws:iot: <i>region</i>:<i>account-id</i> :thing/<i>thing-name</i></code>
iot: ClearDefault Pengotorisasi	ClearDefaultPengotorisasi	Tidak ada

Tindakan kebijakan	AWS IoT API	Sumber daya
IOT: CreateAuthorizer	CreateAuthorizer	arn:aws:iot: <i>region:account-id</i> :authorizer/ <i>authorizer-function-name</i>
IOT: CreateCertificateFromCsr	CreateCertificateFromCsr	*
IOT: CreateDimension	CreateDimension	arn:aws:iot: <i>region:account-id</i> :dimension/ <i>dimension-name</i>
IOT: CreateJob	CreateJob	arn:aws:iot: <i>region:account-id</i> :job/ <i>job-id</i> arn:aws:iot: <i>region:account-id</i> :thinggroup/ <i>thing-group-name</i> arn:aws:iot: <i>region:account-id</i> :thing/ <i>thing-name</i> arn:aws:iot: <i>region:account-id</i> :jobtemplate/ <i>job-template-id</i>
iot: TemplateCreateJob	CreateJobTemplate	arn:aws:iot: <i>region:account-id</i> :job/ <i>job-id</i> arn:aws:iot: <i>region:account-id</i> :jobtemplate/ <i>job-template-id</i>
IOT: CreateKeysAndCertificate	CreateKeysAndCertificate	*
IOT: CreatePolicy	CreatePolicy	arn:aws:iot: <i>region:account-id</i> :policy/ <i>policy-name</i>

Tindakan kebijakan	AWS IoT API	Sumber daya
iot: Versi CreatePolicy	CreatePolicyVersi	arn:aws:iot: <i>region</i> : <i>account-id</i> :policy/ <i>policy-name</i>
<div style="border: 1px solid #add8e6; border-radius: 10px; padding: 10px; background-color: #e6f2ff;"> <p> Note</p> <p>Ini harus menjadi AWS IoT kebijakan, bukan kebijakan IAM.</p> </div>		
iot: CreateRole Alias	CreateRoleAlias	(Parameter: RoleAlias) arn:aws:iot: <i>region</i> : <i>account-id</i> :rolealiases/ <i>role-alias-name</i>
iot: Profil CreateSecurity	CreateSecurityProfil	arn:aws:iot: <i>region</i> : <i>account-id</i> :securityprofile/ <i>security-profile-name</i> arn:aws:iot: <i>region</i> : <i>account-id</i> :dimension/ <i>dimension-name</i>
IOT: CreateThing	CreateThing	arn:aws:iot: <i>region</i> : <i>account-id</i> :thing/ <i>thing-name</i>
iot: Grup CreateThing	CreateThingKelompok	arn:aws:iot: <i>region</i> : <i>account-id</i> :thinggroup/ <i>thing-group-name</i> untuk grup yang sedang dibuat dan untuk grup induk, jika digunakan
iot: Jenis CreateThing	CreateThingTipe	arn:aws:iot: <i>region</i> : <i>account-id</i> :thingtype/ <i>thing-type-name</i>
iot: Aturan CreateTopic	CreateTopicAturan	arn:aws:iot: <i>region</i> : <i>account-id</i> :rule/ <i>rule-name</i>
IOT: DeleteAuthorizer	DeleteAuthorizer	arn:aws:iot: <i>region</i> : <i>account-id</i> :authorizer/ <i>authorizer-name</i>

Tindakan kebijakan	AWS IoT API	Sumber daya
IoT: DeleteCertificate	DeleteCACertificate	arn:aws:iot: <i>region</i> : <i>account-id</i> :cacert/ <i>cert-id</i>
IOT: DeleteCertificate	DeleteCertificate	arn:aws:iot: <i>region</i> : <i>account-id</i> :cert/ <i>cert-id</i>
IOT: DeleteDimension	DeleteDimension	arn:aws:iot: <i>region</i> : <i>account-id</i> :dimension/ <i>dimension-name</i>
IOT: DeleteJob	DeleteJob	arn:aws:iot: <i>region</i> : <i>account-id</i> :job/ <i>job-id</i>
iot: Templat DeleteJob	DeleteJobTemplate	arn:aws:iot: <i>region</i> : <i>account-id</i> :job/ <i>job-template-id</i>
iot: Eksekusi DeleteJob	DeleteJobEksekusi	arn:aws:iot: <i>region</i> : <i>account-id</i> :job/ <i>job-id</i> arn:aws:iot: <i>region</i> : <i>account-id</i> :thing/ <i>thing-name</i>
IOT: DeletePolicy	DeletePolicy	arn:aws:iot: <i>region</i> : <i>account-id</i> :policy/ <i>policy-name</i>
iot: Versi DeletePolicy	DeletePolicyVersi	arn:aws:iot: <i>region</i> : <i>account-id</i> :policy/ <i>policy-name</i>
iot: Kode DeleteRegistration	DeleteRegistrationKode	*
iot: DeleteRoleAlias	DeleteRoleAlias	arn:aws:iot: <i>region</i> : <i>account-id</i> :rolealias/ <i>role-alias-name</i>
iot: Profil DeleteSecurity	DeleteSecurityProfil	arn:aws:iot: <i>region</i> : <i>account-id</i> :securityprofile/ <i>security-profile-name</i> arn:aws:iot: <i>region</i> : <i>account-id</i> :dimension/ <i>dimension-name</i>

Tindakan kebijakan	AWS IoT API	Sumber daya
IOT: DeleteThing	DeleteThing	<code>arn:aws:iot: <i>region</i>:<i>account-id</i> :thing/<i>thing-name</i></code>
iot: Grup DeleteThing	DeleteThingKelompok	<code>arn:aws:iot: <i>region</i>:<i>account-id</i> :thinggroup/<i>thing-group-name</i></code>
iot: Jenis DeleteThing	DeleteThingTipe	<code>arn:aws:iot: <i>region</i>:<i>account-id</i> :thingtype/<i>thing-type-name</i></code>
iot: Aturan DeleteTopic	DeleteTopicAturan	<code>arn:aws:iot: <i>region</i>:<i>account-id</i> :rule/<i>rule-name</i></code>
IoT: DeleteV2 LoggingLevel	HapusEv2 LoggingLevel	<code>arn:aws:iot: <i>region</i>:<i>account-id</i> :thinggroup/<i>thing-group-name</i></code>
iot: Jenis DeprecateThing	DeprecateThingTipe	<code>arn:aws:iot: <i>region</i>:<i>account-id</i> :thingtype/<i>thing-type-name</i></code>
IOT: DescribeAuthorizer	DescribeAuthorizer	<code>arn:aws:iot: <i>region</i>:<i>account-id</i> :authorizer/<i>authorizer-function-name</i></code> (parameter: AuthorizerName) none
IoT: DescribeCACertificate	DescribeCACertificate	<code>arn:aws:iot: <i>region</i>:<i>account-id</i> :cacert/<i>cert-id</i></code>
IOT: DescribeCertificate	DescribeCertificate	<code>arn:aws:iot: <i>region</i>:<i>account-id</i> :cert/<i>cert-id</i></code>
iot: DescribeDefaultPengotorisasi	DescribeDefaultPengotorisasi	Tidak ada
IOT: DescribeEndpoint	DescribeEndpoint	*

Tindakan kebijakan	AWS IoT API	Sumber daya
iot: Konfigurasi DescribeEvent	DescribeEventKonfigurasi	none
IOT: DescribeIndex	DescribeIndex	arn:aws:iot: <i>region</i> : <i>account-id</i> :index/ <i>index-name</i>
IOT: DescribeJob	DescribeJob	arn:aws:iot: <i>region</i> : <i>account-id</i> :job/ <i>job-id</i>
iot: Eksekusi DescribeJob	DescribeJobEksekusi	Tidak ada
iot: Templat DescribeJob	DescribeJobTemplate	arn:aws:iot: <i>region</i> : <i>account-id</i> :job/ <i>job-template-id</i>
iot: DescribeRoleAlias	DescribeRoleAlias	arn:aws:iot: <i>region</i> : <i>account-id</i> :rolealias/ <i>role-alias-name</i>
IOT: DescribeThing	DescribeThing	arn:aws:iot: <i>region</i> : <i>account-id</i> :thing/ <i>thing-name</i>
iot: Grup DescribeThing	DescribeThingKelompok	arn:aws:iot: <i>region</i> : <i>account-id</i> :thinggroup/ <i>thing-group-name</i>
IOT: DescribeThingRegistrationTask	DescribeThingRegistrationTask	Tidak ada
iot: Jenis DescribeThing	DescribeThingTipe	arn:aws:iot: <i>region</i> : <i>account-id</i> :thingtype/ <i>thing-type-name</i>

Tindakan kebijakan	AWS IoT API	Sumber daya
IOT: DetachPolicy	DetachPolicy	arn:aws:iot: <i>region:account-id</i> :cert/ <i>cert-id</i> atau arn:aws:iot: <i>region:account-id</i> :thinggroup/ <i>thing-group-name</i>
iot: Kebijakan DetachPrincipal	DetachPrincipalKebijakan	arn:aws:iot: <i>region:account-id</i> :cert/ <i>cert-id</i>
iot: Profil DetachSecurity	DetachSecurityProfil	arn:aws:iot: <i>region:account-id</i> :securityprofile/ <i>security-profile-name</i> arn:aws:iot: <i>region:account-id</i> :dimension/ <i>dimension-name</i>
iot: DetachThing Prinsipal	DetachThingKepala Sekolah	arn:aws:iot: <i>region:account-id</i> :cert/ <i>cert-id</i>
iot: Aturan DisableTopic	DisableTopicAturan	arn:aws:iot: <i>region:account-id</i> :rule/ <i>rule-name</i>
iot: Aturan EnableTopic	EnableTopicAturan	arn:aws:iot: <i>region:account-id</i> :rule/ <i>rule-name</i>
iot: Kebijakan GetEffective	GetEffectiveKebijakan	arn:aws:iot: <i>region:account-id</i> :cert/ <i>cert-id</i>
iot: Konfigurasi GetIndexing	GetIndexingKonfigurasi	Tidak ada
iot: Dokumen GetJob	GetJobDokumen	arn:aws:iot: <i>region:account-id</i> :job/ <i>job-id</i>

Tindakan kebijakan	AWS IoT API	Sumber daya
iot: Opsi GetLogging	GetLoggingPilihan	*
IOT: GetPolicy	GetPolicy	<code>arn:aws:iot: <i>region</i>:<i>account-id</i> :policy/<i>policy-name</i></code>
iot: Versi GetPolicy	GetPolicyVersi	<code>arn:aws:iot: <i>region</i>:<i>account-id</i> :policy/<i>policy-name</i></code>
iot: Kode GetRegistration	GetRegistrationKode	*
iot: Aturan GetTopic	GetTopicAturan	<code>arn:aws:iot: <i>region</i>:<i>account-id</i> :rule/<i>rule-name</i></code>
iot: Kebijakan ListAttached	ListAttachedKebijakan	<code>arn:aws:iot: <i>region</i>:<i>account-id</i> :thinggroup/<i>thing-group-name</i></code> atau <code>arn:aws:iot: <i>region</i>:<i>account-id</i> :cert/<i>cert-id</i></code>
IOT: ListAuthorizers	ListAuthorizers	Tidak ada
IoT: ListCACertificates	ListCACertificates	*
IOT: ListCertificates	ListCertificates	*
iot: ListCertificates ByCA	ListCertificatesByCA	*
IOT: ListIndices	ListIndices	Tidak ada

Tindakan kebijakan	AWS IoT API	Sumber daya
iot: Job ListJob ExecutionsFor	ListJobExecutionsForJob	Tidak ada
iot: Hal ListJob ExecutionsFor	ListJobExecutionsForHal	Tidak ada
IOT: ListJobs	ListJobs	arn:aws:iot: <i>region</i> : <i>account-id</i> :thinggroup/ <i> thing-group-name</i> jika thingGroupName parameter yang digunakan
iot: Template ListJob	ListJobs	Tidak ada
iot: Sertifikat ListOutgoing	ListOutgoingSertifikat	*
IOT: ListPolicies	ListPolicies	*
iot: Kepala Sekolah ListPolicy	ListPolicyKepala Sekolah	*
iot: Versi ListPolicy	ListPolicyVersi	arn:aws:iot: <i>region</i> : <i>account-id</i> :policy/ <i>policy-name</i>
iot: Kebijakan ListPrincipal	ListPrincipalKebijakan	arn:aws:iot: <i>region</i> : <i>account-id</i> :cert/ <i>cert-id</i>
iot: ListPrincipal Hal-hal	ListPrincipalHal-hal	arn:aws:iot: <i>region</i> : <i>account-id</i> :cert/ <i>cert-id</i>
iot: ListRole Alias	ListRoleAlias	Tidak ada
IOT: ListTargets ForPolicy	ListTargetsForPolicy	arn:aws:iot: <i>region</i> : <i>account-id</i> :policy/ <i>policy-name</i>

Tindakan kebijakan	AWS IoT API	Sumber daya
iot: Grup ListThing	ListThingGrup	Tidak ada
iot: Hal ListThing GroupsFor	ListThing GroupsForHal	arn:aws:iot: <i>region</i> : <i>account-id</i> :thing/ <i>thing-name</i>
iot: Kepala Sekolah ListThing	ListThingKepala Sekolah	arn:aws:iot: <i>region</i> : <i>account-id</i> :thing/ <i>thing-name</i>
iot: Laporan ListThing RegistrationTask	ListThing Registrat ionTaskLaporan	Tidak ada
IOT: ListThing Registrat ionTasks	ListThing Registrat ionTasks	Tidak ada
iot: Jenis ListThing	ListThingJenis	*
IOT: ListThings	ListThings	*
iot: Grup ListThings InThing	ListThing sInThingK elompok	arn:aws:iot: <i>region</i> : <i>account-id</i> :thinggro up/ <i>thing-group-name</i>
iot: Aturan ListTopic	ListTopicAturan	*
IoT: Listv2 LoggingLevels	ListV2 LoggingLevels	Tidak ada
IoT: registerc acertificate	RegisterC ACertificate	*

Tindakan kebijakan	AWS IoT API	Sumber daya
IOT: RegisterCertificate	RegisterCertificate	*
IOT: RegisterThing	RegisterThing	Tidak ada
iot: TransferRejectCertificate	RejectCertificateTransfer	arn:aws:iot: <i>region:account-id</i> :cert/ <i>cert-id</i>
iot: GrupRemoveThingFromThing	RemoveThingFromThingKelompok	arn:aws:iot: <i>region:account-id</i> :thinggroup/ <i>thing-group-name</i> arn:aws:iot: <i>region:account-id</i> :thing/ <i>thing-name</i>
iot: AturanReplaceTopic	ReplaceTopicAturan	arn:aws:iot: <i>region:account-id</i> :rule/ <i>rule-name</i>
IOT: SearchIndex	SearchIndex	arn:aws:iot: <i>region:account-id</i> :index/ <i>index-id</i>
iot: SetDefaultPengotorisasi	SetDefaultPengotorisasi	arn:aws:iot: <i>region:account-id</i> :authorizer/ <i>authorizer-function-name</i>
IOT: SetDefaultPolicyVersion	SetDefaultPolicyVersion	arn:aws:iot: <i>region:account-id</i> :policy/ <i>policy-name</i>
iot: OpsiSetLogging	SetLoggingPilihan	arn:aws:iot: <i>region:account-id</i> :role/ <i>role-name</i>
IoT: Setv2LoggingLevel	SETv2LoggingLevel	arn:aws:iot: <i>region:account-id</i> :thinggroup/ <i>thing-group-name</i>
IoT: Setv2LoggingOptions	SETv2LoggingOptions	arn:aws:iot: <i>region:account-id</i> :role/ <i>role-name</i>

Tindakan kebijakan	AWS IoT API	Sumber daya
IOT: StartThingRegistrationTask	StartThingRegistrationTask	Tidak ada
IOT: StopThingRegistrationTask	StopThingRegistrationTask	Tidak ada
IOT: TestAuthorization	TestAuthorization	arn:aws:iot: <i>region:account-id</i> :cert/ <i>cert-id</i>
iot: TestInvokePengotorisasi	TestInvokePengotorisasi	Tidak ada
IOT: TransferCertificate	TransferCertificate	arn:aws:iot: <i>region:account-id</i> :cert/ <i>cert-id</i>
IOT: UpdateAuthorizer	UpdateAuthorizer	arn:aws:iot: <i>region:account-id</i> :authorizerfunction/ <i>authorizer-function-name</i>
IoT: UpdateCACertificate	UpdateCACertificate	arn:aws:iot: <i>region:account-id</i> :cacert/ <i>cert-id</i>
IOT: UpdateCertificate	UpdateCertificate	arn:aws:iot: <i>region:account-id</i> :cert/ <i>cert-id</i>
IOT: UpdateDimension	UpdateDimension	arn:aws:iot: <i>region:account-id</i> :dimension/ <i>dimension-name</i>
iot: Konfigurasi UpdateEvent	UpdateEventKonfigurasi	Tidak ada
iot: Konfigurasi UpdateIndexing	UpdateIndexingKonfigurasi	Tidak ada
iot: UpdateRoleAlias	UpdateRoleAlias	arn:aws:iot: <i>region:account-id</i> :rolealiases/ <i>role-alias-name</i>

Tindakan kebijakan	AWS IoT API	Sumber daya
iot: Profil UpdateSecurity	UpdateSecurityProfil	arn:aws:iot: <i>region</i> : <i>account-id</i> :securityprofile/ <i>security-profile-name</i> arn:aws:iot: <i>region</i> : <i>account-id</i> :dimension/ <i>dimension-name</i>
IOT: UpdateThing	UpdateThing	arn:aws:iot: <i>region</i> : <i>account-id</i> :thing/ <i>thing-name</i>
iot: Grup UpdateThing	UpdateThingKelompok	arn:aws:iot: <i>region</i> : <i>account-id</i> :thinggroup/ <i>thing-group-name</i>
iot: Hal UpdateThing GroupsFor	UpdateThingGroupsForHal	arn:aws:iot: <i>region</i> : <i>account-id</i> :thing/ <i>thing-name</i> arn:aws:iot: <i>region</i> : <i>account-id</i> :thinggroup/ <i>thing-group-name</i>

Tindakan kebijakan AWS IoT menggunakan awalan berikut sebelum tindakan: `iot:`. Misalnya, untuk memberikan izin kepada seseorang untuk mencantumkan semua hal IoT yang terdaftar di `ListThings` API mereka Akun AWS, Anda menyertakan `iot:ListThings` tindakan tersebut dalam kebijakan mereka. Pernyataan kebijakan harus mencakup salah satu `Action` atau `NotAction` elemen. AWS IoT mendefinisikan serangkaian tindakannya sendiri yang menggambarkan tugas yang dapat Anda lakukan dengan layanan ini.

Untuk menetapkan beberapa tindakan dalam satu pernyataan, pisahkan dengan koma seperti berikut:

```
"Action": [
  "ec2:action1",
  "ec2:action2"
```

Anda dapat menentukan beberapa tindakan menggunakan wildcard (*). Sebagai contoh, untuk menentukan semua tindakan yang dimulai dengan kata `Describe`, sertakan tindakan berikut:

```
"Action": "iot:Describe*"
```

Untuk melihat daftar tindakan, lihat AWS IoT [Tindakan yang Ditentukan oleh AWS IoT](#) dalam Panduan Pengguna IAM.

Tindakan Penasihat Perangkat

Tabel berikut mencantumkan tindakan IAM IoT Device Advisor, Device Advisor API AWS IoT terkait, dan sumber daya yang dimanipulasi tindakan.

Tindakan kebijakan	AWS IoT API	Sumber daya
iotdevice advisor: Definisi CreateSuite	CreateSuiteDefinisi	Tidak ada
iotdevice advisor: Definisi DeleteSuite	DeleteSuiteDefinisi	arn:aws:iotdeviceadvisor: <i>region</i> : <i>account-id</i> :suitedefinition/ <i>suite-definition-id</i>
iotdeviceadvisor: Definisi GetSuite	GetSuiteDefinisi	arn:aws:iotdeviceadvisor: <i>region</i> : <i>account-id</i> :suitedefinition/ <i>suite-definition-id</i>
iotdeviceadvisor: Jalankan GetSuite	GetSuiteJalankan	arn:aws:iotdeviceadvisor: <i>region</i> : <i>account-id</i> :suitedefinition/ <i>suite-run-id</i>
iotdevice advisor: GetSuite RunReport	GetSuiteRunReport	arn:aws:iotdeviceadvisor: <i>region</i> : <i>account-id</i> :suiterun/ <i>suite-definition-id</i> / <i>suite-run-id</i>
iotdeviceadvisor: Definisi ListSuite	ListSuiteDefinisi	Tidak ada
iotdevice advisor: Berjalan ListSuite	ListSuiteBerjalan	arn:aws:iotdeviceadvisor: <i>region</i> : <i>account-id</i> :suitedefinition/ <i>suite-definition-id</i>

Tindakan kebijakan	AWS IoT API	Sumber daya
iotdeviceadvisor: ListTagsForResource	ListTagsForResource	arn:aws:iotdeviceadvisor: <i>region</i> : <i>account-id</i> :suitedefinition/ <i>suite-definition-id</i> arn:aws:iotdeviceadvisor: <i>region</i> : <i>account-id</i> :suiterun/suite-definition-id/ <i>suite-run-id</i>
iotdeviceadvisor: Jalankan StartSuite	StartSuite	arn:aws:iotdeviceadvisor: <i>region</i> : <i>account-id</i> :suitedefinition/ <i>suite-definition-id</i>
iotdeviceadvisor: TagResource	TagResource	arn:aws:iotdeviceadvisor: <i>region</i> : <i>account-id</i> :suitedefinition/ <i>suite-definition-id</i> arn:aws:iotdeviceadvisor: <i>region</i> : <i>account-id</i> :suiterun/suite-definition-id/ <i>suite-run-id</i>
iotdeviceadvisor: UntagResource	UntagResource	arn:aws:iotdeviceadvisor: <i>region</i> : <i>account-id</i> :suitedefinition/ <i>suite-definition-id</i> arn:aws:iotdeviceadvisor: <i>region</i> : <i>account-id</i> :suiterun/suite-definition-id/ <i>suite-run-id</i>
iotdeviceadvisor: Definisi UpdateSuite	UpdateSuite	arn:aws:iotdeviceadvisor: <i>region</i> : <i>account-id</i> :suitedefinition/ <i>suite-definition-id</i>
iotdeviceadvisor: Jalankan StopSuite	StopSuite	arn:aws:iotdeviceadvisor: <i>region</i> : <i>account-id</i> :suiterun/suite-definition-id/ <i>suite-run-id</i>

Tindakan kebijakan di AWS IoT Device Advisor menggunakan awalan berikut sebelum tindakan: `iotdeviceadvisor`. Misalnya, untuk memberikan izin kepada seseorang untuk mencantumkan

semua definisi suite yang terdaftar di ListSuiteDefinitions API mereka Akun AWS , Anda menyertakan `iotdeviceadvisor:ListSuiteDefinitions` tindakan tersebut dalam kebijakan mereka.

Sumber daya

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Artinya, prinsipal manakah yang dapat melakukan tindakan pada sumber daya apa, dan dengan kondisi apa.


Elemen kebijakan JSON `Resource` menentukan objek yang menjadi target penerapan tindakan. Pernyataan harus menyertakan elemen `Resource` atau `NotResource`. Praktik terbaiknya, tentukan sumber daya menggunakan [Amazon Resource Name \(ARN\)](#). Anda dapat melakukan ini untuk tindakan yang mendukung jenis sumber daya tertentu, yang dikenal sebagai izin tingkat sumber daya.

Untuk tindakan yang tidak mendukung izin di tingkat sumber daya, misalnya operasi pencantuman, gunakan wildcard (*) untuk menunjukkan bahwa pernyataan tersebut berlaku untuk semua sumber daya.

```
"Resource": "*"

```

AWS IoT sumber daya

Tindakan kebijakan	AWS IoT API	Sumber daya
iot: Transfer AcceptCertificate	AcceptCertificateTransfer	arn:aws:iot: <i>region</i> : <i>account-id</i> :cert/ <i>cert-id</i>
<div style="border: 1px solid #add8e6; border-radius: 10px; padding: 10px; background-color: #e6f2ff;"> <p> Note</p> <p>Yang Akun AWS ditentukan dalam ARN harus berupa akun tempat sertifikat ditransfer.</p> </div>		
iot: Grup AddThing ToThing	AddThingToThingKelompok	arn:aws:iot: <i>region</i> : <i>account-id</i> :thinggroup/ <i>thing-group-name</i> arn:aws:iot: <i>region</i> : <i>account-id</i> :thing/ <i>thing-name</i>

Tindakan kebijakan	AWS IoT API	Sumber daya
IOT: Associate Targets WithJob	Associate TargetsWithJob	Tidak ada
IOT: AttachPolicy	AttachPolicy	arn:aws:iot: <i>region:account-id</i> :thinggroup/ <i>thing-group-name</i> atau arn:aws:iot: <i>region:account-id</i> :cert/ <i>cert-id</i>
iot: Kebijakan AttachPrincipal	AttachPrincipalKebijakan	arn:aws:iot: <i>region:account-id</i> :cert/ <i>cert-id</i>
iot: AttachThing Prinsipal	AttachThingKepala Sekolah	arn:aws:iot: <i>region:account-id</i> :cert/ <i>cert-id</i>
iot: Transfer CancelCertificate	CancelCertificateTransfer	arn:aws:iot: <i>region:account-id</i> :cert/ <i>cert-id</i>
<div style="border: 1px solid #add8e6; border-radius: 15px; padding: 10px; background-color: #e6f2ff;"> <p> Note Yang Akun AWS ditentukan dalam ARN harus berupa akun tempat sertifikat ditransfer.</p> </div>		
IOT: CancelJob	CancelJob	arn:aws:iot: <i>region:account-id</i> :job/ <i>job-id</i>
iot: Eksekusi CancelJob	CancelJob Eksekusi	arn:aws:iot: <i>region:account-id</i> :job/ <i>job-id</i> arn:aws:iot: <i>region:account-id</i> :thing/ <i>thing-name</i>
iot: ClearDefault Pengotorisasi	ClearDefaultPengotorisasi	Tidak ada

Tindakan kebijakan	AWS IoT API	Sumber daya
IOT: CreateAuthorizer	CreateAuthorizer	arn:aws:iot: <i>region:account-id</i> :authorizer/ <i>authorizer-function-name</i>
IOT: CreateCertificateFromCsr	CreateCertificateFromCsr	*
IOT: CreateJob	CreateJob	arn:aws:iot: <i>region:account-id</i> :job/ <i>job-id</i> arn:aws:iot: <i>region:account-id</i> :thinggroup/ <i>thing-group-name</i> arn:aws:iot: <i>region:account-id</i> :thing/ <i>thing-name</i> arn:aws:iot: <i>region:account-id</i> :jobtemplate/ <i>job-template-id</i>
iot: TemplateCreateJob	CreateJobTemplate	arn:aws:iot: <i>region:account-id</i> :job/ <i>job-id</i> arn:aws:iot: <i>region:account-id</i> :jobtemplate/ <i>job-template-id</i>
IOT: CreateKeysAndCertificate	CreateKeysAndCertificate	*
IOT: CreatePolicy	CreatePolicy	arn:aws:iot: <i>region:account-id</i> :policy/ <i>policy-name</i>
CreatePolicyVersion	iot: VersionCreatePolicy	arn:aws:iot: <i>region:account-id</i> :policy/ <i>policy-name</i>

 **Note**

Ini harus menjadi AWS IoT kebijakan, bukan kebijakan IAM.

Tindakan kebijakan	AWS IoT API	Sumber daya
iot: CreateRole Alias	CreateRoleAlias	(Parameter: RoleAlias) arn:aws:iot: <i>region</i> : <i>account-id</i> :rolealias/ <i>role-alias-name</i>
IOT: CreateThing	CreateThing	arn:aws:iot: <i>region</i> : <i>account-id</i> :thing/ <i>thing-name</i>
iot: Grup CreateThing	CreateThingKelompok	arn:aws:iot: <i>region</i> : <i>account-id</i> :thinggroup/ <i>thing-group-name</i> untuk grup yang sedang dibuat dan untuk grup induk, jika digunakan
iot: Jenis CreateThing	CreateThingTipe	arn:aws:iot: <i>region</i> : <i>account-id</i> :thingtype/ <i>thing-type-name</i>
iot: Aturan CreateTopic	CreateTopicAturan	arn:aws:iot: <i>region</i> : <i>account-id</i> :rule/ <i>rule-name</i>
IOT: DeleteAuthorizer	DeleteAuthorizer	arn:aws:iot: <i>region</i> : <i>account-id</i> :authorizer/ <i>authorizer-name</i>
IoT: DeleteCACertificate	DeleteCACertificate	arn:aws:iot: <i>region</i> : <i>account-id</i> :cacert/ <i>cert-id</i>
IOT: DeleteCertificate	DeleteCertificate	arn:aws:iot: <i>region</i> : <i>account-id</i> :cert/ <i>cert-id</i>
IOT: DeleteJob	DeleteJob	arn:aws:iot: <i>region</i> : <i>account-id</i> :job/ <i>job-id</i>
iot: Eksekusi DeleteJob	DeleteJob Eksekusi	arn:aws:iot: <i>region</i> : <i>account-id</i> :job/ <i>job-id</i> arn:aws:iot: <i>region</i> : <i>account-id</i> :thing/ <i>thing-name</i>

Tindakan kebijakan	AWS IoT API	Sumber daya
iot: Templat DeleteJob	DeleteJob Template	arn:aws:iot: <i>region</i> : <i>account-id</i> :jobtemplate/ <i>job-template-id</i>
IOT: DeletePolicy	DeletePolicy	arn:aws:iot: <i>region</i> : <i>account-id</i> :policy/ <i>policy-name</i>
iot: Versi DeletePolicy	DeletePolicyVersi	arn:aws:iot: <i>region</i> : <i>account-id</i> :policy/ <i>policy-name</i>
iot: Kode DeleteRegistration	DeleteRegistrationKode	*
iot: DeleteRole Alias	DeleteRoleAlias	arn:aws:iot: <i>region</i> : <i>account-id</i> :rolealiases/ <i>role-alias-name</i>
IOT: DeleteThing	DeleteThing	arn:aws:iot: <i>region</i> : <i>account-id</i> :thing/ <i>thing-name</i>
iot: Grup DeleteThing	DeleteThingKelompok	arn:aws:iot: <i>region</i> : <i>account-id</i> :thinggroup/ <i>thing-group-name</i>
iot: Jenis DeleteThing	DeleteThingTipe	arn:aws:iot: <i>region</i> : <i>account-id</i> :thingtype/ <i>thing-type-name</i>
iot: Aturan DeleteTopic	DeleteTopicAturan	arn:aws:iot: <i>region</i> : <i>account-id</i> :rule/ <i>rule-name</i>
IoT: DeleteV2 LoggingLevel	HapusEv2 LoggingLevel	arn:aws:iot: <i>region</i> : <i>account-id</i> :thinggroup/ <i>thing-group-name</i>
iot: Jenis DeprecateThing	DeprecateThingTipe	arn:aws:iot: <i>region</i> : <i>account-id</i> :thingtype/ <i>thing-type-name</i>

Tindakan kebijakan	AWS IoT API	Sumber daya
IOT: DescribeAuthorizer	DescribeAuthorizer	arn:aws:iot: <i>region:account-id</i> :authorizer/ <i>authorizer-function-name</i> (parameter: AuthorizerName) none
IoT: DescribeCACertificate	DescribeCACertificate	arn:aws:iot: <i>region:account-id</i> :cacert/ <i>cert-id</i>
IOT: DescribeCertificate	DescribeCertificate	arn:aws:iot: <i>region:account-id</i> :cert/ <i>cert-id</i>
iot: DescribeDefaultPolicies	DescribeDefaultPolicies	Tidak ada
IOT: DescribeEndpoint	DescribeEndpoint	*
iot: Konfigurasi DescribeEvent	DescribeEventKonfigurasi	none
IOT: DescribeIndex	DescribeIndex	arn:aws:iot: <i>region:account-id</i> :index/ <i>index-name</i>
IOT: DescribeJob	DescribeJob	arn:aws:iot: <i>region:account-id</i> :job/ <i>job-id</i>
iot: Eksekusi DescribeJob	DescribeJobEksekusi	Tidak ada
iot: Templat DescribeJob	DescribeJobTemplate	arn:aws:iot: <i>region:account-id</i> :jobtemplate/ <i>job-template-id</i>
iot: DescribeRoleAlias	DescribeRoleAlias	arn:aws:iot: <i>region:account-id</i> :rolealiases/ <i>role-alias-name</i>

Tindakan kebijakan	AWS IoT API	Sumber daya
IOT: DescribeThing	DescribeThing	arn:aws:iot: <i>region</i> : <i>account-id</i> :thing/ <i>thing-name</i>
iot: Grup DescribeThing	DescribeThingKelompok	arn:aws:iot: <i>region</i> : <i>account-id</i> :thinggroup/ <i>thing-group-name</i>
IOT: DescribeThingRegistrationTask	DescribeThingRegistrationTask	Tidak ada
iot: Jenis DescribeThing	DescribeThingTipe	arn:aws:iot: <i>region</i> : <i>account-id</i> :thingtype/ <i>thing-type-name</i>
IOT: DetachPolicy	DetachPolicy	arn:aws:iot: <i>region</i> : <i>account-id</i> :cert/ <i>cert-id</i> atau arn:aws:iot: <i>region</i> : <i>account-id</i> :thinggroup/ <i>thing-group-name</i>
iot: Kebijakan DetachPrincipal	DetachPrincipalKebijakan	arn:aws:iot: <i>region</i> : <i>account-id</i> :cert/ <i>cert-id</i>
iot: DetachThingPrinsipal	DetachThingKepalaSekolah	arn:aws:iot: <i>region</i> : <i>account-id</i> :cert/ <i>cert-id</i>
iot: Aturan DisableTopic	DisableTopicAturan	arn:aws:iot: <i>region</i> : <i>account-id</i> :rule/ <i>rule-name</i>
iot: Aturan EnableTopic	EnableTopicAturan	arn:aws:iot: <i>region</i> : <i>account-id</i> :rule/ <i>rule-name</i>
iot: Kebijakan GetEffective	GetEffectiveKebijakan	arn:aws:iot: <i>region</i> : <i>account-id</i> :cert/ <i>cert-id</i>

Tindakan kebijakan	AWS IoT API	Sumber daya
iot: Konfigurasi GetIndexing	GetIndexingKonfigurasi	Tidak ada
iot: Dokumen GetJob	GetJobDokumen	arn:aws:iot: <i>region:account-id</i> :job/ <i>job-id</i>
iot: Opsi GetLogging	GetLoggingPilihan	*
IOT: GetPolicy	GetPolicy	arn:aws:iot: <i>region:account-id</i> :policy/ <i>policy-name</i>
iot: Versi GetPolicy	GetPolicyVersi	arn:aws:iot: <i>region:account-id</i> :policy/ <i>policy-name</i>
iot: Kode GetRegistration	GetRegistrationKode	*
iot: Aturan GetTopic	GetTopicAturan	arn:aws:iot: <i>region:account-id</i> :rule/ <i>rule-name</i>
iot: Kebijakan ListAttached	ListAttachedKebijakan	arn:aws:iot: <i>region:account-id</i> :thinggroup/ <i>thing-group-name</i> atau arn:aws:iot: <i>region:account-id</i> :cert/ <i>cert-id</i>
IOT: ListAuthorizers	ListAuthorizers	Tidak ada
IoT: ListCACertificates	ListCACertificates	*
IOT: ListCertificates	ListCertificates	*

Tindakan kebijakan	AWS IoT API	Sumber daya
iot: ListCertificatesByCA	ListCertificatesByCA	*
IOT: ListIndices	ListIndices	Tidak ada
iot: JobListJobExecutionsFor	ListJobExecutionsForJob	Tidak ada
iot: HalListJobExecutionsFor	ListJobExecutionsForHal	Tidak ada
IOT: ListJobs	ListJobs	arn:aws:iot: <i>region</i> : <i>account-id</i> :thinggroup/ <i>thing-group-name</i> jika thingGroupName parameter yang digunakan
iot: TemplateListJob	ListJobTemplate	Tidak ada
iot: SertifikatListOutgoing	ListOutgoingSertifikat	*
IOT: ListPolicies	ListPolicies	*
iot: Kepala Sekolah ListPolicy	ListPolicyKepala Sekolah	arn:aws:iot: <i>region</i> : <i>account-id</i> :policy/ <i>policy-name</i>
iot: Versi ListPolicy	ListPolicyVersi	arn:aws:iot: <i>region</i> : <i>account-id</i> :policy/ <i>policy-name</i>
iot: Kebijakan ListPrincipal	ListPrincipalKebijakan	arn:aws:iot: <i>region</i> : <i>account-id</i> :cert/ <i>cert-id</i>
iot: ListPrincipalHal-hal	ListPrincipalHal-hal	arn:aws:iot: <i>region</i> : <i>account-id</i> :cert/ <i>cert-id</i>

Tindakan kebijakan	AWS IoT API	Sumber daya
iot: ListRole Alias	ListRoleAlias	Tidak ada
IOT: ListTargets ForPolicy	ListTargetsForPolicy	arn:aws:iot: <i>region</i> : <i>account-id</i> :policy/ <i>policy-name</i>
iot: Grup ListThing	ListThingGrup	Tidak ada
iot: Hal ListThing GroupsFor	ListThingGroupsForHal	arn:aws:iot: <i>region</i> : <i>account-id</i> :thing/ <i>thing-name</i>
iot: Kepala Sekolah ListThing	ListThingKepala Sekolah	arn:aws:iot: <i>region</i> : <i>account-id</i> :thing/ <i>thing-name</i>
iot: Laporan ListThing RegistrationTask	ListThingRegistrationTaskLaporan	Tidak ada
IOT: ListThing RegistrationTasks	ListThingRegistrationTasks	Tidak ada
iot: Jenis ListThing	ListThingJenis	*
IOT: ListThings	ListThings	*
iot: Grup ListThings InThing	ListThingInThingKelompok	arn:aws:iot: <i>region</i> : <i>account-id</i> :thinggroup/ <i>thing-group-name</i>
iot: Aturan ListTopic	ListTopicAturan	*

Tindakan kebijakan	AWS IoT API	Sumber daya
IoT: Listv2 LoggingLevels	ListV2 LoggingLevels	Tidak ada
IoT: registercertificate	RegisterCACertificate	*
IOT: RegisterCertificate	RegisterCertificate	*
IOT: RegisterThing	RegisterThing	Tidak ada
iot: TransferRejectCertificate	RejectCertificateTransfer	arn:aws:iot: <i>region:account-id</i> :cert/ <i>cert-id</i>
iot: GroupRemoveThingFromThing	RemoveThingFromThingKelompok	arn:aws:iot: <i>region:account-id</i> :thinggroup/ <i>thing-group-name</i> arn:aws:iot: <i>region:account-id</i> :thing/ <i>thing-name</i>
iot: Aturan ReplaceTopic	ReplaceTopicAturan	arn:aws:iot: <i>region:account-id</i> :rule/ <i>rule-name</i>
IOT: SearchIndex	SearchIndex	arn:aws:iot: <i>region:account-id</i> :index/ <i>index-id</i>
iot: SetDefaultPengotorisasi	SetDefaultPengotorisasi	arn:aws:iot: <i>region:account-id</i> :authorizer/ <i>authorizer-function-name</i>
IOT: SetDefaultPolicyVersion	SetDefaultPolicyVersion	arn:aws:iot: <i>region:account-id</i> :policy/ <i>policy-name</i>
iot: Opsi SetLogging	SetLoggingPilihan	*

Tindakan kebijakan	AWS IoT API	Sumber daya
IoT: SETv2 LoggingLevel	SETv2 LoggingLevel	*
IoT: SETv2 LoggingOptions	SETv2 LoggingOptions	*
IOT: StartThing RegistrationTask	StartThingRegistrationTask	Tidak ada
IOT: StopThing RegistrationTask	StopThingRegistrationTask	Tidak ada
IOT: TestAuthorization	TestAuthorization	arn:aws:iot: <i>region:account-id</i> :cert/ <i>cert-id</i>
iot: TestInvoke Pengotorisasi	TestInvokePengotorisasi	Tidak ada
IOT: TransferCertificate	TransferCertificate	arn:aws:iot: <i>region:account-id</i> :cert/ <i>cert-id</i>
IOT: UpdateAuthorizer	UpdateAuthorizer	arn:aws:iot: <i>region:account-id</i> :authorizerfunction/ <i>authorizer-function-name</i>
IoT: UpdateCertificate	UpdateCACertificate	arn:aws:iot: <i>region:account-id</i> :cacert/ <i>cert-id</i>
IOT: UpdateCertificate	UpdateCertificate	arn:aws:iot: <i>region:account-id</i> :cert/ <i>cert-id</i>
iot: Konfigurasi UpdateEvent	UpdateEventKonfigurasi	Tidak ada
iot: Konfigurasi UpdateIndexing	UpdateIndexingKonfigurasi	Tidak ada

Tindakan kebijakan	AWS IoT API	Sumber daya
iot: UpdateRole Alias	UpdateRoleAlias	arn:aws:iot: <i>region</i> : <i>account-id</i> :rolealias/ <i>role-alias-name</i>
IOT: UpdateThing	UpdateThing	arn:aws:iot: <i>region</i> : <i>account-id</i> :thing/ <i>thing-name</i>
iot: Grup UpdateThing	UpdateThingKelompok	arn:aws:iot: <i>region</i> : <i>account-id</i> :thinggroup/ <i>thing-group-name</i>
iot: Hal UpdateThing GroupsFor	UpdateThingGroupsForHal	arn:aws:iot: <i>region</i> : <i>account-id</i> :thing/ <i>thing-name</i>

Untuk informasi selengkapnya tentang format ARN, lihat [Nama Sumber Daya Amazon \(ARN\) dan Ruang Nama AWS Layanan](#).

Beberapa AWS IoT tindakan, seperti untuk membuat sumber daya, tidak dapat dilakukan pada sumber daya tertentu. Dalam kasus tersebut, Anda harus menggunakan wildcard (*).

```
"Resource": "*"

```

Untuk melihat daftar jenis AWS IoT sumber daya dan ARNnya, lihat [Sumber Daya yang Ditentukan oleh AWS IoT](#) dalam Panduan Pengguna IAM. Untuk mempelajari tindakan mana yang dapat menentukan ARN setiap sumber daya, lihat [Tindakan yang Ditentukan oleh AWS IoT](#).

Sumber daya Penasihat Perangkat

Untuk menentukan batasan tingkat sumber daya untuk kebijakan IAM AWS IoT Device Advisor, gunakan format ARN sumber daya berikut untuk definisi suite dan rangkaian berjalan.

Format ARN sumber daya definisi suite

```
arn:aws:iotdeviceadvisor:region:account-id:suedefinition/suite-definition-id

```

Suite menjalankan format ARN sumber daya

```
arn:aws:iotdeviceadvisor:region:account-id:suiterun/suite-definition-id/suite-run-id
```

Kunci syarat

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Artinya, prinsipal manakah yang dapat melakukan tindakan pada sumber daya apa, dan dengan kondisi apa.

Elemen `Condition` (atau blok `Condition`) akan memungkinkan Anda menentukan kondisi yang menjadi dasar suatu pernyataan berlaku. Elemen `Condition` bersifat opsional. Anda dapat membuat ekspresi bersyarat yang menggunakan [operator kondisi](#), misalnya sama dengan atau kurang dari, untuk mencocokkan kondisi dalam kebijakan dengan nilai-nilai yang diminta.

Jika Anda menentukan beberapa elemen `Condition` dalam sebuah pernyataan, atau beberapa kunci dalam elemen `Condition` tunggal, maka AWS akan mengevaluasinya menggunakan operasi AND logis. Jika Anda menentukan beberapa nilai untuk satu kunci kondisi, AWS mengevaluasi kondisi menggunakan OR operasi logis. Semua kondisi harus dipenuhi sebelum izin pernyataan diberikan.

Anda juga dapat menggunakan variabel placeholder saat menentukan kondisi. Sebagai contoh, Anda dapat memberikan izin kepada pengguna IAM untuk mengakses sumber daya hanya jika izin tersebut mempunyai tag yang sesuai dengan nama pengguna IAM mereka. Untuk informasi selengkapnya, lihat [Elemen kebijakan IAM: variabel dan tag](#) dalam Panduan Pengguna IAM.

AWS mendukung kunci kondisi global dan kunci kondisi khusus layanan. Untuk melihat semua kunci kondisi AWS global, lihat [kunci konteks kondisi AWS global](#) di Panduan Pengguna IAM.

AWS IoT mendefinisikan kumpulan kunci kondisinya sendiri dan juga mendukung penggunaan beberapa kunci kondisi global. Untuk melihat semua kunci kondisi AWS global, lihat [Kunci Konteks Kondisi AWS Global](#) di Panduan Pengguna IAM.

AWS IoT kunci kondisi

AWS IoT kunci kondisi	Deskripsi	Jenis
<code>aws:RequestTag/\${tag-key}</code>	Kunci tag yang ada dalam permintaan yang dibuat pengguna AWS IoT.	String
<code>aws:ResourceTag/\${tag-key}</code>	Komponen kunci tag dari tag yang dilampirkan ke AWS IoT sumber daya.	String
<code>aws:TagKeys</code>	Daftar semua nama kunci tag yang terkait dengan sumber daya dalam permintaan.	String

Untuk melihat daftar kunci AWS IoT kondisi, lihat [Condition Keys untuk AWS IoT](#) di Panduan Pengguna IAM. Untuk mempelajari tindakan dan sumber daya yang dapat Anda gunakan kunci kondisi, lihat [Tindakan yang Ditentukan oleh AWS IoT](#).

Contoh

Untuk melihat contoh kebijakan AWS IoT berbasis identitas, lihat [AWS IoT contoh kebijakan berbasis identitas](#)

AWS IoT Kebijakan berbasis sumber daya

Kebijakan berbasis sumber daya adalah dokumen kebijakan JSON yang menentukan tindakan apa yang dapat dilakukan oleh prinsipal tertentu pada AWS IoT sumber daya dan dalam kondisi apa.

AWS IoT tidak mendukung kebijakan berbasis sumber daya IAM. Namun, itu mendukung kebijakan berbasis AWS IoT sumber daya. Untuk informasi selengkapnya, lihat [AWS IoT Core kebijakan](#).

Otorisasi berdasarkan tanda AWS IoT

Anda dapat melampirkan tag ke AWS IoT sumber daya atau meneruskan tag dalam permintaan AWS IoT. Untuk mengendalikan akses berdasarkan tag, berikan informasi tentang tag di [elemen kondisi](#) dari kebijakan menggunakan kunci kondisi `iot:ResourceTag/key-name`, `aws:RequestTag/key-name`, atau `aws:TagKeys`. Untuk informasi selengkapnya, lihat [Menggunakan tanda dengan kebijakan IAM](#). Untuk informasi selengkapnya tentang menandai AWS IoT sumber daya, lihat [Menandai sumber daya Anda AWS IoT](#).

Untuk melihat contoh kebijakan berbasis identitas untuk membatasi akses ke sumber daya berdasarkan tag pada sumber daya tersebut, lihat [Melihat AWS IoT sumber daya berdasarkan tag](#).

AWS IoT Peran IAM

[Peran IAM](#) adalah entitas di dalam Akun AWS yang memiliki izin khusus.

Menggunakan kredensial sementara dengan AWS IoT

Anda dapat menggunakan kredensial sementara untuk masuk dengan gabungan, menjalankan IAM role, atau menjalankan peran lintas akun. [Anda memperoleh kredensial keamanan sementara dengan memanggil operasi AWS STS API seperti AssumeRole atau GetFederation Token](#).

AWS IoT mendukung menggunakan kredensial sementara.

Peran terkait layanan

[Peran terkait AWS layanan](#) memungkinkan layanan mengakses sumber daya di layanan lain untuk menyelesaikan tindakan atas nama Anda. Peran terkait layanan muncul di akun IAM Anda dan dimiliki oleh layanan tersebut. Administrator IAM dapat melihat tetapi tidak dapat mengedit izin untuk peran terkait layanan.

AWS IoT tidak mendukung peran terkait layanan.

Peran layanan

Fitur ini memungkinkan layanan untuk menerima [peran layanan](#) atas nama Anda. Peran ini mengizinkan layanan untuk mengakses sumber daya di layanan lain untuk menyelesaikan tindakan

atas nama Anda. Peran layanan muncul di akun IAM Anda dan dimiliki oleh akun tersebut. Ini berarti administrator IAM dapat mengubah izin untuk peran ini. Namun, melakukan hal itu dapat merusak fungsionalitas layanan.

AWS IoT contoh kebijakan berbasis identitas

Secara default, pengguna dan peran IAM tidak memiliki izin untuk membuat atau memodifikasi sumber daya AWS IoT. Mereka juga tidak dapat melakukan tugas menggunakan AWS Management Console, AWS CLI, atau AWS API. Administrator IAM harus membuat kebijakan IAM yang memberikan izin kepada pengguna dan peran untuk melakukan operasi API tertentu pada sumber daya yang diperlukan. Administrator kemudian harus melampirkan kebijakan tersebut ke pengguna atau grup yang memerlukan izin tersebut.

Untuk mempelajari cara membuat kebijakan berbasis identitas IAM menggunakan contoh dokumen kebijakan JSON ini, lihat [Membuat Kebijakan pada Tab JSON](#) dalam Panduan Pengguna IAM.

Topik

- [Praktik terbaik kebijakan](#)
- [Menggunakan konsol AWS IoT](#)
- [Izinkan para pengguna untuk melihat izin mereka sendiri](#)
- [Melihat AWS IoT sumber daya berdasarkan tag](#)
- [Melihat sumber daya AWS IoT Device Advisor berdasarkan tag](#)

Praktik terbaik kebijakan

Kebijakan berbasis identitas menentukan apakah seseorang dapat membuat, mengakses, atau menghapus AWS IoT sumber daya di akun Anda. Tindakan ini membuat Akun AWS Anda dikenai biaya. Ketika Anda membuat atau mengedit kebijakan berbasis identitas, ikuti panduan dan rekomendasi ini:

- Mulailah dengan kebijakan AWS terkelola dan beralih ke izin hak istimewa paling sedikit — Untuk mulai memberikan izin kepada pengguna dan beban kerja Anda, gunakan kebijakan AWS terkelola yang memberikan izin untuk banyak kasus penggunaan umum. Mereka tersedia di Akun AWS. Kami menyarankan Anda mengurangi izin lebih lanjut dengan menentukan kebijakan yang dikelola AWS pelanggan yang khusus untuk kasus penggunaan Anda. Untuk informasi selengkapnya, lihat [Kebijakan yang dikelola AWS](#) atau [Kebijakan yang dikelola AWS untuk fungsi tugas](#) dalam Panduan Pengguna IAM.

- Menerapkan izin dengan hak akses paling rendah – Ketika Anda menetapkan izin dengan kebijakan IAM, hanya berikan izin yang diperlukan untuk melakukan tugas. Anda melakukannya dengan mendefinisikan tindakan yang dapat diambil pada sumber daya tertentu dalam kondisi tertentu, yang juga dikenal sebagai izin dengan hak akses paling rendah. Untuk informasi selengkapnya tentang cara menggunakan IAM untuk mengajukan izin, lihat [Kebijakan dan izin dalam IAM](#) dalam Panduan Pengguna IAM.
- Gunakan kondisi dalam kebijakan IAM untuk membatasi akses lebih lanjut – Anda dapat menambahkan suatu kondisi ke kebijakan Anda untuk membatasi akses ke tindakan dan sumber daya. Sebagai contoh, Anda dapat menulis kondisi kebijakan untuk menentukan bahwa semua permintaan harus dikirim menggunakan SSL. Anda juga dapat menggunakan ketentuan untuk memberikan akses ke tindakan layanan jika digunakan melalui yang spesifik Layanan AWS, seperti AWS CloudFormation. Untuk informasi selengkapnya, lihat [Elemen kebijakan JSON IAM: Kondisi](#) dalam Panduan Pengguna IAM.
- Gunakan IAM Access Analyzer untuk memvalidasi kebijakan IAM Anda untuk memastikan izin yang aman dan fungsional – IAM Access Analyzer memvalidasi kebijakan baru dan yang sudah ada sehingga kebijakan tersebut mematuhi bahasa kebijakan IAM (JSON) dan praktik terbaik IAM. IAM Access Analyzer menyediakan lebih dari 100 pemeriksaan kebijakan dan rekomendasi yang dapat ditindaklanjuti untuk membantu Anda membuat kebijakan yang aman dan fungsional. Untuk informasi selengkapnya, lihat [Validasi kebijakan IAM Access Analyzer](#) dalam Panduan Pengguna IAM.
- Memerlukan otentikasi multi-faktor (MFA) - Jika Anda memiliki skenario yang mengharuskan pengguna IAM atau pengguna root di Anda, Akun AWS aktifkan MFA untuk keamanan tambahan. Untuk meminta MFA ketika operasi API dipanggil, tambahkan kondisi MFA pada kebijakan Anda. Untuk informasi selengkapnya, lihat [Mengonfigurasi akses API yang dilindungi MFA](#) dalam Panduan Pengguna IAM.

Untuk informasi selengkapnya tentang praktik terbaik dalam IAM, lihat [Praktik terbaik keamanan dalam IAM](#) dalam Panduan Pengguna IAM.

Menggunakan konsol AWS IoT

Untuk mengakses AWS IoT konsol, Anda harus memiliki set izin minimum. Izin ini harus memungkinkan Anda untuk membuat daftar dan melihat detail tentang AWS IoT sumber daya di Anda Akun AWS. Jika Anda membuat kebijakan berbasis identitas yang lebih ketat daripada izin minimum yang diperlukan, konsol tidak akan berfungsi sebagaimana mestinya untuk entitas (pengguna atau peran) dengan kebijakan tersebut.

Untuk memastikan bahwa entitas tersebut masih dapat menggunakan AWS IoT konsol, lampirkan juga kebijakan AWS terkelola berikut ke entitas:AWSIoTFullAccess. Untuk informasi selengkapnya, lihat [Menambahkan Izin ke Pengguna](#) dalam Panduan Pengguna IAM.

Anda tidak perlu mengizinkan izin konsol minimum untuk pengguna yang melakukan panggilan hanya ke AWS CLI atau AWS API. Sebagai alternatif, hanya izinkan akses ke tindakan yang cocok dengan operasi API yang sedang Anda coba lakukan.

Izinkan para pengguna untuk melihat izin mereka sendiri

Contoh ini menunjukkan cara membuat kebijakan yang mengizinkan pengguna IAM melihat kebijakan inline dan terkelola yang dilampirkan ke identitas pengguna mereka. Kebijakan ini mencakup izin untuk menyelesaikan tindakan ini di konsol atau menggunakan API atau secara terprogram. AWS CLI AWS

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ViewOwnUserInfo",
      "Effect": "Allow",
      "Action": [
        "iam:GetUserPolicy",
        "iam:ListGroupForUser",
        "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": "*"
  }
]
}

```

Melihat AWS IoT sumber daya berdasarkan tag

Anda dapat menggunakan syarat dalam kebijakan berbasis identitas Anda untuk mengontrol akses ke sumber daya AWS IoT berdasarkan tanda. Contoh ini menunjukkan bagaimana Anda dapat membuat kebijakan yang memungkinkan melihat sesuatu. Namun, izin diberikan hanya jika tag benda `Owner` memiliki nilai nama pengguna pengguna tersebut. Kebijakan ini juga memberi izin yang diperlukan untuk menyelesaikan tindakan ini pada konso tersebut.

```

{
  "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}"}
      }
    }
  ]
}

```

Anda dapat melampirkan kebijakan ini ke pengguna IAM di akun Anda. Jika pengguna bernama `richard-roe` mencoba untuk melihat grup AWS IoT penagihan, grup penagihan harus diberi tag `Owner=richard-roe` atau `owner=richard-roe`. Jika tidak, aksesnya akan ditolak. Kunci tanda syarat `Owner` sama dengan kedua `Owner` dan `owner` karena nama kunci syarat tidak terpengaruh huruf besar/kecil. Untuk informasi lebih lanjut, lihat [Elemen Kebijakan IAM JSON: Persyaratan](#) dalam Panduan Pengguna IAM.

Melihat sumber daya AWS IoT Device Advisor berdasarkan tag

Anda dapat menggunakan kondisi dalam kebijakan berbasis identitas untuk mengontrol akses ke sumber daya AWS IoT Device Advisor berdasarkan tag. Contoh berikut menunjukkan bagaimana Anda dapat membuat kebijakan yang memungkinkan melihat definisi suite tertentu. Namun, izin diberikan hanya jika tag definisi suite telah `SuiteType` disetel ke nilai `MQTT`. Kebijakan ini juga memberi izin yang diperlukan untuk menyelesaikan tindakan ini pada konsol tersebut.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ViewSuiteDefinition",
      "Effect": "Allow",
      "Action": "iotdeviceadvisor:GetSuiteDefinition",
      "Resource": "arn:aws:iotdeviceadvisor:*:*:suitedefinition/*",
      "Condition": {
        "StringEquals": {"aws:ResourceTag/SuiteType": "MQTT"}
      }
    }
  ]
}
```

AWS kebijakan terkelola untuk AWS IoT

Untuk menambahkan izin ke pengguna, grup, dan peran, lebih mudah menggunakan kebijakan AWS terkelola daripada menulis kebijakan sendiri. Dibutuhkan waktu dan keahlian untuk [membuat kebijakan yang dikelola pelanggan IAM](#) yang hanya memberi tim Anda izin yang mereka butuhkan. Untuk memulai dengan cepat, Anda dapat menggunakan kebijakan AWS terkelola kami. Kebijakan ini mencakup kasus penggunaan umum dan tersedia di Akun AWS Anda. Untuk informasi selengkapnya tentang kebijakan AWS [AWS terkelola, lihat kebijakan terkelola](#) di Panduan Pengguna IAM.

AWS layanan memelihara dan memperbarui kebijakan AWS terkelola. Anda tidak dapat mengubah izin dalam kebijakan AWS terkelola. Layanan terkadang menambahkan izin tambahan ke kebijakan yang dikelola AWS untuk mendukung fitur-fitur baru. Jenis pembaruan ini akan memengaruhi semua identitas (pengguna, grup, dan peran) di mana kebijakan tersebut dilampirkan. Layanan kemungkinan besar akan memperbarui kebijakan yang dikelola AWS saat ada fitur baru yang diluncurkan atau

saat ada operasi baru yang tersedia. Layanan tidak menghapus izin dari kebijakan AWS terkelola, sehingga pembaruan kebijakan tidak akan merusak izin yang ada.

Selain itu, AWS mendukung kebijakan terkelola untuk fungsi pekerjaan yang mencakup beberapa layanan. Misalnya, kebijakan AWS terkelola `ReadOnlyAccess` menyediakan akses hanya-baca ke semua AWS layanan dan sumber daya. Saat layanan meluncurkan fitur baru, AWS menambahkan izin hanya-baca untuk operasi dan sumber daya baru. Untuk melihat daftar dan deskripsi dari kebijakan fungsi tugas, lihat [kebijakan yang dikelola AWS untuk fungsi tugas](#) di Panduan Pengguna IAM.

Note

AWS IoT bekerja dengan kebijakan keduanya AWS IoT dan IAM. Topik ini hanya membahas kebijakan IAM, yang mendefinisikan tindakan kebijakan untuk operasi API bidang kontrol dan bidang data. Lihat juga [AWS IoT Core kebijakan](#).

AWS kebijakan terkelola: `AWSIoTConfigAccess`

Anda dapat melampirkan kebijakan `AWSIoTConfigAccess` ke identitas IAM Anda.

Kebijakan ini memberikan izin identitas terkait yang memungkinkan akses ke semua operasi AWS IoT konfigurasi. Kebijakan ini dapat memengaruhi pemrosesan dan penyimpanan data. Untuk melihat kebijakan ini di AWS Management Console, lihat [AWSIoTConfigAccess](#).

Detail izin

Kebijakan ini mencakup izin berikut.

- `iot`— Mengambil AWS IoT data dan melakukan tindakan konfigurasi 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 kebijakan terkelola: AWSIoTConfigReadOnlyAccess

Anda dapat melampirkan kebijakan AWSIoTConfigReadOnlyAccess ke identitas IAM Anda.

Kebijakan ini memberikan izin identitas terkait yang memungkinkan akses hanya-baca ke semua operasi konfigurasi. AWS IoT Untuk melihat kebijakan ini di AWS Management Console, lihat [AWSIoTConfigReadOnlyAccess](#).

Detail izin

Kebijakan ini mencakup izin berikut.

- `iot`— Lakukan operasi read-only dari tindakan konfigurasi 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 kebijakan terkelola: AWSIoTDataAccess

Anda dapat melampirkan kebijakan `AWSIoTDataAccess` ke identitas IAM Anda.

Kebijakan ini memberikan izin identitas terkait yang memungkinkan akses ke semua operasi AWS IoT data. Operasi data mengirim data melalui protokol MQTT atau HTTP. Untuk melihat kebijakan ini di AWS Management Console, lihat [AWSIoTDataAccess](#).

Detail izin

Kebijakan ini mencakup izin berikut.

- `iot`— Ambil AWS IoT data dan izinkan akses penuh ke tindakan AWS IoT pengiriman pesan.

```
{
  "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 kebijakan terkelola: `AWSIoTFullAccess`

Anda dapat melampirkan kebijakan `AWSIoTFullAccess` ke identitas IAM Anda.

Kebijakan ini memberikan izin identitas terkait yang memungkinkan akses ke semua operasi AWS IoT konfigurasi dan pesan. Untuk melihat kebijakan ini di AWS Management Console, lihat [AWSIoTFullAccess](#).

Detail izin

Kebijakan ini mencakup izin berikut.

- `iot`— Ambil AWS IoT data dan izinkan akses penuh ke AWS IoT konfigurasi dan tindakan pengiriman pesan.
- `iotjobsdata`— Ambil data AWS IoT Jobs dan izinkan akses penuh ke operasi API pesawat data AWS IoT Jobs.

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

AWS kebijakan terkelola: AWSIoTLogging

Anda dapat melampirkan kebijakan AWSIoTLogging ke identitas IAM Anda.

Kebijakan ini memberikan izin identitas terkait yang memungkinkan akses untuk membuat grup Amazon CloudWatch Logs dan mengalirkan log ke grup. Kebijakan ini dilampirkan pada peran CloudWatch pencatatan Anda. Untuk melihat kebijakan ini di AWS Management Console, lihat [AWSIoTLogging](#).

Detail izin

Kebijakan ini mencakup izin berikut.

- **logs**— Ambil CloudWatch log. Juga memungkinkan pembuatan grup CloudWatch Log dan streaming log ke grup.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:PutMetricFilter",
        "logs:PutRetentionPolicy",
        "logs:GetLogEvents",
        "logs>DeleteLogStream"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

AWS kebijakan terkelola: AWSIoTOTAUpdate

Anda dapat melampirkan kebijakan AWSIoTOTAUpdate ke identitas IAM Anda.

Kebijakan ini memberikan izin identitas terkait yang memungkinkan akses untuk membuat AWS IoT lowongan, pekerjaan penandatanganan AWS IoT kode, dan menjelaskan pekerjaan penandatanganan AWS kode. Untuk melihat kebijakan ini di AWS Management Console, lihat [AWSIoTOTAUpdate](#).

Detail izin

Kebijakan ini mencakup izin berikut.

- `iot`— Buat AWS IoT pekerjaan dan pekerjaan penandatanganan kode.
- `signer`— Lakukan pembuatan pekerjaan penandatanganan AWS kode.

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

AWS kebijakan terkelola: AWSIoTRuleActions

Anda dapat melampirkan kebijakan `AWSIoTRuleActions` ke identitas IAM Anda.

Kebijakan ini memberikan izin identitas terkait yang memungkinkan akses ke semua Layanan AWS yang didukung dalam tindakan AWS IoT aturan. Untuk melihat kebijakan ini di AWS Management Console, lihat [AWSIoTRuleActions](#).

Detail izin

Kebijakan ini mencakup izin berikut.

- `iot`- Lakukan tindakan untuk menerbitkan pesan tindakan aturan.
- `dynamodb`- Masukkan pesan ke dalam tabel DynamoDB atau membagi pesan menjadi beberapa kolom tabel DynamoDB.

- s3- Simpan objek di ember Amazon S3.
- kinesis- Kirim pesan ke objek aliran Amazon Kinesis.
- firehose- Masukkan catatan dalam objek aliran Firehose.
- cloudwatch- Ubah status CloudWatch alarm atau kirim data pesan ke CloudWatch metrik.
- sns- Lakukan operasi untuk mempublikasikan pemberitahuan menggunakan Amazon SNS. Operasi ini mencakup topik AWS IoT SNS.
- sqs- Masukkan pesan untuk ditambahkan ke antrian SQS.
- es- Kirim pesan ke OpenSearch layanan Layanan.

```
{
  "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 kebijakan terkelola: AWSIoTThingsRegistration

Anda dapat melampirkan kebijakan AWSIoTThingsRegistration ke identitas IAM Anda.

Kebijakan ini memberikan izin identitas terkait yang memungkinkan akses untuk mendaftarkan sesuatu secara massal menggunakan API. StartThingRegistrationTask Kebijakan ini dapat

memengaruhi pemrosesan dan penyimpanan data. Untuk melihat kebijakan ini di AWS Management Console, lihat [AWSIoTThingsRegistration](#).

Detail izin

Kebijakan ini mencakup izin berikut.

- `iot-` Lakukan tindakan untuk membuat sesuatu dan melampirkan kebijakan dan sertifikat saat mendaftar secara massal.

```
{
  "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 pembaruan kebijakan AWS terkelola

Lihat detail tentang pembaruan kebijakan AWS terkelola AWS IoT sejak layanan ini mulai melacak perubahan ini. Untuk peringatan otomatis tentang perubahan pada halaman ini, berlangganan umpan RSS di halaman Riwayat AWS IoT dokumen.

Perubahan	Deskripsi	Tanggal
AWSIoTFullAccess — Pembaruan ke kebijakan yang sudah ada	AWS IoT menambahkan izin baru untuk memungkinkan pengguna mengakses operasi API bidang data AWS IoT Jobs menggunakan protokol HTTP. Awalan kebijakan IAM baru, <code>iotjobsdata:</code> , memberi Anda kontrol akses berbutir lebih halus untuk mengakses titik akhir bidang data AWS IoT Jobs. Untuk operasi API bidang kontrol,	Mei 11, 2022

Perubahan	Deskripsi	Tanggal
	Anda masih menggunakan <code>iot: awalan</code> . Untuk informasi selengkapnya, lihat AWS IoT Core kebijakan untuk protokol HTTPS .	
AWS IoT mulai melacak perubahan	AWS IoT mulai melacak perubahan untuk kebijakan AWS terkelolanya.	Mei 11, 2022

Memecahkan masalah AWS IoT identitas dan akses

Gunakan informasi berikut untuk membantu Anda mendiagnosis dan memperbaiki masalah umum yang mungkin Anda temui saat bekerja dengan AWS IoT dan IAM.

Topik

- [Saya tidak berwenang untuk melakukan tindakan di AWS IoT](#)
- [Saya tidak berwenang untuk melakukan `iam: PassRole`](#)
- [Saya ingin mengizinkan orang di luar saya Akun AWS untuk mengakses AWS IoT sumber daya saya](#)

Saya tidak berwenang untuk melakukan tindakan di AWS IoT

Jika Anda menerima pesan kesalahan bahwa Anda tidak memiliki otorisasi untuk melakukan tindakan, kebijakan Anda harus diperbarui agar Anda dapat melakukan tindakan tersebut.

Contoh kesalahan berikut terjadi ketika pengguna IAM, `mateojackson`, mencoba menggunakan konsol untuk melihat detail tentang sumber daya sesuatu tetapi tidak memiliki `iot:DescribeThing` izin.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
iot:DescribeThing on resource: MyIoTThing
```

Dalam hal ini, kebijakan untuk `mateojackson` pengguna harus diperbarui untuk mengizinkan akses ke sumber daya benda dengan menggunakan `iot:DescribeThing` tindakan.

Jika Anda memerlukan bantuan, hubungi AWS administrator Anda. Administrator Anda adalah orang yang memberi Anda kredensial masuk.

Menggunakan AWS IoT Device Advisor

Jika Anda menggunakan AWS IoT Device Advisor, contoh error berikut terjadi saat pengguna mateojackson mencoba menggunakan konsol untuk melihat detail tentang definisi suite tetapi tidak memiliki `iotdeviceadvisor:GetSuiteDefinition` izin.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
iotdeviceadvisor:GetSuiteDefinition on resource: MySuiteDefinition
```

Dalam hal ini, kebijakan untuk mateojackson pengguna harus diperbarui untuk mengizinkan akses ke `MySuiteDefinition` sumber daya menggunakan `iotdeviceadvisor:GetSuiteDefinition` tindakan.

Saya tidak berwenang untuk melakukan iam: PassRole

Jika Anda menerima kesalahan yang tidak diizinkan untuk melakukan `iam:PassRole` tindakan, kebijakan Anda harus diperbarui agar Anda dapat meneruskan peran AWS IoT.

Beberapa Layanan AWS memungkinkan Anda untuk meneruskan peran yang ada ke layanan tersebut alih-alih membuat peran layanan baru atau peran terkait layanan. Untuk melakukannya, Anda harus memiliki izin untuk meneruskan peran ke layanan.

Contoh kesalahan berikut terjadi ketika pengguna IAM bernama marymajor mencoba menggunakan konsol tersebut untuk melakukan tindakan di AWS IoT. Namun, tindakan tersebut memerlukan layanan untuk mendapatkan izin yang diberikan oleh peran layanan. Mary tidak memiliki izin untuk meneruskan peran tersebut pada layanan.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
iam:PassRole
```

Dalam kasus ini, kebijakan Mary harus diperbarui agar dia mendapatkan izin untuk melakukan tindakan `iam:PassRole` tersebut.

Jika Anda memerlukan bantuan, hubungi AWS administrator Anda. Administrator Anda adalah orang yang memberi Anda kredensial masuk.

Saya ingin mengizinkan orang di luar saya Akun AWS untuk mengakses AWS IoT sumber daya saya

Anda dapat membuat peran yang dapat digunakan pengguna di akun lain atau orang-orang di luar organisasi Anda untuk mengakses sumber daya Anda. Anda dapat menentukan siapa saja yang dipercaya untuk mengambil peran tersebut. Untuk layanan yang mendukung kebijakan berbasis sumber daya atau daftar kontrol akses (ACL), Anda dapat menggunakan kebijakan tersebut untuk memberi orang akses ke sumber daya Anda.

Untuk mempelajari selengkapnya, periksa referensi berikut:

- Untuk mempelajari apakah AWS IoT mendukung fitur-fitur ini, lihat [Bagaimana AWS IoT bekerja dengan IAM](#).
- Untuk mempelajari cara menyediakan akses ke sumber daya Anda di seluruh sumber daya Akun AWS yang Anda miliki, lihat [Menyediakan akses ke pengguna IAM di pengguna lain Akun AWS yang Anda miliki](#) di Panduan Pengguna IAM.
- Untuk mempelajari cara menyediakan akses ke sumber daya Anda kepada pihak ketiga Akun AWS, lihat [Menyediakan akses yang Akun AWS dimiliki oleh pihak ketiga](#) dalam Panduan Pengguna IAM.
- Untuk mempelajari cara memberikan akses melalui federasi identitas, lihat [Menyediakan akses ke pengguna terautentikasi eksternal \(federasi identitas\)](#) dalam Panduan Pengguna IAM.
- Untuk mempelajari perbedaan antara menggunakan peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat [Akses sumber daya lintas akun di IAM di Panduan Pengguna IAM](#).

Pembuatan Log dan Pemantauan

Pemantauan adalah bagian penting dari menjaga keandalan, ketersediaan, dan kinerja AWS IoT dan AWS solusi Anda. Anda harus mengumpulkan data pemantauan dari semua bagian AWS solusi Anda sehingga Anda dapat lebih mudah men-debug kegagalan multi-titik, jika terjadi. Untuk informasi tentang prosedur pencatatan dan pemantauan, lihat [Pemantauan AWS IoT](#)

Alat Pemantauan

AWS menyediakan alat yang dapat Anda gunakan untuk memantau AWS IoT. Anda dapat mengonfigurasi beberapa alat ini agar melakukan pemantauan untuk Anda. Beberapa alat memerlukan intervensi manual. Kami menyarankan agar Anda mengotomasi tugas pemantauan sebanyak mungkin.

Alat Pemantauan Otomatis

Anda dapat menggunakan alat pemantauan otomatis berikut untuk menonton AWS IoT dan melaporkan ketika ada sesuatu yang salah:

- **CloudWatch Alarm Amazon** — Tonton satu metrik selama periode waktu yang Anda tentukan, dan lakukan satu atau beberapa tindakan berdasarkan nilai metrik relatif terhadap ambang batas tertentu selama beberapa periode waktu. Tindakannya adalah pemberitahuan yang dikirim ke topik Amazon Simple Notification Service (Amazon SNS) atau kebijakan Amazon EC2 Auto Scaling. CloudWatch alarm tidak memanggil tindakan hanya karena mereka berada dalam keadaan tertentu. Status harus diubah dan dipelihara selama jangka waktu tertentu. Untuk informasi selengkapnya, lihat [Pantau AWS IoT alarm dan metrik menggunakan Amazon CloudWatch](#).
- **Amazon CloudWatch Logs** — Pantau, simpan, dan akses file log Anda dari AWS CloudTrail atau sumber lain. Amazon CloudWatch Logs juga memungkinkan Anda melihat langkah-langkah penting yang diambil oleh kasus pengujian AWS IoT Device Advisor, peristiwa yang dihasilkan, dan pesan MQTT yang dikirim dari perangkat Anda atau AWS IoT Core selama eksekusi pengujian. Log ini memungkinkan untuk men-debug dan mengambil tindakan korektif pada perangkat Anda. Untuk informasi selengkapnya, lihat [Monitor AWS IoT menggunakan CloudWatch Log](#) Untuk informasi selengkapnya tentang menggunakan Amazon CloudWatch, lihat [Memantau File Log](#) di Panduan CloudWatch Pengguna Amazon.
- **CloudWatch Acara Amazon** — Cocokkan acara dan arahkan ke satu atau beberapa fungsi atau aliran target untuk membuat perubahan, menangkap informasi status, dan mengambil tindakan korektif. Untuk informasi selengkapnya, lihat [Apa Itu CloudWatch Acara Amazon](#) di Panduan CloudWatch Pengguna Amazon.
- **AWS CloudTrail Pemantauan Log** - Bagikan file log antar akun, pantau file CloudTrail log secara real time dengan mengirimkannya ke CloudWatch Log, menulis aplikasi pemrosesan log di Java, dan validasi bahwa file log Anda tidak berubah setelah pengiriman oleh CloudTrail. Untuk informasi selengkapnya, lihat [Logging panggilan AWS IoT API menggunakan AWS CloudTrail](#) dan juga [Bekerja dengan File CloudTrail Log](#) di Panduan AWS CloudTrail Pengguna.

Alat Pemantauan Manual

Bagian penting lainnya dari pemantauan AWS IoT melibatkan pemantauan secara manual item-item yang tidak tercakup oleh CloudWatch alarm. Dasbor AWS IoT CloudWatch, dan konsol AWS layanan lainnya memberikan at-a-glance tampilan status AWS lingkungan Anda. Kami menyarankan Anda juga memeriksa file log AWS IoT.

- AWS IoT dasbor menunjukkan:
 - Sertifikat CA
 - Sertifikat
 - Kebijakan
 - Aturan
 - Hal-hal
- CloudWatch halaman rumah menunjukkan:
 - Alarm dan status saat ini.
 - Grafik alarm dan sumber daya.
 - Status kesehatan layanan.

Anda dapat menggunakan CloudWatch untuk melakukan hal berikut:

- Buat [dasbor yang disesuaikan](#) untuk memantau layanan yang Anda pedulikan.
- Data metrik grafik untuk memecahkan masalah dan menemukan tren.
- Cari dan telusuri semua metrik AWS sumber daya Anda.
- Buat dan sunting alarm untuk menerima pemberitahuan tentang masalah.

Validasi kepatuhan untuk Core AWS IoT


Untuk mempelajari apakah an Layanan AWS berada dalam lingkup program kepatuhan tertentu, lihat [Layanan AWS di Lingkup oleh Program Kepatuhan Layanan AWS](#) dan pilih program kepatuhan yang Anda minati. Untuk informasi umum, lihat [Program AWS Kepatuhan Program AWS](#) .

Anda dapat mengunduh laporan audit pihak ketiga menggunakan AWS Artifact. Untuk informasi selengkapnya, lihat [Mengunduh Laporan di AWS Artifact](#) .

Tanggung jawab kepatuhan Anda saat menggunakan Layanan AWS ditentukan oleh sensitivitas data Anda, tujuan kepatuhan perusahaan Anda, dan hukum dan peraturan yang berlaku. AWS menyediakan sumber daya berikut untuk membantu kepatuhan:

- [Panduan Memulai Cepat Keamanan dan Kepatuhan — Panduan](#) penerapan ini membahas pertimbangan arsitektur dan memberikan langkah-langkah untuk menerapkan lingkungan dasar AWS yang berfokus pada keamanan dan kepatuhan.

- [Arsitektur untuk Keamanan dan Kepatuhan HIPAA di Amazon Web Services](#) — Whitepaper ini menjelaskan bagaimana perusahaan dapat menggunakan AWS untuk membuat aplikasi yang memenuhi syarat HIPAA.

 Note

Tidak semua memenuhi Layanan AWS syarat HIPAA. Untuk informasi selengkapnya, lihat [Referensi Layanan yang Memenuhi Syarat HIPAA](#).

- [AWS Sumber Daya AWS](#) — Kumpulan buku kerja dan panduan ini mungkin berlaku untuk industri dan lokasi Anda.
- [AWS Panduan Kepatuhan Pelanggan](#) - Memahami model tanggung jawab bersama melalui lensa kepatuhan. Panduan ini merangkum praktik terbaik untuk mengamankan Layanan AWS dan memetakan panduan untuk kontrol keamanan di berbagai kerangka kerja (termasuk Institut Standar dan Teknologi Nasional (NIST), Dewan Standar Keamanan Industri Kartu Pembayaran (PCI), dan Organisasi Internasional untuk Standardisasi (ISO)).
- [Mengevaluasi Sumber Daya dengan Aturan](#) dalam Panduan AWS Config Pengembang — AWS Config Layanan menilai seberapa baik konfigurasi sumber daya Anda mematuhi praktik internal, pedoman industri, dan peraturan.
- [AWS Security Hub](#)— Ini Layanan AWS memberikan pandangan komprehensif tentang keadaan keamanan Anda di dalamnya AWS. Security Hub menggunakan kontrol keamanan untuk sumber daya AWS Anda serta untuk memeriksa kepatuhan Anda terhadap standar industri keamanan dan praktik terbaik. Untuk daftar layanan dan kontrol yang didukung, lihat [Referensi kontrol Security Hub](#).
- [Amazon GuardDuty](#) — Ini Layanan AWS mendeteksi potensi ancaman terhadap beban kerja Akun AWS, kontainer, dan data Anda dengan memantau lingkungan Anda untuk aktivitas mencurigakan dan berbahaya. GuardDuty dapat membantu Anda mengatasi berbagai persyaratan kepatuhan, seperti PCI DSS, dengan memenuhi persyaratan deteksi intrusi yang diamanatkan oleh kerangka kerja kepatuhan tertentu.
- [AWS Audit Manager](#) Ini Layanan AWS membantu Anda terus mengaudit AWS penggunaan Anda untuk menyederhanakan cara Anda mengelola risiko dan kepatuhan terhadap peraturan dan standar industri.

Ketahanan dalam IoT Core AWS

Infrastruktur AWS global dibangun di sekitar Wilayah AWS s dan Availability Zone. Wilayah AWS s menyediakan beberapa Availability Zone yang terpisah secara fisik dan terisolasi, yang terhubung dengan latensi rendah, throughput tinggi, dan jaringan yang sangat redundan. Dengan Zona Ketersediaan, Anda dapat merancang serta mengoperasikan aplikasi dan basis data yang secara otomatis melakukan failover di antara Zona Ketersediaan tanpa gangguan. Zona Ketersediaan memiliki ketersediaan dan toleransi kesalahan yang lebih baik, dan dapat diskalakan dibandingkan infrastruktur biasa yang terdiri dari satu atau beberapa pusat data.

Untuk informasi selengkapnya tentang Wilayah AWS s dan Availability Zone, lihat [Infrastruktur AWS Global](#).

AWS IoT Core menyimpan informasi tentang perangkat Anda di registri perangkat. Ini juga menyimpan sertifikat CA, sertifikat perangkat, dan data bayangan perangkat. Jika terjadi kegagalan perangkat keras atau jaringan, data ini secara otomatis direplikasi di seluruh Availability Zone tetapi tidak di seluruh Wilayah.

AWS IoT Core menerbitkan peristiwa MQTT saat registri perangkat diperbarui. Anda dapat menggunakan pesan-pesan ini untuk mencadangkan data registri Anda dan menyimpannya di suatu tempat, seperti tabel DynamoDB. Anda bertanggung jawab untuk menyimpan sertifikat yang AWS IoT Core dibuat untuk Anda atau yang Anda buat sendiri. Device shadow menyimpan data status tentang perangkat Anda dan dapat di-resent saat perangkat kembali online. AWS IoT Device Advisor menyimpan informasi tentang konfigurasi rangkaian pengujian Anda. Data ini secara otomatis direplikasi jika terjadi kegagalan perangkat keras atau jaringan.

AWS IoT Core sumber daya bersifat spesifik Wilayah dan tidak direplikasi Wilayah AWS kecuali Anda secara khusus melakukannya.

Untuk informasi tentang praktik terbaik Keamanan, lihat [Praktik terbaik keamanan di AWS IoT Core](#).

Menggunakan AWS IoT Core dengan antarmuka VPC endpoint

[Dengan AWS IoT Core, Anda dapat membuat titik akhir data IoT dalam virtual private cloud \(VPC\) Anda dengan menggunakan titik akhir VPC antarmuka.](#) Endpoint VPC antarmuka didukung oleh AWS PrivateLink, sebuah AWS teknologi yang dapat Anda gunakan untuk mengakses layanan yang berjalan AWS dengan menggunakan alamat IP pribadi. Untuk informasi selengkapnya, lihat [Amazon Virtual Private Cloud](#).

Untuk menghubungkan perangkat di lapangan pada jaringan jarak jauh, seperti jaringan perusahaan ke VPC Amazon Anda, lihat opsi yang tercantum dalam matriks konektivitas VPC [jaringan-ke-Amazon](#).

Daftar Isi

- [Membuat titik akhir VPC untuk bidang data AWS IoT Core](#)
- [Membuat titik akhir VPC untuk penyedia kredensi AWS IoT Core](#)
- [Membuat titik akhir antarmuka VPC Amazon](#)
- [Mengkonfigurasi zona host pribadi](#)
- [Mengontrol Akses ke AWS IoT Core lebih dari titik akhir VPC](#)
- [Batasan](#)
- [Menskalakan titik akhir VPC dengan AWS IoT Core](#)
- [Menggunakan domain khusus dengan titik akhir VPC](#)
- [Ketersediaan titik akhir VPC untuk AWS IoT Core](#)

Membuat titik akhir VPC untuk bidang data AWS IoT Core

Anda dapat membuat titik akhir VPC untuk API bidang AWS IoT Core data untuk menghubungkan perangkat Anda ke AWS IoT layanan dan layanan lainnya. AWS Untuk memulai dengan titik akhir VPC, [buat antarmuka VPC endpoint](#) dan pilih sebagai layanan. AWS IoT Core AWS Jika Anda menggunakan CLI, pertama-tama panggil [describe-vpc-endpoint-services](#) untuk memastikan bahwa Anda memilih Availability Zone yang ada di area khusus Anda. AWS IoT Core Wilayah AWS Misalnya, di us-east-1, perintah ini akan terlihat seperti:

```
aws ec2 describe-vpc-endpoint-services --service-name com.amazonaws.us-east-1.iot.data
```

Note

Fitur VPC untuk membuat catatan DNS secara otomatis dinonaktifkan. Untuk terhubung ke titik akhir ini, Anda harus membuat catatan DNS Pribadi secara manual. Untuk informasi selengkapnya tentang catatan DNS VPC Pribadi, lihat DNS [pribadi untuk titik akhir](#) antarmuka. Untuk informasi selengkapnya tentang batasan AWS IoT Core VPC, lihat. [Batasan](#)

Untuk menghubungkan klien MQTT ke antarmuka titik akhir VPC:

- Anda harus membuat catatan DNS secara manual di zona host pribadi yang dilampirkan ke VPC Anda. Untuk memulai, lihat [Membuat zona host pribadi](#).
- Dalam zona host pribadi Anda, buat catatan alias untuk setiap IP antarmuka network elastis untuk titik akhir VPC. Jika Anda memiliki beberapa IP antarmuka jaringan untuk beberapa titik akhir VPC, buat catatan DNS tertimbang dengan bobot yang sama di semua catatan tertimbang. Alamat IP ini tersedia dari panggilan [DescribeNetworkInterfaces](#) API saat difilter oleh ID titik akhir VPC di bidang deskripsi.

Lihat petunjuk terperinci di bawah ini untuk [Membuat titik akhir antarmuka VPC Amazon](#) dan [Mengonfigurasi zona host pribadi](#) untuk AWS IoT Core bidang data.

Membuat titik akhir VPC untuk penyedia kredensi AWS IoT Core

[Anda dapat membuat titik akhir VPC untuk penyedia AWS IoT Core kredensi untuk menghubungkan perangkat menggunakan otentikasi berbasis sertifikat klien dan mendapatkan kredensyal sementara dalam format Signature Version 4. AWSAWS](#) Untuk memulai dengan titik akhir VPC untuk penyedia AWS IoT Core kredensi, jalankan perintah [CLI create-vpc-endpoint untuk membuat titik akhir VPC antarmuka dan pilih](#) penyedia kredensi sebagai layanan. AWS IoT Core AWS Untuk memastikan bahwa Anda memilih Availability Zone yang AWS IoT Core ada di area khusus Anda Wilayah AWS, pertama-tama Anda menjalankan [perintah describe-vpc-endpoint-services](#). Misalnya, di us-east-1, perintah ini akan terlihat seperti:

```
aws ec2 describe-vpc-endpoint-services --service-name com.amazonaws.us-east-1.iot.credentials
```

Note

Fitur VPC untuk membuat catatan DNS secara otomatis dinonaktifkan. Untuk terhubung ke titik akhir ini, Anda harus membuat catatan DNS Pribadi secara manual. Untuk informasi selengkapnya tentang catatan DNS VPC Pribadi, lihat DNS [pribadi untuk titik akhir](#) antarmuka. Untuk informasi selengkapnya tentang batasan AWS IoT Core VPC, lihat [Batasan](#)

Untuk menghubungkan klien HTTP ke antarmuka titik akhir VPC:

- Anda harus membuat catatan DNS secara manual di zona host pribadi yang dilampirkan ke VPC Anda. Untuk memulai, lihat [Membuat zona host pribadi](#).
- Dalam zona host pribadi Anda, buat catatan alias untuk setiap IP antarmuka network elastis untuk titik akhir VPC. Jika Anda memiliki beberapa IP antarmuka jaringan untuk beberapa titik akhir VPC, buat catatan DNS tertimbang dengan bobot yang sama di semua catatan tertimbang. Alamat IP ini tersedia dari panggilan [DescribeNetworkInterfaces](#) API saat difilter oleh ID titik akhir VPC di bidang deskripsi.

Lihat petunjuk terperinci di bawah ini untuk [Membuat titik akhir antarmuka VPC Amazon](#) dan [Mengonfigurasi zona host pribadi](#) untuk AWS IoT Core penyedia kredensi.

Membuat titik akhir antarmuka VPC Amazon

Anda dapat membuat titik akhir VPC antarmuka untuk terhubung ke AWS layanan yang didukung oleh AWS PrivateLink. Gunakan prosedur berikut untuk membuat titik akhir VPC antarmuka yang terhubung ke bidang AWS IoT Core data atau AWS IoT Core penyedia kredensi. Untuk informasi selengkapnya, lihat [Mengakses AWS layanan menggunakan titik akhir VPC antarmuka](#).

Note

Proses untuk membuat titik akhir antarmuka VPC Amazon untuk bidang AWS IoT Core data dan penyedia AWS IoT Core kredensi serupa, tetapi Anda harus membuat perubahan spesifik titik akhir untuk membuat koneksi berfungsi.

[Untuk membuat antarmuka VPC endpoint menggunakan konsol VPC Endpoints](#)

1. Arahkan ke konsol [VPC](#) Endpoints, di bawah Virtual private cloud di menu sebelah kiri, pilih Endpoints lalu Create Endpoint.
2. Di halaman Buat titik akhir, tentukan informasi berikut.
 - Pilih Layanan AWS s untuk kategori Layanan.
 - Untuk Nama Layanan, cari dengan memasukkan kata kunci `iot`. Dalam daftar `iot` layanan yang ditampilkan, pilih titik akhir.

Jika Anda membuat titik akhir VPC untuk bidang AWS IoT Core data, pilih titik akhir API bidang AWS IoT Core data untuk Wilayah Anda. Titik akhir akan menjadi `formatcom.amazonaws.region.iot.data`.

Jika Anda membuat titik akhir VPC untuk penyedia AWS IoT Core kredensi, pilih titik akhir penyedia AWS IoT Core kredensyal untuk Wilayah Anda. Titik akhir akan menjadi `formatcom.amazonaws.region.iot.credentials`.

Note

Nama layanan untuk pesawat AWS IoT Core data di Wilayah China akan menjadi `formatcn.com.amazonaws.region.iot.data`. Membuat titik akhir VPC untuk penyedia AWS IoT Core kredensi tidak didukung di Wilayah China.

- Untuk VPC dan Subnet, pilih VPC tempat Anda ingin membuat titik akhir, dan Availability Zones (AZ) tempat Anda ingin membuat jaringan endpoint.
 - Untuk Aktifkan nama DNS, pastikan Aktifkan untuk titik akhir ini tidak dipilih. Baik pesawat AWS IoT Core data maupun penyedia AWS IoT Core kredensi belum mendukung nama DNS pribadi.
 - Untuk grup Keamanan, pilih grup keamanan yang ingin Anda kaitkan dengan antarmuka jaringan titik akhir.
 - Secara opsional, Anda dapat menambah atau menghapus tag. Tag adalah pasangan nama-nilai yang Anda gunakan untuk mengasosiasikan dengan titik akhir Anda.
3. Untuk membuat titik akhir VPC Anda, pilih Buat titik akhir.

Setelah Anda membuat AWS PrivateLink titik akhir, di tab Detail titik akhir Anda, Anda akan melihat daftar nama DNS. Anda dapat menggunakan salah satu nama DNS yang Anda buat di bagian ini untuk [mengonfigurasi zona host pribadi Anda](#).

Mengonfigurasi zona host pribadi

Anda dapat menggunakan salah satu nama DNS yang Anda buat di bagian sebelumnya untuk mengonfigurasi zona host pribadi Anda.

Untuk pesawat AWS IoT Core data

Nama DNS harus berupa nama konfigurasi domain atau titik IoT:Data-ATS akhir Anda. Contoh nama DNS dapat berupa: `xxx-ats.data.iot.region.amazonaws.com`.

Untuk penyedia AWS IoT Core kredensi

Nama DNS harus menjadi titik `iot:CredentialProvider` akhir Anda. Contoh nama DNS dapat berupa: `xxxx.credentials.iot.region.amazonaws.com`.

Note

Proses untuk mengonfigurasi zona host pribadi untuk bidang AWS IoT Core data dan penyedia AWS IoT Core kredensi serupa, tetapi Anda harus membuat perubahan spesifik titik akhir untuk membuat koneksi berfungsi.

Buat zona yang dihosting pribadi

Untuk membuat zona host pribadi menggunakan konsol Route 53

1. Arahkan ke konsol [Route 53](#) Zona yang dihosting dan pilih Buat zona yang dihosting.
2. Di halaman Buat zona yang dihosting, tentukan informasi berikut.
 - Untuk nama Domain, masukkan alamat titik akhir untuk titik akhir `iot:Data-ATS` atau titik `iot:CredentialProvider` akhir Anda. Perintah AWS CLI berikut menunjukkan cara mendapatkan titik akhir melalui jaringan publik: `aws iot describe-endpoint --endpoint-type iot:Data-ATS`, atau `aws iot describe-endpoint --endpoint-type iot:CredentialProvider`

Note

Jika Anda menggunakan domain kustom, lihat [Menggunakan domain kustom dengan titik akhir VPC](#). Domain kustom tidak didukung untuk penyedia AWS IoT Core kredensi.

- Untuk Jenis, pilih Zona host pribadi.
 - Secara opsional, Anda dapat menambahkan atau menghapus tag untuk dikaitkan dengan zona yang dihosting.
3. Untuk membuat zona host pribadi Anda, pilih Buat zona yang dihosting.

Untuk informasi selengkapnya, lihat [Membuat zona yang dihosting pribadi](#).

Buat catatan

Setelah Anda membuat zona host pribadi, Anda dapat membuat catatan yang memberi tahu DNS bagaimana Anda ingin lalu lintas dialihkan ke domain tersebut.

Untuk membuat catatan

1. Dalam daftar zona yang dihosting yang ditampilkan, pilih zona host pribadi yang Anda buat sebelumnya dan pilih Buat catatan.
2. Gunakan metode wizard untuk membuat catatan. Jika konsol menyajikan metode Quick create, pilih Beralih ke wizard.
3. Pilih kebijakan Perutean Sederhana untuk Perutean dan kemudian pilih Berikutnya.
4. Di halaman Konfigurasi catatan, pilih Tentukan catatan sederhana.
5. Di halaman Tentukan catatan sederhana:
 - Untuk nama Rekam, masukkan `iot:Data-ATS` titik akhir atau titik `iot:CredentialProvider` akhir. Ini harus sama dengan nama zona host pribadi.
 - Untuk jenis Rekam, pertahankan nilainya sebagai `A - Routes traffic to an IPv4 address and some AWS resources`.
 - Untuk lalu lintas Nilai/Rute ke, pilih Alias ke titik akhir VPC. Kemudian pilih Wilayah Anda dan kemudian pilih titik akhir yang Anda buat sebelumnya, seperti yang dijelaskan [???](#) dari daftar titik akhir yang ditampilkan.
6. Pilih Tentukan catatan sederhana untuk membuat catatan Anda.

Mengontrol Akses ke AWS IoT Core lebih dari titik akhir VPC

[Anda dapat membatasi akses perangkat AWS IoT Core agar diizinkan hanya melalui titik akhir VPC dengan menggunakan tombol konteks kondisi VPC.](#) AWS IoT Core mendukung kunci konteks terkait VPC berikut:

- [SourceVpc](#)
- [SourceVpce](#)
- [VPC SourceIp](#)

Note

AWS IoT Core tidak mendukung [kebijakan Titik Akhir untuk titik akhir VPC](#).

Misalnya, kebijakan berikut memberikan izin untuk terhubung AWS IoT Core menggunakan ID klien yang cocok dengan nama benda, dan memublikasikan ke topik apa pun yang diawali dengan nama benda, bergantung pada perangkat yang terhubung ke titik akhir VPC dengan ID Titik Akhir VPC tertentu. Kebijakan ini akan menolak upaya koneksi ke titik akhir data IoT publik Anda.

```
{
  "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}/*"
      ]
    }
  ]
}
```

Batasan

[Titik akhir VPC saat ini hanya didukung untuk titik akhir AWS IoT Core data dan AWS IoT Core titik akhir penyedia kredensial.](#)

Keterbatasan titik akhir VPC data IoT

Bagian ini mencakup batasan titik akhir VPC data IoT.

- MQTT menjaga periode hidup dibatasi hingga 230 detik. Menjaga periode hidup lebih lama dari itu akan secara otomatis dikurangi menjadi 230 detik.
- Setiap titik akhir VPC mendukung 100.000 total perangkat yang terhubung secara bersamaan. Jika Anda membutuhkan lebih banyak koneksi, lihat [Menskalakan titik akhir VPC dengan AWS IoT Core](#).
- Titik akhir VPC hanya mendukung lalu lintas IPv4.
- Titik akhir VPC hanya akan melayani [sertifikat ATS](#), kecuali untuk domain khusus.
- [Kebijakan titik akhir VPC](#) tidak didukung.
- Untuk titik akhir VPC yang dibuat untuk bidang AWS IoT Core data, AWS IoT Core tidak mendukung penggunaan catatan DNS publik zona atau regional.

Batasan titik akhir penyedia kredensial

Bagian ini mencakup batasan titik akhir VPC penyedia kredensi.

- Titik akhir VPC hanya mendukung lalu lintas IPv4.
- Titik akhir VPC hanya akan melayani sertifikat [ATS](#).
- [Kebijakan titik akhir VPC](#) tidak didukung.
- Domain kustom tidak didukung untuk titik akhir penyedia kredensial.
- Untuk titik akhir VPC yang dibuat untuk penyedia AWS IoT Core kredensi, AWS IoT Core tidak mendukung penggunaan catatan DNS publik zona atau regional.

Menskalakan titik akhir VPC dengan AWS IoT Core

AWS IoT Core Endpoint VPC antarmuka dibatasi hingga 100.000 perangkat yang terhubung melalui satu titik akhir antarmuka. Jika kasus penggunaan Anda memerlukan lebih banyak koneksi

bersamaan ke broker, sebaiknya gunakan beberapa titik akhir VPC dan merutekan perangkat Anda secara manual di seluruh titik akhir antarmuka Anda. Saat membuat catatan DNS pribadi untuk merutekan lalu lintas ke titik akhir VPC Anda, pastikan untuk membuat catatan tertimbang sebanyak Anda memiliki titik akhir VPC untuk mendistribusikan lalu lintas di beberapa titik akhir Anda.

Menggunakan domain khusus dengan titik akhir VPC

Jika Anda ingin menggunakan domain kustom dengan titik akhir VPC, Anda harus membuat catatan nama domain kustom Anda di zona yang dihosting pribadi dan membuat catatan perutean di Route53. Untuk informasi selengkapnya, lihat [Membuat zona yang dihosting pribadi](#).

Note

Domain khusus hanya didukung untuk titik akhir AWS IoT Core data.

Ketersediaan titik akhir VPC untuk AWS IoT Core

AWS IoT Core [Titik akhir VPC antarmuka tersedia di semua AWS IoT Core wilayah yang didukung](#). AWS IoT Core Titik akhir VPC antarmuka untuk penyedia AWS IoT Core kredensi tidak didukung di Wilayah China dan. AWS GovCloud (US) Regions

Keamanan infrastruktur di AWS IoT

Sebagai kumpulan layanan terkelola, AWS IoT dilindungi oleh prosedur keamanan jaringan AWS global yang dijelaskan dalam whitepaper [Amazon Web Services: Tinjauan Proses Keamanan](#).

Anda menggunakan panggilan API yang AWS dipublikasikan untuk mengakses AWS IoT melalui jaringan. Klien harus mendukung Keamanan Lapisan Pengangkutan (TLS) 1.2 atau versi yang lebih baru. Klien juga harus support suite cipher dengan Perfect Forward Secrecy (PFS) seperti Ephemeral Diffie-Hellman (DHE) atau Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). Sebagian besar sistem modern, misalnya Java 7 dan versi yang lebih baru, mendukung mode ini. Untuk informasi selengkapnya, lihat [Keamanan transportasi di AWS IoT Core](#).

Permintaan harus ditandatangani menggunakan access key ID dan secret access key yang terkait dengan prinsipal IAM. Atau Anda dapat menggunakan [AWS Security Token Service](#) (AWS STS) untuk membuat kredensial keamanan sementara untuk menandatangani permintaan.

Pemantauan keamanan armada produksi atau perangkat dengan Core AWS IoT

Armada IoT dapat terdiri dari sejumlah besar perangkat yang memiliki kemampuan beragam, berumur panjang, dan didistribusikan secara geografis. Karakteristik ini membuat pengaturan armada rumit dan rentan terhadap kesalahan. Dan karena perangkat sering dibatasi dalam daya komputasi, memori, dan kemampuan penyimpanan, ini membatasi penggunaan enkripsi dan bentuk keamanan lainnya pada perangkat itu sendiri. Selain itu, perangkat sering menggunakan perangkat lunak dengan kerentanan yang diketahui. Faktor-faktor ini membuat armada IoT menjadi target yang menarik bagi peretas dan menyulitkan untuk mengamankan armada perangkat Anda secara berkelanjutan.

AWS IoT Device Defender mengatasi tantangan ini dengan menyediakan alat untuk mengidentifikasi masalah keamanan dan penyimpangan dari praktik terbaik. Anda dapat menggunakan AWS IoT Device Defender untuk menganalisis, mengaudit, dan memantau perangkat yang terhubung untuk mendeteksi perilaku abnormal, dan mengurangi risiko keamanan. AWS IoT Device Defender dapat mengaudit armada perangkat untuk memastikan mereka mematuhi praktik terbaik keamanan dan mendeteksi perilaku abnormal pada perangkat. Hal ini memungkinkan untuk menerapkan kebijakan keamanan yang konsisten di seluruh armada AWS IoT perangkat Anda dan merespons dengan cepat ketika perangkat dikompromikan. Untuk informasi selengkapnya, lihat [AWS IoT Device Defender](#).

AWS IoT Device Advisor mendorong pembaruan dan menambal armada Anda sesuai kebutuhan. AWS IoT Device Advisor memperbarui kasus uji secara otomatis. Kasus uji yang Anda pilih selalu dengan versi terbaru. Untuk informasi selengkapnya, lihat [Penasihat Perangkat](#).

Praktik terbaik keamanan di AWS IoT Core

Bagian ini berisi informasi tentang praktik terbaik keamanan untuk AWS IoT Core. Untuk informasi tentang aturan keamanan untuk solusi IoT Industri, lihat [Sepuluh aturan emas keamanan untuk solusi IoT Industri](#).

Melindungi koneksi MQTT di AWS IoT

[AWS IoT Core](#) adalah layanan cloud terkelola yang memungkinkan perangkat yang terhubung berinteraksi dengan aplikasi cloud dan perangkat lain dengan mudah dan aman. AWS IoT Core mendukung HTTP, [WebSocket](#), dan [MQTT](#), protokol komunikasi ringan yang dirancang khusus untuk

mentolerir koneksi intermiten. Jika Anda terhubung AWS IoT menggunakan MQTT, setiap koneksi Anda harus dikaitkan dengan pengenal yang dikenal sebagai ID klien. ID klien MQTT secara unik mengidentifikasi koneksi MQTT. Jika koneksi baru dibuat menggunakan ID klien yang sudah diklaim untuk koneksi lain, broker AWS IoT pesan menjatuhkan koneksi lama untuk memungkinkan koneksi baru. ID klien harus unik di masing-masing Akun AWS dan masing-masing Wilayah AWS. Ini berarti bahwa Anda tidak perlu menegakkan keunikan global ID klien di luar Anda Akun AWS atau di seluruh Wilayah di dalam Anda. Akun AWS

Dampak dan tingkat keparahan terputusnya koneksi MQTT pada armada perangkat Anda bergantung pada banyak faktor. Ini termasuk:

- Kasus penggunaan Anda (misalnya, data yang dikirim perangkat Anda AWS IoT, berapa banyak data, dan frekuensi pengiriman data).
- [Konfigurasi klien MQTT Anda \(misalnya, pengaturan penyambungan kembali otomatis, pengaturan waktu mundur terkait, dan penggunaan sesi persisten MQTT\)](#).
- Kendala sumber daya perangkat.
- Akar penyebab pemutusan hubungan, agresivitas, dan ketekunannya.

Untuk menghindari konflik ID klien dan potensi dampak negatifnya, pastikan bahwa setiap perangkat atau aplikasi seluler memiliki kebijakan AWS IoT atau IAM yang membatasi ID klien mana yang dapat digunakan untuk koneksi MQTT ke broker pesan. AWS IoT Misalnya, Anda dapat menggunakan kebijakan IAM untuk mencegah perangkat menutup sambungan perangkat lain secara tidak sengaja dengan menggunakan ID klien yang sudah digunakan. Untuk informasi selengkapnya, lihat [Otorisasi](#).

Semua perangkat di armada Anda harus memiliki kredensial dengan hak istimewa yang hanya mengizinkan tindakan yang dimaksudkan, yang mencakup (namun tidak terbatas pada) tindakan AWS IoT MQTT seperti memublikasikan pesan atau berlangganan topik dengan cakupan dan konteks tertentu. Kebijakan izin khusus dapat bervariasi untuk kasus penggunaan Anda. Identifikasi kebijakan izin yang paling memenuhi persyaratan bisnis dan keamanan Anda.

Untuk menyederhanakan pembuatan dan pengelolaan kebijakan izin, Anda dapat menggunakan [AWS IoT Core variabel kebijakan](#) dan variabel [kebijakan IAM](#). Variabel kebijakan dapat ditempatkan dalam kebijakan dan ketika kebijakan dievaluasi, variabel diganti dengan nilai yang berasal dari permintaan perangkat. Dengan menggunakan variabel kebijakan, Anda dapat membuat satu kebijakan untuk memberikan izin ke beberapa perangkat. Anda dapat mengidentifikasi variabel kebijakan yang relevan untuk kasus penggunaan Anda berdasarkan konfigurasi AWS IoT akun Anda, mekanisme otentikasi, dan protokol jaringan yang digunakan dalam menghubungkan ke broker

AWS IoT pesan. Namun, untuk menulis kebijakan izin terbaik, pertimbangkan secara spesifik kasus penggunaan dan [model ancaman](#) Anda.

Misalnya, jika Anda mendaftarkan perangkat di AWS IoT registri, Anda dapat menggunakan [variabel kebijakan hal](#) dalam AWS IoT kebijakan untuk memberikan atau menolak izin berdasarkan properti benda seperti nama benda, tipe benda, dan nilai atribut benda. Nama benda diperoleh dari ID klien dalam pesan koneksi MQTT yang dikirim ketika sesuatu terhubung ke AWS IoT. [Variabel kebijakan hal diganti ketika sesuatu terhubung ke AWS IoT lebih dari MQTT menggunakan otentikasi timbal balik TLS atau MQTT melalui protokol menggunakan identitas Amazon Cognito yang diautentikasi. WebSocket](#) Anda dapat menggunakan [AttachThingPrincipal](#) API untuk melampirkan sertifikat dan identitas Amazon Cognito yang diautentikasi ke suatu hal. `iot:Connection.Thing.ThingName` adalah variabel kebijakan hal yang berguna untuk menegakkan pembatasan ID klien. Contoh AWS IoT kebijakan berikut memerlukan nama benda terdaftar untuk digunakan sebagai ID klien untuk koneksi MQTT ke broker pesan: AWS IoT

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

Jika Anda ingin mengidentifikasi konflik ID klien yang sedang berlangsung, Anda dapat mengaktifkan dan menggunakan [CloudWatch Log untuk AWS IoT](#). Untuk setiap koneksi MQTT yang terputus oleh broker AWS IoT pesan karena konflik ID klien, catatan log yang serupa dengan berikut ini dihasilkan:

```
{
  "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"
}
```

Anda dapat menggunakan [filter CloudWatch Log](#) seperti `{$.reason="DUPLICATE_CLIENT_ID" }` untuk mencari contoh konflik ID klien atau untuk mengatur [filter CloudWatch metrik](#) dan CloudWatch alarm terkait untuk pemantauan dan pelaporan berkelanjutan.

Anda dapat menggunakan [AWS IoT Device Defender](#) untuk mengidentifikasi kebijakan yang terlalu permisif AWS IoT dan IAM. AWS IoT Device Defender juga menyediakan pemeriksaan audit yang memberi tahu Anda jika beberapa perangkat dalam armada Anda terhubung ke broker AWS IoT pesan menggunakan ID klien yang sama.

Anda dapat menggunakan AWS IoT Device Advisor untuk memvalidasi bahwa perangkat Anda dapat terhubung dengan andal AWS IoT Core dan mengikuti praktik terbaik keamanan.

Lihat juga

- [AWS IoT Core](#)
- [AWS IoT Fitur Keamanan](#)
- [AWS IoT Core variabel kebijakan](#)
- [Variabel Kebijakan IAM](#)
- [Identitas Amazon Cognito](#)
- [AWS IoT Pembela Perangkat](#)
- [CloudWatch Log untuk AWS IoT](#)

Jaga agar jam perangkat Anda tetap sinkron

Penting untuk memiliki waktu yang akurat di perangkat Anda. Sertifikat X.509 memiliki tanggal dan waktu kedaluwarsa. Jam di perangkat Anda digunakan untuk memverifikasi bahwa sertifikat server masih valid. Jika Anda sedang membangun perangkat IoT komersial, ingatlah bahwa produk Anda mungkin disimpan untuk waktu yang lama sebelum dijual. Jam waktu nyata dapat melayang selama waktu ini dan baterai dapat habis, sehingga pengaturan waktu di pabrik tidak cukup.

Untuk sebagian besar sistem, ini berarti bahwa perangkat lunak perangkat harus menyertakan klien protokol waktu jaringan (NTP). Perangkat harus menunggu hingga disinkronkan dengan server NTP sebelum mencoba terhubung. AWS IoT Core Jika ini tidak memungkinkan, sistem harus menyediakan cara bagi pengguna untuk mengatur waktu perangkat sehingga koneksi berikutnya berhasil.

Setelah perangkat disinkronkan dengan server NTP, perangkat dapat membuka koneksi dengan AWS IoT Core. Berapa banyak kemiringan jam yang diizinkan tergantung pada apa yang Anda coba lakukan dengan koneksi.

Validasi sertifikat server

Hal pertama yang dilakukan perangkat untuk berinteraksi AWS IoT adalah membuka koneksi yang aman. Saat Anda menghubungkan perangkat Anda AWS IoT, pastikan bahwa Anda sedang berbicara AWS IoT dan bukan server lain yang meniru identitas AWS IoT. Setiap AWS IoT server disediakan dengan sertifikat yang dikeluarkan untuk domain `iot.amazonaws.com`. Sertifikat ini dikeluarkan AWS IoT oleh otoritas sertifikat tepercaya yang memverifikasi identitas dan kepemilikan domain kami.

Salah satu hal pertama yang AWS IoT Core dilakukan ketika perangkat terhubung adalah mengirim perangkat sertifikat server. Perangkat dapat memverifikasi bahwa mereka mengharapkan untuk terhubung `iot.amazonaws.com` dan bahwa server pada akhir koneksi tersebut memiliki sertifikat dari otoritas tepercaya untuk domain itu.

Sertifikat TLS dalam format X.509 dan mencakup berbagai informasi seperti nama organisasi, lokasi, nama domain, dan masa berlaku. Periode validitas ditentukan sebagai sepasang nilai waktu yang disebut `notBefore` dan `notAfter`. Layanan seperti AWS IoT Core menggunakan periode validitas terbatas (misalnya, satu tahun) untuk sertifikat server mereka dan mulai melayani yang baru sebelum yang lama kedaluwarsa.

Gunakan satu identitas per perangkat

Gunakan satu identitas per klien. Perangkat umumnya menggunakan sertifikat klien X.509. Aplikasi web dan seluler menggunakan Identitas Amazon Cognito. Ini memungkinkan Anda menerapkan izin berbutir halus ke perangkat Anda.

Misalnya, Anda memiliki aplikasi yang terdiri dari perangkat ponsel yang menerima pembaruan status dari dua objek rumah pintar yang berbeda - bola lampu dan termostat. Bola lampu mengirimkan status tingkat baterainya, dan termostat mengirimkan pesan yang melaporkan suhu.

AWS IoT mengautentikasi perangkat secara individual dan memperlakukan setiap koneksi secara individual. Anda dapat menerapkan kontrol akses berbutir halus menggunakan kebijakan otorisasi. Anda dapat menentukan kebijakan untuk termostat yang memungkinkannya mempublikasikan ke ruang topik. Anda dapat menentukan kebijakan terpisah untuk bola lampu yang memungkinkannya mempublikasikan ke ruang topik yang berbeda. Terakhir, Anda dapat menentukan kebijakan untuk aplikasi seluler yang hanya memungkinkannya untuk terhubung dan berlangganan topik termostat dan bola lampu untuk menerima pesan dari perangkat ini.

Terapkan prinsip hak istimewa dan cakupan paling sedikit ke bawah izin per perangkat sebanyak mungkin. Semua perangkat atau pengguna harus memiliki AWS IoT kebijakan AWS IoT yang hanya memungkinkannya terhubung dengan ID klien yang dikenal, dan untuk mempublikasikan dan berlangganan serangkaian topik yang diidentifikasi dan tetap.

Gunakan detik Wilayah AWS sebagai cadangan

Pertimbangkan untuk menyimpan salinan data Anda dalam sedetik Wilayah AWS sebagai cadangan. Perhatikan bahwa AWS solusi bernama [Disaster Recovery untuk AWS IoT](#) tidak lagi tersedia. Meskipun [GitHubperpustakaan](#) terkait tetap dapat diakses, AWS tidak digunakan lagi pada Juli 2023 dan tidak lagi menyediakan pemeliharaan atau dukungan untuknya. Untuk menerapkan solusi Anda sendiri atau untuk menjelajahi opsi dukungan tambahan, kunjungi [Kontak AWS](#). Jika ada Manajer Akun AWS Teknis yang terkait dengan akun Anda, hubungi mereka untuk meminta bantuan.

Gunakan hanya dalam penyediaan waktu

Membuat dan menyediakan setiap perangkat secara manual dapat memakan waktu. AWS IoT menyediakan cara untuk mendefinisikan template ke perangkat penyediaan saat mereka pertama kali terhubung AWS IoT. Untuk informasi selengkapnya, lihat [ust-in-time Penyediaan J](#).

Izin untuk menjalankan pengujian AWS IoT Device Advisor

Templat kebijakan berikut menunjukkan izin minimum dan entitas IAM yang diperlukan untuk menjalankan kasus pengujian AWS IoT Device Advisor. [Anda harus mengganti *perangkat-role-arn* Anda dengan peran perangkat Amazon Resource Name \(ARN\) yang Anda buat di bawah prasyarat.](#)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
```

```

    "Effect": "Allow",
    "Action": "iam:PassRole",
    "Resource": "your-device-role-arn",
    "Condition": {
      "StringEquals": {
        "iam:PassedToService": "iotdeviceadvisor.amazonaws.com"
      }
    }
  },
  {
    "Sid": "VisualEditor1",
    "Effect": "Allow",
    "Action": [
      "execute-api:Invoke*",
      "iam:ListRoles", // Required to list device roles in the Device
Advisor console
      "iot:Connect",
      "iot:CreateJob",
      "iot>DeleteJob",
      "iot:DescribeCertificate",
      "iot:DescribeEndpoint",
      "iot:DescribeJobExecution",
      "iot:DescribeJob",
      "iot:DescribeThing",
      "iot:GetPendingJobExecutions",
      "iot:GetPolicy",
      "iot:ListAttachedPolicies",
      "iot:ListCertificates",
      "iot:ListPrincipalPolicies",
      "iot:ListThingPrincipals",
      "iot:ListThings",
      "iot:Publish",
      "iot:StartNextPendingJobExecution",
      "iot:UpdateJobExecution",
      "iot:UpdateThingShadow",
      "logs:CreateLogGroup",
      "logs:CreateLogStream",
      "logs:DescribeLogGroups",
      "logs:DescribeLogStreams",
      "logs:PutLogEvents",
      "logs:PutRetentionPolicy"
    ],
    "Resource": "*"
  },

```

```
{
  "Sid": "VisualEditor2",
  "Effect": "Allow",
  "Action": "iotdeviceadvisor:*",
  "Resource": "*"
}
```

Pencegahan deputy kebingungan lintas layanan untuk Device Advisor

Masalah deputy yang bingung adalah masalah keamanan di mana entitas yang tidak memiliki izin untuk melakukan tindakan dapat memaksa entitas yang lebih istimewa untuk melakukan tindakan. Pada tahun AWS, peniruan lintas layanan dapat mengakibatkan masalah wakil yang membingungkan. Peniruan identitas lintas layanan dapat terjadi ketika satu layanan (layanan yang dipanggil) memanggil layanan lain (layanan yang dipanggil). Layanan pemanggilan dapat dimanipulasi menggunakan izinnya untuk bertindak pada sumber daya pelanggan lain dengan cara yang seharusnya tidak dilakukannya kecuali bila memiliki izin untuk mengakses. Untuk mencegah hal ini, AWS sediakan alat yang membantu Anda melindungi data Anda untuk semua layanan dengan prinsip layanan yang telah diberikan akses ke sumber daya di akun Anda.

Sebaiknya gunakan kunci konteks kondisi [aws:SourceAccount](#) global [aws:SourceArn](#) dan global dalam kebijakan sumber daya untuk membatasi izin yang diberikan Device Advisor kepada layanan lain ke sumber daya. Jika Anda menggunakan kedua kunci konteks kondisi global, `aws:SourceAccount` nilai dan akun dalam `aws:SourceArn` nilai harus menggunakan ID akun yang sama saat digunakan dalam pernyataan kebijakan yang sama.

Nilai `aws:SourceArn` harus ARN dari sumber definisi suite Anda. Sumber definisi suite mengacu pada rangkaian pengujian yang Anda buat dengan Device Advisor.

Cara paling efektif untuk melindungi dari masalah confused deputy adalah dengan menggunakan kunci konteks kondisi global `aws:SourceArn` dengan ARN lengkap sumber daya. Jika Anda tidak mengetahui ARN lengkap sumber daya atau jika Anda menentukan beberapa sumber daya, gunakan kunci kondisi konteks `aws:SourceArn` global dengan wildcard (*) untuk bagian ARN yang tidak diketahui. Misalnya, `arn:aws:iotdeviceadvisor:*:account-id:suitedefinition/*`

Contoh berikut menunjukkan bagaimana Anda dapat menggunakan `aws:SourceArn` dan kunci konteks kondisi `aws:SourceAccount` global di Device Advisor untuk mencegah masalah wakil yang membingungkan.


```
{
  "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 pelatihan dan sertifikasi

Ikuti kursus berikut untuk mempelajari konsep-konsep kunci untuk AWS IoT keamanan: [AWS IoT Security Primer](#).

Pemantauan AWS IoT

Pemantauan adalah bagian penting dari menjaga keandalan, ketersediaan, dan kinerja AWS IoT dan AWS solusi Anda.

Kami sangat menyarankan Anda untuk mengumpulkan data pemantauan dari semua bagian AWS solusi Anda untuk mempermudah debug kegagalan multi-titik, jika terjadi. Mulailah dengan membuat rencana pemantauan yang menjawab pertanyaan-pertanyaan berikut. Jika Anda tidak yakin bagaimana menjawabnya, Anda masih dapat terus [mengaktifkan logging](#) dan menetapkan baseline kinerja Anda.

- Apa tujuan pemantauan Anda?
- Sumber daya manakah yang akan Anda pantau?
- Seberapa seringkah Anda akan memantau sumber daya ini?
- Apa sajakah alat pemantauan yang akan Anda gunakan?
- Siapa yang akan melakukan tugas pemantauan?
- Siapa yang harus diberi tahu saat terjadi kesalahan?

Langkah Anda selanjutnya adalah [mengaktifkan pencatatan](#) dan menetapkan dasar AWS IoT kinerja normal di lingkungan Anda dengan mengukur kinerja pada berbagai waktu dan dalam kondisi beban yang berbeda. Saat Anda memantau AWS IoT, simpan data pemantauan historis sehingga Anda dapat membandingkannya dengan data kinerja saat ini. Ini akan membantu Anda mengidentifikasi pola kinerja normal dan anomali kinerja, dan merancang metode untuk mengatasi masalah.

Untuk menetapkan kinerja dasar Anda AWS IoT, Anda harus memantau metrik ini untuk memulai. Anda selalu dapat memantau lebih banyak metrik nanti.

- [PublishIn.Success](#)
- [PublishOut.Success](#)
- [Subscribe.Success](#)
- [Ping.Success](#)
- [Connect.Success](#)
- [GetThingShadow.Accepted](#)
- [UpdateThingShadow.Accepted](#)

- [DeleteThingShadow.Accepted](#)
- [RulesExecuted](#)

Topik di bagian ini dapat membantu Anda memulai pencatatan dan pemantauan AWS IoT.

Topik

- [Konfigurasi AWS IoT logging](#)
- [Pantau AWS IoT alarm dan metrik menggunakan Amazon CloudWatch](#)
- [Monitor AWS IoT menggunakan CloudWatch Log](#)
- [Unggah log sisi perangkat ke Amazon CloudWatch](#)
- [Logging panggilan AWS IoT API menggunakan AWS CloudTrail](#)

Konfigurasi AWS IoT logging

Anda harus mengaktifkan logging dengan menggunakan AWS IoT konsol, CLI, atau API sebelum Anda dapat memantau dan mencatat AWS IoT aktivitas.

Anda dapat mengaktifkan pencatatan untuk semua AWS IoT atau hanya grup hal tertentu. Anda dapat mengonfigurasi AWS IoT logging dengan menggunakan AWS IoT konsol, CLI, atau API; namun, Anda harus menggunakan CLI atau API untuk mengonfigurasi logging untuk grup hal tertentu.

Saat mempertimbangkan cara mengonfigurasi AWS IoT logging Anda, konfigurasi logging default menentukan bagaimana AWS IoT aktivitas akan dicatat kecuali ditentukan lain. Mulai, Anda mungkin ingin mendapatkan log terperinci dengan [tingkat log](#) default INFO atau DEBUG. Setelah meninjau log awal, Anda dapat mengubah level log default ke level yang kurang bertele-tele seperti WARN atau ERROR dan menetapkan tingkat log spesifik sumber daya yang lebih bertele-tele pada sumber daya yang mungkin memerlukan perhatian lebih. Level log dapat diubah kapan pun Anda mau.

Topik ini mencakup login sisi cloud. AWS IoT Untuk informasi tentang pencatatan dan pemantauan sisi perangkat, lihat [Mengunggah log sisi perangkat](#) ke CloudWatch

Untuk informasi tentang pencatatan dan pemantauan AWS IoT Greengrass, lihat [Logging dan monitoring in AWS IoT Greengrass](#). Pada 30 Juni 2023, perangkat lunak AWS IoT Greengrass Core telah bermigrasi ke AWS IoT Greengrass Version 2

Konfigurasi peran dan kebijakan pencatatan

Sebelum mengaktifkan login AWS IoT, Anda harus membuat peran IAM dan kebijakan yang memberikan AWS izin untuk memantau AWS IoT aktivitas atas nama Anda. Anda juga dapat membuat peran IAM dengan kebijakan yang diperlukan di [bagian Log AWS IoT konsol](#).

Note

Sebelum mengaktifkan AWS IoT logging, pastikan Anda memahami izin akses CloudWatch Log. Pengguna dengan akses ke CloudWatch Log dapat melihat informasi debugging dari perangkat Anda. Untuk informasi selengkapnya, lihat [Otentikasi dan Kontrol Akses untuk CloudWatch Log Amazon](#).

Jika Anda mengharapkan pola lalu lintas tinggi AWS IoT Core karena pengujian beban, pertimbangkan untuk mematikan pencatatan IoT untuk mencegah pelambatan. Jika lalu lintas tinggi terdeteksi, layanan kami dapat menonaktifkan login di akun Anda.

Berikut ini menunjukkan cara membuat peran dan kebijakan logging untuk AWS IoT Core sumber daya.

Buat peran logging

Untuk membuat peran logging, buka [hub Peran konsol IAM](#) dan pilih Buat peran.

1. Di bawah Pilih entitas tepercaya, pilih AWS Layanan. Kemudian pilih IoT di bawah Kasus penggunaan. Jika Anda tidak melihat IoT, masukkan dan cari IoT dari menu tarik-turun Kasus penggunaan untuk AWS layanan lain. Pilih Selanjutnya.
2. Pada halaman Tambahkan izin, Anda akan melihat kebijakan yang secara otomatis dilampirkan ke peran layanan. Pilih Selanjutnya.
3. Pada halaman Nama, tinjau, dan buat, masukkan nama Peran dan Deskripsi peran untuk peran tersebut, lalu pilih Buat peran.
4. Dalam daftar Peran, temukan peran yang Anda buat, buka, dan salin ARN Peran (**logging-role-arn**) untuk digunakan saat Anda. [Konfigurasi logging default di AWS IoT \(konsol\)](#)

Kebijakan peran pencatatan

Dokumen kebijakan berikut menyediakan kebijakan peran dan kebijakan kepercayaan yang memungkinkan AWS IoT untuk mengirimkan entri log atas nama Anda. CloudWatch Jika Anda juga

mengizinkan LoRa WAN AWS IoT Core untuk mengirimkan entri log, Anda akan melihat dokumen kebijakan yang dibuat untuk Anda yang mencatat kedua aktivitas tersebut.

Note

Dokumen-dokumen ini dibuat untuk Anda saat Anda membuat peran logging. Dokumen memiliki variabel, `${partition}${region}`, dan `${accountId}`, yang harus Anda ganti dengan nilai-nilai Anda.

Kebijakan peran:

```
{
  "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:${partition}:logs:${region}:${accountId}:log-group:AWSIoTLogsV2:*"
      ]
    }
  ]
}
```

Kebijakan kepercayaan untuk mencatat AWS IoT Core aktivitas saja:

```
{
```

```

"Version": "2012-10-17",
"Statement": [
  {
    "Sid": "",
    "Effect": "Allow",
    "Principal": {
      "Service": "iot.amazonaws.com"
    },
    "Action": "sts:AssumeRole"
  }
]
}

```

Konfigurasi logging default di AWS IoT (konsol)

Bagian ini menjelaskan cara menggunakan AWS IoT konsol untuk mengonfigurasi logging untuk semua AWS IoT. Untuk mengonfigurasi logging hanya untuk grup hal tertentu, Anda harus menggunakan CLI atau API. Untuk informasi tentang mengonfigurasi logging untuk grup hal tertentu, lihat [Konfigurasi login khusus sumber daya \(CLI\) AWS IoT](#).

Untuk menggunakan AWS IoT konsol untuk mengonfigurasi logging default untuk semua AWS IoT

1. Masuk ke AWS IoT konsol. Untuk informasi selengkapnya, lihat [Buka AWS IoT konsol](#).
2. Pada panel navigasi kiri, pilih Pengaturan. Di bagian Log pada halaman Pengaturan, pilih Kelola log.

Halaman Log menampilkan peran logging dan tingkat verbositas yang digunakan oleh semua AWS IoT

3. Pada halaman Log, pilih Pilih peran untuk menentukan peran yang Anda buat [Buat peran logging](#), atau Buat Peran untuk membuat peran baru yang akan digunakan untuk pencatatan.

Logs Info

Log role Info

Create or select the role you want to use to log information to CloudWatch Logs.

Select role

loggingrole ▼ Create role

Attach policy to IAM role permitting AWS IoT to publish logs to CloudWatch on your behalf.

Log level Info

Select how detailed you want your logs to be. Selecting Error (least verbose) logs only errors and is the least detailed. Selecting Debug (most verbose) creates the most detailed logs. Collecting more detailed logs can increase logging costs.

Log level

Debug (most verbosity) ▼

Cancel Update

- Pilih tingkat Log yang menjelaskan [tingkat detail](#) entri log yang ingin Anda tampilkan di CloudWatch log.
- Pilih Perbarui untuk menyimpan perubahan.

Setelah mengaktifkan logging, kunjungi [Melihat AWS IoT log di CloudWatch konsol](#) untuk mempelajari lebih lanjut tentang melihat entri log.

Konfigurasi login default AWS IoT (CLI)

Bagian ini menjelaskan cara mengonfigurasi pencatatan global AWS IoT dengan menggunakan CLI.

Note

Anda memerlukan Nama Sumber Daya Amazon (ARN) dari peran yang ingin Anda gunakan. Jika Anda perlu membuat peran yang akan digunakan untuk logging, lihat [Buat peran logging](#) sebelum melanjutkan.

Prinsipal yang digunakan untuk memanggil API harus memiliki [Melewati izin peran](#) peran logging Anda.

Anda juga dapat melakukan prosedur ini dengan API dengan menggunakan metode di AWS API yang sesuai dengan perintah CLI yang ditampilkan di sini.

Untuk menggunakan CLI untuk mengonfigurasi logging default untuk AWS IoT

1. Gunakan [set-v2-logging-options](#) perintah untuk mengatur opsi logging untuk akun Anda.

```
aws iot set-v2-logging-options \  
  --role-arn logging-role-arn \  
  --default-log-level log-level
```

di mana:

`--role-arn`

Peran ARN yang memberikan AWS IoT izin untuk menulis ke log Anda di Log. CloudWatch

`--default-log-level`

[Level log](#) yang akan digunakan. Nilai yang valid adalah:ERROR,WARN,INFO,DEBUG, atau DISABLED

`--no-disable-all-logs`

Parameter opsional yang memungkinkan semua AWS IoT logging. Gunakan parameter ini untuk mengaktifkan logging saat ini dinonaktifkan.

`--disable-all-logs`

Parameter opsional yang menonaktifkan semua AWS IoT logging. Gunakan parameter ini untuk menonaktifkan logging saat ini diaktifkan.

2. Gunakan [get-v2-logging-options](#) perintah untuk mendapatkan opsi logging Anda saat ini.


```
aws iot get-v2-logging-options
```

Setelah mengaktifkan logging, kunjungi [Melihat AWS IoT log di CloudWatch konsol](#) untuk mempelajari lebih lanjut tentang melihat entri log.

Note

AWS IoT terus mendukung perintah lama (`set-logging-options` dan `get-logging-options`) untuk mengatur dan mendapatkan pencatatan global di akun Anda. Ketahuilah bahwa ketika perintah ini digunakan, log yang dihasilkan berisi teks biasa, daripada muatan JSON dan latensi logging umumnya lebih tinggi. Tidak ada perbaikan lebih lanjut yang akan dilakukan untuk implementasi perintah lama ini. Kami menyarankan Anda menggunakan versi "v2" untuk mengonfigurasi opsi pencatatan Anda dan, jika memungkinkan, ubah aplikasi lama yang menggunakan versi yang lebih lama.

Konfigurasi login khusus sumber daya (CLI) AWS IoT

Bagian ini menjelaskan cara mengonfigurasi logging khusus sumber daya dengan AWS IoT menggunakan CLI. [Pencatatan khusus sumber daya memungkinkan Anda menentukan tingkat pencatatan untuk grup hal tertentu.](#)

Kelompok benda dapat berisi kelompok hal lain untuk membuat hubungan hierarkis. Prosedur ini menjelaskan cara mengonfigurasi logging dari satu grup hal. Anda dapat menerapkan prosedur ini ke grup hal induk dalam hierarki untuk mengonfigurasi logging untuk semua grup hal dalam hierarki. Anda juga dapat menerapkan prosedur ini ke grup hal anak untuk mengganti konfigurasi logging induknya.

Selain grup benda, Anda juga dapat mencatat target seperti ID klien perangkat, IP sumber, dan ID utama.

Note

Anda memerlukan Nama Sumber Daya Amazon (ARN) dari peran yang ingin Anda gunakan. Jika Anda perlu membuat peran yang akan digunakan untuk logging, lihat [Buat peran logging](#) sebelum melanjutkan.

Prinsipal yang digunakan untuk memanggil API harus memiliki [Melewati izin peran](#) peran logging Anda.

Anda juga dapat melakukan prosedur ini dengan API dengan menggunakan metode di AWS API yang sesuai dengan perintah CLI yang ditampilkan di sini.

Untuk menggunakan CLI untuk mengonfigurasi logging khusus sumber daya untuk AWS IoT

1. Gunakan [set-v2-logging-options](#) perintah untuk mengatur opsi logging untuk akun Anda.

```
aws iot set-v2-logging-options \  
  --role-arn logging-role-arn \  
  --default-log-level log-level
```

di mana:

`--role-arn`

Peran ARN yang memberikan AWS IoT izin untuk menulis ke log Anda di Log. CloudWatch

`--default-log-level`

[Level log](#) yang akan digunakan. Nilai yang valid adalah:ERROR,WARN,INFO,DEBUG, atau DISABLED

`--no-disable-all-logs`

Parameter opsional yang memungkinkan semua AWS IoT logging. Gunakan parameter ini untuk mengaktifkan logging saat saat ini dinonaktifkan.

`--disable-all-logs`

Parameter opsional yang menonaktifkan semua AWS IoT logging. Gunakan parameter ini untuk menonaktifkan logging saat saat ini diaktifkan.

2. Gunakan [set-v2-logging-level](#) perintah untuk mengonfigurasi logging khusus sumber daya untuk grup sesuatu.

```
aws iot set-v2-logging-level \  
  --log-target targetType=THING_GROUP,targetName=thing_group_name \  
  --log-level log_level
```

--log-target

Jenis dan nama sumber daya yang Anda konfigurasi logging. `target_type` Nilai harus salah satu dari: `THING_GROUP` | `CLIENT_ID` | `SOURCE_IP` | `PRINCIPAL_ID`. Nilai parameter `log-target` dapat berupa teks, seperti yang ditunjukkan pada contoh perintah sebelumnya, atau string JSON, seperti contoh berikut.

```
aws iot set-v2-logging-level \  
    --log-target '{"targetType": "THING_GROUP","targetName":  
    "thing_group_name"}' \  
    --log-level log_level
```

--log-level

Tingkat logging yang digunakan saat membuat log untuk sumber daya yang ditentukan. Nilai yang valid adalah: `DEBUG`, `INFO`, `ERROR`, `WARN`, dan `DISABLED`

```
aws iot set-v2-logging-level \  
    --log-target targetType=CLIENT_ID,targetName=ClientId1 \  
    --log-level DEBUG
```

- Gunakan [list-v2-logging-levels](#) perintah untuk mencantumkan level logging yang saat ini dikonfigurasi.

```
aws iot list-v2-logging-levels
```

- Gunakan [delete-v2-logging-level](#) perintah untuk menghapus tingkat logging khusus sumber daya, seperti contoh berikut.

```
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
```

--targetType

target_type Nilai harus salah satu dari: THING_GROUP | CLIENT_ID | SOURCE_IP | PRINCIPAL_ID.

--targetName

Nama grup benda untuk menghapus level logging.

Setelah mengaktifkan logging, kunjungi [Melihat AWS IoT log di CloudWatch konsol](#) untuk mempelajari lebih lanjut tentang melihat entri log.

Tingkat Log

Level log ini menentukan peristiwa yang dicatat dan berlaku untuk tingkat log default dan khusus sumber daya.

ERROR

Kesalahan apa pun yang menyebabkan operasi gagal.

Log hanya menyertakan informasi ERROR.

PERINGATAN

Apa pun yang berpotensi menyebabkan ketidakkonsistenan dalam sistem, tetapi mungkin tidak menyebabkan operasi gagal.

Log termasuk ERROR dan WARN informasi.

INFO

Informasi tingkat tinggi tentang aliran benda.

Log termasuk informasi INFO, ERROR, dan WARN.

AWAKUTU

Informasi yang mungkin berguna saat men-debug masalah.

Log termasuk informasi DEBUG, INFO, ERROR, dan WARN.

DISABLED

Semua logging dinonaktifkan.

Pantau AWS IoT alarm dan metrik menggunakan Amazon CloudWatch

Anda dapat memantau AWS IoT penggunaan CloudWatch, yang mengumpulkan dan memproses data mentah dari AWS IoT metrik yang dapat dibaca, mendekati waktu nyata. Statistik ini dicatat untuk jangka waktu dua minggu, sehingga Anda dapat mengakses informasi historis dan mendapatkan perspektif yang lebih baik tentang kinerja aplikasi atau layanan web Anda. Secara default, data AWS IoT metrik dikirim secara otomatis ke CloudWatch dalam interval satu menit. Untuk informasi selengkapnya, lihat [Apa Itu Amazon CloudWatch, CloudWatch Acara Amazon, dan CloudWatch Log Amazon?](#) di Panduan CloudWatch Pengguna Amazon.

Menggunakan AWS IoT metrik

Metrik yang dilaporkan oleh AWS IoT memberikan informasi yang dapat Anda analisis dengan berbagai cara. Kasus penggunaan berikut didasarkan pada skenario di mana Anda memiliki sepuluh hal yang terhubung ke internet sekali sehari. Setiap hari:

- Sepuluh hal terhubung AWS IoT pada waktu yang hampir bersamaan.
- Setiap hal berlangganan filter topik, dan kemudian menunggu selama satu jam sebelum memutuskan sambungan. Selama periode ini, hal-hal berkomunikasi satu sama lain dan belajar lebih banyak tentang keadaan dunia.
- Setiap hal menerbitkan beberapa persepsi yang didasarkan pada penggunaan UpdateThingShadow data yang baru ditemukan.
- Setiap hal terputus dari AWS IoT.

Untuk membantu Anda memulai, topik ini mengeksplorasi beberapa pertanyaan yang mungkin Anda miliki.

- [Bagaimana saya bisa diberi tahu jika barang-barang saya tidak terhubung dengan sukses setiap hari?](#)
- [Bagaimana saya bisa diberi tahu jika barang-barang saya tidak mempublikasikan data setiap hari?](#)
- [Bagaimana saya bisa diberi tahu jika pembaruan bayangan barang saya ditolak setiap hari?](#)
- [Bagaimana saya bisa membuat CloudWatch alarm untuk pekerjaan?](#)

Lebih lanjut tentang CloudWatch alarm dan metrik

- [Membuat CloudWatch alarm untuk memantau AWS IoT](#)
- [AWS IoT metrik dan dimensi](#)

Membuat CloudWatch alarm untuk memantau AWS IoT

Anda dapat membuat CloudWatch alarm yang mengirimkan pesan Amazon SNS saat alarm berubah status. Alarm mengawasi satu metrik selama suatu periode waktu yang Anda tentukan. Ketika nilai metrik melebihi ambang batas yang diberikan selama beberapa periode waktu, satu atau lebih tindakan dilakukan. Tindakan tersebut dapat berupa pemberitahuan yang dikirim ke topik Amazon SNS atau kebijakan Auto Scaling. Alarm memicu tindakan untuk perubahan status berkelanjutan saja. CloudWatch alarm tidak memicu tindakan hanya karena mereka berada dalam keadaan tertentu; negara harus telah berubah dan dipertahankan untuk sejumlah periode tertentu.

Topik berikut menjelaskan beberapa contoh penggunaan CloudWatch alarm.

- [Bagaimana saya bisa diberi tahu jika barang-barang saya tidak terhubung dengan sukses setiap hari?](#)
- [Bagaimana saya bisa diberitahu jika barang-barang saya tidak mempublikasikan data setiap hari?](#)
- [Bagaimana saya bisa diberi tahu jika pembaruan bayangan barang saya ditolak setiap hari?](#)
- [Bagaimana saya bisa membuat CloudWatch alarm untuk pekerjaan?](#)

Anda dapat melihat semua metrik yang dapat dipantau oleh CloudWatch alarm. [AWS IoT metrik dan dimensi](#)

Bagaimana saya bisa diberi tahu jika barang-barang saya tidak terhubung dengan sukses setiap hari?

1. Buat topik Amazon SNS bernama `things-not-connecting-successfully`, dan rekam Nama Sumber Daya Amazon (ARN). Prosedur ini akan merujuk ke ARN topik Anda sebagai *sns-topic-arn*

Untuk informasi selengkapnya tentang cara membuat notifikasi Amazon SNS, lihat [Memulai Amazon SNS](#).

2. Buat alarm.

```
aws cloudwatch put-metric-alarm \  
  --alarm-name ConnectSuccessAlarm \  
  --metric-name ConnectSuccessAlarm \  
  --namespace AWS/iot \  
  --dimensions Name=ThingName,Value=ConnectSuccessAlarm \  
  --period 60 \  
  --threshold 1 \  
  --evaluation-periods 3 \  
  --treat-missing-data-as Breach \  
  --sns-topic-arn sns-topic-arn
```

```
--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. Uji alarm.

```
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. Verifikasi bahwa alarm muncul di [CloudWatch konsol](#) Anda.

Bagaimana saya bisa diberitahu jika barang-barang saya tidak mempublikasikan data setiap hari?

1. Buat topik Amazon SNS bernama `things-not-publishing-data`, dan rekam Nama Sumber Daya Amazon (ARN). Prosedur ini akan merujuk ke ARN topik Anda sebagai *sns-topic-arn*

Untuk informasi selengkapnya tentang cara membuat notifikasi Amazon SNS, lihat [Memulai Amazon SNS](#).

2. Buat alarm.

```
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. Uji alarm.

```
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. Verifikasi bahwa alarm muncul di [CloudWatch konsol](#) Anda.

Bagaimana saya bisa diberi tahu jika pembaruan bayangan barang saya ditolak setiap hari?

1. Buat topik Amazon SNS bernama `things-shadow-updates-rejected`, dan rekam Nama Sumber Daya Amazon (ARN). Prosedur ini akan merujuk ke ARN topik Anda sebagai *sns-topic-arn*

Untuk informasi selengkapnya tentang cara membuat notifikasi Amazon SNS, lihat [Memulai Amazon SNS](#).

2. Buat alarm.

```
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. Uji alarm.

```
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. Verifikasi bahwa alarm muncul di [CloudWatch konsol](#) Anda.

Bagaimana saya bisa membuat CloudWatch alarm untuk pekerjaan?

Layanan Jobs menyediakan CloudWatch metrik bagi Anda untuk memantau pekerjaan Anda. Anda dapat membuat CloudWatch alarm untuk memantau apa pun [Metrik pekerjaan](#).

Perintah berikut membuat CloudWatch alarm untuk memantau jumlah total eksekusi pekerjaan yang gagal untuk Contoh Job *Sample0taJob* dan memberi tahu Anda bila lebih dari 20 eksekusi pekerjaan gagal. Alarm memantau metrik Pekerjaan FailedJobExecutionTotalCount dengan memeriksa nilai yang dilaporkan setiap 300 detik. Ini diaktifkan ketika satu nilai yang dilaporkan lebih besar dari 20, yang berarti ada lebih dari 20 eksekusi pekerjaan yang gagal sejak pekerjaan dimulai. Ketika alarm berbunyi, ia mengirimkan pemberitahuan ke topik Amazon SNS yang disediakan.

```
aws cloudwatch put-metric-alarm \  
  --alarm-name TotalFailedJobExecution-Sample0TAJob \  
  --alarm-description "Alarm when total number of failed job execution exceeds the threshold for Sample0TAJob" \  
  --namespace AWS/IoT \  
  --metric-name FailedJobExecutionTotalCount \  
  --dimensions Name=JobId,Value=Sample0TAJob \  
  --statistic Sum \  
  --threshold 20 \  
  --comparison-operator GreaterThanThreshold \  
  --period 300 \  
  --unit Count \  
  --evaluation-periods 1 \  
  --alarm-actions arn:aws:sns:<AWS_REGION>:<AWS_ACCOUNT_ID>:Sample0TAJob-has-too-many-failed-job-eexecutions
```

Perintah berikut membuat CloudWatch alarm untuk memantau jumlah eksekusi pekerjaan yang gagal untuk Job *Sample0taJob* dalam periode tertentu. Ini kemudian memberi tahu Anda ketika

lebih dari lima eksekusi pekerjaan telah gagal selama periode itu. Alarm memantau metrik Pekerjaan FailedJobExecutionCount dengan memeriksa nilai yang dilaporkan setiap 3600 detik. Ini diaktifkan ketika satu nilai yang dilaporkan lebih besar dari 5, yang berarti ada lebih dari 5 eksekusi pekerjaan yang gagal dalam satu jam terakhir. Ketika alarm berbunyi, ia mengirimkan pemberitahuan ke topik Amazon SNS yang disediakan.

```
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 metrik dan dimensi

Saat Anda berinteraksi AWS IoT, layanan mengirimkan metrik dan dimensi berikut ke CloudWatch setiap menit. Anda dapat menggunakan prosedur berikut untuk melihat metrik. AWS IoT

Untuk melihat metrik (CloudWatch konsol)

Metrik dikelompokkan terlebih dahulu berdasarkan namespace layanan, lalu berdasarkan berbagai kombinasi dimensi dalam setiap namespace.

1. Buka [konsol CloudWatch](#).
2. Di panel navigasi, pilih Metrik lalu pilih Semua metrik.
3. Di tab Browse, cari AWS IoT untuk melihat daftar metrik.

Untuk melihat metrik (CLI)

- Pada prompt perintah, gunakan perintah berikut:

```
aws cloudwatch list-metrics --namespace "AWS/IoT"
```

CloudWatch menampilkan grup metrik berikut untuk AWS IoT:

- [AWS IoT metrik](#)
- [AWS IoT Core metrik penyedia kredensi](#)
- [Metrik otentikasi](#)
- [Sertifikat server metrik stapling OCSP](#)
- [Metrik aturan](#)
- [Metrik tindakan aturan](#)
- [Metrik spesifik tindakan HTTP](#)
- [Metrik broker pesan](#)
- [Metrik bayangan perangkat](#)
- [Metrik pekerjaan](#)
- [Metrik audit Device Defender](#)
- [Device Defender mendeteksi metrik](#)
- [Metrik penyediaan perangkat](#)
- [LoRaMetrik WAN](#)
- [Metrik pengindeksan armada](#)
- [Dimensi untuk metrik](#)

AWS IoT metrik

Metrik	Deskripsi
AddThingToDynamicThingGroup sFailed	Jumlah peristiwa kegagalan yang terkait dengan menambahkan sesuatu ke grup hal dinamis. DynamicThingGroupName Dimensi berisi nama grup dinamis yang gagal menambahkan sesuatu.

Metrik	Deskripsi
NumLogBatchesFailedToPublishThrottled	Kumpulan tunggal peristiwa log yang gagal dipublikasikan karena kesalahan pelambatan.
NumLogEventsFailedToPublishThrottled	Jumlah peristiwa log dalam batch yang gagal dipublikasikan karena kesalahan pelambatan.

AWS IoT Core metrik penyedia kredensi

Metrik	Deskripsi
CredentialExchangeSuccess	Jumlah AssumeRoleWithCertificate permintaan yang berhasil ke penyedia AWS IoT Core kredensial.

Metrik otentikasi

Note

Metrik otentikasi ditampilkan di CloudWatch konsol di bawah Protocol Metrics.

Metrik	Deskripsi
Connect.AuthNErrror	Jumlah upaya koneksi yang AWS IoT Core menolak karena kegagalan otentikasi. Metrik ini hanya mempertimbangkan koneksi yang mengirim string Server Name Indication (SNI) yang cocok dengan titik akhir Anda. Akun AWS Metrik ini mencakup upaya koneksi dari sumber eksternal seperti alat pemindaian internet atau aktivitas probing. ProtocolDimensi berisi protokol yang digunakan untuk mengirim upaya koneksi.

Sertifikat server metrik stapling OCSP

Metrik	Deskripsi
StapleDataRetrieveOCSP .Success	Tanggapan OCSP telah diterima dan diproses dengan sukses. Respons ini akan disertakan selama jabat tangan TLS untuk domain yang dikonfigurasi. <code>DomainConfigurationName</code> Dimensi berisi nama domain yang dikonfigurasi dengan sertifikat server diaktifkan OCSP stapling.

Metrik aturan

Metrik	Deskripsi
ParseError	Jumlah kesalahan parse JSON yang terjadi dalam pesan yang dipublikasikan pada topik di mana aturan mendengarkan. <code>RuleName</code> Dimensi berisi nama aturan.
RuleMessageThrottled	Jumlah pesan yang dibatasi oleh mesin aturan karena perilaku jahat atau karena jumlah pesan melebihi batas throttle engine aturan. <code>RuleName</code> Dimensi berisi nama aturan yang akan dipicu.
RuleNotFound	Aturan yang akan dipicu tidak dapat ditemukan. <code>RuleName</code> Dimensi berisi nama aturan.
RulesExecuted	Jumlah AWS IoT aturan yang dieksekusi.
TopicMatch	Jumlah pesan masuk yang diterbitkan pada topik di mana aturan mendengarkan. <code>RuleName</code> Dimensi berisi nama aturan.

Metrik tindakan aturan

Metrik	Deskripsi
Failure	Jumlah pemanggilan tindakan aturan yang gagal. <code>RuleNameDimensi</code> berisi nama aturan yang menentukan tindakan. <code>ActionType Dimensi</code> berisi jenis tindakan yang dipanggil.
Success	Jumlah pemanggilan tindakan aturan yang berhasil. <code>RuleNameDimensi</code> berisi nama aturan yang menentukan tindakan. <code>ActionType Dimensi</code> berisi jenis tindakan yang dipanggil.
ErrorActionFailure	Jumlah tindakan kesalahan yang gagal. <code>RuleNameDimensi</code> berisi nama aturan yang menentukan tindakan. <code>ActionType Dimensi</code> berisi jenis tindakan yang dipanggil.
ErrorActionSuccess	Jumlah tindakan kesalahan yang berhasil. <code>RuleNameDimensi</code> berisi nama aturan yang menentukan tindakan. <code>ActionType Dimensi</code> berisi jenis tindakan yang dipanggil.

Metrik spesifik tindakan HTTP

Metrik	Deskripsi
HttpCode_0ther	Dihasilkan jika kode status respons dari layanan web/aplikasi hilir tidak 2xx, 4xx atau 5xx.
HttpCode_4XX	Dihasilkan jika kode status respons dari layanan web/aplikasi hilir adalah antara 400 dan 499.
HttpCode_5XX	Dihasilkan jika kode status respons dari layanan web/aplikasi hilir adalah antara 500 dan 599.

Metrik	Deskripsi
HttpInvalidUrl	Dihasilkan jika URL titik akhir, setelah templat substitusi diganti, tidak dimulai dengan. https://
HttpRequestTimeout	Dihasilkan jika layanan/aplikasi web hilir tidak mengembalikan respons dalam batas waktu tunggu permintaan. Untuk informasi selengkapnya, lihat Service Quotas .
HttpUnknownHost	Dihasilkan jika URL valid, tetapi layanan tidak ada atau tidak dapat dijangkau.

Metrik broker pesan

Note

Metrik broker pesan ditampilkan di CloudWatch konsol di bawah Metrik Protokol.

Metrik	Deskripsi
Connect.AuthError	Jumlah permintaan koneksi yang tidak dapat disahkan oleh broker pesan. ProtocolDimensi berisi protokol yang digunakan untuk mengirim CONNECT pesan.
Connect.ClientError	Jumlah permintaan koneksi ditolak karena pesan MQTT tidak memenuhi persyaratan yang ditentukan dalam. Kuota AWS IoT ProtocolDimensi berisi protokol yang digunakan untuk mengirim CONNECT pesan.
Connect.ClientIDThrottle	Jumlah permintaan koneksi dibatasi karena klien melebihi tingkat permintaan koneksi yang diizinkan untuk ID klien tertentu. ProtocolDimensi berisi

Metrik	Deskripsi
	protokol yang digunakan untuk mengirim CONNECT pesan.
<code>Connect.ServerError</code>	Jumlah permintaan koneksi yang gagal karena terjadi kesalahan internal. <code>ProtocolDimensi</code> berisi protokol yang digunakan untuk mengirim CONNECT pesan.
<code>Connect.Success</code>	Jumlah koneksi yang berhasil ke broker pesan. <code>ProtocolDimensi</code> berisi protokol yang digunakan untuk mengirim CONNECT pesan.
<code>Connect.Throttle</code>	Jumlah permintaan koneksi yang dibatasi karena akun melebihi tingkat permintaan koneksi yang diizinkan. <code>ProtocolDimensi</code> berisi protokol yang digunakan untuk mengirim CONNECT pesan.
<code>Ping.Success</code>	Jumlah pesan ping yang diterima oleh broker pesan. <code>ProtocolDimensi</code> berisi protokol yang digunakan untuk mengirim pesan ping.
<code>PublishIn.AuthError</code>	Jumlah permintaan publikasi yang tidak dapat diotorisasi oleh broker pesan. <code>ProtocolDimensi</code> berisi protokol yang digunakan untuk mempublikasikan pesan. HTTP Publish tidak mendukung metrik ini.
<code>PublishIn.ClientError</code>	Jumlah permintaan publikasi ditolak oleh broker pesan karena pesan tidak memenuhi persyaratan yang ditentukan dalam Kuota AWS IoT . <code>ProtocolDimensi</code> berisi protokol yang digunakan untuk mempublikasikan pesan. HTTP Publish tidak mendukung metrik ini.

Metrik	Deskripsi
<code>PublishIn.ServerError</code>	Jumlah permintaan publikasi yang gagal diproses oleh broker pesan karena terjadi kesalahan internal. <code>ProtocolDimensi</code> berisi protokol yang digunakan untuk mengirim PUBLISH pesan. HTTP Publish tidak mendukung metrik ini.
<code>PublishIn.Success</code>	Jumlah permintaan publikasi yang berhasil diproses oleh broker pesan. <code>ProtocolDimensi</code> berisi protokol yang digunakan untuk mengirim PUBLISH pesan.
<code>PublishIn.Throttle</code>	Jumlah permintaan publikasi yang dibatasi karena klien melebihi tingkat pesan masuk yang diizinkan. <code>ProtocolDimensi</code> berisi protokol yang digunakan untuk mengirim PUBLISH pesan. HTTP Publish tidak mendukung metrik ini.
<code>PublishOut.AuthError</code>	Jumlah permintaan publikasi yang dibuat oleh broker pesan yang tidak dapat diotorisasi oleh AWS IoT. <code>ProtocolDimensi</code> berisi protokol yang digunakan untuk mengirim PUBLISH pesan.
<code>PublishOut.ClientError</code>	Jumlah permintaan publikasi yang dibuat oleh broker pesan yang ditolak karena pesan tidak memenuhi persyaratan yang ditentukan dalam Kuota AWS IoT . <code>ProtocolDimensi</code> berisi protokol yang digunakan untuk mengirim PUBLISH pesan.
<code>PublishOut.Success</code>	Jumlah permintaan publikasi yang berhasil dibuat oleh broker pesan. <code>ProtocolDimensi</code> berisi protokol yang digunakan untuk mengirim PUBLISH pesan.
<code>PublishOut.Throttle</code>	Jumlah permintaan publikasi yang dibatasi karena klien melebihi tingkat pesan keluar yang diizinkan. <code>ProtocolDimensi</code> berisi protokol yang digunakan untuk mengirim PUBLISH pesan.

Metrik	Deskripsi
<code>PublishRetained.AuthError</code>	Jumlah permintaan publikasi dengan RETAIN tanda yang ditetapkan bahwa broker pesan tidak dapat mengotorisasi. <code>ProtocolDimensi</code> berisi protokol yang digunakan untuk mengirim PUBLISH pesan.
<code>PublishRetained.ServerError</code>	Jumlah permintaan publikasi yang ditahan yang gagal diproses oleh broker pesan karena terjadi kesalahan internal. <code>ProtocolDimensi</code> berisi protokol yang digunakan untuk mengirim PUBLISH pesan.
<code>PublishRetained.Success</code>	Jumlah permintaan publikasi dengan set RETAIN bendera yang berhasil diproses oleh broker pesan. <code>ProtocolDimensi</code> berisi protokol yang digunakan untuk mengirim PUBLISH pesan.
<code>PublishRetained.Throttle</code>	Jumlah permintaan publikasi dengan set RETAIN tanda yang dibatasi karena klien melebihi tingkat pesan masuk yang diizinkan. <code>ProtocolDimensi</code> berisi protokol yang digunakan untuk mengirim PUBLISH pesan.
<code>Queued.Success</code>	Jumlah pesan tersimpan yang berhasil diproses oleh broker pesan untuk klien yang terputus dari sesi persisten mereka. Pesan dengan QoS 1 disimpan sementara klien dengan sesi persisten terputus.
<code>Queued.Throttle</code>	Jumlah pesan yang tidak dapat disimpan dan dibatasi saat klien dengan sesi persisten terputus. Ini terjadi ketika klien melebihi batas pesan Antrian per detik per akun . Pesan dengan QoS 1 disimpan sementara klien dengan sesi persisten terputus.
<code>Queued.ServerError</code>	Jumlah pesan yang belum disimpan untuk sesi persisten karena kesalahan internal. Ketika klien dengan sesi persisten terputus, pesan dengan Quality of Service (QoS) 1 disimpan.

Metrik	Deskripsi
<code>Subscribe.AuthError</code>	Jumlah permintaan berlangganan yang dibuat oleh klien yang tidak dapat diotorisasi. <code>ProtocolDimensi</code> berisi protokol yang digunakan untuk mengirim <code>SUBSCRIBE</code> pesan.
<code>Subscribe.ClientError</code>	Jumlah permintaan berlangganan yang ditolak karena <code>SUBSCRIBE</code> pesan tidak memenuhi persyaratan yang ditentukan dalam Kuota AWS IoT . <code>ProtocolDimensi</code> berisi protokol yang digunakan untuk mengirim <code>SUBSCRIBE</code> pesan.
<code>Subscribe.ServerError</code>	Jumlah permintaan berlangganan yang ditolak karena terjadi kesalahan internal. <code>ProtocolDimensi</code> berisi protokol yang digunakan untuk mengirim <code>SUBSCRIBE</code> pesan.
<code>Subscribe.Success</code>	Jumlah permintaan berlangganan yang berhasil diproses oleh broker pesan. <code>ProtocolDimensi</code> berisi protokol yang digunakan untuk mengirim <code>SUBSCRIBE</code> pesan.
<code>Subscribe.Throttle</code>	Jumlah permintaan berlangganan yang dibatasi karena batas tarif permintaan berlangganan yang diizinkan terlampaui untuk Anda. Akun AWS Batasan ini termasuk Langganan per detik per akun, Langganan per akun, dan Langganan per koneksi yang dijelaskan dalam broker AWS IoT Core pesan dan batas protokol dan kuota . <code>ProtocolDimensi</code> berisi protokol yang digunakan untuk mengirim <code>SUBSCRIBE</code> pesan.
<code>Throttle.Exceeded</code>	Metrik ini akan ditampilkan CloudWatch ketika klien MQTT dibatasi pada paket per detik per batas tingkat koneksi. Metrik ini tidak berlaku untuk koneksi HTTP.

Metrik	Deskripsi
<code>Unsubscribe.ClientError</code>	Jumlah permintaan berhenti berlangganan yang ditolak karena UNSUBSCRIBE pesan tidak memenuhi persyaratan yang ditentukan dalam Kuota AWS IoT . <code>ProtocolDimensi</code> berisi protokol yang digunakan untuk mengirim UNSUBSCRIBE pesan.
<code>Unsubscribe.ServerError</code>	Jumlah permintaan berhenti berlangganan yang ditolak karena terjadi kesalahan internal. <code>ProtocolDimensi</code> berisi protokol yang digunakan untuk mengirim UNSUBSCRIBE pesan.
<code>Unsubscribe.Success</code>	Jumlah permintaan berhenti berlangganan yang berhasil diproses oleh broker pesan. <code>ProtocolDimensi</code> berisi protokol yang digunakan untuk mengirim UNSUBSCRIBE pesan.
<code>Unsubscribe.Throttle</code>	Jumlah permintaan berhenti berlangganan yang ditolak karena klien melebihi tingkat permintaan berhenti berlangganan yang diizinkan. <code>ProtocolDimensi</code> berisi protokol yang digunakan untuk mengirim UNSUBSCRIBE pesan.

Metrik bayangan perangkat

Note

Metrik bayangan perangkat ditampilkan di CloudWatch konsol di bawah Metrik Protokol.

Metrik	Deskripsi
<code>DeleteThingShadow.Accepted</code>	Jumlah <code>DeleteThingShadow</code> permintaan yang berhasil diproses. <code>ProtocolDimensi</code> berisi protokol yang digunakan untuk membuat permintaan.

Metrik	Deskripsi
<code>GetThingShadow.Accepted</code>	Jumlah <code>GetThingShadow</code> permintaan yang berhasil diproses. <code>ProtocolDimensi</code> berisi protokol yang digunakan untuk membuat permintaan.
<code>ListThingShadow.Accepted</code>	Jumlah <code>ListThingShadow</code> permintaan yang berhasil diproses. <code>ProtocolDimensi</code> berisi protokol yang digunakan untuk membuat permintaan.
<code>UpdateThingShadow.Accepted</code>	Jumlah <code>UpdateThingShadow</code> permintaan yang berhasil diproses. <code>ProtocolDimensi</code> berisi protokol yang digunakan untuk membuat permintaan.

Metrik pekerjaan

Metrik	Deskripsi
<code>CanceledJobExecutionCount</code>	Jumlah eksekusi pekerjaan yang statusnya telah berubah menjadi CANCELED dalam jangka waktu yang ditentukan oleh CloudWatch. (Untuk informasi selengkapnya tentang CloudWatch metrik, lihat CloudWatchMetrik Amazon .) <code>JobIdDimensi</code> berisi ID pekerjaan.
<code>CanceledJobExecutionTotalCount</code>	Jumlah total eksekusi pekerjaan yang statusnya CANCELED untuk pekerjaan yang diberikan. <code>JobIdDimensi</code> berisi ID pekerjaan.
<code>ClientErrorCount</code>	Jumlah kesalahan klien yang dihasilkan saat menjalankan pekerjaan. <code>JobIdDimensi</code> berisi ID pekerjaan.
<code>FailedJobExecutionCount</code>	Jumlah eksekusi pekerjaan yang statusnya telah berubah menjadi FAILED dalam jangka waktu yang ditentukan oleh CloudWatch. (Untuk informasi selengkapnya tentang CloudWatch metrik, lihat

Metrik	Deskripsi
FailedJobExecutionTotalCount	<p>(CloudWatch Metrik Amazon.) JobIdDimensi berisi ID pekerjaan.</p> <p>Jumlah total eksekusi pekerjaan yang statusnya FAILED untuk pekerjaan yang diberikan. JobIdDimensi berisi ID pekerjaan.</p>
InProgressJobExecutionCount	<p>Jumlah eksekusi pekerjaan yang statusnya telah berubah menjadi IN_PROGRESS dalam jangka waktu yang ditentukan oleh CloudWatch. (Untuk informasi selengkapnya tentang CloudWatch metrik, lihat CloudWatchMetrik Amazon.) JobIdDimensi berisi ID pekerjaan.</p>
InProgressJobExecutionTotalCount	<p>Jumlah total eksekusi pekerjaan yang statusnya IN_PROGRESS untuk pekerjaan yang diberikan. JobIdDimensi berisi ID pekerjaan.</p>
RejectedJobExecutionTotalCount	<p>Jumlah total eksekusi pekerjaan yang statusnya REJECTED untuk pekerjaan yang diberikan. JobIdDimensi berisi ID pekerjaan.</p>
RemovedJobExecutionTotalCount	<p>Jumlah total eksekusi pekerjaan yang statusnya REMOVED untuk pekerjaan yang diberikan. JobIdDimensi berisi ID pekerjaan.</p>
QueuedJobExecutionCount	<p>Jumlah eksekusi pekerjaan yang statusnya telah berubah menjadi QUEUED dalam jangka waktu yang ditentukan oleh CloudWatch. (Untuk informasi selengkapnya tentang CloudWatch metrik, lihat CloudWatch Metrik Amazon.) JobIdDimensi berisi ID pekerjaan.</p>
QueuedJobExecutionTotalCount	<p>Jumlah total eksekusi pekerjaan yang statusnya QUEUED untuk pekerjaan yang diberikan. JobIdDimensi berisi ID pekerjaan.</p>

Metrik	Deskripsi
RejectedJobExecutionCount	Jumlah eksekusi pekerjaan yang statusnya telah berubah menjadi REJECTED dalam jangka waktu yang ditentukan oleh CloudWatch. (Untuk informasi selengkapnya tentang CloudWatch metrik, lihat CloudWatchMetrik Amazon .) JobIdDimensi berisi ID pekerjaan.
RemovedJobExecutionCount	Jumlah eksekusi pekerjaan yang statusnya telah berubah menjadi REMOVED dalam jangka waktu yang ditentukan oleh CloudWatch. (Untuk informasi selengkapnya tentang CloudWatch metrik, lihat CloudWatchMetrik Amazon .) JobIdDimensi berisi ID pekerjaan.
ServerErrorCount	Jumlah kesalahan server yang dihasilkan saat menjalankan pekerjaan. JobIdDimensi berisi ID pekerjaan.
SucceededJobExecutionCount	Jumlah eksekusi pekerjaan yang statusnya telah berubah menjadi SUCCESS dalam jangka waktu yang ditentukan oleh CloudWatch. (Untuk informasi selengkapnya tentang CloudWatch metrik, lihat CloudWatchMetrik Amazon .) JobIdDimensi berisi ID pekerjaan.
SucceededJobExecutionTotalCount	Jumlah total eksekusi pekerjaan yang statusnya SUCCESS untuk pekerjaan yang diberikan. JobIdDimensi berisi ID pekerjaan.

Metrik audit Device Defender

Metrik	Deskripsi
NonCompliantResources	Jumlah sumber daya yang ditemukan tidak sesuai dengan cek. Sistem melaporkan jumlah sumber daya

Metrik	Deskripsi
	yang tidak sesuai untuk setiap pemeriksaan setiap audit yang dilakukan.
ResourcesEvaluated	Jumlah sumber daya yang dievaluasi untuk kepatuhan. Sistem melaporkan jumlah sumber daya yang dievaluasi untuk setiap pemeriksaan setiap audit yang dilakukan.
MisconfiguredDeviceDefenderNotification	Memberi tahu Anda ketika konfigurasi SNS Anda AWS IoT Device Defender salah dikonfigurasi. Dimensi

Device Defender mendeteksi metrik

Metrik	Deskripsi
NumOfMetricsExported	Jumlah metrik yang diekspor untuk metrik sisi cloud, sisi perangkat, atau kustom. Sistem melaporkan jumlah metrik yang diekspor untuk akun, untuk metrik tertentu. Metrik ini hanya tersedia untuk pelanggan yang menggunakan ekspor metrik.
NumOfMetricsSkipped	Jumlah metrik yang dilewati untuk metrik sisi cloud, sisi perangkat, atau kustom. Sistem melaporkan jumlah metrik yang dilewati untuk akun, untuk metrik tertentu karena izin yang tidak memadai yang diberikan kepada Device Defender Detect untuk dipublikasikan ke topik mqtt. Metrik ini hanya tersedia untuk pelanggan yang menggunakan ekspor metrik.
NumOfMetricsExceedingSizeLimit	Jumlah metrik yang dilewati untuk ekspor untuk metrik sisi cloud, sisi perangkat, atau kustom karena ukuran melebihi batasan ukuran pesan MQTT. Sistem melaporkan jumlah metrik yang dilewati untuk ekspor akun, untuk metrik tertentu karena ukuran

Metrik	Deskripsi
	melebihi batasan ukuran pesan MQTT. Metrik ini hanya tersedia untuk pelanggan yang menggunakan ekspor metrik.
Violations	Jumlah pelanggaran baru perilaku profil keamanan yang telah ditemukan sejak terakhir kali evaluasi dilakukan. Sistem melaporkan jumlah pelanggaran baru untuk akun, untuk profil keamanan tertentu, dan untuk perilaku tertentu dari profil keamanan tertentu.
ViolationsCleared	Jumlah pelanggaran perilaku profil keamanan yang telah diselesaikan sejak terakhir kali evaluasi dilakukan. Sistem melaporkan jumlah pelanggaran yang diselesaikan untuk akun, untuk profil keamanan tertentu, dan untuk perilaku tertentu dari profil keamanan tertentu.
ViolationsInvalidated	Jumlah pelanggaran perilaku profil keamanan yang informasinya tidak lagi tersedia sejak terakhir kali evaluasi dilakukan (karena perangkat pelaporan berhenti melaporkan, atau tidak lagi dipantau karena alasan tertentu). Sistem melaporkan jumlah pelanggaran yang tidak valid untuk seluruh akun, untuk profil keamanan tertentu, dan untuk perilaku tertentu dari profil keamanan tertentu.
MisconfiguredDeviceDefenderNotification	Memberi tahu Anda ketika konfigurasi SNS Anda AWS IoT Device Defender salah dikonfigurasi. Dimensi

Metrik penyediaan perangkat

AWS IoT Metrik penyediaan armada

Metrik	Deskripsi
<p><code>ApproximateNumberOfThingsRegistered</code></p>	<p>Hitungan hal-hal yang telah didaftarkan oleh Fleet Provisioning.</p> <p>Meskipun hitungannya umumnya akurat, arsitektur terdistribusi AWS IoT Core membuatnya sulit untuk mempertahankan jumlah yang tepat dari hal-hal yang terdaftar.</p> <p>Statistik yang digunakan untuk metrik ini adalah:</p> <ul style="list-style-type: none"> • Max untuk melaporkan jumlah total barang yang telah terdaftar. Untuk hitungan hal-hal yang terdaftar selama jendela CloudWatch agregasi, lihat <code>RegisterThingFailed</code> metrik. <p>Dimensi: ClaimCertificateId</p>
<p><code>CreateKeysAndCertificateFailed</code></p>	<p>Jumlah kegagalan yang terjadi oleh panggilan ke <code>CreateKeysAndCertificate</code> MQTT API.</p> <p>Metrik dipancarkan dalam kasus Sukses (nilai = 0) dan Kegagalan (nilai = 1). Metrik ini dapat digunakan untuk melacak jumlah sertifikat yang dibuat dan didaftarkan selama jendela agregasi yang CloudWatch didukung, seperti 5 menit. atau 1 jam.</p> <p>Statistik yang tersedia untuk metrik ini adalah:</p> <ul style="list-style-type: none"> • Jumlahkan untuk melaporkan jumlah panggilan yang gagal. • <code>SampleCount</code> untuk melaporkan jumlah total panggilan yang berhasil dan gagal.

Metrik	Deskripsi
<code>CreateCertificateFromCsrfailed</code>	<p>Jumlah kegagalan yang terjadi oleh panggilan ke <code>CreateCertificateFromCsrfailed</code> MQTT API.</p> <p>Metrik dipancarkan dalam kasus Sukses (nilai = 0) dan Kegagalan (nilai = 1). Metrik ini dapat digunakan untuk melacak jumlah hal yang terdaftar selama jendela agregasi yang CloudWatch didukung, seperti 5 menit. atau 1 jam.</p> <p>Statistik yang tersedia untuk metrik ini adalah:</p> <ul style="list-style-type: none">• Jumlahkan untuk melaporkan jumlah panggilan yang gagal.• <code>SampleCount</code> untuk melaporkan jumlah total panggilan yang berhasil dan gagal.
<code>RegisterThingFailed</code>	<p>Jumlah kegagalan yang terjadi oleh panggilan ke <code>RegisterThing</code> MQTT API.</p> <p>Metrik dipancarkan dalam kasus Sukses (nilai = 0) dan Kegagalan (nilai = 1). Metrik ini dapat digunakan untuk melacak jumlah hal yang terdaftar selama jendela agregasi yang CloudWatch didukung, seperti 5 menit. atau 1 jam. Untuk jumlah total barang yang terdaftar, lihat <code>ApproximateNumberOfThingsRegistered</code> metrik.</p> <p>Statistik yang tersedia untuk metrik ini adalah:</p> <ul style="list-style-type: none">• Jumlahkan untuk melaporkan jumlah panggilan yang gagal.• <code>SampleCount</code> untuk melaporkan jumlah total panggilan yang berhasil dan gagal. <p>Dimensi: TemplateName</p>

ust-in-time Metrik penyediaan J

Metrik	Deskripsi
<code>ProvisionThing.ClientError</code>	Berapa kali perangkat gagal menyediakan karena kesalahan klien. Misalnya, kebijakan yang ditentukan dalam templat tidak ada.
<code>ProvisionThing.ServerError</code>	Berapa kali perangkat gagal menyediakan karena kesalahan server. Pelanggan dapat mencoba lagi untuk menyediakan perangkat setelah menunggu dan mereka dapat menghubungi AWS IoT jika masalahnya tetap sama.
<code>ProvisionThing.Success</code>	Berapa kali perangkat berhasil disediakan.

LoRaMetrik WAN

Tabel berikut menunjukkan metrik AWS IoT Core untuk LoRa WAN. Untuk informasi selengkapnya, lihat [AWS IoT Core metrik LoRa WAN](#).

AWS IoT Core untuk metrik LoRa WAN

Metrik	Deskripsi
Perangkat/gateway aktif	Jumlah perangkat LoRa WAN aktif dan gateway di akun Anda.
Jumlah pesan uplink	Jumlah pesan uplink yang dikirim dalam durasi waktu tertentu untuk semua gateway dan perangkat aktif di perangkat Anda. Akun AWS Pesan Uplink adalah pesan yang dikirim dari perangkat Anda ke WAN AWS IoT Core . LoRa
Jumlah pesan downlink	Jumlah pesan downlink yang dikirim dalam durasi waktu tertentu untuk semua gateway dan perangkat aktif di Anda. Akun AWS Pesan downlink adalah pesan yang dikirim dari AWS IoT Core LoRa WAN ke perangkat Anda.

Metrik	Deskripsi
Tingkat kehilangan pesan	Setelah menambahkan perangkat dan terhubung ke AWS IoT Core LoRa WAN, perangkat Anda dapat memulai pesan uplink untuk mulai bertukar pesan dengan cloud. Anda dapat menggunakan metrik ini untuk kemudian melacak tingkat pesan uplink yang hilang.
Bergabunglah dengan metrik	Setelah menambahkan perangkat dan gateway, Anda melakukan prosedur bergabung sehingga perangkat Anda dapat mengirim data uplink dan berkomunikasi dengan WAN AWS IoT Core . LoRa Anda dapat menggunakan metrik ini untuk mendapatkan informasi tentang metrik gabungan untuk semua perangkat aktif di perangkat Anda Akun AWS.
Indikator kekuatan sinyal yang diterima rata-rata (RSSI)	Anda dapat menggunakan metrik ini untuk memantau RSSI rata-rata (indikator kekuatan sinyal yang diterima) dalam durasi waktu yang ditentukan. RSSI adalah pengukuran yang menunjukkan apakah sinyal cukup kuat untuk koneksi nirkabel yang baik. Nilai ini negatif dan harus mendekati nol untuk koneksi yang kuat.
Rasio sinyal terhadap noise rata-rata (SNR)	Anda dapat menggunakan metrik ini untuk memantau rata-rata SNR (signal-to-noise rasio S) dalam durasi waktu yang ditentukan. SNR adalah pengukuran yang menunjukkan apakah sinyal yang diterima cukup kuat dibandingkan dengan tingkat kebisingan untuk koneksi nirkabel yang baik. Nilai SNR positif dan harus lebih besar dari nol untuk menunjukkan bahwa daya sinyal lebih kuat dari daya noise.

Metrik	Deskripsi
Ketersediaan gateway	Anda dapat menggunakan metrik ini untuk mendapatkan informasi tentang ketersediaan gateway ini dalam durasi waktu yang ditentukan. Metrik ini menampilkan waktu koneksi websocket gateway ini untuk durasi waktu tertentu.

ust-in-time Metrik penyediaan J

Metrik	Deskripsi
<code>ProvisionThing.ClientError</code>	Berapa kali perangkat gagal menyediakan karena kesalahan klien. Misalnya, kebijakan yang ditentukan dalam templat tidak ada.
<code>ProvisionThing.ServerError</code>	Berapa kali perangkat gagal menyediakan karena kesalahan server. Pelanggan dapat mencoba lagi untuk menyediakan perangkat setelah menunggu dan mereka dapat menghubungi AWS IoT jika masalahnya tetap sama.
<code>ProvisionThing.Success</code>	Berapa kali perangkat berhasil disediakan.

Metrik pengindeksan armada

AWS IoT metrik pengindeksan armada

Metrik	Deskripsi
<code>NamedShadowCountForDynamicGroupQueryLimitExceeded</code>	Maksimal 25 bayangan bernama per benda diproses untuk istilah kueri yang bukan sumber data spesifik dalam grup benda dinamis. Ketika batas ini dilanggar untuk suatu hal, jenis <code>NamedShadowCountForDynamicGroupQueryLimitExceeded</code> acara akan dipancarkan.

Dimensi untuk metrik

Metrik menggunakan namespace dan menyediakan metrik untuk dimensi berikut

Dimensi	Deskripsi
ActionType	Jenis tindakan yang ditentukan oleh aturan yang memicu permintaan.
BehaviorName	Nama Device Defender Deteksi perilaku profil keamanan yang sedang dipantau.
ClaimCertificateId	Klaim yang digunakan untuk menyediakan perangkat .certificateId
CheckName	Nama pemeriksaan audit Device Defender yang hasilnya sedang dipantau.
JobId	ID pekerjaan yang kemajuan/kegagalan koneksi pesannya sedang dipantau.
Protocol	Protokol yang digunakan untuk membuat permintaan. Nilai yang valid adalah: MQTT atau HTTP
RuleName	Nama aturan dipicu oleh permintaan.
ScheduledAuditName	Nama audit terjadwal Device Defender yang hasil pemeriksaannya sedang dipantau. Ini memiliki nilai OnDemand jika hasil yang dilaporkan adalah untuk audit yang dilakukan sesuai permintaan.
SecurityProfileName	Nama profil keamanan Device Defender Detect yang perilakunya sedang dipantau.
TemplateName	Nama template penyediaan.
SourceArn	Mengacu pada profil keamanan untuk mendeteksi atau akun arn untuk audit.

Dimensi	Deskripsi
RoleArn	Mengacu pada peran Device Defender mencoba untuk mengambil alih.
TopicArn	Mengacu pada topik SNS Device Defender mencoba untuk mempublikasikan ke.
Error	<p>Memberikan deskripsi singkat tentang Kesalahan yang diterima saat mencoba mempublikasikan ke topik SNS. Kemungkinan nilainya adalah:</p> <ul style="list-style-type: none"> • “KMS KeyNot Found”: menunjukkan kunci KMS tidak ada untuk topik tersebut. • “InvalidTopicName”: menunjukkan Topik SNS tidak valid. • “KMS AccessDenied “: menunjukkan bahwa peran tidak memiliki izin untuk kunci KMS untuk Topik. • “AuthorizationError“: menunjukkan bahwa peran yang diberikan tidak mengizinkan Device Defender untuk mempublikasikan ke topik SNS. • “SNS TopicNot Found”: menunjukkan topik SNS yang disediakan tidak ada. • “FailureToAssumeRole“: menunjukkan bahwa peran yang diberikan tidak mengizinkan Device Defender untuk mengambil peran tersebut. • “CrossRegionSNSTOPic“: menunjukkan bahwa topik SNS ada di wilayah yang berbeda.

Monitor AWS IoT menggunakan CloudWatch Log

Saat [AWS IoT logging diaktifkan](#), AWS IoT kirimkan peristiwa kemajuan tentang setiap pesan saat diteruskan dari perangkat Anda melalui broker pesan dan mesin aturan. Di [CloudWatch konsol](#), CloudWatch log muncul di grup log bernama AWSIoTLogs.

Untuk informasi selengkapnya tentang CloudWatch Log, lihat [CloudWatch Log](#). Untuk informasi tentang AWS IoT CloudWatch Log yang didukung, lihat [CloudWatch Entri AWS IoT log log](#).

Melihat AWS IoT log di CloudWatch konsol

Note

Grup `AWSIoTLogsV2` log tidak terlihat di CloudWatch konsol hingga:

- Anda telah mengaktifkan login AWS IoT. Untuk info selengkapnya tentang cara mengaktifkan login AWS IoT, lihat [Konfigurasi AWS IoT logging](#)
- Beberapa entri log telah ditulis oleh AWS IoT operasi.

Untuk melihat AWS IoT log Anda di CloudWatch konsol

1. Jelajahi <https://console.aws.amazon.com/cloudwatch/>. Pada panel navigasi, pilih Grup log.
2. Di kotak teks Filter, masukkan `AWSIoTLogsV2`, lalu tekan Enter.
3. Klik dua kali grup `AWSIoTLogsV2` log.
4. Pilih Cari Semua. Daftar lengkap AWS IoT log yang dihasilkan untuk akun Anda ditampilkan.
5. Pilih ikon perluas untuk melihat aliran individual.

Anda juga dapat memasukkan kueri di kotak teks Filter peristiwa. Berikut adalah beberapa pertanyaan menarik untuk dicoba:

- `{ $.logLevel = "INFO" }`

Temukan semua log yang memiliki tingkat log `INFO`.

- `{ $.status = "Success" }`

Temukan semua log yang memiliki status `Success`.

- `{ $.status = "Success" && $.eventType = "GetThingShadow" }`

Temukan semua log yang memiliki status `Success` dan jenis acara `GetThingShadow`.

Untuk informasi selengkapnya tentang membuat ekspresi filter, lihat [Kueri CloudWatch Log](#).

CloudWatch Entri AWS IoT log log

Setiap komponen AWS IoT menghasilkan entri lognya sendiri. Setiap entri log memiliki event Type yang menentukan operasi yang menyebabkan entri log dihasilkan. Bagian ini menjelaskan entri log yang dihasilkan oleh AWS IoT komponen berikut.

Topik

- [Entri log broker pesan](#)
- [Sertifikat server entri log OCSP](#)
- [Entri log Device Shadow](#)
- [Aturan entri log mesin](#)
- [Entri log pekerjaan](#)
- [Entri log penyediaan perangkat](#)
- [Entri log grup hal dinamis](#)
- [Entri log pengindeksan armada](#)
- [Atribut CloudWatch Log Umum](#)

Entri log broker pesan

Broker AWS IoT pesan menghasilkan entri log untuk peristiwa berikut:

Topik

- [Connect entri log](#)
- [Putuskan entri log](#)
- [GetRetainedMessage entri log](#)
- [ListRetainedMessage entri log](#)
- [Publikasikan entri log](#)
- [Entri log Publikasi-Keluar](#)
- [Entri log antrian](#)
- [Berlangganan entri log](#)

Connect entri log

Broker AWS IoT pesan menghasilkan entri log dengan eventType Connect kapan klien MQTT terhubung.

Connect contoh entri log

```
{
  "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
}
```

Selain [Atribut CloudWatch Log Umum](#), entri Connect log berisi atribut berikut:

clientId

ID klien yang membuat permintaan.

principalId

ID kepala sekolah yang membuat permintaan.

protocol

Protokol yang digunakan untuk membuat permintaan. Nilai-nilai yang valid adalah MQTT atau HTTP.

sourceIp

Alamat IP tempat permintaan berasal.

sourcePort

Port tempat permintaan berasal.

Putuskan entri log

Broker AWS IoT pesan menghasilkan entri log dengan Disconnect kapan eventType klien MQTT terputus.

Putuskan contoh entri log

```
{
  "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"
}
```

Selain [Atribut CloudWatch Log Umum](#), entri Disconnect log berisi atribut berikut:

clientId

ID klien yang membuat permintaan.

principalId

ID kepala sekolah yang membuat permintaan.

protokol

Protokol yang digunakan untuk membuat permintaan. Nilai-nilai yang valid adalah MQTT atau HTTP.

sourceIp

Alamat IP tempat permintaan berasal.

sourcePort

Port tempat permintaan berasal.

akal budi

Alasan mengapa klien terputus.

detail

Penjelasan singkat tentang kesalahan tersebut.

disconnectReason

Alasan mengapa klien terputus.

GetRetainedMessage entri log

Broker AWS IoT pesan menghasilkan entri log dengan eventType GetRetainedMessage kapan [GetRetainedMessage](#) dipanggil.

GetRetainedMessage contoh entri log

```
{
  "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"
}
```

Selain [Atribut CloudWatch Log Umum](#), entri GetRetainedMessage log berisi atribut berikut:

terakhir ModifiedDate

Tanggal dan waktu Epoch, dalam milidetik, ketika pesan yang disimpan disimpan oleh. AWS IoT protokol

Protokol yang digunakan untuk membuat permintaan. Nilai valid: HTTP.

qos

Tingkat Kualitas Layanan (QoS) yang digunakan dalam permintaan publikasi. Nilai-nilai yang valid adalah 0 atau 1.

topicName

Nama topik berlangganan.

ListRetainedMessage entri log

Broker AWS IoT pesan menghasilkan entri log dengan eventType ListRetainedMessage kapan [ListRetainedMessages](#) dipanggil.

ListRetainedMessage contoh entri log

```
{
  "timestamp": "2017-08-07 18:47:56.664",
  "logLevel": "INFO",
  "traceId": "1a60d02e-15b9-605b-7096-a9f584a6ad3f",
  "accountId": "123456789012",
  "status": "Success",
  "eventType": "ListRetainedMessage",
  "protocol": "HTTP"
}
```

Selain [Atribut CloudWatch Log Umum](#), entri ListRetainedMessage log berisi atribut berikut:

protokol

Protokol yang digunakan untuk membuat permintaan. Nilai valid: HTTP.

Publikasikan entri log

Ketika broker AWS IoT pesan menerima pesan MQTT, itu menghasilkan entri log dengan eventType Publish-In

Contoh entri log 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"
}
```

Selain [Atribut CloudWatch Log Umum](#), entri Publish-In log berisi atribut berikut:

clientId

ID klien yang membuat permintaan.

principalId

ID kepala sekolah yang membuat permintaan.

protokol

Protokol yang digunakan untuk membuat permintaan. Nilai-nilai yang valid adalah MQTT atau HTTP.

melestarikan

Atribut yang digunakan saat pesan memiliki flag RETAIN yang disetel dengan nilai `True`. Jika pesan tidak memiliki tanda RETAIN disetel, atribut ini tidak muncul di entri log. Untuk informasi selengkapnya, lihat [Pesan yang disimpan MQTT](#).

sourceIp

Alamat IP tempat permintaan berasal.

sourcePort

Port tempat permintaan berasal.

topicName

Nama topik berlangganan.

Entri log Publikasi-Keluar

Ketika broker pesan menerbitkan pesan MQTT, itu menghasilkan entri log dengan eventType Publish-Out

Contoh entri log 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
}
```

Selain [Atribut CloudWatch Log Umum](#), entri Publish-Out log berisi atribut berikut:

clientId

ID klien berlangganan yang menerima pesan tentang topik MQTT tersebut.

principalId

ID kepala sekolah yang membuat permintaan.

protokol

Protokol yang digunakan untuk membuat permintaan. Nilai-nilai yang valid adalah MQTT atau HTTP.

sourceIp

Alamat IP tempat permintaan berasal.

sourcePort

Port tempat permintaan berasal.

topicName

Nama topik berlangganan.

Entri log antrian

Ketika perangkat dengan sesi persisten terputus, broker pesan MQTT menyimpan pesan perangkat dan AWS IoT menghasilkan entri log dengan EventType. Queued Untuk informasi lebih lanjut tentang sesi persisten MQTT, lihat. [Sesi persisten MQTT](#)

Contoh entri log kesalahan server antrian

```
{
  "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"
}
```

Selain [Atribut CloudWatch Log Umum](#), entri log kesalahan Queued server berisi atribut berikut:

clientId

ID klien tempat pesan diantrian.

detail

Server Error

Kesalahan server mencegah pesan disimpan.

protokol

Protokol yang digunakan untuk membuat permintaan. Nilainya akan selalu begituMQTT.

qos

Tingkat Quality of Service (QoS) permintaan. Nilai akan selalu 1 karena pesan dengan QoS 0 tidak disimpan.

topicName

Nama topik berlangganan.

Contoh entri log sukses yang diantrian

```
{
  "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"
}
```

Selain [Atribut CloudWatch Log Umum](#), entri log Queued sukses berisi atribut berikut:

clientId

ID klien tempat pesan diantrian.

protokol

Protokol yang digunakan untuk membuat permintaan. Nilainya akan selalu begitu MQTT.

qos

Tingkat Quality of Service (QoS) permintaan. Nilai akan selalu 1 karena pesan dengan QoS 0 tidak disimpan.

topicName

Nama topik berlangganan.

Contoh entri log terhambat antrian

```
{
  "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"
}
```

Selain Queued entri log yang dibatasi berisi atribut berikut: [Atribut CloudWatch Log Umum](#)

clientId

ID klien tempat pesan diantrian.

detail

Throttled while queueing offline message

Klien melebihi [Queued messages per second per account](#) batas, sehingga pesan tidak disimpan.

protokol

Protokol yang digunakan untuk membuat permintaan. Nilainya akan selalu begitu MQTT.

qos

Tingkat Quality of Service (QoS) permintaan. Nilai akan selalu 1 karena pesan dengan QoS 0 tidak disimpan.

topicName

Nama topik berlangganan.

Berlangganan entri log

Broker AWS IoT pesan menghasilkan entri log dengan eventType Subscribe kapan klien MQTT berlangganan suatu topik.

MQTT 3 Berlangganan contoh entri log

```
{
  "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
}
```

Selain [Atribut CloudWatch Log Umum](#), entri Subscribe log berisi atribut berikut:

clientId

ID klien yang membuat permintaan.

principalId

ID kepala sekolah yang membuat permintaan.

protokol

Protokol yang digunakan untuk membuat permintaan. Nilai-nilai yang valid adalah MQTT atau HTTP.

sourceIp

Alamat IP tempat permintaan berasal.

sourcePort

Port tempat permintaan berasal.

topicName

Nama topik berlangganan.

MQTT 5 Berlangganan contoh entri log

```
{
  "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
}
```

Untuk operasi Berlangganan MQTT 5, selain atribut entri log [Berlangganan MQTT 3, entri log MQTT 5 berisi atribut](#) berikut: [Atribut CloudWatch Log Umum](#) Subscribe

langganan

Daftar pemetaan antara topik yang diminta dalam permintaan Berlangganan dan kode alasan MQTT 5 individu. Untuk informasi lebih lanjut, lihat kode [alasan MQTT](#).

Sertifikat server entri log OCSP

AWS IoT Core menghasilkan entri log untuk acara berikut:

Topik

- [Ambil entri log StapleData OCSP](#)

Ambil entri log StapleData OCSP

AWS IoT Core menghasilkan entri log dengan eventType dari RetrieveOCSPStapleData ketika server mengambil data pokok OCSP.

Ambil contoh entri log StapleData OCSP

Berikut ini adalah contoh entri log dariSuccess.

```
{
  "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"
}
}

```

Berikut ini adalah contoh entri log dari `Failure`.

```

{
  "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:"
  }
}

```

Untuk `RetrieveOCSPStaple` operasi, selain [Atribut CloudWatch Log Umum](#), entri log berisi atribut berikut:

akal budi

Alasan mengapa operasi gagal.

domain ConfigName

Nama konfigurasi domain Anda.

Detail Koneksi

Penjelasan singkat tentang detail koneksi.

- `http StatusCode`

Kode status HTTP yang dikembalikan oleh responder OCSP sebagai tanggapan atas permintaan klien yang dibuat ke server.

- `ocsp ResponderUri`

URI responder OCSP yang AWS IoT Core mengambil dari sertifikat server.

- `sourceIp`

Alamat IP sumber AWS IoT Core server.

- `TargetIp`

Alamat IP target responden OCSP.

`ocsp RequestDetails`

Detail permintaan OCSP.

- `requesterName`

Identifer untuk AWS IoT Core server yang mengirimkan permintaan ke responder OCSP.

- `permintaan CertId`

ID sertifikat permintaan. Ini adalah ID sertifikat yang meminta respons OCSP.

`ocsp ResponseDetails`

Detail respons OCSP.

- `respon CertId`

ID sertifikat dari respons OCSP.

- `ocsp ResponseStatus`

Status respons OCSP.

- `SertStatus`

Status sertifikat.

- `tanda tangan`

Tanda tangan yang diterapkan pada respons oleh entitas terpercaya.

- `ini UpdateTime`

Waktu di mana status yang ditunjukkan diketahui benar.

- berikutnya UpdateTime

Waktu pada atau sebelum informasi yang lebih baru akan tersedia tentang status sertifikat.

- diproduksi AtTime

Waktu di mana responden OCSP menandatangani tanggapan ini.

- Ukuran dijepit DataPayload

Ukuran muatan dari data yang dijepit.

Entri log Device Shadow

Layanan AWS IoT Device Shadow menghasilkan entri log untuk peristiwa berikut:

Topik

- [DeleteThingShadow entri log](#)
- [GetThingShadow entri log](#)
- [UpdateThingShadow entri log](#)

DeleteThingShadow entri log

Layanan Device Shadow menghasilkan entri log dengan nilai eventType dari DeleteThingShadow saat permintaan untuk menghapus bayangan perangkat diterima.

DeleteThingShadow contoh entri log

```
{
  "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"
}
```

Selain [Atribut CloudWatch Log Umum](#), entri DeleteThingShadow log berisi atribut berikut:

perangkat ShadowName

Nama bayangan untuk diperbarui.

protokol

Protokol yang digunakan untuk membuat permintaan. Nilai-nilai yang valid adalah MQTT atau HTTP.

topicName

Nama topik di mana permintaan itu diterbitkan.

GetThingShadow entri log

Layanan Device Shadow menghasilkan entri log dengan nilai eventType dari GetThingShadow saat permintaan get untuk bayangan diterima.

GetThingShadow contoh entri log

```
{
  "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"
}
```

Selain [Atribut CloudWatch Log Umum](#), entri GetThingShadow log berisi atribut berikut:

perangkat ShadowName

Nama bayangan yang diminta.

protokol

Protokol yang digunakan untuk membuat permintaan. Nilai-nilai yang valid adalah MQTT atau HTTP.

topicName

Nama topik di mana permintaan itu diterbitkan.

UpdateThingShadow entri log

Layanan Device Shadow menghasilkan entri log dengan nilai eventType dari UpdateThingShadow saat permintaan untuk memperbarui bayangan perangkat diterima.

UpdateThingShadow contoh entri log

```
{
  "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"
}
```

Selain [Atribut CloudWatch Log Umum](#), entri UpdateThingShadow log berisi atribut berikut:

perangkat ShadowName

Nama bayangan untuk diperbarui.

protokol

Protokol yang digunakan untuk membuat permintaan. Nilai-nilai yang valid adalah MQTT atau HTTP.

topicName

Nama topik di mana permintaan itu diterbitkan.

Aturan entri log mesin

Mesin AWS IoT aturan menghasilkan log untuk peristiwa berikut:

Topik

- [FunctionExecution entri log](#)
- [RuleExecution entri log](#)
- [RuleMatch entri log](#)
- [RuleExecutionThrottled entri log](#)
- [RuleNotFound entri log](#)
- [StartingRuleExecution entri log](#)

FunctionExecution entri log

Mesin aturan menghasilkan entri log dengan FunctionExecution ketika eventType kueri SQL aturan memanggil fungsi eksternal. Fungsi eksternal dipanggil ketika tindakan aturan membuat permintaan HTTP ke AWS IoT atau layanan web lain (misalnya, memanggil `get_thing_shadow` atau `machinelearning_predict`).

FunctionExecution contoh entri log

```
{
  "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"
}
```

Selain [Atribut CloudWatch Log Umum](#), entri FunctionExecution log berisi atribut berikut:

clientId

N/A untuk FunctionExecution log.

principalId

ID kepala sekolah yang membuat permintaan.

sumber daya

Kumpulan sumber daya yang digunakan oleh tindakan aturan.

ruleName

Nama aturan pencocokan.

topicName

Nama topik berlangganan.

RuleExecution entri log

Ketika mesin AWS IoT aturan memicu tindakan aturan, itu menghasilkan entri `RuleExecution` log.

RuleExecution contoh entri log

```
{
  "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"
}
```

Selain [Atribut CloudWatch Log Umum](#), entri `RuleExecution` log berisi atribut berikut:

clientId

ID klien yang membuat permintaan.

principalId

ID kepala sekolah yang membuat permintaan.

sumber daya

Kumpulan sumber daya yang digunakan oleh tindakan aturan.

ruleAction

Nama tindakan dipicu.

ruleName

Nama aturan pencocokan.

topicName

Nama topik berlangganan.

RuleMatch entri log

Mesin AWS IoT aturan menghasilkan entri log dengan event Type RuleMatch ketika broker pesan menerima pesan yang cocok dengan aturan.

RuleMatch contoh entri log

```
{
  "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"
}
```

Selain [Atribut CloudWatch Log Umum](#), entri RuleMatch log berisi atribut berikut:

clientId

ID klien yang membuat permintaan.

principalId

ID kepala sekolah yang membuat permintaan.

ruleName

Nama aturan pencocokan.

topicName

Nama topik berlangganan.

RuleExecutionThrottled entri log

Ketika eksekusi dibatasi, mesin AWS IoT aturan menghasilkan entri log dengan `darieventType`.

RuleExecutionThrottled

RuleExecutionThrottled contoh entri log

```
{
  "timestamp": "2017-10-04 19:25:46.070",
  "logLevel": "ERROR",
  "traceId": "30aa7ccc-1d23-0b97-aa7b-76196d83537e",
  "accountId": "123456789012",
  "status": "Failure",
  "eventType": "RuleMessageThrottled",
  "clientId": "abf27092886e49a8a5c1922749736453",
  "topicName": "$aws/rules/example_rule",
  "ruleName": "example_rule",
  "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
  "reason": "RuleExecutionThrottled",
  "details": "Exection of Rule example_rule throttled"
}
```

Selain [Atribut CloudWatch Log Umum](#), entri `RuleExecutionThrottled` log berisi atribut berikut:

clientId

ID klien yang membuat permintaan.

detail

Penjelasan singkat tentang kesalahan tersebut.

principalId

ID kepala sekolah yang membuat permintaan.

akal budi

Tali "RuleExecutionDilapisi".

ruleName

Nama aturan yang akan dipicu.

topicName

Nama topik yang dipublikasikan.

RuleNotFound entri log

Ketika mesin AWS IoT aturan tidak dapat menemukan aturan dengan nama tertentu, itu menghasilkan entri log dengan eventType dariRuleNotFound.

RuleNotFound contoh entri log

```
{
  "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"
}
```

Selain [Atribut CloudWatch Log Umum](#), entri RuleNotFound log berisi atribut berikut:

clientId

ID klien yang membuat permintaan.

detail

Penjelasan singkat tentang kesalahan tersebut.

principalId

ID kepala sekolah yang membuat permintaan.

akal budi

String "RuleNotDitemukan".

ruleName

Nama aturan yang tidak dapat ditemukan.

topicName

Nama topik yang dipublikasikan.

StartingRuleExecution entri log

Ketika mesin AWS IoT aturan mulai memicu tindakan aturan, itu menghasilkan entri log dengan `eventType` dari `StartingRuleExecution`.

StartingRuleExecution contoh entri log

```
{
  "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"
}
```

Selain [Atribut CloudWatch Log Umum](#), entri `rule- log` berisi atribut berikut:

clientId

ID klien yang membuat permintaan.

principalId

ID kepala sekolah yang membuat permintaan.

ruleAction

Nama tindakan dipicu.

ruleName

Nama aturan pencocokan.

topicName

Nama topik berlangganan.

Entri log pekerjaan

Layanan AWS IoT Job menghasilkan entri log untuk peristiwa berikut. Entri log dihasilkan ketika permintaan MQTT atau HTTP diterima dari perangkat.

Topik

- [DescribeJobExecution entri log](#)
- [GetPendingJobExecution entri log](#)
- [ReportFinalJobExecutionCount entri log](#)
- [StartNextPendingJobExecution entri log](#)
- [UpdateJobExecution entri log](#)

DescribeJobExecution entri log

Layanan AWS IoT Jobs menghasilkan entri log dengan eventType `DescribeJobExecution` ketika layanan menerima permintaan untuk menggambarkan pelaksanaan pekerjaan.

DescribeJobExecution contoh entri log

```
{
  "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."
}
```

Selain [Atribut CloudWatch Log Umum](#), entri `GetJobExecution` log berisi atribut berikut:

`clientId`

ID klien yang membuat permintaan.

`clientToken`

Pengidentifikasi unik dan peka huruf besar/kecil untuk memastikan idempotensi permintaan.

Untuk informasi lebih lanjut, lihat [Cara Memastikan Idempotensi](#).

`detail`

Informasi lain dari layanan Jobs.

`jobId`

ID pekerjaan untuk eksekusi pekerjaan.

`protokol`

Protokol yang digunakan untuk membuat permintaan. Nilai-nilai yang valid adalah MQTT atau HTTP.

`topicName`

Topik yang digunakan untuk membuat permintaan.

GetPendingJobExecution entri log

Layanan AWS IoT Jobs menghasilkan entri log dengan `eventType` `GetPendingJobExecution` ketika layanan menerima permintaan eksekusi pekerjaan.

GetPendingJobExecution contoh entri log

```
{
  "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."
}
```

Selain [Atribut CloudWatch Log Umum](#), entri `GetPendingJobExecution` log berisi atribut berikut:

`clientId`

ID klien yang membuat permintaan.

`clientToken`

Pengidentifikasi unik dan peka huruf besar/kecil untuk memastikan idempotensi permintaan. Untuk informasi lebih lanjut, lihat [Cara Memastikan Idempotensi](#).

`detail`

Informasi lain dari layanan Jobs.

`protokol`

Protokol yang digunakan untuk membuat permintaan. Nilai-nilai yang valid adalah MQTT atau HTTP.

`topicName`

Nama topik berlangganan.

`ReportFinalJobExecutionCount` entri log

Layanan AWS IoT Jobs menghasilkan entri log dengan `ReportFinalJobExecutionCount` kapan pekerjaan selesai. `entryType`

`ReportFinalJobExecutionCount` contoh entri log

```
{
  "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"
  }

```

Selain [Atribut CloudWatch Log Umum](#), entri `ReportFinalJobExecutionCount` log berisi atribut berikut:

detail

Informasi lain dari layanan Jobs.

jobId

ID pekerjaan untuk eksekusi pekerjaan.

StartNextPendingJobExecution entri log

Ketika menerima permintaan untuk memulai eksekusi pekerjaan tertunda berikutnya, layanan AWS IoT Jobs menghasilkan entri log dengan `eventType` dari `StartNextPendingJobExecution`.

StartNextPendingJobExecution contoh entri log

```

{
  "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."
}

```

Selain [Atribut CloudWatch Log Umum](#), entri `StartNextPendingJobExecution` log berisi atribut berikut:

clientId

ID klien yang membuat permintaan.

clientToken

Pengidentifikasi unik dan peka huruf besar/kecil untuk memastikan idempotensi permintaan.

Untuk informasi lebih lanjut, lihat [Cara Memastikan Idempotensi](#).

detail

Informasi lain dari layanan Jobs.

protokol

Protokol yang digunakan untuk membuat permintaan. Nilai-nilai yang valid adalah MQTT atau HTTP.

topicName

Topik yang digunakan untuk membuat permintaan.

UpdateJobExecution entri log

Layanan AWS IoT Jobs menghasilkan entri log dengan eventType UpdateJobExecution ketika layanan menerima permintaan untuk memperbarui eksekusi pekerjaan.

UpdateJobExecution contoh entri log

```
{
  "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."
}
```

Selain [Atribut CloudWatch Log Umum](#), entri UpdateJobExecution log berisi atribut berikut:

clientId

ID klien yang membuat permintaan.

clientToken

Pengidentifikasi unik dan peka huruf besar/kecil untuk memastikan idempotensi permintaan. Untuk informasi lebih lanjut, lihat [Cara Memastikan Idempotensi](#).

detail

Informasi lain dari layanan Jobs.

jobId

ID pekerjaan untuk eksekusi pekerjaan.

protokol

Protokol yang digunakan untuk membuat permintaan. Nilai-nilai yang valid adalah MQTT atau HTTP.

topicName

Topik yang digunakan untuk membuat permintaan.

versionNumber

Versi eksekusi pekerjaan.

Entri log penyediaan perangkat

Layanan Penyediaan AWS IoT Perangkat menghasilkan log untuk peristiwa berikut.

Topik

- [GetDeviceCredentials entri log](#)
- [ProvisionDevice entri log](#)

GetDeviceCredentials entri log

Layanan Penyediaan AWS IoT Perangkat menghasilkan entri log dengan eventType jumlah GetDeviceCredential saat klien memanggil GetDeviceCredential

GetDeviceContoh entri log kredensial

```
{
  "timestamp" : "2019-02-20 20:31:22.932",
  "logLevel" : "INFO",
  "traceId" : "8d9c016f-6cc7-441e-8909-7ee3d5563405",
  "accountId" : "123456789101",
  "status" : "Success",
  "eventType" : "GetDeviceCredentials",
  "deviceCertificateId" :
  "e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855",
  "details" : "Additional details about this log."
}
```

Selain [Atribut CloudWatch Log Umum](#), entri GetDeviceCredentials log berisi atribut berikut:

detail

Penjelasan singkat tentang kesalahan tersebut.

perangkat CertificateId

ID sertifikat perangkat.

ProvisionDevice entri log

Layanan Penyediaan AWS IoT Perangkat menghasilkan entri log dengan eventType jumlah ProvisionDevice saat klien memanggil ProvisionDevice

ProvisionDevice contoh entri log

```
{
  "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" :
  "e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855",
  "details" : "Additional details about this log."
}
```



```
}
```

Selain [Atribut CloudWatch Log Umum](#), entri ProvisionDevice log berisi atribut berikut:

detail

Penjelasan singkat tentang kesalahan tersebut.

perangkat CertificateId

ID sertifikat perangkat.

penyediaan TemplateName

Nama template penyediaan.

Entri log grup hal dinamis

AWS IoT Dynamic Thing Groups menghasilkan log untuk acara berikut.

Topik

- [AddThingToDynamicThingGroupsFailed entri log](#)

AddThingToDynamicThingGroupsFailed entri log

AWS IoT Ketika tidak dapat menambahkan sesuatu ke grup dinamis yang ditentukan, itu menghasilkan entri log dengan eventType dariAddThingToDynamicThingGroupsFailed. Ini terjadi ketika sesuatu memenuhi kriteria untuk berada dalam kelompok hal dinamis; Namun, itu tidak dapat ditambahkan ke grup dinamis atau dihapus dari grup dinamis. Ini bisa terjadi karena:

- Benda itu sudah termasuk dalam jumlah kelompok maksimum.
- --override-dynamic-groupsOpsi ini digunakan untuk menambahkan benda ke grup benda statis. Itu dihapus dari kelompok hal yang dinamis untuk memungkinkan hal itu.

Untuk informasi selengkapnya, lihat [Batasan dan Konflik Grup Hal Dinamis](#).

AddThingToDynamicThingGroupsFailed contoh entri log

Contoh ini menunjukkan entri log AddThingToDynamicThingGroupsFailed kesalahan.

Dalam contoh ini, TestThingmemenuhi kriteria untuk berada dalam kelompok benda dinamis yang

tercantum `dynamicThingGroupNames`, tetapi tidak dapat ditambahkan ke grup dinamis tersebut, seperti yang dijelaskan dalam `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."
}
```

Selain [Atribut CloudWatch Log Umum](#), entri `AddThingToDynamicThingGroupsFailed` log berisi atribut berikut:

ThingGroupName dinamis

Sebuah array dari kelompok benda dinamis yang benda itu tidak dapat ditambahkan.

akal budi

Alasan mengapa hal itu tidak dapat ditambahkan ke kelompok benda dinamis.

thingName

Nama benda yang tidak bisa ditambahkan ke grup benda dinamis.

Entri log pengindeksan armada

AWS IoT pengindeksan armada menghasilkan entri log untuk peristiwa berikut.

Topik

- [NamedShadowCountForDynamicGroupQueryLimitExceeded entri log](#)

NamedShadowCountForDynamicGroupQueryLimitExceeded entri log

Maksimal 25 bayangan bernama per benda diproses untuk istilah kueri yang bukan sumber data spesifik dalam grup dinamis. Ketika batas ini dilanggar untuk suatu hal, jenis NamedShadowCountForDynamicGroupQueryLimitExceeded acara akan dipancarkan.

NamedShadowCountForDynamicGroupQueryLimitExceeded contoh entri log

Contoh ini menunjukkan entri log NamedShadowCountForDynamicGroupQueryLimitExceeded kesalahan. Dalam contoh ini, DynamicGroup hasil berdasarkan semua nilai bisa tidak akurat, seperti yang dijelaskan di lapangan. 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."
}
```

Atribut CloudWatch Log Umum

Semua entri CloudWatch log Log menyertakan atribut ini:

accountId

Akun AWS ID Anda.

eventType

Jenis peristiwa yang log dihasilkan. Nilai jenis acara tergantung pada peristiwa yang menghasilkan entri log. Setiap deskripsi entri log mencakup nilai eventType untuk entri log tersebut.

logLevel

Tingkat log yang digunakan. Untuk informasi selengkapnya, lihat [the section called “Tingkat Log”](#).

status

Status HTTP dari permintaan.

timestamp

Stempel waktu UTC yang dapat dibaca manusia saat klien terhubung ke broker pesan. AWS IoT

traceld

Pengidentifikasi yang dibuat secara acak yang dapat digunakan untuk mengkorelasikan semua log untuk permintaan tertentu.

Unggah log sisi perangkat ke Amazon CloudWatch

Anda dapat mengunggah log historis sisi perangkat ke Amazon CloudWatch untuk memantau dan menganalisis aktivitas perangkat di lapangan. Log sisi perangkat dapat mencakup file log sistem, aplikasi, dan perangkat. [Proses ini menggunakan parameter tindakan aturan CloudWatch Log untuk mempublikasikan log sisi perangkat ke dalam grup log yang ditentukan pelanggan.](#)

Cara kerjanya

Proses dimulai ketika AWS IoT perangkat mengirim pesan MQTT yang berisi file log diformat ke topik. AWS IoT Aturan memantau topik pesan dan mengirimkan file log ke grup CloudWatch Log yang Anda tentukan. Anda kemudian dapat meninjau dan menganalisis informasi.

Topik

- [Topik MQTT](#)
- [Tindakan aturan](#)

Topik MQTT

Pilih ruang nama topik MQTT yang akan Anda gunakan untuk mempublikasikan log. Sebaiknya gunakan format ini untuk ruang topik umum, `$aws/rules/things/thing_name/logs`, dan format ini untuk topik kesalahan, `$aws/rules/things/thing_name/logs/errors`. Struktur penamaan untuk log dan topik kesalahan direkomendasikan, tetapi tidak diperlukan. Untuk informasi selengkapnya, lihat [Merancang Topik MQTT](#) untuk AWS IoT Core

Dengan menggunakan ruang topik umum yang direkomendasikan, Anda menggunakan topik yang dicadangkan AWS IoT Basic Ingest. AWS IoT Basic Ingest mengirim data perangkat dengan aman ke AWS layanan yang didukung oleh tindakan AWS IoT aturan. Ini menghapus broker pesan terbitkan/berlangganan dari jalur konsumsi, membuatnya lebih hemat biaya. Untuk informasi selengkapnya, lihat [Mengurangi biaya pengiriman pesan dengan Basic Ingest](#).

Jika Anda menggunakan BatchMode untuk mengunggah file log, pesan Anda harus mengikuti format tertentu yang menyertakan stempel waktu dan pesan UNIX. [Untuk informasi selengkapnya, lihat persyaratan format pesan MQTT untuk topik BatchMode dalam tindakan aturan Log. CloudWatch](#)

Tindakan aturan

Saat AWS IoT menerima pesan MQTT dari perangkat klien, AWS IoT aturan memantau topik yang ditentukan pelanggan dan memublikasikan konten ke dalam grup log yang Anda tentukan. CloudWatch Proses ini menggunakan tindakan aturan CloudWatch Log untuk memantau MQTT untuk kumpulan file log. Untuk informasi selengkapnya, lihat tindakan AWS IoT aturan [CloudWatch Log](#).

Mode Batch

`batchMode` adalah parameter Boolean dalam tindakan aturan AWS IoT CloudWatch Log. Parameter ini opsional dan off (`false`) secara default. Untuk mengunggah file log sisi perangkat dalam batch, Anda harus mengaktifkan parameter ini (`true`) saat membuat aturan. AWS IoT Untuk informasi selengkapnya, lihat [CloudWatch Log](#) di bagian [tindakan AWS IoT aturan](#).

Mengunggah log sisi perangkat dengan menggunakan aturan AWS IoT

Anda dapat menggunakan mesin AWS IoT aturan untuk mengunggah catatan log dari file log sisi perangkat yang ada (log sistem, aplikasi, dan perangkat-klien) ke Amazon. CloudWatch Saat log sisi perangkat dipublikasikan ke topik MQTT, tindakan aturan Log akan mentransfer pesan ke CloudWatch Log. CloudWatch Proses ini menguraikan cara mengunggah log perangkat dalam batch menggunakan `batchMode` parameter tindakan aturan yang diaktifkan (disetel ke `true`).

Untuk mulai mengunggah log sisi perangkat CloudWatch, lengkapi prasyarat berikut.

Prasyarat

Sebelum memulai, lakukan hal berikut:

- Buat setidaknya satu perangkat IoT target yang terdaftar AWS IoT Core sebagai sesuatu AWS IoT . Untuk informasi selengkapnya, lihat [Membuat objek benda](#).
- Tentukan ruang topik MQTT untuk konsumsi dan kesalahan. [Untuk informasi selengkapnya tentang topik MQTT dan konvensi penamaan yang direkomendasikan, lihat bagian topik `MQTTMQTT` di Unggah log sisi perangkat ke Amazon. CloudWatch](#)

Untuk informasi selengkapnya tentang prasyarat ini, lihat [Mengunggah](#) log sisi perangkat ke CloudWatch

Membuat grup CloudWatch log

Untuk membuat grup CloudWatch log, selesaikan langkah-langkah berikut. Pilih tab yang sesuai tergantung pada apakah Anda lebih suka melakukan langkah-langkah melalui AWS Management Console atau AWS Command Line Interface (AWS CLI).

AWS Management Console

Untuk membuat grup CloudWatch log dengan menggunakan AWS Management Console

1. Buka AWS Management Console dan navigasikan ke [CloudWatch](#).
2. Pada bilah navigasi, pilih Log, lalu Log grup.
3. Pilih Buat grup log.
4. Perbarui nama grup Log dan, secara opsional, perbarui bidang pengaturan Retensi.
5. Pilih Buat.

AWS CLI

Untuk membuat grup CloudWatch log dengan menggunakan AWS CLI

1. Untuk membuat grup log, jalankan perintah berikut. Untuk informasi selengkapnya, lihat [create-log-group](#) di Referensi Perintah AWS CLI v2.

Ganti nama grup log di contoh (`uploadLogsGroup`) dengan nama pilihan Anda.

```
aws logs create-log-group --log-group-name uploadLogsGroup
```

2. Untuk mengonfirmasi bahwa grup log dibuat dengan benar, jalankan perintah berikut.

```
aws logs describe-log-groups --log-group-name-prefix uploadLogsGroup
```

Contoh output:

```
{
  "logGroups": [
    {
```

```
        "logGroupName": "uploadLogsGroup",
        "creationTime": 1674521804657,
        "metricFilterCount": 0,
        "arn": "arn:aws:logs:us-east-1:111122223333:log-
group:uploadLogsGroup:*",
        "storedBytes": 0
    }
]
}
```

Membuat aturan topik

Untuk membuat AWS IoT aturan, selesaikan langkah-langkah berikut. Pilih tab yang sesuai tergantung pada apakah Anda lebih suka melakukan langkah-langkah melalui AWS Management Console atau AWS Command Line Interface (AWS CLI).

AWS Management Console

Untuk membuat aturan topik dengan menggunakan AWS Management Console

1. Buka hub Aturan.
 - a. Buka AWS Management Console dan navigasikan ke [AWS IoT](#).
 - b. Pada bilah navigasi, pilih Perutean pesan dan kemudian Aturan.
 - c. Pilih Buat aturan.
2. Masukkan properti aturan.
 - a. Masukkan nama Aturan alfanumerik.
 - b. (Opsional) Masukkan deskripsi Aturan dan Tag.
 - c. Pilih Selanjutnya.
3. Masukkan pernyataan SQL.
 - a. Masukkan pernyataan SQL menggunakan topik MQTT yang Anda tetapkan untuk konsumsi.

Misalnya, `SELECT * FROM '$aws/rules/things/thing_name/logs'`
 - b. Pilih Selanjutnya.
4. Masukkan tindakan aturan.

- a. Pada menu Action 1, pilih CloudWatchlog.
- b. Pilih nama grup Log lalu pilih grup log yang Anda buat.
- c. Pilih Gunakan mode batch.
- d. Tentukan peran IAM untuk aturan tersebut.

Jika Anda memiliki peran IAM untuk aturan tersebut, lakukan hal berikut.

1. Pada menu peran IAM, pilih peran IAM Anda.

Jika Anda tidak memiliki peran IAM untuk aturan tersebut, lakukan hal berikut.

1. Pilih Buat peran baru.
2. Untuk nama Peran, masukkan nama unik dan pilih Buat.
3. Konfirmasikan bahwa nama peran IAM sudah benar di bidang peran IAM.

- e. Pilih Selanjutnya.

5. Tinjau konfigurasi template.
 - a. Tinjau pengaturan untuk template Job untuk memverifikasi bahwa mereka benar.
 - b. Setelah selesai, pilih Buat.

AWS CLI

Untuk membuat peran IAM dan aturan topik dengan menggunakan AWS CLI

1. Buat peran IAM yang memberikan hak atas aturan. AWS IoT
 - a. Buat kebijakan IAM.

Untuk membuat kebijakan IAM, jalankan perintah berikut. Pastikan Anda memperbarui nilai `policy-name` parameter. Untuk informasi selengkapnya, lihat [create-policy](#) di Referensi Perintah AWS CLI v2.

Note

Jika Anda menggunakan sistem operasi Microsoft Windows, Anda mungkin perlu mengganti penanda akhir baris (`\n`) dengan tanda centang (```) atau karakter lain.


```
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. Salin kebijakan ARN dari output Anda ke editor teks.

Contoh output:

```
{  
  "Policy": {  
    "PolicyName": "uploadLogsPolicy",  
    "PermissionsBoundaryUsageCount": 0,  
    "CreateDate": "2023-01-23T18:30:10Z",  
    "AttachmentCount": 0,  
    "IsAttachable": true,  
    "PolicyId": "AAABBBCCDDDEEEFFFGGG",  
    "DefaultVersionId": "v1",  
    "Path": "/",  
    "Arn": "arn:aws:iam:111122223333:policy/uploadLogsPolicy",  
    "UpdateDate": "2023-01-23T18:30:10Z"  
  }  
}
```

- c. Buat peran IAM dan kebijakan kepercayaan.

Untuk membuat kebijakan IAM, jalankan perintah berikut. Pastikan Anda memperbarui nilai `role-name` parameter. Untuk informasi selengkapnya, lihat [create-roledi](#) Referensi Perintah AWS CLI v2.

```
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. Lampirkan kebijakan IAM ke aturan.

Untuk membuat kebijakan IAM, jalankan perintah berikut. Pastikan Anda memperbarui nilai `role-name` dan `policy-arn` parameter. Untuk informasi selengkapnya, lihat [attach-role-policy](#) di Referensi Perintah AWS CLI v2.

```
aws iam attach-role-policy \  
--role-name uploadLogsRole \  
--policy-arn arn:aws:iam::111122223333:policy/uploadLogsPolicy
```

- e. Tinjau perannya.

Untuk mengonfirmasi bahwa peran IAM dibuat dengan benar, jalankan perintah berikut. Pastikan Anda memperbarui nilai `role-name` parameter. Untuk informasi selengkapnya, lihat [get-roledi](#) di Referensi Perintah AWS CLI v2.

```
aws iam get-role --role-name uploadLogsRole
```

Contoh output:

```

{
  "Role": {
    "Path": "/",
    "RoleName": "uploadLogsRole",
    "RoleId": "AAABBBCCDDDEEEFFFGG",
    "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. Buat aturan AWS IoT topik di AWS CLI.

- a. Untuk membuat aturan AWS IoT topik, jalankan perintah berikut. Pastikan Anda memperbarui nilai `--rule-name`, `sql` pernyataan, `description`, `roleARN`, dan `logGroupName` parameter. Untuk informasi selengkapnya, lihat [create-topic-rule di Referensi Perintah v2](#). AWS CLI

```

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. Untuk mengonfirmasi bahwa aturan dibuat dengan benar, jalankan perintah berikut. Pastikan Anda memperbarui nilai `role-name` parameter. Untuk informasi selengkapnya, lihat [get-topic-rule di Referensi Perintah v2](#). AWS CLI

```
aws iot get-topic-rule --rule-name uploadLogsRule
```

Contoh output:

```
{
  "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"
  }
}
```

Mengirim log sisi perangkat ke AWS IoT

Untuk mengirim log sisi perangkat ke AWS IoT

1. Untuk mengirim log historis ke AWS IoT, berkomunikasi dengan perangkat Anda untuk memastikan hal berikut.
 - Informasi log dikirim ke namespace topik yang benar sebagaimana ditentukan dalam bagian Prasyarat dari prosedur ini.

Misalnya, `$aws/rules/things/thing_name/logs`

- Payload pesan MQTT diformat dengan benar. Untuk informasi lebih lanjut tentang topik MQTT dan konvensi penamaan yang direkomendasikan, lihat bagian di dalamnya. [Topik MQTT](#)
[Unggah log sisi perangkat ke Amazon CloudWatch](#)
2. Konfirmasikan bahwa pesan MQTT diterima dalam klien MQTT. AWS IoT
 - a. Buka AWS Management Console dan navigasikan ke [AWS IoT](#).
 - b. Untuk melihat klien pengujian MQTT, pada bilah navigasi, pilih Uji, klien uji MQTT.
 - c. Untuk Berlangganan topik, Filter topik, masukkan namespace topik.
 - d. Pilih Langganan.

Pesan MQTT muncul di tabel Langganan dan Topik, seperti yang terlihat di bawah ini. Pesan-pesan ini dapat memakan waktu hingga lima menit untuk muncul.

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

Message payload

► **Additional configuration**

Publish

Subscriptions	topic/test/
topic/test/  	<p>▼ topic/test/</p> <pre>[{ "timestamp": 1673520691123, "message": "Test message 1" }, { "timestamp": 1673520692321, "message": "Test message 2" }, { "timestamp": 1673520693322, "message": "Test message 3" }]</pre>

Melihat data log

Untuk meninjau catatan log Anda di CloudWatch Log

1. Buka AWS Management Console, dan arahkan ke [CloudWatch](#).
2. Pada bilah navigasi, pilih Log, Wawasan Log.
3. Pada menu Pilih grup log, pilih grup log yang Anda tentukan dalam AWS IoT aturan.
4. Pada halaman wawasan Log, pilih Jalankan kueri.

Logging panggilan AWS IoT API menggunakan AWS CloudTrail

AWS IoT terintegrasi dengan AWS CloudTrail, layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau AWS layanan di AWS IoT. CloudTrail menangkap semua panggilan API untuk AWS IoT sebagai peristiwa, termasuk panggilan dari AWS IoT konsol dan dari panggilan kode ke AWS IoT API. Jika Anda membuat jejak, Anda dapat mengaktifkan pengiriman CloudTrail acara secara berkelanjutan ke bucket Amazon S3, termasuk acara untuk AWS IoT. Jika Anda tidak mengonfigurasi jejak, Anda masih dapat melihat peristiwa terbaru di CloudTrail konsol dalam Riwayat acara. Dengan menggunakan informasi yang dikumpulkan oleh CloudTrail, Anda dapat menentukan permintaan yang dibuat AWS IoT, alamat IP dari mana permintaan dibuat, siapa yang membuat permintaan, kapan dibuat, dan detail lainnya.

Untuk mempelajari selengkapnya CloudTrail, lihat [Panduan AWS CloudTrail Pengguna](#).

AWS IoT informasi di CloudTrail

CloudTrail diaktifkan pada Akun AWS saat Anda membuat akun. Ketika aktivitas terjadi di AWS IoT, aktivitas tersebut dicatat dalam suatu CloudTrail peristiwa bersama dengan peristiwa AWS layanan lainnya dalam riwayat Acara. Anda dapat melihat, mencari, dan mengunduh acara terbaru di situs Anda Akun AWS. Untuk informasi selengkapnya, lihat [Melihat Acara dengan Riwayat CloudTrail Acara](#).

Untuk catatan berkelanjutan tentang peristiwa di Akun AWS, termasuk peristiwa untuk AWS IoT, buat jejak. Jejak memungkinkan CloudTrail untuk mengirimkan file log ke bucket Amazon S3. Secara default, saat Anda membuat jejak di konsol, jejak berlaku untuk semua Wilayah AWS s. Jejak mencatat peristiwa dari semua Wilayah AWS s di AWS partisi dan mengirimkan file log ke bucket Amazon S3 yang Anda tentukan. Anda dapat mengonfigurasi AWS layanan lain untuk menganalisis lebih lanjut dan menindaklanjuti data peristiwa yang dikumpulkan dalam CloudTrail log. Lihat informasi yang lebih lengkap di:

- [Gambaran Umum untuk Membuat Jejak](#)
- [CloudTrail Layanan dan Integrasi yang Didukung](#)
- [Mengkonfigurasi Notifikasi Amazon SNS untuk CloudTrail](#)
- [Menerima File CloudTrail Log dari Beberapa Wilayah](#) dan [Menerima File CloudTrail Log dari Beberapa Akun](#)

Note

AWS IoT tindakan bidang data (sisi perangkat) tidak dicatat oleh CloudTrail. Gunakan CloudWatch untuk memantau tindakan ini.

Secara umum, tindakan bidang AWS IoT kontrol yang membuat perubahan dicatat oleh CloudTrail. Panggilan seperti `CreateThing`, `CreateKeysAndCertificate`, dan `UpdateCertificate` meninggalkan CloudTrail entri, sementara panggilan seperti `ListThings` dan `ListTopicAttaches` tidak.

Setiap entri peristiwa atau log berisi informasi tentang siapa yang membuat permintaan tersebut. Informasi identitas membantu Anda menentukan hal berikut ini:

- Apakah permintaan tersebut dibuat dengan kredensial root atau pengguna IAM.
- Apakah permintaan tersebut dibuat dengan kredensial keamanan sementara untuk satu peran atau pengguna terfederasi.
- Apakah permintaan itu dibuat oleh AWS layanan lain.

Untuk informasi selengkapnya, lihat Elemen [CloudTrail UserIdentity](#).

AWS IoT tindakan didokumentasikan dalam [Referensi AWS IoT API](#). AWS IoT Tindakan nirkabel didokumentasikan dalam [Referensi API AWS IoT Nirkabel](#).

Memahami entri file AWS IoT log

Trail adalah konfigurasi yang memungkinkan pengiriman peristiwa sebagai file log ke bucket Amazon S3 yang Anda tentukan. CloudTrail file log berisi satu atau lebih entri log. Peristiwa mewakili permintaan tunggal dari sumber mana pun dan mencakup informasi tentang tindakan yang diminta, tanggal dan waktu tindakan, parameter permintaan, dan sebagainya. CloudTrail file log bukanlah jejak tumpukan yang diurutkan dari panggilan API publik, sehingga file tersebut tidak muncul dalam urutan tertentu.

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan AttachPolicy tindakan.

```
{
  "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",
  "RequestParameters": {
    "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":""
}
```

Aturan untuk AWS IoT

Aturan memberi perangkat Anda kemampuan untuk berinteraksi Layanan AWS. Aturan dianalisis dan tindakan dilakukan berdasarkan aliran topik MQTT. Anda dapat menggunakan aturan untuk mendukung tugas-tugas berikut:

- Menambah atau memfilter data yang diterima dari perangkat.
- Menulis data yang diterima dari perangkat ke database Amazon DynamoDB.
- Simpan file ke Amazon S3.
- Kirim pemberitahuan push ke semua pengguna yang menggunakan Amazon SNS.
- Publikasikan data ke antrian Amazon SQS.
- Memanggil fungsi Lambda untuk mengekstrak data.
- Memproses pesan dari sejumlah besar perangkat menggunakan Amazon Kinesis.
- Kirim data ke OpenSearch Layanan Amazon.
- Tangkap CloudWatch metrik.
- Ubah CloudWatch alarm.
- Kirim data dari pesan MQTT ke Amazon SageMaker untuk membuat prediksi berdasarkan model pembelajaran mesin (ML).
- Kirim pesan kepada Salesforce IoT Input Stream.
- Kirim data pesan ke AWS IoT Analytics saluran.
- Mulai proses mesin status Step Functions.
- Kirim data pesan ke AWS IoT Events input.
- Kirim data pesan ke properti aset di AWS IoT SiteWise.
- Kirim data pesan ke aplikasi atau layanan web.

Aturan Anda dapat menggunakan pesan MQTT yang melewati protokol terbitkan/berlangganan yang didukung oleh [the section called “Protokol komunikasi perangkat”](#) [Anda juga dapat menggunakan fitur Basic Ingest untuk mengirim data perangkat dengan aman ke Layanan AWS daftar sebelumnya, tanpa menimbulkan biaya pengiriman pesan.](#) Fitur [Basic Ingest](#) mengoptimalkan aliran data dengan menghapus broker pesan terbitkan/berlangganan dari jalur konsumsi. Ini membuatnya hemat biaya sambil tetap menjaga keamanan dan fitur pemrosesan data AWS IoT.

Sebelum AWS IoT dapat melakukan tindakan ini, Anda harus memberikan izin untuk mengakses AWS sumber daya Anda atas nama Anda. Ketika tindakan dilakukan, Anda dikenakan biaya standar untuk Layanan AWS yang Anda gunakan.

Daftar Isi

- [Memberikan AWS IoT aturan akses yang dibutuhkannya](#)
- [Melewati izin peran](#)
- [Buat aturan](#)
- [Melihat aturan Anda](#)
- [Menghapus aturan](#)
- [AWS IoT tindakan aturan](#)
- [Memecahkan masalah aturan](#)
- [Mengakses sumber daya lintas akun menggunakan aturan AWS IoT](#)
- [Penanganan kesalahan \(tindakan kesalahan\)](#)
- [Mengurangi biaya pengiriman pesan dengan Basic Ingest](#)
- [AWS IoT Referensi SQL](#)

Memberikan AWS IoT aturan akses yang dibutuhkannya

Gunakan peran IAM untuk mengontrol AWS sumber daya yang dapat diakses oleh setiap aturan. Sebelum membuat aturan, Anda harus membuat peran IAM dengan kebijakan yang memungkinkan akses ke AWS sumber daya yang diperlukan. AWS IoT mengasumsikan peran ini saat menerapkan aturan.

Selesaikan langkah-langkah berikut untuk membuat peran dan AWS IoT kebijakan IAM yang memberikan AWS IoT aturan akses yang diperlukan (AWS CLI).

1. Simpan dokumen kebijakan kepercayaan berikut, yang memberikan AWS IoT izin untuk mengambil peran, ke file bernama `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"
        }
    }
}
]
}

```

Gunakan perintah [create-role](#) untuk membuat peran IAM yang menentukan file: `iot-role-trust.json`

```
aws iam create-role --role-name my-iot-role --assume-role-policy-document
file://iot-role-trust.json
```

Output dari perintah ini terlihat seperti berikut:

```

{
  "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. Simpan JSON berikut ke dalam file bernama `my-iot-policy.json`.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",

```

```
"Action": "dynamodb:*",
"Resource": "*"
}
]
}
```

JSON ini adalah contoh dokumen kebijakan yang memberikan akses AWS IoT administrator ke DynamoDB.

Gunakan perintah [create-policy](#) untuk memberikan AWS IoT akses ke AWS sumber daya Anda setelah mengambil peran, meneruskan file: `my-iot-policy.json`

```
aws iam create-policy --policy-name my-iot-policy --policy-document file://my-iot-policy.json
```

Untuk informasi selengkapnya tentang cara memberikan akses ke Layanan AWS dalam kebijakan AWS IoT, lihat [Buat aturan](#).

Output dari perintah [create-policy](#) berisi ARN kebijakan. Lampirkan kebijakan ke peran.

```
{
  "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. Gunakan perintah [attach-role-policy](#) untuk melampirkan kebijakan Anda ke peran Anda:

```
aws iam attach-role-policy --role-name my-iot-role --policy-arn
arn:aws:iam::123456789012:policy/my-iot-policy
```

Melewati izin peran

Bagian dari definisi aturan adalah peran IAM yang memberikan izin untuk mengakses sumber daya yang ditentukan dalam tindakan aturan. Mesin aturan mengasumsikan peran itu ketika tindakan aturan dipanggil. Peran harus didefinisikan Akun AWS sama dengan aturan.

Saat membuat atau mengganti aturan, Anda, pada dasarnya, meneruskan peran ke mesin aturan. `iam:PassRole` izin diperlukan untuk melakukan operasi ini. Untuk memverifikasi bahwa Anda memiliki izin ini, buat kebijakan yang memberikan `iam:PassRole` izin dan melampirkannya ke pengguna IAM Anda. Kebijakan berikut menunjukkan cara mengizinkan `iam:PassRole` izin untuk peran.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt1",
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": [
        "arn:aws:iam::123456789012:role/myRole"
      ]
    }
  ]
}
```

Dalam contoh kebijakan ini, izin `iam:PassRole` diberikan untuk peran `myRole`. Peran ditentukan dengan menggunakan ARN peran. Lampirkan kebijakan ini ke pengguna IAM Anda atau peran yang dimiliki pengguna Anda. Untuk informasi lebih lanjut, lihat [Bekerja dengan Kebijakan Terkelola](#).

Note

Fungsi Lambda menggunakan kebijakan berbasis sumber daya, dan kebijakan tersebut melekat langsung ke fungsi Lambda itu sendiri. Saat Anda membuat aturan yang memanggil fungsi Lambda, Anda tidak meneruskan peran, sehingga pengguna yang membuat aturan tidak memerlukan `iam:PassRole` izin. Untuk informasi selengkapnya tentang otorisasi fungsi Lambda, lihat [Memberikan Izin Menggunakan Kebijakan Sumber Daya](#).

Buat aturan

Anda dapat membuat AWS IoT aturan untuk merutekan data dari hal-hal yang terhubung untuk berinteraksi dengan AWS layanan lain. AWS IoT Aturan terdiri dari komponen-komponen berikut:

Komponen aturan

Komponen	Deskripsi	Diperlukan atau Opsional
Nama aturan	Nama aturan. Perhatikan bahwa kami tidak merekomendasikan penggunaan informasi identitas pribadi dalam nama aturan Anda.	Wajib.
Deskripsi aturan	Deskripsi tekstual aturan. Perhatikan bahwa kami tidak merekomendasikan penggunaan informasi identitas pribadi dalam deskripsi aturan Anda.	Tidak wajib.
Pernyataan SQL	Sintaks SQL yang disederhanakan untuk memfilter pesan yang diterima pada topik MQTT dan mendorong data ke tempat lain. Untuk informasi selengkapnya, lihat AWS IoT Referensi SQL .	Wajib.
Versi SQL	Versi mesin aturan SQL untuk digunakan saat mengevaluasi aturan. Meskipun properti ini opsional, kami sangat menyarankan Anda menentukan versi SQL. AWS IoT Core Konsol menyetel properti ini secara default ke 2016-03-23. Jika properti ini tidak diatur, seperti dalam AWS CLI perintah atau AWS CloudFormation template, 2015-10-08 digunakan. Untuk informasi selengkapnya, lihat Versi SQL .	Wajib.
Satu atau lebih tindakan	Tindakan AWS IoT dilakukan saat memberlakukan aturan. Misalnya, Anda dapat menyisipkan data ke dalam tabel DynamoDB, menulis data ke bucket Amazon S3, memublikasikan ke topik Amazon SNS, atau menjalankan fungsi Lambda.	Wajib.

Komponen	Deskripsi	Diperlukan atau Opsional
Tindakan kesalahan	Tindakan AWS IoT dilakukan ketika tidak dapat melakukan tindakan aturan.	Tidak wajib.

Sebelum membuat AWS IoT aturan, Anda harus membuat peran IAM dengan kebijakan yang memungkinkan akses ke AWS sumber daya yang diperlukan. AWS IoT mengasumsikan peran ini saat menerapkan aturan. Untuk informasi selengkapnya, lihat [Memberikan AWS IoT aturan akses yang diperlukan](#) dan [Melewati izin peran](#).

Saat Anda membuat aturan, perhatikan berapa banyak data yang Anda publikasikan tentang topik. Jika Anda membuat aturan yang menyertakan pola topik wildcard, aturan tersebut mungkin cocok dengan sebagian besar pesan Anda. Jika ini masalahnya, Anda mungkin perlu meningkatkan kapasitas AWS sumber daya yang digunakan oleh tindakan target. Selain itu, jika Anda membuat aturan penerbitan ulang yang menyertakan pola topik wildcard, Anda dapat berakhir dengan aturan melingkar yang menyebabkan loop tak terbatas.

Note

Membuat dan memperbarui aturan adalah tindakan tingkat administrator. Setiap pengguna yang memiliki izin untuk membuat atau memperbarui aturan dapat mengakses data yang diproses oleh aturan.

Buat aturan (Konsol)

Untuk membuat aturan (AWS Management Console)

Gunakan [AWS Management Console](#) perintah untuk membuat aturan:

1. Buka [konsol AWS IoT](#).
2. Di navigasi kiri, pilih Perutean pesan dari bagian Kelola. Kemudian pilih Aturan.
3. Pada halaman Aturan, pilih Buat aturan.
4. Pada halaman Tentukan properti aturan, masukkan nama untuk aturan Anda. Deskripsi aturan dan Tag adalah opsional. Pilih Selanjutnya.
5. Pada halaman pernyataan Configure SQL, pilih versi SQL dan masukkan pernyataan SQL. Contoh pernyataan SQL bisa. `SELECT temperature FROM 'iot/topic' WHERE`

temperature > 50 Untuk informasi selengkapnya, lihat [versi SQL](#) dan referensi [AWS IoT SQL](#).

6. Pada halaman Lampirkan tindakan aturan, tambahkan tindakan aturan untuk merutekan data ke AWS layanan lain.
 1. Dalam Tindakan aturan, pilih tindakan aturan dari daftar drop-down. Misalnya, Anda dapat memilih Kinesis Stream. Untuk informasi selengkapnya tentang tindakan aturan, lihat [tindakan AWS IoT aturan](#).
 2. Bergantung pada tindakan aturan yang Anda pilih, masukkan detail konfigurasi terkait. Misalnya, jika Anda memilih Kinesis Stream, Anda harus memilih atau membuat sumber daya aliran data, dan secara opsional memasukkan detail konfigurasi seperti kunci Partisi, yang digunakan untuk mengelompokkan data dengan pecahan dalam Steam.
 3. Dalam peran IAM, pilih atau buat peran untuk memberikan AWS IoT akses ke titik akhir Anda. Perhatikan bahwa secara otomatis AWS IoT akan membuat kebijakan dengan awalan `aws-iot-rule` di bawah peran IAM Anda dipilih. Anda dapat memilih Lihat untuk melihat peran IAM dan kebijakan dari konsol IAM. Tindakan kesalahan adalah opsional. Anda dapat menemukan informasi lebih lanjut di [Penanganan kesalahan \(tindakan kesalahan\)](#). Untuk informasi selengkapnya tentang membuat peran IAM untuk aturan Anda, lihat [Memberikan aturan akses yang diperlukan](#). Pilih Selanjutnya.
7. Pada halaman Tinjau dan buat, tinjau semua konfigurasi dan lakukan pengeditan jika diperlukan. Pilih Buat.

Setelah Anda membuat aturan berhasil, Anda akan melihat aturan yang tercantum di halaman Aturan. Anda dapat memilih aturan untuk membuka halaman Detail tempat Anda dapat melihat aturan, mengedit aturan, menonaktifkan aturan, dan menghapus aturan.

Buat aturan (CLI)

Untuk membuat aturan (AWS CLI)

Gunakan perintah [create-topic-rule untuk membuat aturan](#):

```
aws iot create-topic-rule --rule-name myrule --topic-rule-payload file://myrule.json
```

Berikut ini adalah contoh file payload dengan aturan yang menyisipkan semua pesan yang dikirim ke `iot/test` topik ke dalam tabel DynamoDB yang ditentukan. Pernyataan SQL menyaring pesan dan peran ARN AWS IoT memberikan izin untuk menulis ke tabel 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()}"
      }
    }
  ]
}
```

Berikut ini adalah contoh file payload dengan aturan yang menyisipkan semua pesan yang dikirim ke `iot/test` topik ke dalam bucket S3 yang ditentukan. Pernyataan SQL memfilter pesan, dan peran ARN AWS IoT memberikan izin untuk menulis ke bucket Amazon S3.

```
{
  "awsIotSqlVersion": "2016-03-23",
  "sql": "SELECT * FROM 'iot/test'",
  "ruleDisabled": false,
  "actions": [
    {
      "s3": {
        "roleArn": "arn:aws:iam::123456789012:role/aws_iot_s3",
        "bucketName": "my-bucket",
        "key": "myS3Key"
      }
    }
  ]
}
```

Berikut ini adalah contoh file payload dengan aturan yang mendorong data ke 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()}"
    }
  }
]
}

```

Berikut ini adalah contoh file payload dengan aturan yang memanggil fungsi Lambda:

```

{
  "sql": "expression",
  "ruleDisabled": false,
  "awsIotSqlVersion": "2016-03-23",
  "actions": [
    {
      "lambda": {
        "functionArn": "arn:aws:lambda:us-west-2:123456789012:function:my-lambda-function"
      }
    }
  ]
}

```

Berikut ini adalah contoh file payload dengan aturan yang menerbitkan ke topik 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"
      }
    }
  ]
}

```

```
]
}
```

Berikut ini adalah contoh file payload dengan aturan yang menerbitkan kembali pada topik MQTT yang berbeda:

```
{
  "sql": "expression",
  "ruleDisabled": false,
  "awsIotSqlVersion": "2016-03-23",
  "actions": [
    {
      "republish": {
        "topic": "my-mqtt-topic",
        "roleArn": "arn:aws:iam::123456789012:role/my-iot-role"
      }
    }
  ]
}
```

Berikut ini adalah contoh file payload dengan aturan yang mendorong data ke aliran Amazon 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"
      }
    }
  ]
}
```

Berikut ini adalah contoh file payload dengan aturan yang menggunakan SageMaker `machinelearning_predict` fungsi Amazon untuk menerbitkan ulang ke topik jika data dalam payload MQTT diklasifikasikan sebagai 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"
    }
  }
]
}

```

Berikut ini adalah contoh file payload dengan aturan yang menerbitkan pesan ke aliran input Salesforce IoT Cloud.

```

{
  "sql": "expression",
  "ruleDisabled": false,
  "awsIotSqlVersion": "2016-03-23",
  "actions": [
    {
      "salesforce": {
        "token": "ABCDEFGH123456789abcdefghi123456789",
        "url": "https://ingestion-cluster-id.my-env.sfdcnw.com/streams/stream-id/
connection-id/my-event"
      }
    }
  ]
}

```

Berikut ini adalah contoh file payload dengan aturan yang memulai eksekusi mesin status 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"
  }
}
]
}

```

Menandai aturan Anda

Untuk menambahkan lapisan spesifisitas lain ke aturan baru atau yang sudah ada, Anda dapat menerapkan penandaan. Penandaan memanfaatkan pasangan nilai kunci dalam aturan Anda untuk memberi Anda kontrol yang lebih besar atas bagaimana dan di mana aturan Anda diterapkan pada sumber daya dan layanan Anda AWS IoT . Misalnya, Anda dapat membatasi cakupan aturan agar hanya berlaku di lingkungan beta untuk pengujian pra rilis (Key=environment, Value=beta) atau menangkap semua pesan yang dikirim ke `iot/test` topik hanya dari titik akhir tertentu dan menyimpannya di bucket Amazon S3.

Contoh kebijakan IAM

Untuk contoh yang menunjukkan cara memberikan izin penandaan untuk aturan, pertimbangkan pengguna yang menjalankan perintah berikut untuk membuat aturan dan menandainya agar hanya diterapkan ke lingkungan beta mereka.

Dalam contoh, ganti:

- *MyTopicRuleName* dengan nama aturan.
- *myrule.json* dengan nama dokumen kebijakan.

```

aws iot create-topic-rule
  --rule-name MyTopicRuleName
  --topic-rule-payload file://myrule.json
  --tags "environment=beta"

```

Untuk contoh ini, Anda harus menggunakan kebijakan IAM berikut:

```

{
  "Version": "2012-10-17",
  "Statement":
  {
    "Action": [ "iot:CreateTopicRule", "iot:TagResource" ],

```

```
    "Effect": "Allow",
    "Resource": [
        "arn:aws:iot:us-east-1:123456789012:rule/MyTopicRuleName"
    ]
}
```

Contoh di atas menunjukkan aturan yang baru dibuat MyTopicRuleName yang disebut yang hanya berlaku untuk lingkungan beta Anda. `iot:TagResource` Dalam pernyataan kebijakan dengan panggilan MyTopicRuleName khusus memungkinkan penandaan saat membuat atau memperbarui MyTopicRuleName. Parameter yang `--tags "environment=beta"` digunakan saat membuat aturan membatasi cakupan hanya MyTopicRuleName untuk lingkungan beta Anda. Jika Anda menghapus parameter `--tags "environment=beta"`, maka MyTopicRuleName akan berlaku untuk semua lingkungan.

Untuk informasi selengkapnya tentang membuat peran dan kebijakan IAM khusus untuk AWS IoT aturan, lihat [Memberikan AWS IoT aturan akses yang dibutuhkannya](#)

Untuk informasi umum tentang menandai sumber daya Anda, lihat [Menandai sumber daya Anda AWS IoT](#).

Melihat aturan Anda

Gunakan perintah [list-topic-rules untuk membuat daftar aturan](#) Anda:

```
aws iot list-topic-rules
```

Gunakan perintah [get-topic-rule](#) untuk mendapatkan informasi tentang aturan:

```
aws iot get-topic-rule --rule-name myrule
```

Menghapus aturan

Ketika Anda selesai dengan aturan, Anda dapat menghapusnya.

Untuk menghapus aturan (AWS CLI)

Gunakan perintah [delete-topic-rule untuk menghapus aturan](#):

```
aws iot delete-topic-rule --rule-name myrule
```


AWS IoT tindakan aturan

AWS IoT tindakan aturan menentukan apa yang harus dilakukan ketika aturan dipanggil. Anda dapat menentukan tindakan untuk mengirim data ke database Amazon DynamoDB, mengirim data ke Amazon Kinesis Data Streams, memanggil fungsi, AWS Lambda dan sebagainya. AWS IoT mendukung tindakan berikut di Wilayah AWS mana layanan tindakan tersedia.

Tindakan aturan	Deskripsi	Nama di API
Apache Kafka	Mengirim pesan ke cluster Apache Kafka.	kafka
CloudWatch alarm	Mengubah status CloudWatch alarm Amazon.	cloudwatchAlarm
CloudWatch Log	Mengirim pesan ke Amazon CloudWatch Logs.	cloudwatchLogs
CloudWatch metrik	Mengirim pesan ke CloudWatch metrik.	cloudwatchMetric
DynamoDB	Mengirim pesan ke tabel DynamoDB.	dynamoDB
DynamoDBv2	Mengirim data pesan ke beberapa kolom dalam tabel DynamoDB.	dynamoDBv2
Elasticsearch	Mengirim pesan ke OpenSearch titik akhir.	OpenSearch
HTTP	Memposting pesan ke titik akhir HTTPS.	http
IoT Analytics	Mengirim pesan ke AWS IoT Analytics saluran.	iotAnalytics
AWS IoT Events	Mengirim pesan ke AWS IoT Events input.	iotEvents

Tindakan aturan	Deskripsi	Nama di API
AWS IoT SiteWise	Mengirim data pesan ke properti AWS IoT SiteWise aset.	iotSiteWise
Firehose	Mengirim pesan ke aliran pengiriman Firehose.	firehose
Kinesis Data Streams	Mengirim pesan ke aliran data Kinesis.	kinesis
Lambda	Memanggil fungsi Lambda dengan data pesan sebagai input.	lambda
Lokasi	Mengirim data lokasi ke Amazon Location Service.	location
OpenSearch	Mengirim pesan ke titik akhir OpenSearch Layanan Amazon.	OpenSearch
Publikasikan ulang	Menerbitkan kembali pesan ke topik MQTT lain.	republish
S3	Menyimpan pesan di bucket Amazon Simple Storage Service (Amazon S3).	s3
Salesforce IoT	Mengirim pesan ke aliran input IoT Salesforce.	salesforce
SNS	Menerbitkan pesan sebagai notifikasi push Amazon Simple Notification Service (Amazon SNS).	sns

Tindakan aturan	Deskripsi	Nama di API
SQS	Mengirim pesan ke antrian Amazon Simple Queue Service (Amazon SQS).	sqs
Step Functions	Memulai mesin AWS Step Functions negara.	stepFunctions
the section called “Timestream”	Mengirim pesan ke tabel database Amazon Timestream.	timestream

Catatan

- Tentukan aturan Wilayah AWS sama dengan sumber daya layanan lain sehingga tindakan aturan dapat berinteraksi dengan sumber daya tersebut.
- Mesin AWS IoT aturan mungkin melakukan beberapa upaya untuk melakukan tindakan jika terjadi kesalahan intermiten. Jika semua upaya gagal, pesan akan dibuang dan kesalahan tersedia di Log Anda CloudWatch . Anda dapat menentukan tindakan kesalahan untuk setiap aturan yang dipanggil setelah terjadi kegagalan. Untuk informasi selengkapnya, lihat [Penanganan kesalahan \(tindakan kesalahan\)](#).
- Beberapa tindakan aturan mengaktifkan tindakan dalam layanan yang terintegrasi dengan AWS Key Management Service (AWS KMS) untuk mendukung enkripsi data saat istirahat. Jika Anda menggunakan kunci AWS KMS yang dikelola pelanggan untuk mengenkripsi data saat istirahat, layanan harus memiliki izin untuk menggunakan kunci KMS atas nama pemanggil. Untuk mempelajari cara mengelola izin untuk kunci KMS yang dikelola pelanggan, lihat topik enkripsi data di panduan layanan yang sesuai. Untuk informasi selengkapnya tentang kunci KMS yang dikelola pelanggan, lihat [AWS Key Management Service konsep](#) di Panduan AWS Key Management Service Pengembang.

Apache Kafka

Tindakan Apache Kafka (Kafka) mengirim pesan langsung ke Amazon Managed Streaming for Apache Kafka (Amazon MSK), cluster Apache Kafka yang dikelola oleh penyedia pihak ketiga [seperti Confluent Cloud](#), atau cluster Apache Kafka yang dikelola sendiri untuk analisis dan visualisasi data.

Note

Topik ini mengasumsikan keakraban dengan platform Apache Kafka dan konsep terkait. Untuk informasi lebih lanjut tentang Apache Kafka, lihat [Apache Kafka](#). [MSK Tanpa Server tidak didukung](#). MSK Kluster tanpa server hanya dapat dilakukan melalui otentikasi IAM, yang saat ini tidak didukung oleh tindakan aturan Apache Kafka.

Persyaratan

Tindakan aturan ini memiliki persyaratan sebagai berikut:

- Peran IAM yang AWS IoT dapat diasumsikan untuk melakukan `ec2:CreateNetworkInterface`, `ec2:DescribeNetworkInterfaces`, `ec2:CreateNetworkInterface`, `ec2:DescribeVpc`, `ec2:DescribeVpcAttribute`, dan `ec2:DescribeSecurityGroups` operasi. Peran ini menciptakan dan mengelola antarmuka jaringan elastis ke Amazon Virtual Private Cloud Anda untuk menjangkau broker Kafka Anda. Untuk informasi selengkapnya, lihat [Memberikan AWS IoT aturan akses yang dibutuhkannya](#).

Di AWS IoT konsol, Anda dapat memilih atau membuat peran untuk memungkinkan AWS IoT Core untuk melakukan tindakan aturan ini.

Untuk informasi selengkapnya tentang antarmuka jaringan, lihat [Antarmuka jaringan elastis](#) di Panduan Pengguna Amazon EC2.

Kebijakan yang dilampirkan pada peran yang Anda tentukan akan terlihat seperti contoh berikut.

```
{
  "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": "*"
}
]
}

```

- Jika Anda menggunakan AWS Secrets Manager untuk menyimpan kredensial yang diperlukan untuk terhubung ke broker Kafka Anda, Anda harus membuat peran IAM yang AWS IoT Core dapat diasumsikan untuk melakukan dan operasi. `secretsmanager:GetSecretValue`
`secretsmanager:DescribeSecret`

Kebijakan yang dilampirkan pada peran yang Anda tentukan akan terlihat seperti contoh berikut.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetSecretValue",
        "secretsmanager:DescribeSecret"
      ],
      "Resource": [
        "arn:aws:secretsmanager:region:123456789012:secret:kafka_client_truststore-*",
        "arn:aws:secretsmanager:region:123456789012:secret:kafka_keytab-*"
      ]
    }
  ]
}

```

- Anda dapat menjalankan cluster Apache Kafka Anda di dalam Amazon Virtual Private Cloud (Amazon VPC). Anda harus membuat tujuan VPC Amazon dan menggunakan gateway NAT di subnet Anda untuk meneruskan pesan dari AWS IoT ke cluster Kafka publik. Mesin AWS

IoT aturan membuat antarmuka jaringan di setiap subnet yang tercantum di tujuan VPC untuk mengarahkan lalu lintas langsung ke VPC. Saat Anda membuat tujuan VPC, mesin AWS IoT aturan secara otomatis membuat tindakan aturan VPC. Untuk informasi selengkapnya tentang tindakan aturan VPC, lihat [Tujuan Virtual Private Cloud \(VPC\)](#)

- Jika Anda menggunakan pelanggan yang dikelola AWS KMS key (kunci KMS) untuk mengenkripsi data saat istirahat, layanan harus memiliki izin untuk menggunakan kunci KMS atas nama pemanggil. Untuk informasi selengkapnya, lihat [enkripsi MSK Amazon](#) di Panduan Pengembang Amazon Managed Streaming for Apache Kafka.

Parameter

Saat Anda membuat AWS IoT aturan dengan tindakan ini, Anda harus menentukan informasi berikut:

DestinasiARN

Nama Sumber Daya Amazon (ARN) dari tujuan VPC. Untuk informasi tentang membuat tujuan VPC, lihat [Tujuan Virtual Private Cloud \(VPC\)](#)

topik

Topik Kafka untuk pesan yang akan dikirim ke broker Kafka.

Anda dapat mengganti bidang ini menggunakan templat substitusi. Untuk informasi selengkapnya, lihat [the section called “Templat substitusi”](#).

kunci (opsional)

Kunci pesan Kafka.

Anda dapat mengganti bidang ini menggunakan templat substitusi. Untuk informasi selengkapnya, lihat [the section called “Templat substitusi”](#).

header (opsional)

Daftar header Kafka yang Anda tentukan. Setiap header adalah pasangan kunci-nilai yang dapat Anda tentukan saat Anda membuat tindakan Kafka. Anda dapat menggunakan header ini untuk merutekan data dari klien IoT ke cluster Kafka hilir tanpa mengubah payload pesan Anda.

Anda dapat mengganti bidang ini menggunakan templat substitusi. [Untuk memahami cara meneruskan fungsi Aturan sebaris sebagai templat substitusi di header Kafka Action, lihat Contoh.](#) Untuk informasi selengkapnya, lihat [the section called “Templat substitusi”](#).

Note

Header dalam format biner tidak didukung.

partisi (opsional)

Partisi pesan Kafka.

Anda dapat mengganti bidang ini menggunakan templat substitusi. Untuk informasi selengkapnya, lihat [the section called “Templat substitusi”](#).

ClientProperties

Objek yang mendefinisikan properti klien produsen Apache Kafka.

acks (opsional)

Jumlah ucapan terima kasih yang harus diterima oleh produsen sebelum mempertimbangkan permintaan lengkap.

Jika Anda menentukan 0 sebagai nilai, produsen tidak akan menunggu pengakuan apa pun dari server. Jika server tidak menerima pesan, produser tidak akan mencoba lagi untuk mengirim pesan.

Nilai yang valid: `-1,0,1,all`. Nilai default-nya adalah 1.

bootstrap.servers

Daftar pasangan host dan port (misalnya, `host1:port1,host2:port2`) yang digunakan untuk membuat koneksi awal ke cluster Kafka Anda.

kompresi.type (opsional)

Jenis kompresi untuk semua data yang dihasilkan oleh produsen.

Nilai yang valid: `none,gzip,snappy,lz4,zstd`. Nilai default-nya adalah `none`.

security.protocol

Protokol keamanan yang digunakan untuk melampirkan ke broker Kafka Anda.

Nilai-nilai yang valid: `SSL, SASL_SSL`. Nilai default-nya adalah `SSL`.

key.serializer

Menentukan cara mengubah objek kunci yang Anda berikan dengan `ProducerRecord` menjadi byte.

Nilai valid: `StringSerializer`.

value.serializer

Menentukan cara mengubah objek nilai yang Anda berikan dengan `ProducerRecord` menjadi byte.

Nilai valid: `ByteBufferSerializer`.

ssl.truststore

File truststore dalam format base64 atau lokasi file truststore di. [AWS Secrets Manager](#) Nilai ini tidak diperlukan jika truststore Anda dipercaya oleh otoritas sertifikat Amazon (CA).

Bidang ini mendukung template substitusi. Jika Anda menggunakan Secrets Manager untuk menyimpan kredensial yang diperlukan untuk terhubung ke broker Kafka Anda, Anda dapat menggunakan fungsi `get_secret` SQL untuk mengambil nilai untuk bidang ini. Untuk informasi selengkapnya tentang templat substitusi, lihat [the section called “Templat substitusi”](#). Untuk informasi selengkapnya tentang fungsi `get_secret` SQL, lihat [the section called “get_secret \(secretID, secretType, kunci, roLearn\)”](#). Jika truststore dalam bentuk file, gunakan parameternya `SecretBinary`. Jika truststore dalam bentuk string, gunakan parameternya `SecretString`.

Ukuran maksimum nilai ini adalah 65 KB.

ssl.truststore.password

Kata sandi untuk truststore. Nilai ini diperlukan hanya jika Anda telah membuat kata sandi untuk truststore.

ssl.keystore

File keystore. Nilai ini diperlukan saat Anda menentukan SSL sebagai nilai untuk `security.protocol`.

Bidang ini mendukung template substitusi. Gunakan Secrets Manager untuk menyimpan kredensial yang diperlukan untuk terhubung ke broker Kafka Anda. Untuk mengambil nilai untuk

bidang ini, gunakan fungsi `get_secret` SQL. Untuk informasi selengkapnya tentang templat substitusi, lihat [the section called “Templat substitusi”](#). Untuk informasi selengkapnya tentang fungsi `get_secret` SQL, lihat [the section called “get_secret \(secretID, secretType, kunci, roLearn\)”](#). Gunakan parameter `SecretBinary`.

`ssl.keystore.password`

Kata sandi toko untuk file keystore. Nilai ini diperlukan jika Anda menentukan nilai untuk `ssl.keystore`.

Nilai bidang ini bisa berupa plaintext. Bidang ini juga mendukung template substitusi. Gunakan Secrets Manager untuk menyimpan kredensial yang diperlukan untuk terhubung ke broker Kafka Anda. Untuk mengambil nilai untuk bidang ini, gunakan fungsi `get_secret` SQL. Untuk informasi selengkapnya tentang templat substitusi, lihat [the section called “Templat substitusi”](#). Untuk informasi selengkapnya tentang fungsi `get_secret` SQL, lihat [the section called “get_secret \(secretID, secretType, kunci, roLearn\)”](#). Gunakan parameter `SecretString`.

`ssl.key.password`


Kata sandi kunci pribadi di file keystore Anda.

Bidang ini mendukung template substitusi. Gunakan Secrets Manager untuk menyimpan kredensial yang diperlukan untuk terhubung ke broker Kafka Anda. Untuk mengambil nilai untuk bidang ini, gunakan fungsi `get_secret` SQL. Untuk informasi selengkapnya tentang templat substitusi, lihat [the section called “Templat substitusi”](#). Untuk informasi selengkapnya tentang fungsi `get_secret` SQL, lihat [the section called “get_secret \(secretID, secretType, kunci, roLearn\)”](#). Gunakan parameter `SecretString`.

`sasl.mekanisme`

Mekanisme keamanan yang digunakan untuk terhubung ke broker Kafka Anda. Nilai ini diperlukan saat Anda menentukan `SASL_SSL` untuk `security.protocol`.

Nilai-nilai yang valid: PLAIN, SCRAM-SHA-512, GSSAPI.

 Note

SCRAM-SHA-512 adalah satu-satunya mekanisme keamanan yang didukung di Wilayah `cn-north-1`, `cn-northwest-1`, `-1`, dan `-1.us-gov-east us-gov-west`

sasl.plain.username

Nama pengguna yang digunakan untuk mengambil string rahasia dari Secrets Manager. Nilai ini diperlukan saat Anda menentukan SASL_SSL untuk `security.protocol` dan PLAIN untuk `sasl.mechanism`.

sasl.plain.password

Kata sandi yang digunakan untuk mengambil string rahasia dari Secrets Manager. Nilai ini diperlukan saat Anda menentukan SASL_SSL untuk `security.protocol` dan PLAIN untuk `sasl.mechanism`.

sasl.scram.username

Nama pengguna yang digunakan untuk mengambil string rahasia dari Secrets Manager. Nilai ini diperlukan saat Anda menentukan SASL_SSL untuk `security.protocol` dan SCRAM-SHA-512 untuk `sasl.mechanism`.

sasl.scram.password

Kata sandi yang digunakan untuk mengambil string rahasia dari Secrets Manager. Nilai ini diperlukan saat Anda menentukan SASL_SSL untuk `security.protocol` dan SCRAM-SHA-512 untuk `sasl.mechanism`.

sasl.kerberos.keytab

File keytab untuk otentikasi Kerberos di Secrets Manager. Nilai ini diperlukan saat Anda menentukan SASL_SSL untuk `security.protocol` dan GSSAPI untuk `sasl.mechanism`.

Bidang ini mendukung template substitusi. Gunakan Secrets Manager untuk menyimpan kredensial yang diperlukan untuk terhubung ke broker Kafka Anda. Untuk mengambil nilai untuk bidang ini, gunakan fungsi `get_secret` SQL. Untuk informasi selengkapnya tentang templat substitusi, lihat [the section called “Templat substitusi”](#). Untuk informasi selengkapnya tentang fungsi `get_secret` SQL, lihat [the section called “get_secret \(secretID, secretType, kunci, roLearn\)”](#). Gunakan parameter `SecretBinary`.

sasl.kerberos.service.name

Nama utama Kerberos di mana Apache Kafka berjalan. Nilai ini diperlukan saat Anda menentukan SASL_SSL untuk `security.protocol` dan GSSAPI untuk `sasl.mechanism`.

sasl.kerberos.krb5.kdc

Nama host dari pusat distribusi kunci (KDC) yang terhubung dengan klien produsen Apache Kafka Anda. Nilai ini diperlukan saat Anda menentukan SASL_SSL untuk `security.protocol` dan GSSAPI untuk `sasl.mechanism`.

sasl.kerberos.krb5.realm

Ranah yang terhubung dengan klien produser Apache Kafka Anda. Nilai ini diperlukan saat Anda menentukan SASL_SSL untuk `security.protocol` dan GSSAPI untuk `sasl.mechanism`.

sasl.kerberos.principal

Identitas unik Kerberos tempat Kerberos dapat menetapkan tiket untuk mengakses layanan yang sadar Kerberos. Nilai ini diperlukan saat Anda menentukan SASL_SSL untuk `security.protocol` dan GSSAPI untuk `sasl.mechanism`.

Contoh

Contoh JSON berikut mendefinisikan tindakan Apache Kafka dalam sebuah aturan. AWS IoT Contoh berikut meneruskan fungsi inline [sourceIP \(\)](#) sebagai template [substitusi](#) di header 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()}"
    }
  ]
}
]
}
}
}
}
}
}
}
}
}

```

Catatan penting tentang pengaturan Kerberos Anda

- Pusat distribusi kunci (KDC) Anda harus dapat diselesaikan melalui Private Domain Name System (DNS) dalam VPC target Anda. Salah satu pendekatan yang mungkin adalah menambahkan entri DNS KDC ke zona host pribadi. Untuk informasi selengkapnya tentang pendekatan ini, lihat [Bekerja dengan zona yang dihosting pribadi](#).
- Setiap VPC harus memiliki resolusi DNS yang diaktifkan. Untuk informasi selengkapnya, lihat [Menggunakan DNS dengan VPC Anda](#).
- Grup keamanan antarmuka jaringan dan grup keamanan tingkat instans di tujuan VPC harus mengizinkan lalu lintas dari dalam VPC Anda pada port berikut.
 - Lalu lintas TCP pada port pendengar broker bootstrap (seringkali 9092, tetapi harus berada dalam kisaran 9000—9100)
 - Lalu lintas TCP dan UDP di port 88 untuk KDC
- SCRAM-SHA-512 adalah satu-satunya mekanisme keamanan yang didukung di Wilayah cn-north-1, cn-northwest-1, -1, dan -1. us-gov-east us-gov-west

Tujuan Virtual Private Cloud (VPC)

Tindakan aturan Apache Kafka merutekan data ke cluster Apache Kafka di Amazon Virtual Private Cloud (Amazon VPC). Konfigurasi VPC yang digunakan oleh tindakan aturan Apache Kafka diaktifkan secara otomatis saat Anda menentukan tujuan VPC untuk tindakan aturan Anda.

Tujuan VPC berisi daftar subnet di dalam VPC. Mesin aturan membuat elastic network interface di setiap subnet yang Anda tentukan dalam daftar ini. Untuk informasi selengkapnya tentang antarmuka jaringan, lihat [Antarmuka jaringan elastis](#) di Panduan Pengguna Amazon EC2.

Persyaratan dan pertimbangan

- Jika Anda menggunakan cluster Apache Kafka yang dikelola sendiri yang akan diakses menggunakan endpoint publik di internet:
 - Buat gateway NAT untuk instance di subnet Anda. Gateway NAT memiliki alamat IP publik yang dapat terhubung ke internet, yang memungkinkan mesin aturan untuk meneruskan pesan Anda ke kluster Kafka publik.
 - Alokasikan alamat IP elastis dengan antarmuka jaringan elastis (ENI) yang dibuat oleh tujuan VPC. Grup keamanan yang Anda gunakan harus dikonfigurasi untuk memblokir lalu lintas masuk.

Note

Jika tujuan VPC dinonaktifkan dan kemudian diaktifkan kembali, Anda harus mengaitkan kembali IP elastis dengan ENI baru.

- Jika tujuan aturan topik VPC tidak menerima lalu lintas apa pun selama 30 hari berturut-turut, itu akan dinonaktifkan.
- Jika ada sumber daya yang digunakan oleh tujuan VPC berubah, tujuan akan dinonaktifkan dan tidak dapat digunakan.
- Beberapa perubahan yang dapat menonaktifkan tujuan VPC meliputi: menghapus VPC, subnet, grup keamanan, atau peran yang digunakan; memodifikasi peran agar tidak lagi memiliki izin yang diperlukan; dan menonaktifkan tujuan.

Harga

Untuk tujuan penetapan harga, tindakan aturan VPC diukur selain tindakan yang mengirim pesan ke sumber daya saat sumber daya ada di VPC Anda. Untuk informasi harga, lihat [Harga AWS IoT Core](#).

Membuat tujuan aturan topik virtual private cloud (VPC)

Anda membuat tujuan virtual private cloud (VPC) dengan menggunakan [CreateTopicRuleDestination](#) API atau konsol. AWS IoT Core

Saat Anda membuat tujuan VPC, Anda harus menentukan informasi berikut.

vpclId

ID unik dari tujuan VPC.

subnetIds

Daftar subnet di mana mesin aturan menciptakan antarmuka jaringan elastis. Mesin aturan mengalokasikan antarmuka jaringan tunggal untuk setiap subnet dalam daftar.

SecurityGroups (opsional)

Daftar grup keamanan untuk diterapkan ke antarmuka jaringan.

roleArn

Nama Sumber Daya Amazon (ARN) dari peran yang memiliki izin untuk membuat antarmuka jaringan atas nama Anda.

ARN ini harus memiliki kebijakan yang melekat padanya yang terlihat seperti contoh berikut.

```
{
  "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"
      }
    }
  }
]
}

```

Membuat tujuan VPC dengan menggunakan AWS CLI

Contoh berikut menunjukkan cara membuat tujuan VPC dengan menggunakan 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"}'

```

Setelah Anda menjalankan perintah ini, status tujuan VPC akan menjadi IN_PROGRESS Setelah beberapa menit, statusnya akan berubah menjadi ERROR (jika perintah tidak berhasil) atau ENABLED. Ketika status tujuan ENABLED, itu siap digunakan.

Anda dapat menggunakan perintah berikut untuk mendapatkan status tujuan VPC Anda.

```
aws --region region iot get-topic-rule-destination --arn "VPCDestinationARN"
```

Membuat tujuan VPC dengan menggunakan konsol AWS IoT Core

Langkah-langkah berikut menjelaskan cara membuat tujuan VPC dengan menggunakan konsol AWS IoT Core

1. Arahkan ke AWS IoT Core konsol. Di panel kiri, pada tab Act, pilih Destinasi.
2. Masukkan nilai untuk bidang berikut.
 - ID VPC
 - ID Subnet
 - Grup Keamanan
3. Pilih peran yang memiliki izin yang diperlukan untuk membuat antarmuka jaringan. Kebijakan contoh sebelumnya berisi izin ini.

Ketika status tujuan VPC DIAKTIFKAN, itu siap digunakan.

CloudWatch alarm

Tindakan CloudWatch alarm (`cloudwatch:Alarm`) mengubah status CloudWatch alarm Amazon. Anda dapat menentukan alasan dan nilai perubahan status dalam panggilan ini.

Persyaratan

Tindakan aturan ini memiliki persyaratan sebagai berikut:

- Peran IAM yang AWS IoT dapat diasumsikan untuk melakukan `cloudwatch:SetAlarmState` operasi. Untuk informasi selengkapnya, lihat [Memberikan AWS IoT aturan akses yang dibutuhkannya](#).

Di AWS IoT konsol, Anda dapat memilih atau membuat peran untuk memungkinkan AWS IoT untuk melakukan tindakan aturan ini.

Parameter

Saat Anda membuat AWS IoT aturan dengan tindakan ini, Anda harus menentukan informasi berikut:

alarmName

Nama CloudWatch alarm.

Mendukung [template substitusi](#): API dan hanya AWS CLI

stateReason

Alasan perubahan alarm.

Mendukung [template substitusi](#): Ya

stateValue

Nilai status alarm. Nilai-nilai yang valid: OK, ALARM, INSUFFICIENT_DATA.

Mendukung [template substitusi](#): Ya

roleArn

Peran IAM yang memungkinkan akses ke CloudWatch alarm. Untuk informasi selengkapnya, lihat [Persyaratan](#).

Mendukung [template substitusi](#): Tidak

Contoh

Contoh JSON berikut mendefinisikan tindakan CloudWatch alarm AWS IoT dalam aturan.

```
{
  "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"
        }
      }
    ]
  }
}
```

```
}
```

Lihat juga

- [Apa itu Amazon CloudWatch?](#) di Panduan CloudWatch Pengguna Amazon
- [Menggunakan CloudWatch alarm Amazon](#) di CloudWatch Panduan Pengguna Amazon

CloudWatch Log

Tindakan CloudWatch Logs (`cloudwatchLogs`) mengirimkan data ke Amazon CloudWatch Logs. Anda dapat menggunakan `batchMode` untuk mengunggah dan stempel waktu beberapa catatan log perangkat dalam satu pesan. Anda juga dapat menentukan grup log tempat tindakan mengirim data.

Persyaratan

Tindakan aturan ini memiliki persyaratan sebagai berikut:

- Peran IAM yang AWS IoT dapat diasumsikan untuk melakukan `logs:CreateLogStream`, `logs:DescribeLogStreams`, dan `logs:PutLogEvents` operasi. Untuk informasi selengkapnya, lihat [Memberikan AWS IoT aturan akses yang dibutuhkannya](#).

Di AWS IoT konsol, Anda dapat memilih atau membuat peran untuk memungkinkan AWS IoT untuk melakukan tindakan aturan ini.

- Jika Anda menggunakan pelanggan yang dikelola AWS KMS key (kunci KMS) untuk mengenkripsi data CloudWatch log di Log, layanan harus memiliki izin untuk menggunakan kunci KMS atas nama pemanggil. Untuk informasi selengkapnya, lihat [Mengekripsi data CloudWatch log di Log menggunakan AWS KMS](#) Panduan Pengguna Amazon CloudWatch Logs.

Persyaratan format pesan MQTT untuk **batchMode**

Jika Anda menggunakan tindakan aturan CloudWatch Log dengan `batchMode` dimatikan, tidak ada persyaratan pemformatan pesan MQTT. (Catatan: nilai default `batchMode` parameter adalah `false`.) Namun, jika Anda menggunakan tindakan aturan CloudWatch Log dengan `batchMode` diaktifkan (nilai parameternya `true`), pesan MQTT yang berisi log sisi perangkat harus diformat agar berisi stempel waktu dan muatan pesan. Catatan: `timestamp` mewakili waktu terjadinya peristiwa dan dinyatakan sebagai sejumlah milidetik setelah 1 Januari 1970 00:00:00 UTC.

Berikut ini adalah contoh format publikasi:

```
[
  {"timestamp": 1673520691093, "message": "Test message 1"},
  {"timestamp": 1673520692879, "message": "Test message 2"},
  {"timestamp": 1673520693442, "message": "Test message 3"}
]
```

Bergantung pada bagaimana log sisi perangkat dihasilkan, log tersebut mungkin perlu difilter dan diformat ulang sebelum dikirim untuk memenuhi persyaratan ini. Untuk informasi selengkapnya, lihat [Muatan Pesan MQTT](#).

Terlepas dari `batchMode` parameter, `message` konten harus mematuhi batasan ukuran AWS IoT pesan. Untuk informasi lebih lanjut, lihat [AWS IoT Core kuota dan titik akhir](#).

Parameter

Saat Anda membuat AWS IoT aturan dengan tindakan ini, Anda harus menentukan informasi berikut:

`logGroupName`

Grup CloudWatch log tempat tindakan mengirimkan data.

Mendukung [template substitusi](#): API dan hanya AWS CLI

`roleArn`

Peran IAM yang memungkinkan akses ke grup CloudWatch log. Untuk informasi selengkapnya, lihat [Persyaratan](#).

Mendukung [template substitusi](#): Tidak

(opsional) `batchMode`

Menunjukkan apakah kumpulan catatan log akan diekstraksi dan diunggah ke dalam. CloudWatch Nilai termasuk `true` atau `false` (default). Untuk informasi selengkapnya, lihat [Persyaratan](#).

Mendukung [template substitusi](#): Tidak

Contoh

Contoh JSON berikut mendefinisikan tindakan CloudWatch Log dalam aturan. AWS IoT

```
{
  "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
        }
      }
    ]
  }
}
```

Lihat juga

- [Apa itu Amazon CloudWatch Logs?](#) di Panduan Pengguna CloudWatch Log Amazon

CloudWatch metrik

Tindakan CloudWatch metrik (`cloudwatchMetric`) menangkap CloudWatch metrik Amazon. Anda dapat menentukan namespace metrik, nama, nilai, unit, dan stempel waktu.

Persyaratan

Tindakan aturan ini memiliki persyaratan sebagai berikut:

- Peran IAM yang AWS IoT dapat diasumsikan untuk melakukan `cloudwatch:PutMetricData` operasi. Untuk informasi selengkapnya, lihat [Memberikan AWS IoT aturan akses yang dibutuhkannya](#).

Di AWS IoT konsol, Anda dapat memilih atau membuat peran untuk memungkinkan AWS IoT untuk melakukan tindakan aturan ini.

Parameter

Saat Anda membuat AWS IoT aturan dengan tindakan ini, Anda harus menentukan informasi berikut:

metricName

Nama CloudWatch metrik.

Mendukung [template substitusi](#): Ya

metricNamespace

Nama namespace CloudWatch metrik.

Mendukung [template substitusi](#): Ya

metricUnit

Unit metrik yang didukung oleh CloudWatch.

Mendukung [template substitusi](#): Ya

metricValue

String yang berisi nilai CloudWatch metrik.

Mendukung [template substitusi](#): Ya

metricTimestamp

(Opsional) String yang berisi stempel waktu, dinyatakan dalam detik dalam waktu epoch Unix. Default ke waktu epoch Unix saat ini.

Mendukung [template substitusi](#): Ya

roleArn

Peran IAM yang memungkinkan akses ke CloudWatch metrik. Untuk informasi selengkapnya, lihat [Persyaratan](#).

Mendukung [template substitusi](#): Tidak

Contoh

Contoh JSON berikut mendefinisikan tindakan CloudWatch metrik dalam aturan. AWS IoT

```
{
  "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"
        }
      }
    ]
  }
}

```

Contoh JSON berikut mendefinisikan tindakan CloudWatch metrik dengan template substitusi dalam aturan. AWS IoT

```

{
  "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"
        }
      }
    ]
  }
}

```

Lihat juga

- [Apa itu Amazon CloudWatch?](#) di Panduan CloudWatch Pengguna Amazon

- [Menggunakan CloudWatch metrik Amazon](#) di CloudWatch Panduan Pengguna Amazon

DynamoDB

Tindakan DynamoDB dynamoDB () menulis semua atau sebagian pesan MQTT ke tabel Amazon DynamoDB.

Anda dapat mengikuti tutorial yang menunjukkan cara membuat dan menguji aturan dengan tindakan DynamoDB. Untuk informasi selengkapnya, lihat [Tutorial: Menyimpan data perangkat dalam tabel DynamoDB](#).

Note

Aturan ini menulis data non-JSON ke DynamoDB sebagai data biner. Konsol DynamoDB menampilkan data sebagai teks yang disandikan base64.

Persyaratan

Tindakan aturan ini memiliki persyaratan sebagai berikut:

- Peran IAM yang AWS IoT dapat diasumsikan untuk melakukan dynamodb:PutItem operasi. Untuk informasi selengkapnya, lihat [Memberikan AWS IoT aturan akses yang dibutuhkannya](#).

Di AWS IoT konsol, Anda dapat memilih atau membuat peran untuk memungkinkan AWS IoT untuk melakukan tindakan aturan ini.

- Jika Anda menggunakan pelanggan yang dikelola AWS KMS key (kunci KMS) untuk mengenkripsi data saat istirahat di DynamoDB, layanan harus memiliki izin untuk menggunakan kunci KMS atas nama pemanggil. Untuk informasi selengkapnya, lihat [kunci KMS yang Dikelola Pelanggan](#) di Panduan Memulai Amazon DynamoDB.

Parameter

Saat Anda membuat AWS IoT aturan dengan tindakan ini, Anda harus menentukan informasi berikut:

`tableName`

Nama dari tabel DynamoDB.

Mendukung [template substitusi](#): API dan hanya AWS CLI

hashKeyField

Nama kunci hash (juga disebut kunci partisi).

Mendukung [template substitusi](#): API dan hanya AWS CLI

hashKeyType

(Opsional) Tipe data dari kunci hash (juga disebut kunci partisi). Nilai-nilai yang valid: STRING, NUMBER.

Mendukung [template substitusi](#): API dan hanya AWS CLI

hashKeyValue

Nilai kunci hash. Pertimbangkan untuk menggunakan template substitusi seperti `${topic()}` atau `${timestamp()}`.

Mendukung [template substitusi](#): Ya

rangeKeyField

(Opsional) Nama tombol rentang (juga disebut tombol sortir).

Mendukung [template substitusi](#): API dan hanya AWS CLI

rangeKeyType

(Opsional) Tipe data dari tombol rentang (juga disebut tombol sortir). Nilai-nilai yang valid: STRING, NUMBER.

Mendukung [template substitusi](#): API dan hanya AWS CLI

rangeKeyValue

(Opsional) Nilai tombol rentang. Pertimbangkan untuk menggunakan template substitusi seperti `${topic()}` atau `${timestamp()}`.

Mendukung [template substitusi](#): Ya

payloadField

(Opsional) Nama kolom tempat muatan ditulis. Jika Anda menghilangkan nilai ini, payload ditulis ke kolom bernama. `payload`

Mendukung [template substitusi](#): Ya

operation

(Opsional) Jenis operasi yang akan dilakukan. Nilai-nilai yang valid: INSERT, UPDATE, DELETE.

Mendukung [template substitusi](#): Ya

roleARN

Peran IAM yang memungkinkan akses ke tabel DynamoDB. Untuk informasi selengkapnya, lihat [Persyaratan](#).

Mendukung [template substitusi](#): Tidak

Data yang ditulis ke tabel DynamoDB adalah hasil dari pernyataan SQL aturan.

Contoh

Contoh JSON berikut mendefinisikan tindakan DynamoDB dalam aturan. 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"
        }
      }
    ]
  }
}
```

Lihat juga

- [Apa itu Amazon DynamoDB?](#) di Panduan Pengembang Amazon DynamoDB

- [Memulai DynamoDB](#) di Panduan Pengembang Amazon DynamoDB
- [Tutorial: Menyimpan data perangkat dalam tabel DynamoDB](#)

DynamoDBv2

Tindakan DynamoDBv2 (`dynamoDBv2`) menulis semua atau sebagian pesan MQTT ke tabel Amazon DynamoDB. Setiap atribut dalam payload ditulis ke kolom terpisah dalam database DynamoDB.

Persyaratan

Tindakan aturan ini memiliki persyaratan sebagai berikut:

- Peran IAM yang AWS IoT dapat diasumsikan untuk melakukan `dynamodb:PutItem` operasi. Untuk informasi selengkapnya, lihat [Memberikan AWS IoT aturan akses yang dibutuhkannya](#).

Di AWS IoT konsol, Anda dapat memilih atau membuat peran untuk memungkinkan AWS IoT untuk melakukan tindakan aturan ini.

- Payload pesan MQTT harus berisi kunci tingkat root yang cocok dengan kunci partisi utama tabel dan kunci tingkat root yang cocok dengan kunci sortir utama tabel, jika ditentukan.
- Jika Anda menggunakan pelanggan yang dikelola AWS KMS key (kunci KMS) untuk mengenkripsi data saat istirahat di DynamoDB, layanan harus memiliki izin untuk menggunakan kunci KMS atas nama pemanggil. Untuk informasi selengkapnya, lihat [kunci KMS yang Dikelola Pelanggan](#) di Panduan Memulai Amazon DynamoDB.

Parameter

Saat Anda membuat AWS IoT aturan dengan tindakan ini, Anda harus menentukan informasi berikut:

`putItem`

Objek yang menentukan tabel DynamoDB yang data pesan akan ditulis. Objek ini harus berisi informasi berikut:

`tableName`

Nama dari tabel DynamoDB.

Mendukung [template substitusi](#): API dan hanya AWS CLI

roleARN

Peran IAM yang memungkinkan akses ke tabel DynamoDB. Untuk informasi selengkapnya, lihat [Persyaratan](#).

Mendukung [template substitusi](#): Tidak

Data yang ditulis ke tabel DynamoDB adalah hasil dari pernyataan SQL aturan.

Contoh

Contoh JSON berikut mendefinisikan tindakan DynamoDBv2 dalam aturan. AWS IoT

```
{
  "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_iam_dynamoDBv2",
        }
      }
    ]
  }
}
```

Contoh JSON berikut mendefinisikan tindakan DynamoDB dengan template substitusi dalam aturan. 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"
        }
    ]
}
```

Lihat juga

- [Apa itu Amazon DynamoDB?](#) di Panduan Pengembang Amazon DynamoDB
- [Memulai DynamoDB](#) di Panduan Pengembang Amazon DynamoDB

Elasticsearch

Tindakan Elasticsearch (`elasticsearch`) menulis data dari pesan MQTT ke domain Layanan Amazon. OpenSearch Anda kemudian dapat menggunakan alat seperti OpenSearch Dasbor untuk menanyakan dan memvisualisasikan data di OpenSearch Layanan.

Warning

ElasticsearchTindakan hanya dapat digunakan oleh tindakan aturan yang ada. Untuk membuat tindakan aturan baru atau memperbarui tindakan aturan yang ada, gunakan tindakan OpenSearch aturan sebagai gantinya. Untuk informasi selengkapnya, lihat [OpenSearch](#).

Persyaratan

Tindakan aturan ini memiliki persyaratan sebagai berikut:

- Peran IAM yang AWS IoT dapat diasumsikan untuk melakukan `es:ESHttpPost` operasi. Untuk informasi selengkapnya, lihat [Memberikan AWS IoT aturan akses yang dibutuhkannya](#).

Di AWS IoT konsol, Anda dapat memilih atau membuat peran untuk memungkinkan AWS IoT untuk melakukan tindakan aturan ini.

- Jika Anda menggunakan pelanggan yang dikelola AWS KMS key (kunci KMS) untuk mengenkripsi data saat istirahat OpenSearch, layanan harus memiliki izin untuk menggunakan kunci KMS atas nama pemanggil. Untuk informasi selengkapnya, lihat [Enkripsi data saat istirahat untuk OpenSearch Layanan Amazon](#) di Panduan Pengembang OpenSearch Layanan Amazon.

Parameter

Saat Anda membuat AWS IoT aturan dengan tindakan ini, Anda harus menentukan informasi berikut:

endpoint

Titik akhir domain layanan Anda.

Mendukung [template substitusi](#): API dan hanya AWS CLI

index

Indeks tempat Anda ingin menyimpan data Anda.

Mendukung [template substitusi](#): Ya

type

Jenis dokumen yang Anda simpan.

Mendukung [template substitusi](#): Ya

id

Pengidentifikasi unik untuk setiap dokumen.

Mendukung [template substitusi](#): Ya

roleARN

Peran IAM yang memungkinkan akses ke domain OpenSearch Layanan. Untuk informasi selengkapnya, lihat [Persyaratan](#).

Mendukung [template substitusi](#): Tidak

Contoh

Contoh JSON berikut mendefinisikan tindakan Elasticsearch dalam AWS IoT aturan dan bagaimana Anda dapat menentukan bidang untuk tindakan tersebut. `elasticsearch` Untuk informasi lebih lanjut, lihat [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_iam_es"
        }
      }
    ]
  }
}
```

Contoh JSON berikut mendefinisikan tindakan Elasticsearch dengan template substitusi dalam aturan. 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_iam_es"
        }
      }
    ]
  }
}
```

Lihat juga

- [OpenSearch](#)
- [Apa itu OpenSearch Layanan Amazon?](#)

HTTP

Tindakan HTTPS (`https`) mengirimkan data dari pesan MQTT ke aplikasi atau layanan web.

Persyaratan

Tindakan aturan ini memiliki persyaratan sebagai berikut:

- Anda harus mengonfirmasi dan mengaktifkan HTTPS titik akhir sebelum mesin aturan dapat menggunakannya. Untuk informasi selengkapnya, lihat [Bekerja dengan tujuan aturan topik HTTP](#).

Parameter

Saat Anda membuat AWS IoT aturan dengan tindakan ini, Anda harus menentukan informasi berikut:

`url`

Titik akhir HTTPS tempat pesan dikirim menggunakan metode HTTP POST. Jika Anda menggunakan alamat IP sebagai pengganti nama host, itu harus berupa alamat IPv4. Alamat IPv6 tidak didukung.

Mendukung [template substitusi](#): Ya

`confirmationUrl`

(Opsional) Jika ditentukan, AWS IoT gunakan URL konfirmasi untuk membuat tujuan aturan topik yang cocok. Anda harus mengaktifkan tujuan aturan topik sebelum menggunakannya dalam tindakan HTTP. Untuk informasi selengkapnya, lihat [Bekerja dengan tujuan aturan topik HTTP](#). Jika Anda menggunakan templat substitusi, Anda harus membuat tujuan aturan topik secara manual sebelum `http` tindakan dapat digunakan. `confirmationUrl` harus menjadi awalan dari `url`

Hubungan antara `url` dan `confirmationUrl` dijelaskan sebagai berikut:

- Jika `url` di-hardcode dan tidak `confirmationUrl` disediakan, kami secara implisit memperlakukan bidang tersebut sebagai `url confirmationUrl` AWS IoT membuat tujuan aturan topik untuk `url`.
- Jika `url` dan `confirmationUrl` di-hardcode, `url` harus dimulai dengan `confirmationUrl` AWS IoT membuat tujuan aturan topik untuk `confirmationUrl`.
- Jika `url` berisi template substitusi, Anda harus menentukan `confirmationUrl` dan `url` harus mulai dengan `confirmationUrl`. Jika `confirmationUrl` berisi templat substitusi, Anda harus membuat tujuan aturan topik secara manual sebelum http tindakan dapat digunakan. Jika `confirmationUrl` tidak berisi templat substitusi, AWS IoT buat tujuan aturan topik untuk `confirmationUrl`.

Mendukung [template substitusi](#): Ya

headers

(Opsional) Daftar header untuk disertakan dalam permintaan HTTP ke titik akhir. Setiap header harus berisi informasi berikut:

key

Kunci header.

Mendukung [template substitusi](#): Tidak

value

Nilai header.

Mendukung [template substitusi](#): Ya

Note

Jenis konten default adalah `application/json` ketika payload dalam format JSON. Jika tidak, itu adalah `application/octet-stream`. Anda dapat menimpa dengan menentukan jenis konten yang tepat di header dengan tipe konten kunci (case insensitive).

auth

(Opsional) Otentikasi yang digunakan oleh mesin aturan untuk terhubung ke URL titik akhir yang ditentukan dalam argumen. `url` Saat ini, Signature Version 4 adalah satu-satunya jenis otentikasi yang didukung. Untuk informasi selengkapnya, lihat [Otorisasi HTTP](#).

Mendukung [template substitusi](#): Tidak

Contoh

Contoh JSON berikut mendefinisikan AWS IoT aturan dengan tindakan 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}"
            }
          ]
        }
      }
    ]
  }
}
```

Logika coba lagi tindakan HTTP

Mesin AWS IoT aturan mencoba ulang tindakan HTTP sesuai dengan aturan ini:

- Mesin aturan mencoba mengirim pesan setidaknya sekali.
- Mesin aturan mencoba ulang paling banyak dua kali. Jumlah percobaan maksimum adalah tiga.
- Mesin aturan tidak mencoba lagi jika:
 - Percobaan sebelumnya memberikan respons yang lebih besar dari 16.384 byte.

- Layanan web hilir atau aplikasi menutup koneksi TCP setelah mencoba.
- Total waktu untuk menyelesaikan permintaan dengan percobaan ulang melebihi batas waktu permintaan.
- Permintaan mengembalikan kode status HTTP selain 429, 500-599.

 Note

[Biaya transfer data standar](#) berlaku untuk percobaan ulang.

Lihat juga

- [Bekerja dengan tujuan aturan topik HTTP](#)
- [Rutekan data langsung dari AWS IoT Core ke layanan web Anda](#) di Internet of Things di AWS blog

Bekerja dengan tujuan aturan topik HTTP

Tujuan aturan topik HTTP adalah layanan web tempat mesin aturan dapat merutekan data dari aturan topik. AWS IoT Core Sumber daya menjelaskan layanan web untuk AWS IoT. Sumber daya tujuan aturan topik dapat dibagikan dengan aturan yang berbeda.

Sebelum AWS IoT Core dapat mengirim data ke layanan web lain, itu harus mengkonfirmasi bahwa ia dapat mengakses titik akhir layanan.

Ikhtisar tujuan aturan topik HTTP

Tujuan aturan topik HTTP mengacu pada layanan web yang mendukung URL konfirmasi dan satu atau beberapa URL pengumpulan data. Sumber daya tujuan aturan topik HTTP berisi URL konfirmasi layanan web Anda. Saat mengonfigurasi tindakan aturan topik HTTP, Anda menentukan URL sebenarnya dari titik akhir yang seharusnya menerima data bersama dengan URL konfirmasi layanan web. Setelah tujuan Anda dikonfirmasi, aturan topik mengirimkan hasil pernyataan SQL ke titik akhir HTTPS (dan bukan ke URL konfirmasi).

Tujuan aturan topik HTTP dapat berada di salah satu status berikut:

ENABLED

Tujuan telah dikonfirmasi dan dapat digunakan oleh tindakan aturan. Tujuan harus berada di ENABLED negara bagian agar dapat digunakan dalam suatu aturan. Anda hanya dapat mengaktifkan tujuan yang dalam status DISABLED.

DISABLED

Tujuan telah dikonfirmasi tetapi tidak dapat digunakan oleh tindakan aturan. Ini berguna jika Anda ingin mencegah sementara lalu lintas ke titik akhir Anda tanpa harus melalui proses konfirmasi lagi. Anda hanya dapat menonaktifkan tujuan yang berada dalam status ENABLED.

IN_PROGRESS

Konfirmasi tujuan sedang berlangsung.

ERROR

Waktu konfirmasi tujuan habis.

Setelah tujuan aturan topik HTTP telah dikonfirmasi dan diaktifkan, itu dapat digunakan dengan aturan apa pun di akun Anda.

Bagian berikut menjelaskan tindakan umum pada tujuan aturan topik HTTP.

Membuat dan mengonfirmasi tujuan aturan topik HTTP

Anda membuat tujuan aturan topik HTTP dengan memanggil `CreateTopicRuleDestination` operasi atau dengan menggunakan AWS IoT konsol.

Setelah Anda membuat tujuan, AWS IoT kirimkan permintaan konfirmasi ke URL konfirmasi. Permintaan konfirmasi memiliki format berikut:

```
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"
}
```

Isi permintaan konfirmasi mencakup informasi berikut:

`arn`

Nama Sumber Daya Amazon (ARN) untuk tujuan aturan topik untuk mengonfirmasi.

`confirmationToken`

Token konfirmasi yang dikirim oleh AWS IoT Core. Token dalam contoh terpotong. Token Anda akan lebih lama. Anda memerlukan token ini untuk mengonfirmasi tujuan Anda AWS IoT Core.

`enableUrl`

URL yang Anda telusuri untuk mengonfirmasi tujuan aturan topik.

`messageType`

Jenis pesan.

Untuk menyelesaikan proses konfirmasi endpoint, Anda harus melakukan salah satu hal berikut setelah URL konfirmasi Anda menerima permintaan konfirmasi.

- `enableUrl` Panggil permintaan konfirmasi, lalu panggil `UpdateTopicRuleDestination` untuk mengatur status aturan topik `ENABLED`.
- Panggil `ConfirmTopicRuleDestination` operasi dan meneruskan `confirmationToken` dari permintaan konfirmasi.
- Salin `confirmationToken` dan tempel ke dialog konfirmasi tujuan di AWS IoT konsol.

Mengirim permintaan konfirmasi baru

Untuk mengaktifkan pesan konfirmasi baru untuk tujuan, panggil `UpdateTopicRuleDestination` dan setel status tujuan aturan topik ke `IN_PROGRESS`.

Ulangi proses konfirmasi setelah Anda mengirim permintaan konfirmasi baru.


Menonaktifkan dan menghapus tujuan aturan topik

Untuk menonaktifkan tujuan, panggil `UpdateTopicRuleDestination` dan setel status tujuan aturan topik ke `DISABLED`. Aturan topik dalam status `DISABLED` dapat diaktifkan lagi tanpa perlu mengirim permintaan konfirmasi baru.

Untuk menghapus tujuan aturan topik, hubungi `DeleteTopicRuleDestination`.

Otoritas sertifikat didukung oleh titik akhir HTTPS di tujuan aturan topik

Otoritas sertifikat berikut didukung oleh titik akhir HTTPS di tujuan aturan topik. Anda dapat memilih salah satu dari otoritas sertifikat yang didukung ini. Tanda tangan adalah untuk referensi. Perhatikan bahwa Anda tidak dapat menggunakan sertifikat yang ditandatangani sendiri karena tidak akan berfungsi.

 Bantu kami meningkatkan topik ini

[Beritahu kami pendapat Anda.](#)

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: usertrustsacertificationauthority

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: turktrustelektroniksertifikahizmetisaglayicisi5
```

```
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: certsingrootca

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: tubitakkamussslkoksertifikasisurum1

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:E

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: xrapglobalcaroot
```

```
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: affirmtrustpremiuemecc

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: globalsegrootcar3
```

```
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: globalsegrootcar2
```

```
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: comodoaaca

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:6

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
```

IoT Analytics

Tindakan AWS IoT Analytics (`iotAnalytics`) mengirimkan data dari pesan MQTT ke saluran. AWS IoT Analytics

Persyaratan

Tindakan aturan ini memiliki persyaratan sebagai berikut:

- Peran IAM yang AWS IoT dapat diasumsikan untuk melakukan `iotanalytics:BatchPutMessage` operasi. Untuk informasi selengkapnya, lihat [Memberikan AWS IoT aturan akses yang dibutuhkannya](#).

Di AWS IoT konsol, Anda dapat memilih atau membuat peran untuk memungkinkan AWS IoT untuk melakukan tindakan aturan ini.

Kebijakan yang dilampirkan pada peran yang Anda tentukan akan terlihat seperti contoh berikut.

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

```
"Statement": [  
  {  
    "Effect": "Allow",  
    "Action": "iotanalytics:BatchPutMessage",  
    "Resource": [  
      "arn:aws:iotanalytics:us-west-2:account-id:channel/mychannel"  
    ]  
  }  
]
```

Parameter

Saat Anda membuat AWS IoT aturan dengan tindakan ini, Anda harus menentukan informasi berikut:

batchMode

(Opsional) Apakah akan memproses tindakan sebagai batch. Nilai default-nya adalah `false`.

`batchMode` kapan `true` dan pernyataan aturan SQL mengevaluasi ke Array, setiap elemen Array dikirim sebagai pesan terpisah ketika diteruskan [BatchPutMessage](#) ke saluran AWS IoT Analytics. Array yang dihasilkan tidak dapat memiliki lebih dari 100 pesan.

Mendukung [template substitusi](#): Tidak

channelName

Nama AWS IoT Analytics saluran untuk menulis data.

Mendukung [template substitusi](#): API dan hanya AWS CLI

roleArn

Peran IAM yang memungkinkan akses ke AWS IoT Analytics saluran. Untuk informasi selengkapnya, lihat [Persyaratan](#).

Mendukung [template substitusi](#): Tidak

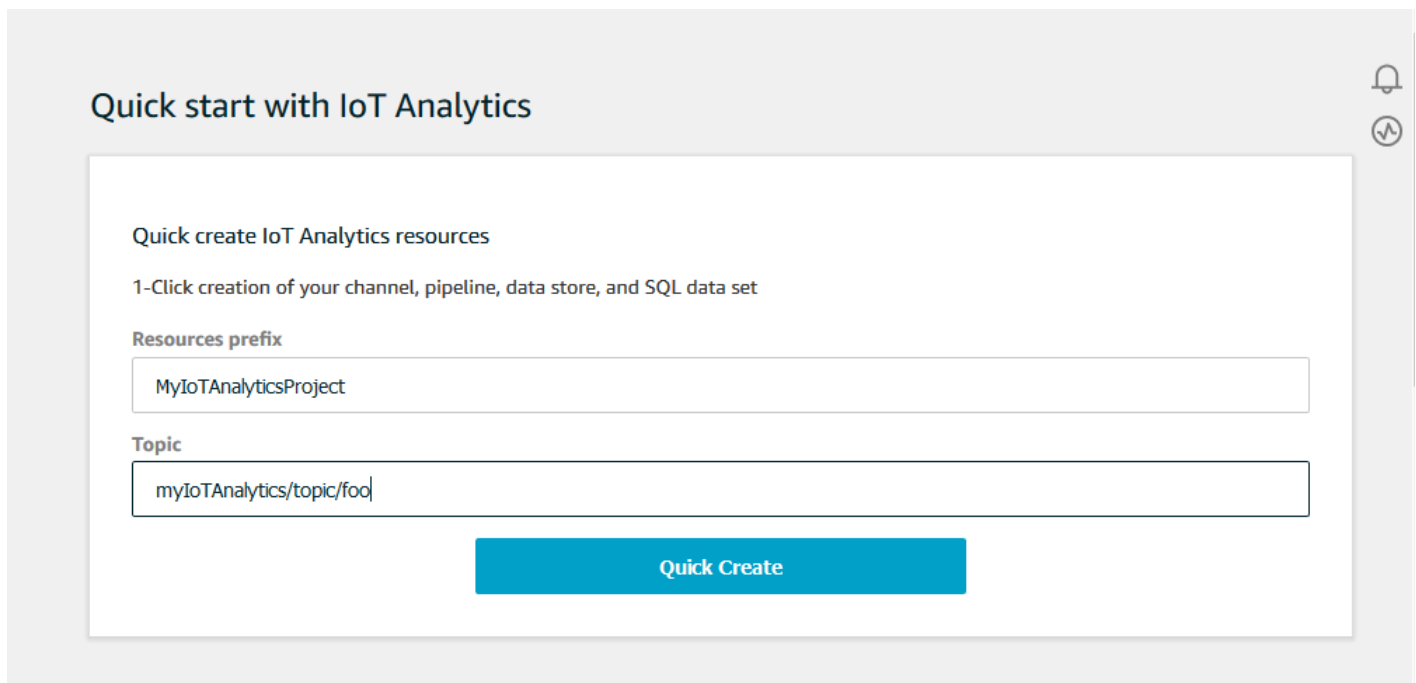
Contoh

Contoh JSON berikut mendefinisikan AWS IoT Analytics tindakan dalam aturan. AWS IoT


```
{
  "topicRulePayload": {
    "sql": "SELECT * FROM 'some/topic'",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [
      {
        "iotAnalytics": {
          "channelName": "mychannel",
          "roleArn": "arn:aws:iam::123456789012:role/analyticsRole",
        }
      }
    ]
  }
}
```

Lihat juga

- [Apa itu AWS IoT Analytics?](#) di Panduan AWS IoT Analytics Pengguna
- AWS IoT Analytics Konsol ini juga memiliki fitur Mulai cepat yang memungkinkan Anda membuat saluran, penyimpanan data, pipeline, dan penyimpanan data dengan satu klik. Untuk informasi selengkapnya, lihat [panduan mulai cepat AWS IoT Analytics konsol](#) di Panduan AWS IoT Analytics Pengguna.



Quick start with IoT Analytics

Quick create IoT Analytics resources

1-Click creation of your channel, pipeline, data store, and SQL data set

Resources prefix

Topic

Quick Create

AWS IoT Events

Tindakan AWS IoT Events (`iotEvents`) mengirimkan data dari pesan MQTT ke input. AWS IoT Events

Important

Jika payload dikirim ke AWS IoT Core tanpa `Input` attribute Key, atau jika kunci tidak berada di jalur JSON yang sama yang ditentukan dalam kunci, itu akan menyebabkan aturan IoT gagal dengan kesalahan. `Failed to send message to Iot Events`

Persyaratan

Tindakan aturan ini memiliki persyaratan sebagai berikut:

- Peran IAM yang AWS IoT dapat diasumsikan untuk melakukan `iotEvents:BatchPutMessage` operasi. Untuk informasi selengkapnya, lihat [Memberikan AWS IoT aturan akses yang dibutuhkannya](#).

Di AWS IoT konsol, Anda dapat memilih atau membuat peran untuk memungkinkan AWS IoT untuk melakukan tindakan aturan ini.

Parameter

Saat Anda membuat AWS IoT aturan dengan tindakan ini, Anda harus menentukan informasi berikut:

`batchMode`

(Opsional) Apakah akan memproses tindakan acara sebagai batch. Nilai default-nya adalah `false`.

`batchModeKapan true` dan pernyataan aturan SQL mengevaluasi ke Array, setiap elemen Array diperlakukan sebagai pesan terpisah ketika dikirim ke AWS IoT Acara dengan memanggil [BatchPutMessage](#) Array yang dihasilkan tidak dapat memiliki lebih dari 10 pesan.

`batchModeKapan true`, Anda tidak dapat menentukan `messageId`.

Mendukung [template substitusi](#): Tidak

inputName

Nama AWS IoT Events input.

Mendukung [template substitusi](#): API dan hanya AWS CLI

messageId

(Opsional) Gunakan ini untuk memverifikasi bahwa hanya satu input (pesan) dengan yang `messageId` diberikan diproses oleh AWS IoT Events detektor. Anda dapat menggunakan template `${newuuid()}` substitusi untuk menghasilkan ID unik untuk setiap permintaan.

Saat `batchMode` adalah `true`, Anda tidak dapat menentukan `messageId`—nilai UUID baru akan diberikan.

Mendukung [template substitusi](#): Ya

roleArn

Peran IAM yang memungkinkan AWS IoT untuk mengirim input ke AWS IoT Events detektor. Untuk informasi selengkapnya, lihat [Persyaratan](#).

Mendukung [template substitusi](#): Tidak

Contoh

Contoh JSON berikut mendefinisikan tindakan IoT Events dalam sebuah aturan. 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"
        }
      }
    ]
  }
}
```

Lihat juga

- [Apa itu AWS IoT Events?](#) di Panduan AWS IoT Events Pengembang

AWS IoT SiteWise

Tindakan AWS IoT SiteWise (`iotSiteWise`) mengirimkan data dari pesan MQTT ke properti aset di AWS IoT SiteWise

Anda dapat mengikuti tutorial yang menunjukkan cara menelan data dari AWS IoT berbagai hal. Untuk informasi selengkapnya, lihat tutorial [Menelan data ke AWS IoT SiteWise dari AWS IoT hal-hal](#) atau bagian [Menyerap data menggunakan aturan AWS IoT Inti](#) di AWS IoT SiteWise Panduan Pengguna.

Persyaratan

Tindakan aturan ini memiliki persyaratan sebagai berikut:

- Peran IAM yang AWS IoT dapat diasumsikan untuk melakukan `iotsitewise:BatchPutAssetPropertyValue` operasi. Untuk informasi selengkapnya, lihat [Memberikan AWS IoT aturan akses yang dibutuhkannya](#).

Anda dapat melampirkan contoh kebijakan kepercayaan berikut ke peran tersebut.

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

Untuk meningkatkan keamanan, Anda dapat menentukan jalur hierarki AWS IoT SiteWise aset di `Condition` properti. Contoh berikut adalah kebijakan kepercayaan yang menentukan jalur hierarki aset.

```
{
```

```

"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Allow",
    "Action": "iotsitewise:BatchPutAssetPropertyValue",
    "Resource": "*",
    "Condition": {
      "StringLike": {
        "iotsitewise:assetHierarchyPath": [
          "/root node asset ID",
          "/root node asset ID/*"
        ]
      }
    }
  }
]
}

```

- Saat Anda mengirim data AWS IoT SiteWise dengan tindakan ini, data Anda harus memenuhi persyaratan BatchPutAssetPropertyValue operasi. Untuk informasi selengkapnya, lihat [BatchPutAssetPropertyNilai](#) dalam Referensi AWS IoT SiteWise API.

Parameter

Saat Anda membuat AWS IoT aturan dengan tindakan ini, Anda harus menentukan informasi berikut:

putAssetPropertyValueEntries

Daftar entri nilai properti aset yang masing-masing berisi informasi berikut:

propertyAlias

(Opsional) Alias properti yang terkait dengan properti aset Anda. Tentukan salah satu `propertyAlias` atau keduanya `assetId` dan `apropertyId`. Untuk informasi selengkapnya tentang alias properti, lihat [Memetakan aliran data industri ke properti aset di Panduan Pengguna.AWS IoT SiteWise](#)

Mendukung [template substitusi](#): Ya

assetId

(Opsional) ID AWS IoT SiteWise aset. Tentukan salah satu `propertyAlias` atau keduanya `assetId` dan `apropertyId`.

Mendukung [template substitusi](#): Ya

propertyId

(Opsional) ID properti aset. Tentukan salah satu `propertyAlias` atau keduanya `assetId` dan `apropertyId`.

Mendukung [template substitusi](#): Ya

entryId

(Opsional) Pengidentifikasi unik untuk entri ini. Tentukan `entryId` untuk melacak pesan mana yang menyebabkan kesalahan dengan lebih baik jika terjadi kegagalan. Default untuk UUID baru.

Mendukung [template substitusi](#): Ya

propertyValues

Daftar nilai properti untuk disisipkan yang masing-masing berisi stempel waktu, kualitas, dan nilai (TQV) dalam format berikut:

timestamp

Struktur stempel waktu yang berisi informasi berikut:

timeInSeconds

Sebuah string yang berisi waktu dalam detik dalam waktu epoch Unix. Jika payload pesan Anda tidak memiliki stempel waktu, Anda dapat menggunakan [stempel waktu \(\)](#), yang mengembalikan waktu saat ini dalam milidetik. Untuk mengonversi waktu itu menjadi detik, Anda dapat menggunakan templat substitusi berikut: `${floor(timestamp() / 1E3)}`.

Mendukung [template substitusi](#): Ya

offsetInNanos

(Opsional) String yang berisi offset waktu nanodetik dari waktu dalam detik. Jika payload pesan Anda tidak memiliki stempel waktu, Anda dapat menggunakan [stempel waktu \(\)](#), yang mengembalikan waktu saat ini dalam milidetik. Untuk menghitung offset nanodetik sejak saat itu, Anda dapat menggunakan templat substitusi berikut: `${(timestamp() % 1E3) * 1E6}`

Mendukung [template substitusi](#): Ya

Mengenai waktu epoch Unix, hanya AWS IoT SiteWise menerima entri yang memiliki stempel waktu hingga 7 hari di masa lalu hingga 5 menit di masa depan.

quality

(Opsional) String yang menggambarkan kualitas nilai. Nilai-nilai yang valid: GOOD, BAD, UNCERTAIN.

Mendukung [template substitusi](#): Ya

value

Struktur nilai yang berisi salah satu bidang nilai berikut, tergantung pada tipe data properti aset:

booleanValue

(Opsional) Sebuah string yang berisi nilai Boolean dari entri nilai.

Mendukung [template substitusi](#): Ya

doubleValue

(Opsional) String yang berisi nilai ganda dari entri nilai.

Mendukung [template substitusi](#): Ya

integerValue

(Opsional) Sebuah string yang berisi nilai integer dari entri nilai.

Mendukung [template substitusi](#): Ya

stringValue

(Opsional) Nilai string dari entri nilai.

Mendukung [template substitusi](#): Ya

roleArn

ARN dari peran IAM yang memberikan AWS IoT izin untuk mengirim nilai properti aset ke. AWS IoT SiteWise Untuk informasi selengkapnya, lihat [Persyaratan](#).

Mendukung [template substitusi](#): Tidak

Contoh

Contoh JSON berikut mendefinisikan SiteWise tindakan IoT dasar dalam sebuah aturan. AWS IoT

```
{
  "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_iam_siteWise"
        }
      ]
    ]
  }
}
```

Contoh JSON berikut mendefinisikan SiteWise tindakan IoT dalam sebuah aturan. AWS IoT Contoh ini menggunakan topik sebagai alias properti dan `timestamp()` fungsi. Misalnya, jika Anda mempublikasikan data ke `/company/windfarm/3/turbine/7/rpm`, tindakan ini mengirimkan data ke properti aset dengan alias properti yang sama dengan topik yang Anda tentukan.

```
{
  "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"
    }
  }
]
}

```

Lihat juga

- [Apa itu AWS IoT SiteWise?](#) dalam AWS IoT SiteWise Panduan Pengguna
- [Menelan data menggunakan AWS IoT Core aturan](#) di AWS IoT SiteWise Panduan Pengguna
- [Menelan data AWS IoT SiteWise dari AWS IoT hal-hal](#) di AWS IoT SiteWise Panduan Pengguna
- [Memecahkan masalah tindakan AWS IoT SiteWise aturan](#) di Panduan Pengguna AWS IoT SiteWise

Firehose

Tindakan Firehose (`firehose`) mengirimkan data dari pesan MQTT ke aliran Amazon Data Firehose.

Persyaratan

Tindakan aturan ini memiliki persyaratan sebagai berikut:

- Peran IAM yang AWS IoT dapat diasumsikan untuk melakukan `firehose:PutRecord` operasi. Untuk informasi selengkapnya, lihat [Memberikan AWS IoT aturan akses yang dibutuhkannya](#).

Di AWS IoT konsol, Anda dapat memilih atau membuat peran untuk memungkinkan AWS IoT untuk melakukan tindakan aturan ini.

- Jika Anda menggunakan Firehose untuk mengirim data ke bucket Amazon S3, dan Anda menggunakan pelanggan AWS KMS yang AWS KMS key berhasil mengenkripsi data saat istirahat di Amazon S3, Firehose harus memiliki akses ke bucket Anda dan izin untuk menggunakan atas nama pemanggil. AWS KMS key Untuk informasi selengkapnya, lihat [Memberikan akses Firehose ke tujuan Amazon S3](#) di Panduan Pengembang Amazon Data Firehose.

Parameter

Saat Anda membuat AWS IoT aturan dengan tindakan ini, Anda harus menentukan informasi berikut:

`batchMode`

(Opsional) Apakah akan mengirimkan aliran Firehose sebagai batch dengan menggunakan [PutRecordBatch](#) Nilai default-nya adalah `false`.

`batchMode`Kapan `true` dan pernyataan SQL aturan mengevaluasi ke Array, setiap elemen Array membentuk satu catatan dalam permintaan. `PutRecordBatch` Array yang dihasilkan tidak dapat memiliki lebih dari 500 record.

Mendukung [template substitusi](#): Tidak

`deliveryStreamName`

Aliran Firehose untuk menulis data pesan.

Mendukung [template substitusi](#): API dan hanya AWS CLI

`separator`

(Opsional) Pemisah karakter yang digunakan untuk memisahkan catatan yang ditulis ke aliran Firehose. Jika Anda menghilangkan parameter ini, aliran tidak menggunakan pemisah. Nilai yang valid: `,` (koma), `\t` (tab), `\n` (baris baru), `\r\n` (baris baru Windows).

Mendukung [template substitusi](#): Tidak
roleArn

Peran IAM yang memungkinkan akses ke aliran Firehose. Untuk informasi selengkapnya, lihat [Persyaratan](#).

Mendukung [template substitusi](#): Tidak

Contoh

Contoh JSON berikut mendefinisikan tindakan Firehose dalam aturan. AWS IoT

```
{
  "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_iam_firehose"
        }
      }
    ]
  }
}
```

Contoh JSON berikut mendefinisikan tindakan Firehose dengan template substitusi dalam aturan. AWS IoT

```
{
  "topicRulePayload": {
    "sql": "SELECT * FROM 'some/topic'",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [
      {
        "firehose": {
          "deliveryStreamName": "${topic()}",
          "roleArn": "arn:aws:iam::123456789012:role/aws_iam_firehose"
        }
      }
    ]
  }
}
```

```
}  
  }  
] }  
} }
```

Lihat juga

- [Apa itu Amazon Data Firehose?](#) di Panduan Pengembang Firehose Data Amazon

Kinesis Data Streams

Tindakan Kinesis Data `kinesis Streams ()` menulis data dari pesan MQTT ke Amazon Kinesis Data Streams.

Persyaratan

Tindakan aturan ini memiliki persyaratan sebagai berikut:

- Peran IAM yang AWS IoT dapat diasumsikan untuk melakukan `kinesis:PutRecord` operasi. Untuk informasi selengkapnya, lihat [Memberikan AWS IoT aturan akses yang dibutuhkannya](#).

Di AWS IoT konsol, Anda dapat memilih atau membuat peran untuk memungkinkan AWS IoT untuk melakukan tindakan aturan ini.

- Jika Anda menggunakan kunci AWS KMS key KMS yang AWS KMS dikelola pelanggan untuk mengenkripsi data saat istirahat di Kinesis Data Streams, layanan harus memiliki izin untuk menggunakan atas nama pemanggil. AWS KMS key Untuk informasi selengkapnya, lihat [Izin untuk menggunakan buatan pengguna AWS KMS keys di Panduan Pengembang](#) Amazon Kinesis Data Streams.

Parameter

Saat Anda membuat AWS IoT aturan dengan tindakan ini, Anda harus menentukan informasi berikut:

`stream`

Aliran data Kinesis untuk menulis data.

Mendukung [template substitusi](#): API dan hanya AWS CLI

partitionKey

Kunci partisi yang digunakan untuk menentukan pecahan data mana yang ditulis. Kunci partisi biasanya terdiri dari ekspresi (misalnya, `${topic()}` atau `${timestamp()}`).

Mendukung [template substitusi](#): Ya

roleArn

ARN dari peran IAM yang memberikan AWS IoT izin untuk mengakses aliran data Kinesis. Untuk informasi selengkapnya, lihat [Persyaratan](#).

Mendukung [template substitusi](#): Tidak

Contoh

Contoh JSON berikut mendefinisikan tindakan Kinesis Data Streams dalam sebuah aturan. AWS IoT

```
{
  "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"
        }
      }
    ]
  }
}
```

Contoh JSON berikut mendefinisikan tindakan Kinesis dengan template substitusi dalam sebuah aturan. AWS IoT

```
{
  "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"
    }
  }
]
```

Lihat juga

- [Apa itu Amazon Kinesis Data Streams?](#) di Panduan Pengembang Amazon Kinesis Data Streams

Lambda

Tindakan Lambda (lambda) memanggil AWS Lambda fungsi, meneruskan pesan MQTT. AWS IoT memanggil fungsi Lambda secara asinkron.

Anda dapat mengikuti tutorial yang menunjukkan cara membuat dan menguji aturan dengan tindakan Lambda. Untuk informasi selengkapnya, lihat [Tutorial: Memformat notifikasi dengan menggunakan fungsi AWS Lambda](#).

Persyaratan

Tindakan aturan ini memiliki persyaratan sebagai berikut:

- AWS IoT Untuk menjalankan fungsi Lambda, Anda harus mengonfigurasi kebijakan yang memberikan izin `lambda:InvokeFunction`. AWS IoT Anda hanya dapat menjalankan fungsi Lambda yang ditentukan di tempat yang Wilayah AWS sama di mana kebijakan Lambda Anda ada. Fungsi Lambda menggunakan kebijakan berbasis sumber daya, jadi Anda harus melampirkan kebijakan ke fungsi Lambda itu sendiri.

Gunakan AWS CLI perintah berikut untuk melampirkan kebijakan yang memberikan `lambda:InvokeFunction` izin.

```
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"
```

add-permission Perintah mengharapkan parameter berikut:

--function-name

Nama fungsi Lambda. Anda menambahkan izin baru untuk memperbarui kebijakan sumber daya fungsi.

--region

Wilayah AWS Fungsi.

--principal

Kepala sekolah yang mendapat izin. Ini harus memungkinkan AWS IoT izin `iot.amazonaws.com` untuk memanggil fungsi Lambda.

--source-arn

ARN dari aturan tersebut. Anda dapat menggunakan `get-topic-rule` AWS CLI perintah untuk mendapatkan ARN aturan.

--source-account

Di Akun AWS mana aturan didefinisikan.

--statement-id

Pengidentifikasi pernyataan unik.

--action

Tindakan Lambda yang ingin Anda izinkan dalam pernyataan ini. AWS IoT Untuk memungkinkan menjalankan fungsi Lambda, tentukan. `lambda:InvokeFunction`

Important

Jika Anda menambahkan izin untuk AWS IoT prinsipal tanpa memberikan `source-arn` atau `source-account`, apa pun Akun AWS yang membuat aturan dengan tindakan

Lambda Anda dapat mengaktifkan aturan untuk menjalankan fungsi Lambda Anda. AWS IoT

Untuk informasi selengkapnya, lihat [izin AWS Lambda](#).

- Jika Anda menggunakan AWS KMS pelanggan yang AWS KMS key berhasil mengenkripsi data saat istirahat di Lambda, layanan harus memiliki izin untuk menggunakan atas AWS KMS key nama penelepon. Untuk informasi lebih lanjut, lihat [Enkripsi saat diam](#) di Panduan Developer AWS Lambda .

Parameter

Saat Anda membuat AWS IoT aturan dengan tindakan ini, Anda harus menentukan informasi berikut:

functionArn

ARN dari fungsi Lambda untuk memanggil. AWS IoT harus memiliki izin untuk menjalankan fungsi. Untuk informasi selengkapnya, lihat [Persyaratan](#).

Jika Anda tidak menentukan versi atau alias untuk fungsi Lambda Anda, versi terbaru dari fungsi tersebut dimatikan. Anda dapat menentukan versi atau alias jika Anda ingin mematikan versi tertentu dari fungsi Lambda Anda. Untuk menentukan versi atau alias, tambahkan versi atau alias ke ARN dari fungsi Lambda.

```
arn:aws:lambda:us-east-2:123456789012:function:myLambdaFunction:someAlias
```

Untuk informasi selengkapnya tentang pembuatan versi dan alias, dan lihat [versi AWS Lambda fungsi](#) dan alias.

Mendukung [template substitusi](#): API dan hanya AWS CLI

Contoh

Contoh JSON berikut mendefinisikan tindakan Lambda dalam aturan. AWS IoT

```
{
  "topicRulePayload": {
    "sql": "SELECT * FROM 'some/topic'",
  }
}
```



```

    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [
      {
        "lambda": {
          "functionArn": "arn:aws:lambda:us-
east-2:123456789012:function:myLambdaFunction"
        }
      }
    ]
  }
}

```

Contoh JSON berikut mendefinisikan tindakan Lambda dengan template substitusi dalam aturan. AWS IoT

```

{
  "topicRulePayload": {
    "sql": "SELECT * FROM 'some/topic'",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [
      {
        "lambda": {
          "functionArn": "arn:aws:lambda:us-east-1:123456789012:function:
${topic()}"
        }
      }
    ]
  }
}

```

Lihat juga

- [Apa itu AWS Lambda?](#) di Panduan AWS Lambda Pengembang
- [Tutorial: Memformat notifikasi dengan menggunakan fungsi AWS Lambda](#)

Lokasi

Tindakan Location (location) mengirimkan data lokasi geografis Anda ke [Amazon Location Service](#).

Persyaratan

Tindakan aturan ini memiliki persyaratan sebagai berikut:

- Peran IAM yang AWS IoT dapat diasumsikan untuk melakukan `geo:BatchUpdateDevicePosition` operasi. Untuk informasi selengkapnya, lihat [Memberikan AWS IoT aturan akses yang dibutuhkannya](#).

Di AWS IoT konsol, Anda dapat memilih atau membuat peran untuk memungkinkan AWS IoT untuk melakukan tindakan aturan ini.

Parameter

Saat Anda membuat AWS IoT aturan dengan tindakan ini, Anda harus menentukan informasi berikut:

`deviceId`

ID unik perangkat yang menyediakan data lokasi. Untuk informasi selengkapnya, lihat [DeviceId](#) dari Referensi API Amazon Location Service.

Mendukung [template substitusi](#): Ya

`latitude`

String yang mengevaluasi nilai ganda yang mewakili garis lintang lokasi perangkat.

Mendukung [template substitusi](#): Ya

`longitude`

String yang mengevaluasi nilai ganda yang mewakili bujur lokasi perangkat.

Mendukung [template substitusi](#): Ya

`roleArn`

Peran IAM yang memungkinkan akses ke domain Amazon Location Service. Untuk informasi selengkapnya, lihat [Persyaratan](#).

`timestamp`

Waktu data lokasi diambil sampelnya. Nilai default adalah waktu pesan MQTT diproses.

`timestamp` Nilai terdiri dari dua nilai berikut:

- `value`: Ekspresi yang mengembalikan nilai waktu epoch yang panjang. Anda dapat menggunakan [the section called "time_to_epoch \(String, String\)"](#) fungsi ini untuk membuat stempel waktu yang valid dari nilai tanggal atau waktu yang diteruskan dalam payload pesan. Mendukung [template substitusi](#): Ya.
- `unit`: (Opsional) Ketepatan nilai stempel waktu yang dihasilkan dari ekspresi yang dijelaskan dalam `value`. Nilai yang valid: SECONDS | MILLISECONDS | MICROSECONDS | NANOSECONDS. Nilai default-nya MILLISECONDS. Mendukung [template substitusi](#): API dan AWS CLI hanya.

trackerName

Nama sumber daya pelacak di Amazon Lokasi di mana lokasi diperbarui. Untuk informasi selengkapnya, lihat [Tracker](#) dari Amazon Location Service Developer Guide.

Mendukung [template substitusi](#): API dan hanya AWS CLI

Contoh

Contoh JSON berikut mendefinisikan tindakan Lokasi dalam aturan. AWS IoT

```
{
  "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"
        }
      }
    ]
  }
}
```

Contoh JSON berikut mendefinisikan tindakan Lokasi dengan template substitusi dalam aturan. AWS IoT

```
{
  "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)}"
        }
      }
    ]
  }
}
```

Contoh payload MQTT berikut menunjukkan bagaimana template substitusi dalam contoh sebelumnya mengakses data. Anda dapat menggunakan perintah [get-device-position-history](#) CLI untuk memverifikasi bahwa data payload MQTT dikirimkan di pelacak lokasi Anda.

```
{
  "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"
    }
  ]
}
```

Lihat juga

- [Apa itu Amazon Location Service?](#) di Panduan Pengembang Layanan Lokasi Amazon.

OpenSearch

Tindakan OpenSearch (openSearch) menulis data dari pesan MQTT ke domain Layanan Amazon OpenSearch. Anda kemudian dapat menggunakan alat seperti OpenSearch Dasbor untuk menanyakan dan memvisualisasikan data di OpenSearch Layanan.

Persyaratan

Tindakan aturan ini memiliki persyaratan sebagai berikut:

- Peran IAM yang AWS IoT dapat diasumsikan untuk melakukan es:ESHttpPost operasi. Untuk informasi selengkapnya, lihat [Memberikan AWS IoT aturan akses yang dibutuhkannya](#).

Di AWS IoT konsol, Anda dapat memilih atau membuat peran untuk memungkinkan AWS IoT untuk melakukan tindakan aturan ini.

- Jika Anda menggunakan pelanggan yang AWS KMS key berhasil mengenkripsi data saat istirahat di OpenSearch Layanan, layanan harus memiliki izin untuk menggunakan kunci KMS atas nama pemanggil. Untuk informasi selengkapnya, lihat [Enkripsi data saat istirahat untuk OpenSearch Layanan Amazon](#) di Panduan Pengembang OpenSearch Layanan Amazon.

Parameter

Saat Anda membuat AWS IoT aturan dengan tindakan ini, Anda harus menentukan informasi berikut:

endpoint

Titik akhir domain OpenSearch Layanan Amazon Anda.

Mendukung [template substitusi](#): API dan hanya AWS CLI

index

OpenSearch Indeks tempat Anda ingin menyimpan data Anda.

Mendukung [template substitusi](#): Ya

type

Jenis dokumen yang Anda simpan.

Note

Untuk OpenSearch versi yang lebih lambat dari 1.0, nilai `type` parameter harus `_doc`. Untuk informasi lebih lanjut, lihat [OpenSearch dokumentasi](#).

Mendukung [template substitusi](#): Ya

id

Pengidentifikasi unik untuk setiap dokumen.

Mendukung [template substitusi](#): Ya

roleARN

Peran IAM yang memungkinkan akses ke domain OpenSearch Layanan. Untuk informasi selengkapnya, lihat [Persyaratan](#).

Mendukung [template substitusi](#): Tidak

Batasan

Tindakan OpenSearch (`openSearch`) tidak dapat digunakan untuk mengirimkan data ke kluster VPC Elasticsearch.

Contoh

Contoh JSON berikut mendefinisikan OpenSearch tindakan dalam AWS IoT aturan dan bagaimana Anda dapat menentukan bidang untuk tindakan. OpenSearch Untuk informasi selengkapnya, lihat [OpenSearchTindakan](#).

```
{
  "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"
        }
      }
    ]
  }
}
```

Contoh JSON berikut mendefinisikan OpenSearch tindakan dengan template substitusi dalam aturan. AWS IoT

```
{
  "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

`typeBidang` yang diganti berfungsi untuk OpenSearch versi 1.0. Untuk versi apa pun yang lebih lambat dari 1.0, nilai `type` harus `_doc`.

Lihat juga

[Apa itu OpenSearch Layanan Amazon?](#) di Panduan Pengembang OpenSearch Layanan Amazon

Publikasikan ulang

Tindakan `republish` (`republish`) menerbitkan kembali pesan MQTT ke topik MQTT lainnya.

Persyaratan

Tindakan aturan ini memiliki persyaratan sebagai berikut:

- Peran IAM yang AWS IoT dapat diasumsikan untuk melakukan `iot:Publish` operasi. Untuk informasi selengkapnya, lihat [Memberikan AWS IoT aturan akses yang dibutuhkannya](#).

Di AWS IoT konsol, Anda dapat memilih atau membuat peran untuk memungkinkan AWS IoT untuk melakukan tindakan aturan ini.

Parameter

Saat Anda membuat AWS IoT aturan dengan tindakan ini, Anda harus menentukan informasi berikut:

`headers`


Informasi header MQTT Versi 5.0.

Untuk informasi selengkapnya, lihat [RepublishAction](#) dan [MqttHeaders](#) di Referensi AWS API.

`topic`

Topik MQTT untuk menerbitkan ulang pesan.

Untuk menerbitkan ulang ke topik yang dicadangkan, yang dimulai dengan\$, gunakan \$\$ sebagai gantinya. Misalnya, untuk menerbitkan ulang ke topik bayangan perangkat\$aws/things/MyThing/shadow/update, tentukan topik sebagai\$\$aws/things/MyThing/shadow/update.

 Note

Penerbitan ulang ke [topik pekerjaan yang dipesan](#) tidak didukung.
AWS IoT Device Defender topik cadangan tidak mendukung publikasi HTTP.

Mendukung [template substitusi](#): Ya

qos

(Opsional) Tingkat Kualitas Layanan (QoS) yang akan digunakan saat menerbitkan ulang pesan. Nilai-nilai yang valid: 0, 1. Nilai default-nya adalah 0. Untuk informasi selengkapnya tentang QoS MQTT, lihat. [MQTT](#)

Mendukung [template substitusi](#): Tidak

roleArn

Peran IAM yang memungkinkan AWS IoT untuk mempublikasikan ke topik MQTT. Untuk informasi selengkapnya, lihat [Persyaratan](#).

Mendukung [template substitusi](#): Tidak

Contoh

Contoh JSON berikut mendefinisikan tindakan republish dalam aturan. 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"
    }
  ]
}
}

```

Contoh JSON berikut mendefinisikan tindakan republish dengan template substitusi dalam aturan. AWS IoT

```

{
  "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"
        }
      }
    ]
  }
}

```

Contoh JSON berikut mendefinisikan tindakan republish dengan headers dalam aturan. AWS IoT

```

{
  "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

IP sumber asli tidak akan diteruskan melalui [tindakan Republish](#).

S3

Tindakan S3 (`s3`) menulis data dari pesan MQTT ke bucket Amazon Simple Storage Service (Amazon S3).

Persyaratan

Tindakan aturan ini memiliki persyaratan sebagai berikut:

- Peran IAM yang AWS IoT dapat diasumsikan untuk melakukan `s3:PutObject` operasi. Untuk informasi selengkapnya, lihat [Memberikan AWS IoT aturan akses yang dibutuhkannya](#).

Di AWS IoT konsol, Anda dapat memilih atau membuat peran untuk memungkinkan AWS IoT untuk melakukan tindakan aturan ini.

- Jika Anda menggunakan AWS KMS customermanaged AWS KMS key untuk mengenkripsi data saat istirahat di Amazon S3, layanan harus memiliki izin untuk menggunakan AWS KMS key atas nama pemanggil. Untuk informasi selengkapnya, lihat [AWS dikelola AWS KMS keys dan dikelola pelanggan AWS KMS keys](#) di Panduan Pengembang Layanan Penyimpanan Sederhana Amazon.

Parameter

Saat Anda membuat AWS IoT aturan dengan tindakan ini, Anda harus menentukan informasi berikut:

bucket

Bucket Amazon S3 untuk menulis data.

Mendukung [template substitusi](#): API dan hanya AWS CLI

cannedacl

(Opsional) Amazon S3 memasukkan ACL yang mengontrol akses ke objek yang diidentifikasi oleh kunci objek. Untuk informasi selengkapnya, termasuk nilai yang diizinkan, lihat [ACL Kalengan](#).

Mendukung [template substitusi](#): Tidak

key

Jalur ke file tempat data ditulis.

Pertimbangkan contoh di mana parameter ini berada `${topic()}/${timestamp()}` dan aturan menerima pesan di mana topiknya beradasome/topic. Jika stempel waktu saat ini1460685389, maka tindakan ini menulis data ke file yang dipanggil 1460685389 di some/topic folder bucket S3.

Note

Jika Anda menggunakan kunci statis, AWS IoT timpa satu file setiap kali aturan dipanggil. Kami menyarankan Anda menggunakan stempel waktu pesan atau pengenal pesan unik lainnya sehingga file baru disimpan di Amazon S3 untuk setiap pesan yang diterima.

Mendukung [template substitusi](#): Ya

roleArn

Peran IAM yang memungkinkan akses ke bucket Amazon S3. Untuk informasi selengkapnya, lihat [Persyaratan](#).

Mendukung [template substitusi](#): Tidak

Contoh

Contoh JSON berikut mendefinisikan tindakan S3 dalam aturan. AWS IoT

```
{
  "topicRulePayload": {
    "sql": "SELECT * FROM 'some/topic'",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [
      {
        "s3": {
          "bucketName": "my-bucket",
          "cannedacl": "public-read",
          "key": "${topic()}/${timestamp()}",
          "roleArn": "arn:aws:iam::123456789012:role/aws_iot_s3"
        }
      }
    ]
  }
}
```

Lihat juga

- [Apa itu Amazon S3?](#) di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon

Salesforce IoT

Tindakan Salesforce IoT (`salesforce`) mengirimkan data dari pesan MQTT yang memicu aturan ke aliran input IoT Salesforce.

Parameter

Saat Anda membuat AWS IoT aturan dengan tindakan ini, Anda harus menentukan informasi berikut:

`url`

URL yang diekspos oleh aliran input IoT Salesforce. URL tersedia dari platform IoT Salesforce saat Anda membuat aliran input. Untuk informasi selengkapnya, lihat dokumentasi IoT Salesforce.

Mendukung [template substitusi](#): Tidak

token

Token yang digunakan untuk mengautentikasi akses ke aliran input IoT Salesforce yang ditentukan. Token tersedia dari platform IoT Salesforce saat Anda membuat aliran input. Untuk informasi selengkapnya, lihat dokumentasi IoT Salesforce.

Mendukung [template substitusi](#): Tidak

Contoh

Contoh JSON berikut mendefinisikan tindakan IoT Salesforce dalam sebuah aturan. AWS IoT

```
{
  "topicRulePayload": {
    "sql": "SELECT * FROM 'some/topic'",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [
      {
        "salesforce": {
          "token": "ABCDEFGHII123456789abcdefghi123456789",
          "url": "https://ingestion-cluster-id.my-env.sfdcnw.com/streams/
stream-id/connection-id/my-event"
        }
      }
    ]
  }
}
```

SNS

Tindakan SNS (sns) mengirimkan data dari pesan MQTT sebagai notifikasi push Amazon Simple Notification Service (Amazon SNS).

Anda dapat mengikuti tutorial yang menunjukkan cara membuat dan menguji aturan dengan tindakan SNS. Untuk informasi selengkapnya, lihat [Tutorial: Mengirim notifikasi Amazon SNS](#).

Note

Tindakan SNS tidak mendukung topik [Amazon SNS FIFO \(First-In-First-Out\)](#). Karena mesin aturan adalah layanan terdistribusi penuh, tidak ada jaminan pesanan pesan ketika tindakan SNS dipanggil.

Persyaratan

Tindakan aturan ini memiliki persyaratan sebagai berikut:

- Peran IAM yang AWS IoT dapat diasumsikan untuk melakukan `sns:Publish` operasi. Untuk informasi selengkapnya, lihat [Memberikan AWS IoT aturan akses yang dibutuhkannya](#).

Di AWS IoT konsol, Anda dapat memilih atau membuat peran untuk memungkinkan AWS IoT untuk melakukan tindakan aturan ini.

- Jika Anda menggunakan pengelolaan-dikelola AWS KMS pelanggan AWS KMS key untuk mengenkripsi data saat istirahat di Amazon SNS, layanan harus memiliki izin untuk menggunakan AWS KMS key atas nama pemanggil. Untuk informasi selengkapnya, lihat [Manajemen kunci](#) di Panduan Pengembang Layanan Pemberitahuan Sederhana Amazon.

Parameter

Saat Anda membuat AWS IoT aturan dengan tindakan ini, Anda harus menentukan informasi berikut:

`targetArn`

Topik SNS atau perangkat individual tempat notifikasi push dikirim.

Mendukung [template substitusi](#): API dan hanya AWS CLI

`messageFormat`

(Opsional) Format pesan. Amazon SNS menggunakan pengaturan ini untuk menentukan apakah payload harus diuraikan dan apakah bagian khusus platform yang relevan dari payload harus diekstraksi. Nilai-nilai yang valid: JSON, RAW. Default ke RAW.

Mendukung [template substitusi](#): Tidak

`roleArn`

Peran IAM yang memungkinkan akses ke SNS. Untuk informasi selengkapnya, lihat [Persyaratan](#).

Mendukung [template substitusi](#): Tidak

Contoh

Contoh JSON berikut mendefinisikan tindakan SNS dalam aturan. AWS IoT

```
{
  "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"
        }
      }
    ]
  }
}
```

Contoh JSON berikut mendefinisikan tindakan SNS dengan template substitusi dalam aturan. AWS IoT

```
{
  "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"
        }
      }
    ]
  }
}
```


Lihat juga

- [Apakah Amazon Simple Notification Service?](#) dalam Panduan Developer Amazon Simple Notification Service
- [Tutorial: Mengirim notifikasi Amazon SNS](#)

SQS

Tindakan SQS (sqs) mengirimkan data dari pesan MQTT ke antrian Amazon Simple Queue Service (Amazon SQS).

Note

Tindakan SQS tidak mendukung antrian [Amazon SQS FIFO \(First-In-First-Out\)](#). Karena mesin aturan adalah layanan terdistribusi penuh, tidak ada jaminan urutan pesan ketika tindakan SQS dipicu.

Persyaratan

Tindakan aturan ini memiliki persyaratan sebagai berikut:

- Peran IAM yang AWS IoT dapat diasumsikan untuk melakukan sqs :SendMessage operasi. Untuk informasi selengkapnya, lihat [Memberikan AWS IoT aturan akses yang dibutuhkannya](#).

Di AWS IoT konsol, Anda dapat memilih atau membuat peran untuk memungkinkan AWS IoT untuk melakukan tindakan aturan ini.

- Jika Anda menggunakan AWS KMS pelanggan yang AWS KMS key berhasil mengenkripsi data saat istirahat di Amazon SQS, layanan harus memiliki izin untuk menggunakan atas AWS KMS key nama pemanggil. Untuk informasi selengkapnya, lihat [Manajemen kunci](#) di Panduan Pengembang Layanan Antrian Sederhana Amazon.

Parameter

Saat Anda membuat AWS IoT aturan dengan tindakan ini, Anda harus menentukan informasi berikut:

queueUrl

URL antrian Amazon SQS untuk menulis data. Wilayah di URL ini tidak harus Wilayah AWS sama dengan [AWS IoT aturan](#) Anda.

Note

Mungkin ada biaya tambahan untuk transfer data silang Wilayah AWS menggunakan tindakan aturan SQS. Untuk informasi selengkapnya, lihat [harga Amazon SQS](#).

Mendukung [template substitusi](#): API dan hanya AWS CLI

useBase64

Setel parameter ini `true` untuk mengonfigurasi tindakan aturan ke base64-menyandikan data pesan sebelum menulis data ke antrian Amazon SQS. Default ke `false`.

Mendukung [template substitusi](#): Tidak

roleArn

Peran IAM yang memungkinkan akses ke antrian Amazon SQS. Untuk informasi selengkapnya, lihat [Persyaratan](#).

Mendukung [template substitusi](#): Tidak

Contoh

Contoh JSON berikut mendefinisikan tindakan SQS dalam aturan. AWS IoT

```
{
  "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"
      }
    ]
  }
}

```

Contoh JSON berikut mendefinisikan tindakan SQS dengan template substitusi dalam aturan. AWS IoT

```

{
  "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"
        }
      }
    ]
  }
}

```

Lihat juga

- [Apa itu Layanan Antrian Sederhana Amazon?](#) di Panduan Pengembang Layanan Antrian Sederhana Amazon

Step Functions

Tindakan Step Functions (`stepFunctions`) memulai mesin AWS Step Functions status.

Persyaratan

Tindakan aturan ini memiliki persyaratan sebagai berikut:

- Peran IAM yang AWS IoT dapat diasumsikan untuk melakukan `states:StartExecution` operasi. Untuk informasi selengkapnya, lihat [Memberikan AWS IoT aturan akses yang dibutuhkannya](#).

Di AWS IoT konsol, Anda dapat memilih atau membuat peran untuk memungkinkan AWS IoT untuk melakukan tindakan aturan ini.

Parameter

Saat Anda membuat AWS IoT aturan dengan tindakan ini, Anda harus menentukan informasi berikut:

`stateMachineName`

Nama mesin status Step Functions untuk memulai.

Mendukung [template substitusi](#): API dan hanya AWS CLI

`executionNamePrefix`

(Opsional) Nama yang diberikan untuk eksekusi mesin negara terdiri dari awalan ini diikuti oleh UUID. Step Functions membuat nama unik untuk setiap eksekusi mesin negara jika tidak disediakan.

Mendukung [template substitusi](#): Ya

`roleArn`

ARN dari peran yang memberikan AWS IoT izin untuk memulai mesin negara. Untuk informasi selengkapnya, lihat [Persyaratan](#).

Mendukung [template substitusi](#): Tidak

Contoh

Contoh JSON berikut mendefinisikan tindakan Step Functions dalam aturan. AWS IoT

```
{
  "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_iam_step_functions"
  }
}
]
```

Lihat juga

- [Apa itu AWS Step Functions?](#) di Panduan AWS Step Functions Pengembang

Timestream

Tindakan aturan Timestream menulis atribut (ukuran) dari pesan MQTT ke dalam tabel Amazon Timestream. Untuk informasi selengkapnya tentang Amazon Timestream, lihat [Apa itu Amazon Timestream?](#).

Note

Amazon Timestream tidak tersedia di semua Wilayah AWS s. Jika Amazon Timestream tidak tersedia di Wilayah Anda, Amazon Timestream tidak akan muncul dalam daftar tindakan aturan.

Atribut yang disimpan aturan ini dalam database Timestream adalah atribut yang dihasilkan dari pernyataan kueri aturan. Nilai setiap atribut dalam hasil pernyataan query diuraikan untuk menyimpulkan tipe datanya (seperti dalam [the section called “DynamoDBv2”](#) tindakan). Setiap nilai atribut ditulis ke catatannya sendiri di tabel Timestream. Untuk menentukan atau mengubah tipe data atribut, gunakan [cast\(\)](#) fungsi dalam pernyataan query. Untuk informasi selengkapnya tentang isi setiap catatan Timestream, lihat [the section called “Konten rekaman Timestream”](#).

Note

Dengan SQL V2 (2016-03-23), nilai numerik yang merupakan bilangan bulat, seperti `10.0`, dikonversi representasi Integer mereka (`10`). Secara eksplisit mentransmisikan mereka

ke Decimal nilai, seperti dengan menggunakan fungsi [cast \(\)](#), tidak mencegah perilaku ini—hasilnya masih merupakan nilai Integer. Hal ini dapat menyebabkan kesalahan jenis ketidakcocokan yang mencegah data direkam dalam database Timestream. Untuk memproses nilai numerik bilangan bulat sebagai Decimal nilai, gunakan SQL V1 (2015-10-08) untuk pernyataan kueri aturan.

Note

Jumlah maksimum nilai yang dapat ditulis oleh tindakan aturan Timestream ke dalam tabel Amazon Timestream adalah 100. Untuk informasi selengkapnya, lihat Referensi [Kuota Amazon Timestream](#).

Persyaratan

Tindakan aturan ini memiliki persyaratan sebagai berikut:

- Peran IAM yang AWS IoT dapat diasumsikan untuk melakukan `timestream:DescribeEndpoints` dan `timestream:WriteRecords` operasi. Untuk informasi selengkapnya, lihat [Memberikan AWS IoT aturan akses yang dibutuhkannya](#).

Di AWS IoT konsol, Anda dapat memilih, memperbarui, atau membuat peran AWS IoT untuk memungkinkan melakukan tindakan aturan ini.

- Jika Anda menggunakan pelanggan- AWS KMS untuk mengenkripsi data saat istirahat di Timestream, layanan harus memiliki izin untuk menggunakan AWS KMS key atas nama pemanggil. Untuk informasi selengkapnya, lihat [Cara AWS layanan menggunakan AWS KMS](#).

Parameter

Saat Anda membuat AWS IoT aturan dengan tindakan ini, Anda harus menentukan informasi berikut:

databaseName

Nama database Amazon Timestream yang memiliki tabel untuk menerima catatan yang dibuat tindakan ini. Lihat juga **tableName**.

Mendukung [template substitusi](#): API dan hanya AWS CLI

dimensions

Atribut metadata dari deret waktu yang ditulis dalam setiap catatan ukuran. Misalnya, nama dan Availability Zone dari instans EC2 atau nama produsen turbin angin adalah dimensi.

name

Nama dimensi metadata. Ini adalah nama kolom dalam catatan tabel database.

Dimensi tidak dapat diberi nama: `measure_name`, `measure_value`, atau `time`. Nama-nama ini dicadangkan. Nama dimensi tidak dapat dimulai dengan `ts_` atau `measure_value` dan tidak dapat berisi karakter titik dua (:).

Mendukung [template substitusi](#): Tidak

value

Nilai untuk ditulis dalam kolom ini dari catatan basis data.

Mendukung [template substitusi](#): Ya

roleArn

Amazon Resource Name (ARN) peran yang memberikan AWS IoT izin untuk menulis ke tabel basis data Timestream. Untuk informasi selengkapnya, lihat [Persyaratan](#).

Mendukung [template substitusi](#): Tidak

tableName

Nama tabel database untuk menulis catatan ukuran. Lihat juga **databaseName**.

Mendukung [template substitusi](#): API dan hanya AWS CLI

timestamp

Nilai yang akan digunakan untuk stempel waktu entri. Jika kosong, waktu untuk memproses entri yang akan digunakan.

unit

Ketepatan nilai stempel waktu yang dihasilkan dari ekspresi yang dijelaskan dalam `value`.

Nilai yang valid: `SECONDS` | `MILLISECONDS` | `MICROSECONDS` | `NANOSECONDS`. Nilai default-nya `MILLISECONDS`.

value

Ekspresi yang mengembalikan nilai waktu jangka waktu panjang.

Anda dapat menggunakan [the section called “time_to_epoch \(String, String\)”](#) fungsi ini untuk membuat stempel waktu yang valid dari nilai tanggal atau waktu yang diteruskan dalam payload pesan.

Konten rekaman Timestream

Data yang ditulis ke tabel Amazon Timestream oleh tindakan ini mencakup stempel waktu, metadata dari tindakan aturan Timestream, dan hasil dari pernyataan kueri aturan.

Untuk setiap atribut (ukuran) dalam hasil pernyataan kueri, tindakan aturan ini menulis catatan ke tabel Timestream yang ditentukan dengan kolom ini.

Nama kolom	Jenis atribut	Nilai	Komentar
<i>dimensi-nama</i>	UKURAN	Nilai yang ditentukan dalam entri tindakan aturan Timestream.	Setiap Dimensi yang ditentukan dalam entri tindakan aturan membuat kolom dalam database Timestream dengan nama dimensi.
ukuran_nama	UKURAN_NAMA	Nama atribut	Nama atribut dalam hasil pernyataan query yang nilainya ditentukan dalam <code>measure_value:: data-type</code> kolom.
<i>ukuran_nilai:: tipe data</i>	UKURAN_NILAI	Nilai atribut dalam hasil pernyataan query. Nama atribut	Nilai ditafsirkan* dan dilemparkan sebagai kecocokan yang paling cocok dari:bigint,,boolean,

Nama kolom	Jenis atribut	Nilai	Komentar
		ada di <code>measure_name</code> kolom.	<code>double</code> atau <code>varchar</code> . Amazon Timestream membuat kolom terpisah untuk setiap tipe data. Nilai dalam pesan dapat ditransmisikan ke tipe data lain dengan menggunakan an <code>cast()</code> fungsi dalam pernyataan kueri aturan.
Waktu	TIMESTAMP	Tanggal dan waktu catatan dalam database.	Nilai ini ditetapkan oleh mesin aturan atau <code>timestamp</code> properti, jika didefinisikan.

* Nilai atribut yang dibaca dari payload pesan ditafsirkan sebagai berikut. Lihat [the section called “Contoh”](#) untuk ilustrasi dari masing-masing kasus ini.

- Nilai yang tidak dikutip dari `true` atau `false` ditafsirkan sebagai tipe `boolean`
- Numerik desimal ditafsirkan sebagai tipe `double`
- Nilai numerik tanpa titik desimal ditafsirkan sebagai tipe `bigint`
- String yang dikutip ditafsirkan sebagai `varchar` tipe.
- Objek dan nilai array dikonversi ke string JSON dan disimpan sebagai `varchar` tipe.

Contoh

Contoh JSON berikut mendefinisikan tindakan aturan Timestream dengan template substitusi dalam aturan. 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"
      }
    }
  ]
}

```

Menggunakan tindakan aturan topik Timestream yang ditentukan dalam contoh sebelumnya dengan payload pesan berikut menghasilkan catatan Amazon Timestream yang ditulis dalam tabel berikut.

```

{
  "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"]
  }
}

```

```

}
}

```

Tabel berikut menampilkan kolom database dan catatan yang menggunakan tindakan aturan topik tertentu untuk memproses payload pesan sebelumnya dibuat. `device_id` Kolom `device_firmware_sku` dan adalah DIMENSI yang didefinisikan dalam tindakan aturan topik. Tindakan aturan topik Timestream membuat `time` kolom dan `measure_value::*` kolom `measure_name` dan kolom, yang diisi dengan nilai dari hasil pernyataan kueri tindakan aturan topik.

device_firmware_sku	device_id	ukuran_nama	ukuran_nilai:bigint	ukuran_nilai:varchar	ukuran_nilai:ganda	ukuran_nilai:boolean	Waktu
Metadata Statis Saya	IOTCONSC E-159Example738-0	complex_value	-	{"simple_element":4 "array_of_integers": [23,36,56,72], "array_string": ["red", "green", "blue "]}	-	-	2020-08-26 22:42:16.423 000000
Metadata Statis Saya	IOTCONSC E-159Example738-0	integer_value_as_string	-	123456789012	-	-	2020-08-26 22:42:16.423 000000
Metadata Statis Saya	IOTCONSC E-159Example738-0	boolean_nilai	-	-	-	BETUL	2020-08-26 22:42:16.423 000000

device_firmware_sku	device_id	ukuran_nama	ukuran_nilai: bigint	ukuran_nilai: varchar	ukuran_nilai: ganda	ukuran_nilai: boolean	Waktu
Metadata Statis Saya	IOTCONSOLE-159Example738-0	integer_nilai	123456789012	-	-	-	2020-08-26 22:42:16.423 000000
Metadata Statis Saya	iotconsole-159example738-0	string_nilai	-	Nilai string	-	-	2020-08-26 22:42:16.423 000000
Metadata Statis Saya	iotconsole-159example738-0	array_of_integer	-	[23,36,56,72]	-	-	2020-08-26 22:42:16.423 000000
Metadata Statis Saya	iotconsole-159example738-0	array_string	-	["merah", "hijau", "biru"]	-	-	2020-08-26 22:42:16.423 000000
Metadata Statis Saya	iotconsole-159example738-0	boolean_value_as_string	-	BETUL	-	-	2020-08-26 22:42:16.423 000000

device_firmware_sku	device_id	ukuran_nama	ukuran_nilai:bigint	ukuran_nilai:varchar	ukuran_nilai:ganda	ukuran_nilai:boolea	Waktu
Metadata Statis Saya	iotconsole-159example738-0	nilai ganda	-	-	123.456789012	-	2020-08-26 22:42:16.423 000000
Metadata Statis Saya	iotconsole-159example738-0	double_value_as_string	-	123.45679	-	-	2020-08-26 22:42:16.423 000000

Memecahkan masalah aturan

Jika Anda memiliki masalah dengan aturan Anda, kami sarankan Anda mengaktifkan CloudWatch Log. Anda dapat menganalisis log untuk menentukan apakah masalahnya adalah otorisasi atau, misalnya, kondisi klausa WHERE tidak cocok. Untuk informasi selengkapnya, lihat [Menyiapkan CloudWatch Log](#).

Mengakses sumber daya lintas akun menggunakan aturan AWS IoT

Anda dapat mengonfigurasi AWS IoT aturan untuk akses lintas akun sehingga data yang dicerna pada topik MQTT dari satu akun dapat dialihkan ke layanan, AWS seperti Amazon SQS dan Lambda, dari akun lain. Berikut ini menjelaskan cara mengatur AWS IoT aturan untuk konsumsi data lintas akun, dari topik MQTT di satu akun, hingga tujuan di akun lain.

Aturan lintas akun dapat dikonfigurasi menggunakan [izin berbasis sumber daya](#) pada sumber daya tujuan. Oleh karena itu, hanya tujuan yang mendukung izin berbasis sumber daya yang dapat diaktifkan untuk akses lintas akun dengan aturan. AWS IoT Tujuan yang didukung termasuk Amazon SQS, Amazon SNS, Amazon S3, dan. AWS Lambda

Note

Untuk tujuan yang didukung, kecuali Amazon SQS, Anda harus menentukan aturan yang Wilayah AWS sama dengan sumber daya layanan lain sehingga tindakan aturan dapat berinteraksi dengan sumber daya tersebut. Untuk informasi selengkapnya tentang tindakan AWS IoT aturan, lihat [tindakan AWS IoT aturan](#). Untuk informasi selengkapnya tentang tindakan SQS aturan, lihat [???](#).

Prasyarat

- [Keakraban dengan AWS IoT aturan](#)
- Pemahaman tentang [pengguna IAM](#), [peran](#), dan izin berbasis sumber [daya](#)
- Setelah [AWS CLI](#) menginstal

Penyiapan lintas akun untuk Amazon SQS

Skenario: Akun A mengirimkan data dari pesan MQTT ke antrian Amazon SQS akun B.

Akun AWS	Akun disebut sebagai	Deskripsi
<i>1111-1111-1111</i>	Akun A	Tindakan aturan: sqs:SendMessage
<i>2222-2222-2222</i>	Akun B	Antrean Amazon SQS <ul style="list-style-type: none"> • ARN: <i>arn:aws:sqs:region:2222-2222-2222:ExampleQueue</i> • URL: <i>https://sqs.region.amazonaws.com/2222-2222-2222/ExampleQueue</i>

Note

[Antrian Amazon SQS tujuan Anda tidak harus Wilayah AWS sama dengan aturan Anda.](#) [AWS IoT](#) Untuk informasi selengkapnya tentang tindakan SQS aturan, lihat [???](#).

Lakukan tugas Akun A

Catatan

Untuk menjalankan perintah berikut, pengguna IAM Anda harus memiliki izin untuk `iot:CreateTopicRule` menggunakan Amazon Resource Name (ARN) aturan sebagai sumber daya, dan izin untuk `iam:PassRole` bertindak dengan sumber daya sebagai ARN peran.

1. [Konfigurasi AWS CLI](#) menggunakan pengguna IAM akun A.
2. Buat peran IAM yang mempercayai mesin AWS IoT aturan, dan lampirkan kebijakan yang memungkinkan akses ke antrian Amazon SQS akun B. Lihat contoh perintah dan dokumen kebijakan di [Pemberian akses AWS IoT yang diperlukan](#).
3. Untuk membuat aturan yang dilampirkan ke topik, jalankan [create-topic-rule perintah](#).

```
aws iot create-topic-rule --rule-name myRule --topic-rule-payload file:///./my-rule.json
```

Berikut ini adalah contoh file payload dengan aturan yang menyisipkan semua pesan yang dikirim ke `iot/test` topik ke dalam antrian Amazon SQS yang ditentukan. Pernyataan SQL memfilter pesan dan peran ARN AWS IoT memberikan izin untuk menambahkan pesan ke antrian 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
  }
}
]
}

```

Untuk informasi selengkapnya tentang cara menentukan tindakan Amazon SQS dalam AWS IoT aturan, lihat [tindakan AWS IoT aturan - Amazon SQS](#).

Lakukan tugas Akun B

1. [Konfigurasi AWS CLI](#) menggunakan pengguna IAM akun B.
2. [Untuk memberikan izin sumber daya antrian Amazon SQS ke akun A, jalankan perintah `add-permission`](#).

```

aws sqs add-permission --queue-url https://sqs.region.amazonaws.com/2222-2222-2222/ExampleQueue --label SendMessageToMyQueue --aws-account-ids 1111-1111-1111 --actions SendMessage

```

Penyiapan lintas akun untuk Amazon SNS

Skenario: Akun A mengirim data dari pesan MQTT ke topik akun Amazon SNS B.

Akun AWS	Akun disebut sebagai	Deskripsi
<i>1111-1111-1111</i>	Akun A	Tindakan aturan: <code>sns:Publish</code>
<i>2222-2222-2222</i>	Akun B	Topik Amazon SNS ARN: <code>arn:aws:sns:region:2222-2222-2222:ExampleTopic</code>

Lakukan tugas Akun A

Catatan

Untuk menjalankan perintah berikut, pengguna IAM Anda harus memiliki izin `iot:CreateTopicRule` dengan aturan ARN sebagai sumber daya dan izin untuk `iam:PassRole` tindakan dengan sumber daya sebagai peran ARN.

1. [Konfigurasi AWS CLI](#) menggunakan pengguna IAM akun A.
2. Buat peran IAM yang mempercayai mesin AWS IoT aturan, dan lampirkan kebijakan yang memungkinkan akses ke topik Amazon SNS akun B. Misalnya perintah dan dokumen kebijakan, lihat [Memberikan akses AWS IoT yang diperlukan](#).
3. Untuk membuat aturan yang dilampirkan ke topik, jalankan [create-topic-rule perintah](#).

```
aws iot create-topic-rule --rule-name myRule --topic-rule-payload file:///./my-rule.json
```

Berikut ini adalah contoh file payload dengan aturan yang menyisipkan semua pesan yang dikirim ke `iot/test` topik ke topik Amazon SNS yang ditentukan. Pernyataan SQL memfilter pesan, dan peran ARN AWS IoT memberikan izin untuk mengirim pesan ke topik 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"
      }
    }
  ]
}
```

Untuk informasi selengkapnya tentang cara menentukan tindakan Amazon SNS dalam AWS IoT aturan, lihat [tindakan AWS IoT aturan - Amazon SNS](#).

Lakukan tugas Akun B

1. [Konfigurasi AWS CLI](#) menggunakan pengguna IAM akun B.
2. Untuk memberikan izin pada sumber daya topik Amazon SNS ke akun A, jalankan perintah [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
```

Penyiapan lintas akun untuk Amazon S3

Skenario: Akun A mengirimkan data dari pesan MQTT ke ember akun Amazon S3 B.

Akun AWS	Akun disebut sebagai	Deskripsi
<i>1111-1111-1111</i>	Akun A	Tindakan aturan: <code>s3:PutObject</code>
<i>2222-2222-2222</i>	Akun B	Ember Amazon S3 ARN: <i>arn:aws:s3:::ExampleBucket</i>

Lakukan tugas Akun A

Catatan

Untuk menjalankan perintah berikut, pengguna IAM Anda harus memiliki izin `iot:CreateTopicRule` dengan aturan ARN sebagai sumber daya dan izin untuk `iam:PassRole` bertindak dengan sumber daya sebagai peran ARN.

1. [Konfigurasi AWS CLI](#) menggunakan pengguna IAM akun A.
2. Buat peran IAM yang mempercayai mesin AWS IoT aturan dan melampirkan kebijakan yang memungkinkan akses ke bucket Amazon S3 akun B. Misalnya perintah dan dokumen kebijakan, lihat [Memberikan akses AWS IoT yang diperlukan](#).

3. Untuk membuat aturan yang dilampirkan ke bucket S3 target Anda, jalankan [create-topic-rule perintah](#).

```
aws iot create-topic-rule --rule-name my-rule --topic-rule-payload file://./my-rule.json
```

Berikut ini adalah contoh file payload dengan aturan yang menyisipkan semua pesan yang dikirim ke `iot/test` topik ke dalam bucket Amazon S3 yang ditentukan. Pernyataan SQL memfilter pesan, dan peran ARN AWS IoT memberikan izin untuk menambahkan pesan ke bucket Amazon S3.

```
{
  "sql": "SELECT * FROM 'iot/test'",
  "ruleDisabled": false,
  "awsIotSqlVersion": "2016-03-23",
  "actions": [
    {
      "s3": {
        "bucketName": "ExampleBucket",
        "key": "${topic()}/${timestamp()}",
        "roleArn": "arn:aws:iam::1111-1111-1111:role/my-iot-role"
      }
    }
  ]
}
```

Untuk informasi selengkapnya tentang cara menentukan tindakan Amazon S3 dalam AWS IoT aturan, lihat [tindakan AWS IoT aturan - Amazon S3](#).

Lakukan tugas Akun B

1. [Konfigurasi AWS CLI](#) menggunakan pengguna IAM akun B.
2. Buat kebijakan bucket yang mempercayai prinsipal akun A.

Berikut ini adalah contoh file payload yang mendefinisikan kebijakan bucket yang mempercayai prinsipal akun lain.

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

```

"Statement": [
  {
    "Sid": "AddCannedAcl",
    "Effect": "Allow",
    "Principal": {
      "AWS": [
        "arn:aws:iam::1111-1111-1111:root"
      ]
    },
    "Action": "s3:PutObject",
    "Resource": "arn:aws:s3:::ExampleBucket/*"
  }
]
}

```

Untuk informasi selengkapnya, lihat [contoh kebijakan bucket](#).

3. Untuk melampirkan kebijakan bucket ke bucket yang ditentukan, jalankan [put-bucket-policy perintah](#).

```
aws s3api put-bucket-policy --bucket ExampleBucket --policy file:///./my-bucket-policy.json
```

4. Untuk membuat akses lintas akun berfungsi, pastikan Anda memiliki pengaturan Blokir semua akses publik yang benar. Untuk informasi selengkapnya, lihat [Praktik Terbaik Keamanan untuk Amazon S3](#).

Pengaturan lintas akun untuk AWS Lambda

Skenario: Akun A memanggil AWS Lambda fungsi akun B, meneruskan pesan MQTT.

Akun AWS	Akun disebut sebagai	Deskripsi
1111-1111-1111	Akun A	Tindakan aturan: <code>lambda:InvokeFunction</code>
2222-2222-2222	Akun B	Fungsi Lambda ARN: <code>arn:aws:lambda:region:2222-2222-2222:function:example-function</code>

Lakukan tugas Akun A

Catatan

Untuk menjalankan perintah berikut, pengguna IAM Anda harus memiliki izin `iot:CreateTopicRule` dengan aturan ARN sebagai sumber daya, dan izin untuk `iam:PassRole` bertindak dengan sumber daya sebagai peran ARN.

1. [Konfigurasi AWS CLI](#) menggunakan pengguna IAM akun A.
2. Jalankan [create-topic-rule perintah](#) untuk membuat aturan yang mendefinisikan akses lintas akun ke fungsi Lambda akun B.

```
aws iot create-topic-rule --rule-name my-rule --topic-rule-payload file:///./my-rule.json
```

Berikut ini adalah contoh file payload dengan aturan yang menyisipkan semua pesan yang dikirim ke `iot/test` topik ke dalam fungsi Lambda yang ditentukan. Pernyataan SQL memfilter pesan dan peran ARN AWS IoT memberikan izin untuk meneruskan data ke fungsi 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"
      }
    }
  ]
}
```

Untuk informasi lebih lanjut tentang cara mendefinisikan AWS Lambda tindakan dalam AWS IoT aturan, baca [tindakan AWS IoT aturan - Lambda](#).

Lakukan tugas Akun B

1. [Konfigurasi AWS CLI](#) menggunakan pengguna IAM akun B.

2. Jalankan [perintah add-permission Lambda](#) untuk memberikan izin AWS IoT aturan untuk mengaktifkan fungsi Lambda. Untuk menjalankan perintah berikut, pengguna IAM Anda harus memiliki izin untuk `lambda:AddPermission` bertindak.

```
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"
```

Pilihan:

--kepala sekolah

Bidang ini memberikan izin untuk AWS IoT (diwakili oleh `iot.amazonaws.com`) untuk memanggil fungsi Lambda.

--sumber-arn

Bidang ini mengonfirmasi bahwa hanya `arn:aws:iot:region:1111-1111-1111:rule/example-rule` dalam AWS IoT memicu fungsi Lambda ini dan tidak ada aturan lain di akun yang sama atau berbeda yang dapat mengaktifkan fungsi Lambda ini.

--sumber-akun

Bidang ini mengonfirmasi bahwa AWS IoT mengaktifkan fungsi Lambda ini hanya atas nama akun. `1111-1111-1111`

Catatan

Jika Anda melihat pesan kesalahan "Aturan tidak dapat ditemukan" dari konsol AWS Lambda fungsi Anda di bawah Konfigurasi, abaikan pesan kesalahan dan lanjutkan untuk menguji koneksi.

Penanganan kesalahan (tindakan kesalahan)

Saat AWS IoT menerima pesan dari perangkat, mesin aturan memeriksa untuk melihat apakah pesan tersebut cocok dengan aturan. Jika demikian, pernyataan kueri aturan dievaluasi dan tindakan aturan diaktifkan, meneruskan hasil pernyataan kueri.

Jika masalah terjadi saat mengaktifkan suatu tindakan, mesin aturan mengaktifkan tindakan kesalahan, jika ada yang ditentukan untuk aturan tersebut. Ini mungkin terjadi ketika:

- Aturan tidak memiliki izin untuk mengakses bucket Amazon S3.
- Kesalahan pengguna menyebabkan throughput yang disediakan DynamoDB terlampaui.

Note

Penanganan kesalahan yang tercakup dalam topik ini adalah untuk [tindakan aturan](#). Untuk men-debug masalah SQL, termasuk fungsi eksternal, Anda dapat mengatur AWS IoT logging. Untuk informasi selengkapnya, lihat [???](#).

Format pesan tindakan kesalahan

Satu pesan dihasilkan per aturan dan pesan. Misalnya, jika dua tindakan aturan dalam aturan yang sama gagal, tindakan kesalahan menerima satu pesan yang berisi kedua kesalahan.

Pesan tindakan kesalahan terlihat seperti contoh berikut.

```
{
  "ruleName": "TestAction",
  "topic": "testme/action",
  "cloudwatchTraceId": "7e146a2c-95b5-6caf-98b9-50e3969734c7",
  "clientId": "iotconsole-1511213971966-0",
  "base64OriginalPayload":
  "ewogICJtZXNzYWdlIjogIkh1bGxvIHZyb20gQVdTIElvVCBjb25zb2xlIgp9",
  "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:
yMah1cwPhqTH267QLPhTKeVPKJB8B05ndBHz0mWtxLTM6uAvwYYuqieAKyb6qRPTxP1tHXCoR4Y=).
Message arrived on: error/action, Action: s3, Bucket: us-
east-1-s3-verify-user, Key: \"aaa\". Value of x-amz-id-2:
yMah1cwPhqTH267QLPhTKeVPKJB8B05ndBHz0mWtxLTM6uAvwYYuqieAKyb6qRPTxP1tHXCoR4Y="
    }
  ]
}
```

```
}
```

ruleName

Nama aturan yang memicu tindakan kesalahan.

topik

Topik di mana pesan asli diterima.

cloudwatch Traceld

Identitas unik yang mengacu pada log kesalahan CloudWatch.

clientId

ID klien dari penerbit pesan.

base64 OriginalPayload

Pesan asli payload Base64-dikodekan.

kegagalan

failedAction

Nama tindakan yang gagal diselesaikan (misalnya, "S3Action").

failedResource

Nama sumber daya (misalnya, nama ember S3).

errorMessage

Deskripsi dan penjelasan kesalahan.

Contoh tindakan kesalahan

Berikut adalah contoh aturan dengan tindakan kesalahan tambahan. Aturan berikut memiliki tindakan yang menulis data pesan ke tabel DynamoDB dan tindakan kesalahan yang menulis data ke bucket Amazon S3:

```
{
  "sql" : "SELECT * FROM ..."
  "actions" : [{
    "dynamoDB" : {
      "table" : "PoorlyConfiguredTable",
      "hashKeyField" : "AConstantString",
```



```

        "hashKeyValue" : "AHashKey"}}
    ],
    "errorAction" : {
        "s3" : {
            "roleArn": "arn:aws:iam::123456789012:role/aws_iam_s3",
            "bucketName" : "message-processing-errors",
            "key" : "${replace(topic(), '/', '-') + '-' + timestamp() + '-' +
newuuid()}"
        }
    }
}

```

Anda dapat menggunakan [fungsi](#) atau [template substitusi](#) dalam pernyataan SQL tindakan kesalahan termasuk fungsi eksternal: [aws_lambda\(\)](#), [get_dynamodb\(\)](#), [get_thing_shadow\(\)](#), [get_secret\(\)](#), [machinelearning_predict\(\)](#), dan [decode\(\)](#). Jika tindakan kesalahan mengharuskan untuk memanggil fungsi eksternal, maka menjalankan tindakan kesalahan dapat mengakibatkan tagihan tambahan untuk fungsi eksternal.

Fungsi eksternal berikut ditagih setara dengan tindakan aturan: [aws_lambda](#), [get_dynamodb\(\)](#), dan [get_thing_shadow\(\)](#). Anda juga ditagih untuk [decode\(\)](#) fungsi hanya ketika Anda [mendekode pesan Protobuf ke JSON](#). Untuk lebih jelasnya, lihat [halaman AWS IoT Core harga](#).

Untuk informasi selengkapnya tentang aturan dan cara menentukan tindakan kesalahan, lihat [Membuat AWS IoT Aturan](#).

Untuk informasi lebih lanjut tentang penggunaan CloudWatch untuk memantau keberhasilan atau kegagalan aturan, lihat [AWS IoT metrik dan dimensi](#).

Mengurangi biaya pengiriman pesan dengan Basic Ingest

[Anda dapat menggunakan Basic Ingest, untuk mengirim data perangkat dengan aman ke perangkat yang Layanan AWS didukung oleh AWS IoT tindakan aturan, tanpa menimbulkan biaya pengiriman pesan.](#) Basic Ingest mengoptimalkan aliran data dengan menghapus broker pesan terbitkan/berlangganan dari jalur konsumsi.

Basic Ingest dapat mengirim pesan dari perangkat atau aplikasi Anda. Pesan memiliki nama topik yang dimulai dengan `$aws/rules/rule_name` untuk tiga level pertama mereka, di mana *rule_name* adalah nama AWS IoT aturan yang ingin Anda panggil.

Anda dapat menggunakan aturan yang ada dengan Basic Ingest dengan menambahkan awalan Basic Ingest (`$aws/rules/rule_name`) ke topik pesan yang akan Anda gunakan untuk memanggil

aturan. Misalnya, jika Anda memiliki aturan bernama `BuildingManager` yang dipanggil oleh pesan dengan topik seperti `Buildings/Building5/Floor2/Room201/Lights` (`"sql": "SELECT * FROM 'Buildings/#'"`), Anda dapat memanggil aturan yang sama dengan Basic Ingest dengan mengirimkan pesan dengan topik `$aws/rules/BuildingManager/Buildings/Building5/Floor2/Room201/Lights`

Catatan:

- Perangkat dan aturan Anda tidak dapat berlangganan topik yang dicadangkan Basic Ingest. Untuk informasi selengkapnya, lihat [Topik yang dipesan](#).
- Jika Anda memerlukan broker terbitkan/berlangganan untuk mendistribusikan pesan ke beberapa pelanggan (misalnya, untuk mengirimkan pesan ke perangkat lain dan mesin aturan), Anda harus terus menggunakan broker AWS IoT pesan untuk menangani distribusi pesan. Namun, pastikan Anda mempublikasikan pesan Anda tentang topik selain topik Basic Ingest.

Menggunakan Basic Ingest

Sebelum Anda menggunakan Basic Ingest, verifikasi bahwa perangkat atau aplikasi Anda menggunakan [kebijakan](#) yang memiliki izin publikasi. `$aws/rules/*` Atau, Anda dapat menentukan izin untuk aturan individu dengan `$aws/rules/rule_name/*` dalam kebijakan. Jika tidak, perangkat dan aplikasi Anda dapat terus menggunakan koneksi yang ada AWS IoT Core.

Ketika pesan mencapai mesin aturan, tidak ada perbedaan dalam implementasi atau penanganan kesalahan antara aturan yang dipanggil dari Basic Ingest dan yang dipanggil melalui langganan broker pesan.

Anda dapat membuat aturan untuk digunakan dengan Basic Ingest. Ingatlah hal berikut:

- Awalan awal topik Basic Ingest (`$aws/rules/rule_name`) tidak tersedia untuk fungsi tersebut. [topik \(Desimal\)](#)
- Jika Anda mendefinisikan aturan yang dipanggil hanya dengan Basic Ingest, `FROM` klausa adalah opsional di `sql` bidang definisi. `rule` ini masih diperlukan jika aturan juga dipanggil oleh pesan lain yang harus dikirim melalui broker pesan (misalnya, karena pesan lain tersebut harus didistribusikan ke beberapa pelanggan). Untuk informasi selengkapnya, lihat [AWS IoT Referensi SQL](#).
- Tiga level pertama dari topik Ingest Dasar (`$aws/rules/rule_name`) tidak dihitung terhadap batas panjang 8 segmen atau menuju batas karakter 256 total untuk suatu topik. Jika tidak, pembatasan yang sama berlaku seperti yang didokumentasikan dalam [AWS IoT Batas](#).

- Jika pesan diterima dengan topik Ingest Dasar yang menentukan aturan tidak aktif atau aturan yang tidak ada, log kesalahan dibuat di CloudWatch log Amazon untuk membantu Anda melakukan debug. Untuk informasi selengkapnya, lihat [Aturan entri log mesin](#). `RuleNotFoundMetric` ditunjukkan dan Anda dapat membuat alarm pada metrik ini. Untuk informasi selengkapnya, lihat [Metrik Aturan](#) di [Metrik aturan](#).
- Anda masih dapat mempublikasikan dengan QoS 1 tentang topik Basic Ingest. Anda menerima PUBACK setelah pesan berhasil dikirim ke mesin aturan. Menerima PUBACK tidak berarti bahwa tindakan aturan Anda berhasil diselesaikan. Anda dapat mengonfigurasi tindakan kesalahan untuk menangani kesalahan saat tindakan dijalankan. Lihat informasi yang lebih lengkap di [Penanganan kesalahan \(tindakan kesalahan\)](#).

AWS IoT Referensi SQL

Dalam AWS IoT, aturan didefinisikan menggunakan sintaks seperti SQL. Pernyataan SQL terdiri dari tiga jenis klausa:

SELECT

(Wajib) Mengekstrak informasi dari muatan pesan yang masuk dan melakukan transformasi pada informasi. Pesan yang akan digunakan diidentifikasi oleh [filter topik](#) yang ditentukan dalam klausa FROM.

Klausa SELECT mendukung [Jenis dataOperator](#), [Fungsi](#), [Literal](#), [Pernyataan kasus](#), [Ekstensi JSON](#), [Templat substitusiKueri objek bersarang](#), dan [Muatan biner](#).

FROM

[Filter topik](#) pesan MQTT yang mengidentifikasi pesan untuk mengekstrak data. Aturan diaktifkan untuk setiap pesan yang dikirim ke topik MQTT yang cocok dengan filter topik yang ditentukan di sini. Diperlukan untuk aturan yang diaktifkan oleh pesan yang melewati broker pesan. Opsional untuk aturan yang hanya diaktifkan menggunakan fitur [Basic Ingest](#).

WHERE

(Opsional) Menambahkan logika bersyarat yang menentukan apakah tindakan yang ditentukan oleh aturan dilakukan.

Klausa WHERE mendukung [Jenis dataOperator](#), [Fungsi](#), [Literal](#), [Pernyataan kasus](#), [Ekstensi JSON](#) [Templat substitusi](#), dan [Kueri objek bersarang](#).

Contoh pernyataan SQL terlihat seperti ini:

```
SELECT color AS rgb FROM 'topic/subtopic' WHERE temperature > 50
```

Contoh pesan MQTT (juga disebut payload masuk) terlihat seperti ini:

```
{
  "color":"red",
  "temperature":100
}
```

Jika pesan ini diterbitkan pada 'topic/subtopic' topik, aturan dipicu dan pernyataan SQL dievaluasi. Pernyataan SQL mengekstrak nilai `color` properti jika `temperature` properti lebih besar dari 50. Klausula `WHERE` menentukan kondisi. `temperature > 50` kata kunci mengganti nama `color` properti menjadi. `rgb` Hasilnya (juga disebut payload keluar) terlihat seperti ini:

```
{
  "rgb":"red"
}
```

Data ini kemudian diteruskan ke tindakan aturan, yang mengirimkan data untuk diproses lebih lanjut. Untuk informasi selengkapnya tentang tindakan aturan, lihat [AWS IoT tindakan aturan](#).

Note

Komentar saat ini tidak didukung dalam sintaks AWS IoT SQL.

Nama atribut dengan spasi di dalamnya tidak dapat digunakan sebagai nama bidang dalam pernyataan SQL. Meskipun payload yang masuk dapat memiliki nama atribut dengan spasi di dalamnya, nama tersebut tidak dapat digunakan dalam pernyataan SQL. Namun, mereka akan diteruskan ke payload keluar jika Anda menggunakan spesifikasi nama bidang wildcard (*).

Klausula SELECT

Klausula AWS IoT SELECT pada dasarnya sama dengan klausula ANSI SQL SELECT, dengan beberapa perbedaan kecil.

Klausula SELECT mendukung [Jenis data Operator](#), [Fungsi Literal](#), [Pernyataan kasus](#), [Ekstensi JSON](#), [Templat substitusi Kueri objek bersarang](#), dan [Muatan biner](#).

Anda dapat menggunakan klausula SELECT untuk mengekstrak informasi dari pesan MQTT yang masuk. Anda juga dapat menggunakan SELECT * untuk mengambil seluruh payload pesan masuk. Sebagai contoh:

```
Incoming payload published on topic 'topic/subtopic': {"color":"red", "temperature":50}
SQL statement: SELECT * FROM 'topic/subtopic'
Outgoing payload: {"color":"red", "temperature":50}
```

Jika payload adalah objek JSON, Anda dapat referensi kunci dalam objek. Payload keluar Anda berisi pasangan kunci-nilai. Sebagai contoh:

```
Incoming payload published on topic 'topic/subtopic': {"color":"red", "temperature":50}
SQL statement: SELECT color FROM 'topic/subtopic'
Outgoing payload: {"color":"red"}
```

Anda dapat menggunakan kata kunci AS untuk mengganti nama kunci. Sebagai contoh:

```
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"}
```

Anda dapat memilih beberapa item dengan memisahkannya dengan koma. Sebagai contoh:

```
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}
```

Anda dapat memilih beberapa item termasuk "*" untuk menambahkan item ke muatan yang masuk. Sebagai contoh:

```
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}
```

Anda dapat menggunakan "VALUE" kata kunci untuk menghasilkan payload keluar yang bukan objek JSON. Dengan versi SQL2015-10-08, Anda hanya dapat memilih satu item. Dengan versi

SQL 2016-03-23 atau yang lebih baru, Anda juga dapat memilih array untuk output sebagai objek tingkat atas.

Example

```
Incoming payload published on topic 'topic/subtopic': {"color":"red", "temperature":50}
SQL: SELECT VALUE color FROM 'topic/subtopic'
Outgoing payload: "red"
```

Anda dapat menggunakan '.' sintaks untuk menelusuri objek JSON bersarang di payload yang masuk. Sebagai contoh:

```
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}
```

Untuk informasi tentang cara menggunakan objek JSON dan nama properti yang menyertakan karakter cadangan, seperti angka atau karakter tanda hubung (minus), lihat [Ekstensi JSON](#)

Anda dapat menggunakan fungsi (lihat [Fungsi](#)) untuk mengubah payload yang masuk. Anda dapat menggunakan tanda kurung untuk pengelompokan. Sebagai contoh:

```
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"}
```

Klausula FROM

Klausula FROM berlangganan aturan Anda ke filter [topik](#) atau [topik](#). Lampirkan topik atau filter topik dalam tanda kutip tunggal ('). Aturan dipicu untuk setiap pesan yang dikirim ke topik MQTT yang cocok dengan filter topik yang ditentukan di sini. Anda dapat berlangganan sekelompok topik serupa menggunakan filter topik.

Contoh:

Payload masuk dipublikasikan pada topik: 'topic/subtopic' {temperature: 50}

Payload masuk dipublikasikan pada topik: 'topic/subtopic-2' {temperature: 50}

```
SQL:"SELECT temperature AS t FROM 'topic/subtopic'".
```

Aturan berlangganan 'topic/subtopic', sehingga muatan yang masuk diteruskan ke aturan. Muatan keluar, diteruskan ke tindakan aturan, adalah: {t: 50} Aturannya tidak berlangganan 'topic/subtopic-2', jadi aturan tidak dipicu untuk pesan yang dipublikasikan 'topic/subtopic-2'.

Wildcard Contoh:

Anda dapat menggunakan karakter wildcard '#' (multi-level) untuk mencocokkan satu atau lebih elemen jalur tertentu:

Muatan masuk diterbitkan dengan topik 'topic/subtopic': {temperature: 50}

Muatan masuk diterbitkan dengan topik 'topic/subtopic-2': {temperature: 60}

Muatan masuk diterbitkan dengan topik 'topic/subtopic-3/details': {temperature: 70}

Muatan masuk diterbitkan dengan topik 'topic-2/subtopic-x': {temperature: 80}

```
SQL:"SELECT temperature AS t FROM 'topic/#'".
```

Aturan berlangganan topik apa pun yang dimulai 'topic', jadi dieksekusi tiga kali, mengirimkan muatan keluar (untuk topik/subtopik), (untuk topik/subtopik-2), {t: 60} dan {t: 50} (untuk topik/subtopik-3/detail) ke tindakannya. {t: 70} Itu tidak berlangganan 'topic-2/subtopic-x', jadi aturan tidak dipicu untuk {temperature: 80} pesan.

+ Contoh Wildcard:

Anda dapat menggunakan karakter wildcard '+' (tingkat tunggal) untuk mencocokkan salah satu elemen jalur tertentu:

Muatan masuk diterbitkan dengan topik 'topic/subtopic': {temperature: 50}

Muatan masuk diterbitkan dengan topik 'topic/subtopic-2': {temperature: 60}

Muatan masuk diterbitkan dengan topik 'topic/subtopic-3/details': {temperature: 70}

Muatan masuk diterbitkan dengan topik 'topic-2/subtopic-x': {temperature: 80}

```
SQL:"SELECT temperature AS t FROM 'topic/+'".
```

Aturan berlangganan semua topik dengan dua elemen jalur di mana elemen pertama berada 'topic'. Aturan dijalankan untuk pesan yang dikirim ke 'topic/subtopic' dan 'topic/subtopic-2', tetapi tidak 'topic/subtopic-3/details' (memiliki level lebih dari filter topik) atau 'topic-2/subtopic-x' (tidak dimulai dengan topic).

Klausula WHERE

Klausula WHERE menentukan apakah tindakan yang ditentukan oleh aturan dilakukan. Jika klausula WHERE mengevaluasi ke true, tindakan aturan dilakukan. Kalau tidak, tindakan aturan tidak dilakukan.

Klausula WHERE mendukung [Jenis data Operator](#), [Fungsi](#), [Literal](#), [Pernyataan kasus](#), [Ekstensi JSON Templat substitusi](#), dan [Kueri objek bersarang](#).

Contoh:

Muatan masuk dipublikasikan pada topic/subtopic: {"color":"red", "temperature":40}

```
SQL:SELECT color AS my_color FROM 'topic/subtopic' WHERE temperature > 50
AND color <> 'red'.
```

Dalam hal ini, aturan akan dipicu, tetapi tindakan yang ditentukan oleh aturan tidak akan dilakukan. Tidak akan ada muatan keluar.

Anda dapat menggunakan fungsi dan operator dalam klausula WHERE. Namun, Anda tidak dapat mereferensikan alias apa pun yang dibuat dengan kata kunci AS di SELECT. Klausula WHERE dievaluasi terlebih dahulu, untuk menentukan apakah SELECT dievaluasi.

Contoh dengan muatan non-JSON:

Payload non-JSON yang masuk diterbitkan pada `topic/subtopic`: `80`


```
SQL: `SELECT decode(encode(*, 'base64'), 'base64') AS value FROM 'topic/
subtopic' WHERE decode(encode(*, 'base64'), 'base64') > 50`
```

Dalam hal ini, aturan akan dipicu, dan tindakan yang ditentukan oleh aturan akan dilakukan. Payload keluar akan diubah oleh klausula SELECT sebagai payload JSON. {"value":80}

Jenis data

Mesin AWS IoT aturan mendukung semua tipe data JSON.

Jenis data yang didukung

Tipe	Arti
Int	Sebuah diskritInt. Maksimum 34 digit.
Decimal	<p>A Decimal dengan presisi 34 digit, dengan magnitudo bukan nol minimum 1E-999 dan magnitudo maksimum 9, 999... E999.</p> <div style="border: 1px solid #00a0e3; border-radius: 10px; padding: 10px; margin-top: 10px;"> <p> Note</p> <p>Beberapa fungsi mengembalikan Decimal nilai dengan presisi ganda daripada presisi 34 digit. Dengan SQL V2 (2016-03-23), nilai numerik yang merupakan bilangan bulat, seperti 10.0, diproses sebagai nilai (), bukan Int nilai yang diharapkan (10). Decimal 10.0 Untuk memproses nilai numerik bilangan bulat secara andal sebagai Decimal nilai, gunakan SQL V1 (2015-10-08) untuk pernyataan kueri aturan.</p> </div>
Boolean	True atau False.
String	Rangkaian UTF-8.
Array	Serangkaian nilai yang tidak harus memiliki tipe yang sama.
Object	Nilai JSON yang terdiri dari kunci dan nilai. Kunci harus berupa string. Nilai dapat berupa jenis apa saja.
Null	NullSeperti yang didefinisikan oleh JSON. Ini adalah nilai aktual yang mewakili tidak adanya nilai. Anda dapat secara eksplisit membuat

Tipe	Arti
	<p>Null nilai dengan menggunakan Null kata kunci dalam pernyataan SQL Anda. Misalnya: "SELECT NULL AS n FROM 'topic/su btopic'"</p>
Undefined	<p>Bukan nilai. Ini tidak dapat direpresentasikan secara eksplisit di JSON kecuali dengan menghilangkan nilainya. Misalnya, dalam objek{"foo": null}, kunci "foo" mengembalikan NULL, tetapi kunci "bar" kembali.</p> <p>Undefined Secara internal, bahasa SQL memperlakukan Undefined sebagai nilai, tetapi tidak dapat direpresentasikan dalam JSON, jadi ketika diserialkan ke JSON, hasilnya adalah. Undefined</p> <pre data-bbox="829 940 1507 1016">{"foo":null, "bar":undefined}</pre> <p>diserialisasikan ke JSON sebagai:</p> <pre data-bbox="829 1129 1507 1205">{"foo":null}</pre> <p>Demikian pula, Undefined dikonversi ke string kosong ketika diserialisasikan dengan sendirinya. Fungsi yang dipanggil dengan argumen yang tidak valid (misalnya, jenis yang salah, jumlah argumen yang salah, dan sebagainya) kembali. Undefined</p>

Konversi

Tabel berikut mencantumkan hasil ketika nilai dari satu jenis dikonversi ke jenis lain (ketika nilai dari jenis yang salah diberikan ke fungsi). Misalnya, jika fungsi nilai absolut "abs" (yang mengharapkan Int atauDecimal) diberikan aString, ia mencoba mengonversi String ke aDecimal, mengikuti aturan ini. Dalam hal ini, 'abs ("-5.123")' diperlakukan sebagai 'abs (-5.123)'.

Note

Tidak ada upaya konversi keArray,, ObjectNull, atau. Undefined

Untuk desimal

Jenis Argumen	Hasil
Int	A Decimal tanpa titik desimal.
Decimal	Nilai sumbernya.
Boolean	Undefined . (Anda dapat secara eksplisit menggunakan fungsi cast untuk mengubah true = 1.0, false = 0.0.)
String	Mesin SQL mencoba mengurai string sebagai file. Decimal AWS IoT mencoba mengurai string yang cocok dengan ekspresi reguler: $^-?\d+(\.\d+)?((?i)E-?\d+)?\$$ "0", "-1.2", "5E-12" adalah semua contoh string yang dikonversi secara otomatis ke s. Decimal
Susunan	Undefined .
Objek	Undefined .
Null	Null.
Tidak terdefinisi	Undefined .

Untuk int

Jenis Argumen	Hasil
Int	Nilai sumbernya.
Decimal	Nilai sumber dibulatkan ke yang terdekatInt.

Jenis Argumen	Hasil
Boolean	Undefined . (Anda dapat secara eksplisit menggunakan fungsi cast untuk mengubah true = 1.0, false = 0.0.)
String	Mesin SQL mencoba mengurai string sebagai file. Decimal AWS IoT mencoba mengurai string yang cocok dengan ekspresi reguler: $^-?\d+(\.\d+)?((?i)E-?\d+)?\$$ "0", "-1.2", "5E-12" adalah semua contoh string yang dikonversi secara otomatis menjadi Decimal s. AWS IoT mencoba untuk mengonversi String ke aDecimal, dan kemudian memotong tempat desimal untuk membuat. Decimal Int
Susunan	Undefined .
Objek	Undefined .
Null	Null.
Tidak terdefinisi	Undefined .

Untuk Boolean

Jenis Argumen	Hasil
Int	Undefined . (Anda dapat secara eksplisit menggunakan cast fungsi untuk mengubah 0 = False, any_nonzero_value = True.)
Decimal	Undefined . (Anda dapat secara eksplisit menggunakan fungsi cast untuk mengubah 0 = False, any_nonzero_value = True.)
Boolean	Nilai aslinya.

Jenis Argumen	Hasil
String	"True"=true dan "false"=false (case insensitive). Nilai string lainnya adalah Undefined .
Susunan	Undefined .
Objek	Undefined .
Null	Undefined .
Tidak terdefinisi	Undefined .

Untuk string

Jenis Argumen	Hasil
Int	Sebuah representasi string dari Int dalam notasi standar.
Decimal	String yang mewakili Decimal nilai, mungkin dalam notasi ilmiah.
Boolean	"benar" atau "salah". Semua huruf kecil.
String	Nilai aslinya.
Array	ArraySerial ke JSON. String yang dihasilkan adalah daftar yang dipisahkan koma, terlampir dalam tanda kurung siku. A String dikutip. ADecimal,Int,Boolean, dan Null tidak.
Objek	Objek diserialisasikan ke JSON. String yang dihasilkan adalah daftar pasangan kunci-nilai yang dipisahkan koma dan dimulai dan diakhiri dengan kurawal kurawal. A String dikutip. ADecimal,Int,Boolean, dan Null tidak.
Null	Undefined .

Jenis Argumen	Hasil
Tidak terdefinisi	Tidak terdefinisi.

Operator

Operator berikut dapat digunakan dalam klausa SELECT dan WHERE.

DAN operator

Mengembalikan Boolean hasil. Melakukan operasi dan logis. Mengembalikan nilai true jika operan kiri dan kanan benar. Jika tidak, mengembalikan false. Boolean diperlukan operan atau operan string “true” atau “false” yang tidak peka huruf besar/kecil.

Sintaks: *expression* AND *expression*.

DAN operator

Operan kiri	Operan kanan	Output
Boolean	Boolean	Boolean. Benar jika kedua operan benar. Kalau tidak, salah.
String/Boolean	String/Boolean	Jika semua string “benar” atau “salah” (case insensitive), mereka dikonversi ke Boolean dan diproses secara normal sebagai. <i>boolean</i> AND <i>boolean</i>
Nilai lainnya	Nilai lainnya	Undefined .

ATAU operator

Mengembalikan Boolean hasil. Melakukan operasi OR logis. Mengembalikan nilai true jika operan kiri atau kanan benar. Jika tidak, mengembalikan false. Boolean diperlukan operan atau operan string “true” atau “false” yang tidak peka huruf besar/kecil.

Sintaks: *expression* OR *expression*.

ATAU operator

Operan kiri	Operan kanan	Output
Boolean	Boolean	Boolean. Benar jika salah satu operan benar. Kalau tidak, salah.
String/Boolean	String/Boolean	Jika semua string “benar” atau “salah” (case insensiti ve), mereka dikonversi ke Boolean dan diproses secara normal sebagai. <i>boolean</i> OR <i>boolean</i>
Nilai lainnya	Nilai lainnya	Undefined .

BUKAN operator

Mengembalikan Boolean hasil. Melakukan operasi NOT logis. Mengembalikan nilai true jika operan palsu. Jika tidak, mengembalikan true. Diperlukan Boolean operan atau operan string “true” atau “false” yang tidak peka huruf besar/kecil.

Sintaks: NOT *expression*.

BUKAN operator

operan	Output
Boolean	Boolean. Benar jika operan salah. Kalau tidak, benar.
String	Jika string adalah “true” atau “false” (case insensitive), itu dikonversi ke nilai Boolean yang sesuai, dan nilai sebaliknya dikembalikan.
Nilai lainnya	Undefined .

Di operator

Mengembalikan Boolean hasil. Anda dapat menggunakan operator IN dalam klausa WHERE untuk memeriksa apakah nilai cocok dengan nilai apa pun dalam array. Ia mengembalikan true jika kecocokan ditemukan, dan false sebaliknya.

Sintaks: *expression* IN *expression*.

Di operator

Operan kiri	Operan kanan	Output
Int/Decimal/String/A	Array	Benar jika Object elemen IntegerDecimal/String/Array//ditemukan dalam array. Kalau tidak, salah.

Contoh:

```
SQL: "select * from 'a/b' where 3 in arr"
```

```
JSON: {"arr":[1, 2, 3, "three", 5.7, null]}
```

Dalam contoh ini, klausa kondisi `where 3 in arr` akan mengevaluasi ke `true` karena 3 hadir dalam array bernama `arr`. Oleh karena itu dalam pernyataan SQL, `select * from 'a/b'` akan mengeksekusi. Contoh ini juga menunjukkan bahwa array bisa heterogen.

Operator EXISTS

Mengembalikan Boolean hasil. Anda dapat menggunakan operator EXISTS dalam klausa bersyarat untuk menguji keberadaan elemen dalam subquery. Ia mengembalikan `true` jika subquery mengembalikan satu atau lebih elemen dan `false` jika subquery mengembalikan tidak ada elemen.

Sintaks: *expression*.

Contoh:

```
SQL: "select * from 'a/b' where exists (select * from arr as a where a = 3)"
```

```
JSON: {"arr":[1, 2, 3]}
```

Dalam contoh ini, klausa kondisi `where exists (select * from arr as a where a = 3)` akan mengevaluasi ke `true` karena 3 hadir dalam array bernama `arr`. Oleh karena itu dalam pernyataan SQL, `select * from 'a/b'` akan mengeksekusi.

Contoh:

```
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}]}
```

Dalam contoh ini, klausa kondisi `where exists (select * from e as e where foo = 2)` akan mengevaluasi ke `true` karena array `e` dalam objek JSON berisi objek `{"foo":2}`. Oleh karena itu dalam pernyataan SQL, `select * from 'a/b'` akan mengeksekusi.

> operator

Mengembalikan Boolean hasil. Mengembalikan nilai `true` jika operan kiri lebih besar dari operan kanan. Kedua operan diubah menjadi `aDecimal`, dan kemudian dibandingkan.

Sintaks: *expression* > *expression*.

> operator

Operan kiri	Operan kanan	Output
Int/Decimal	Int/Decimal	Boolean. Benar jika operan kiri lebih besar dari operan kanan. Kalau tidak, salah.
String/Int/Deci	String/Int/Deci	Jika semua string dapat dikonversi ke <code>Decimal</code> , maka <code>Boolean</code> . Mengembalikan nilai <code>true</code> jika operan kiri lebih besar dari operan kanan. Kalau tidak, salah.
Nilai lainnya	Undefined .	Undefined .

>= operator

Mengembalikan Boolean hasil. Mengembalikan nilai `true` jika operan kiri lebih besar dari atau sama dengan operan kanan. Kedua operan diubah menjadi `aDecimal`, dan kemudian dibandingkan.

Sintaks: *expression* >= *expression*.

>= operator

Operan kiri	Operan kanan	Output
Int/Decimal	Int/Decimal	Boolean. Benar jika operan kiri lebih besar dari atau sama dengan operan kanan. Kalau tidak, salah.
String/Int/Decimal	String/Int/Decimal	Jika semua string dapat dikonversi keDecimal, makaBoolean. Mengembalikan nilai true jika operan kiri lebih besar dari atau sama dengan operan kanan. Kalau tidak, salah.
Nilai lainnya	Undefined .	Undefined .

< operator

Mengembalikan Boolean hasil. Mengembalikan nilai true jika operan kiri kurang dari operan kanan. Kedua operan diubah menjadi aDecimal, dan kemudian dibandingkan.

Sintaks:*expression* < *expression*.

< operator

Operan kiri	Operan kanan	Output
Int/Decimal	Int/Decimal	Boolean. Benar jika operan kiri kurang dari operan kanan. Kalau tidak, salah.
String/Int/Decimal	String/Int/Decimal	Jika semua string dapat dikonversi keDecimal, makaBoolean. Mengembalikan nilai true jika operan kiri kurang dari operan kanan. Kalau tidak, salah.
Nilai lainnya	Undefined	Undefined

<= operator

Mengembalikan Boolean hasil. Mengembalikan nilai true jika operan kiri kurang dari atau sama dengan operan kanan. Kedua operan diubah menjadi aDecimal, dan kemudian dibandingkan.

Sintaks:*expression* <= *expression*.

<= operator

Operan kiri	Operan kanan	Output
Int/Decimal	Int/Decimal	Boolean. Benar jika operan kiri kurang dari atau sama dengan operan kanan. Kalau tidak, salah.
String/Int/Decimal	String/Int/Decimal	Jika semua string dapat dikonversi keDecimal, makaBoolean. Mengembalikan nilai true jika operan kiri kurang dari atau sama dengan operan kanan. Kalau tidak, salah.
Nilai lainnya	Undefined	Undefined

<> operator

Mengembalikan Boolean hasil. Mengembalikan nilai true jika operan kiri dan kanan tidak sama. Jika tidak, mengembalikan false.

Sintaks: *expression* <> *expression*.

<> operator

Operan kiri	Operan kanan	Output
Int	Int	Benar jika operan kiri tidak sama dengan operan kanan. Kalau tidak, salah.
Decimal	Decimal	Benar jika operan kiri tidak sama dengan operan kanan. Kalau tidak, salah. Int diubah menjadi Decimal sebelum dibandingkan.
String	String	Benar jika operan kiri tidak sama dengan operan kanan. Kalau tidak, salah.
Array	Array	Benar jika item di setiap operan tidak sama dan tidak dalam urutan yang sama. Jika tidak, salah
Objek	Objek	Benar jika kunci dan nilai masing-masing operan tidak sama. Kalau tidak, salah. Urutan kunci/nilai tidak penting.

Operan kiri	Operan kanan	Output
Null	Null	Salah.
Nilai apa pun	Undefined	Tidak terdefinisi.
Undefined	Nilai apa pun	Tidak terdefinisi.
Jenis tidak cocok	Jenis tidak cocok	Benar.

= operator

Mengembalikan Boolean hasil. Mengembalikan nilai true jika kedua operan kiri dan kanan sama. Jika tidak, mengembalikan false.

Sintaks: *expression* = *expression*.

= operator

Operan kiri	Operan kanan	Output
Int	Int	Benar jika operan kiri sama dengan operan kanan. Kalau tidak, salah.
Decimal	Decimal	Benar jika operan kiri sama dengan operan kanan. Kalau tidak, salah. Int diubah menjadi Decimal sebelum dibandingkan.
String	String	Benar jika operan kiri sama dengan operan kanan. Kalau tidak, salah.
Array	Array	Benar jika item di setiap operan sama dan dalam urutan yang sama. Kalau tidak, salah.
Objek	Objek	Benar jika kunci dan nilai masing-masing operan sama. Kalau tidak, salah. Urutan kunci/nilai tidak penting.
Nilai apa pun	Undefined	Undefined .
Undefined	Nilai apa pun	Undefined .

Operan kiri	Operan kanan	Output
Jenis tidak cocok	Jenis tidak cocok	Salah.

+ operator

“+” adalah operator yang kelebihan beban. Ini dapat digunakan untuk penggabungan string atau penambahan.

Sintaks: *expression* + *expression*.

+ operator

Operan kiri	Operan kanan	Output
String	Nilai apa pun	Mengkonversi operan kanan ke string dan menggabungkan itu ke akhir operan kiri.
Nilai apa pun	String	Mengkonversi operan kiri ke string dan menggabungkan operan kanan ke akhir operan kiri dikonversi.
Int	Int	Intnilai. Menambahkan operan bersama-sama.
Int/Decimal	Int/Decimal	DecimaInilai. Menambahkan operan bersama-sama.
Nilai lainnya	Nilai lainnya	Undefined .

- operator

Mengurangi operan kanan dari operan kiri.

Sintaks: *expression* - *expression*.

- operator

Operan kiri	Operan kanan	Output
Int	Int	Intnilai. Mengurangi operan kanan dari operan kiri.
Int/Decimal	Int/Decimal	DecimaInilai. Mengurangi operan kanan dari operan kiri.

Operan kiri	Operan kanan	Output
String/Int/Deci	String/Int/Deci	Jika semua string dikonversi ke desimal dengan benar, Decimal nilai dikembalikan. Mengurangi operan kanan dari operan kiri. Jika tidak, mengembalikan Undefined .
Nilai lainnya	Nilai lainnya	Undefined .
Nilai lainnya	Nilai lainnya	Undefined .

* Operator

Mengalikan operan kiri dengan operan kanan.

Sintaks: *expression* * *expression*.

* Operator

Operan kiri	Operan kanan	Output
Int	Int	Intnilai. Mengalikan operan kiri dengan operan kanan.
Int/Decimal	Int/Decimal	Decimalnilai. Mengalikan operan kiri dengan operan kanan.
String/Int/Deci	String/Int/Deci	Jika semua string dikonversi ke desimal dengan benar, Decimal nilai dikembalikan. Mengalikan operan kiri dengan operan kanan. Jika tidak, mengembalikan Undefined .
Nilai lainnya	Nilai lainnya	Undefined .

/operator

Membagi operan kiri dengan operan kanan.

Sintaks: *expression* / *expression*.

/operator

Operan kiri	Operan kanan	Output
Int	Int	Intnilai. Membagi operan kiri dengan operan kanan.
Int/Decimal	Int/Decimal	Decimalnilai. Membagi operan kiri dengan operan kanan.
String/Int/Deci	String/Int/Deci	Jika semua string dikonversi ke desimal dengan benar, Decimal nilai dikembalikan. Membagi operan kiri dengan operan kanan. Jika tidak, mengembalikan Undefined .
Nilai lainnya	Nilai lainnya	Undefined .

% operator

Mengembalikan sisanya dari membagi operan kiri dengan operan kanan.

Sintaks: *expression* % *expression*.

% operator

Operan kiri	Operan kanan	Output
Int	Int	Intnilai. Mengembalikan sisanya dari membagi operan kiri dengan operan kanan.
String/Int/Deci	String/Int/Deci	Jika semua string dikonversi ke desimal dengan benar, Decimal nilai dikembalikan. Mengembalikan sisanya dari membagi operan kiri dengan operan kanan. Atau, Undefined .
Nilai lainnya	Nilai lainnya	Undefined .

Fungsi

Anda dapat menggunakan fungsi bawaan berikut dalam klausa SELECT atau WHERE dari ekspresi SQL Anda.

abs (Desimal)

Mengembalikan nilai absolut dari sebuah angka. Didukung oleh SQL versi 2015-10-08 dan yang lebih baru.

Contoh: `abs(-5)` mengembalikan 5.

Jenis Argumen	Hasil
Int	Int, nilai absolut dari argumen.
Decimal	Decimal, nilai absolut dari argumen.
Boolean	Undefined .
String	Decimal. Hasilnya adalah nilai absolut dari argumen. Jika string tidak dapat dikonversi, hasilnya adalah Undefined .
Susunan	Undefined .
Objek	Undefined .
Null	Undefined .
Tidak terdefinisi	Undefined .

akuntid ()

Mengembalikan ID akun yang memiliki aturan ini sebagaiString. Didukung oleh SQL versi 2015-10-08 dan yang lebih baru.

Contoh:

```
accountid() = "123456789012"
```

acos (Desimal)

Mengembalikan cosinus terbalik dari angka dalam radian. DecimaIargumen dibulatkan ke presisi ganda sebelum aplikasi fungsi. Didukung oleh SQL versi 2015-10-08 dan yang lebih baru.

Contoh: $\text{acos}(0) = 1,5707963267948966$

Jenis Argumen	Hasil
Int	Decimal(dengan presisi ganda), kosinus terbalik dari argumen. Hasil imajiner dikembalikan sebagai Undefined .
Decimal	Decimal(dengan presisi ganda), kosinus terbalik dari argumen. Hasil imajiner dikembalikan sebagai Undefined .
Boolean	Undefined .
String	Decimal, kosinus terbalik dari argumen. Jika string tidak dapat dikonversi, hasilnya adalah Undefined . Hasil imajiner dikembalikan sebagai Undefined .
Susunan	Undefined .
Objek	Undefined .
Null	Undefined .
Tidak terdefinisi	Undefined .

asin (Desimal)

Mengembalikan sinus terbalik dari angka dalam radian. Decimal argumen dibulatkan ke presisi ganda sebelum aplikasi fungsi. Didukung oleh SQL versi 2015-10-08 dan yang lebih baru.

Contoh: $\text{asin}(0) = 0.0$

Jenis Argumen	Hasil
Int	Decimal(dengan presisi ganda), sinus terbalik dari argumen. Hasil imajiner dikembalikan sebagai Undefined .

Jenis Argumen	Hasil
Decimal	Decimal(dengan presisi ganda), sinus terbalik dari argumen. Hasil imajiner dikembalikan sebagaiUndefined .
Boolean	Undefined .
String	Decimal(dengan presisi ganda), sinus terbalik dari argumen. Jika string tidak dapat dikonversi, hasilnya adalahUndefined . Hasil imajiner dikembalikan sebagaiUndefined .
Susunan	Undefined .
Objek	Undefined .
Null	Undefined .
Tidak terdefinisi	Undefined .

atan (Desimal)

Mengembalikan tangen terbalik dari angka dalam radian. Decimal argumen dibulatkan ke presisi ganda sebelum aplikasi fungsi. Didukung oleh SQL versi 2015-10-08 dan yang lebih baru.

Contoh: $\text{atan}(0) = 0.0$

Jenis Argumen	Hasil
Int	Decimal(dengan presisi ganda), singgung terbalik dari argumen. Hasil imajiner dikembalikan sebagaiUndefined .
Decimal	Decimal(dengan presisi ganda), singgung terbalik dari argumen. Hasil imajiner dikembalikan sebagaiUndefined .

Jenis Argumen	Hasil
Boolean	Undefined .
String	Decimal, singgung kebalikan dari argumen. Jika string tidak dapat dikonversi, hasilnya adalah Undefined . Hasil imajiner dikembalikan sebagai Undefined .
Susunan	Undefined .
Objek	Undefined .
Null	Undefined .
Tidak terdefinisi	Undefined .

atan2 (Desimal, Desimal)

Mengembalikan sudut, dalam radian, antara sumbu x positif dan titik (x, y) yang didefinisikan dalam dua argumen. Sudutnya positif untuk sudut berlawanan arah jarum jam (setengah bidang atas, $y > 0$), dan negatif untuk sudut searah jarum jam (setengah bidang bawah, $y < 0$). Decimal argumen dibulatkan ke presisi ganda sebelum aplikasi fungsi. Didukung oleh SQL versi 2015-10-08 dan yang lebih baru.

Contoh: $\text{atan2}(1, 0) = 1,5707963267948966$

Jenis Argumen	Jenis Argumen	Hasil
Int/Decimal	Int/Decimal	Decimal(dengan presisi ganda antara sumbu x dan titik (x, y) yang ditentukan).
Int/Decimal/String	Int/Decimal/String	Decimal, singgung terbalik dan dijelaskan. Jika string tidak dapat diinterpretasikan sebagai angka, hasilnya adalah Undefined .
Nilai lainnya	Nilai lainnya	Undefined .

aws_lambda (FunctionARN, InputJson)

Memanggil fungsi Lambda yang ditentukan lewat `inputJson` ke fungsi Lambda dan mengembalikan JSON yang dihasilkan oleh fungsi Lambda.

Pendapat

Pendapat	Deskripsi
<code>functionArn</code>	ARN dari fungsi Lambda untuk memanggil. Fungsi Lambda harus mengembalikan data JSON.
<code>inputJson</code>	Input JSON diteruskan ke fungsi Lambda. Untuk meneruskan kueri dan literal objek bersarang, Anda harus menggunakan versi SQL 2016-03-23.

Anda harus memberikan `AWS IoT lambda:InvokeFunction` izin untuk menjalankan fungsi Lambda yang ditentukan. Contoh berikut menunjukkan cara memberikan `lambda:InvokeFunction` izin menggunakan 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"
```

Berikut ini adalah argumen untuk `add-permission` perintah:

`--fungsi-nama`

Nama fungsi Lambda. Anda menambahkan izin baru untuk memperbarui kebijakan sumber daya fungsi.

`--wilayah`

Akun Anda. Wilayah AWS

--kepala sekolah

Kepala sekolah yang mendapatkan izin. Ini harus `iot.amazonaws.com` untuk mengizinkan AWS IoT izin memanggil fungsi Lambda.

--sumber-arn

ARN dari aturan tersebut. Anda dapat menggunakan `get-topic-rule` AWS CLI perintah untuk mendapatkan ARN aturan.

--source-akun

Di Akun AWS mana aturan didefinisikan.

--pernyataan-id

Pengidentifikasi pernyataan unik.

--tindakan

Tindakan Lambda yang ingin Anda izinkan dalam pernyataan ini. AWS IoT Untuk memungkinkan menjalankan fungsi Lambda, tentukan `lambda:InvokeFunction`

⚠ Important

Jika Anda menambahkan izin untuk AWS IoT prinsipal tanpa memberikan `source-arn` atau `source-account`, aturan apa pun Akun AWS yang membuat aturan dengan tindakan Lambda Anda dapat memicu aturan untuk menjalankan fungsi Lambda Anda. AWS IoT Untuk informasi selengkapnya, lihat Model [Izin Lambda](#).

Diberikan muatan pesan JSON seperti:

```
{
  "attribute1": 21,
  "attribute2": "value"
}
```

`aws_lambdaFungsi` tersebut dapat digunakan untuk memanggil fungsi Lambda sebagai berikut.

```
SELECT
```

```
aws_lambda("arn:aws:lambda:us-east-1:account_id:function:lambda_function",
{"payload":attribute1}) as output FROM 'topic-filter'
```

Jika Anda ingin meneruskan muatan pesan MQTT lengkap, Anda dapat menentukan payload JSON menggunakan '*', seperti contoh berikut.

```
SELECT
aws_lambda("arn:aws:lambda:us-east-1:account_id:function:lambda_function", *) as output
FROM 'topic-filter'
```

`payload.inner.element` memilih data dari pesan yang dipublikasikan pada topik 'topik/subtopik'.

`some.value` memilih data dari output yang dihasilkan oleh fungsi Lambda.

Note

Mesin aturan membatasi durasi eksekusi fungsi Lambda. Panggilan fungsi Lambda dari aturan harus diselesaikan dalam 2000 milidetik.

bit (Int, Int)

Melakukan bitwise AND pada representasi bit dari dua argumen Int (-convert). Didukung oleh SQL versi 2015-10-08 dan yang lebih baru.

Contoh: `bitand(13, 5) = 5`

Jenis Argumen	Jenis Argumen	Hasil
Int	Int	Int, bitwise DAN dari dua argumen
Int/Decimal	Int/Decimal	Int, bitwise DAN dari dua argumen Semua non-Int angka dibulatkan ke bawah ke yang terdekat Int. Jika satu argumen tidak dapat dikonversi ke Int, hasilnya adalah Undefined
Int/Decimal/String	Int/Decimal/String	Int, bitwise DAN dari dua argumen Semua string dikonversi ke desimal

Jenis Argumen	Jenis Argumen	Hasil
		dibulatkan ke bawah ke terdekat konversi gagal, hasilnya adalah .
Nilai lainnya	Nilai lainnya	Undefined .

bitor (Int, Int)

Melakukan bitwise OR dari representasi bit dari dua argumen. Didukung oleh SQL versi 2015-10-08 dan yang lebih baru.

Contoh: `bitor(8, 5) = 13`

Jenis Argumen	Jenis Argumen	Hasil
Int	Int	Int, bitwise OR dari dua argumen
Int/Decimal	Int/Decimal	Int, bitwise OR dari dua argumen non- Int angka dibulatkan ke b yang terdekatInt. Jika konversi hasilnya adalahUndefined .
Int/Decimal/String	Int/Decimal/String	Int, bitwise OR pada dua argumen Semua string dikonversi ke desimal dibulatkan ke bawah ke terdekat konversi gagal, hasilnya adalah .
Nilai lainnya	Nilai lainnya	Undefined .

bitxor (Int, Int)

Melakukan XOR bitwise pada representasi bit dari dua argumen Int (-convert). Didukung oleh SQL versi 2015-10-08 dan yang lebih baru.

Contoh: `bitxor(13, 5) = 8`

Jenis Argumen	Jenis Argumen	Hasil
Int	Int	Int, XOR bitwise pada dua arg
Int/Decimal	Int/Decimal	Int, XOR bitwise pada dua arg Int angka dibulatkan ke bawah terdekatInt.
Int/Decimal/String	Int/Decimal/String	Int, XOR bitwise pada dua arg diubah menjadi desimal dan dib bawah ke terdekat. Int Jika ad yang gagal, hasilnya adalahUnd
Nilai lainnya	Nilai lainnya	Undefined .

bitnot (Int)

Melakukan bitwise NOT pada representasi bit dari argumen Int (-convert). Didukung oleh SQL versi 2015-10-08 dan yang lebih baru.

Contoh: `bitnot(13) = 2`

Jenis Argumen	Hasil
Int	Int, sedikit demi sedikit dari argumen.
Decimal	Int, sedikit demi sedikit dari argumen. DecimalNilai dibulatkan ke bawah ke yang terdekatInt.
String	Int, sedikit demi sedikit dari argumen. String dikonversi menjadi desimal dan dibulatkan ke bawah ke yang terdekat. Int Jika ada konversi yang gagal, hasilnya adalahUndefined .
Nilai lainnya	Nilai lainnya.

pemeran ()

Mengkonversi nilai dari satu tipe data ke yang lain. Pemeran berperilaku sebagian besar seperti konversi standar, dengan penambahan kemampuan untuk mentransmisikan angka ke atau dari Booleans. Jika AWS IoT tidak dapat menentukan cara melemparkan satu jenis ke jenis lainnya, hasilnya adalah `Undefined`. Didukung oleh SQL versi 2015-10-08 dan yang lebih baru. Format: `cast(nilai sebagai tipe)`.

Contoh:

```
cast(true as Int) = 1
```

Kata kunci berikut mungkin muncul setelah “as” saat menelepon `cast`:

Untuk versi SQL 2015-10-08 dan 2016-03-23

Kata Kunci	Hasil
String	Memberikan nilai ke <code>String</code> .
Nvarchar	Memberikan nilai ke <code>String</code> .
Teks	Memberikan nilai ke <code>String</code> .
Nteks	Memberikan nilai ke <code>String</code> .
varchar	Memberikan nilai ke <code>String</code> .
Int	Memberikan nilai ke <code>Int</code> .
Bilangan Bulat	Memberikan nilai ke <code>Int</code> .
Ganda	Memberikan nilai ke <code>Decimal</code> (dengan presisi ganda).


Selain itu, untuk versi SQL 2016-03-23

Kata Kunci	Hasil
<code>Decimal</code>	Memberikan nilai ke <code>Decimal</code> .

Kata Kunci	Hasil
Bool	Memberikan nilai keBoolean.
Boolean	Memberikan nilai keBoolean.

Aturan casting:

Cast ke desimal

Jenis Argumen	Hasil
Int	A Decimal tanpa titik desimal.
Decimal	<p>Nilai sumbernya.</p> <div data-bbox="511 844 1136 1449" style="border: 1px solid #add8e6; border-radius: 10px; padding: 10px;"> <p> Note</p> <p>Dengan SQL V2 (2016-03-23), nilai numerik yang merupakan bilangan bulat, seperti <code>10.0</code>, mengembalikan Int nilai (<code>10</code>) alih-alih nilai yang diharapkan (<code>()</code>). <code>Decimal 10.0</code> Untuk secara andal melemparkan nilai numerik bilangan bulat sebagai Decimal nilai, gunakan SQL V1 (2015-10-08) untuk pernyataan kueri aturan.</p> </div>
Boolean	benar = 1,0, salah = 0,0.
String	Mencoba mengurai string sebagai. <code>Decimal</code> AWS IoT mencoba mengurai string yang cocok dengan regex: <code>^-? \d+ (\. \d+)? ((? i) E-? \d+)? \$</code> . "0", "-1.2", "5E-12" adalah semua contoh string yang dikonversi secara otomatis menjadi desimal.

Jenis Argumen	Hasil
Susunan	Undefined .
Objek	Undefined .
Null	Undefined .
Tidak terdefinisi	Undefined .

Cast ke int

Jenis Argumen	Hasil
Int	Nilai sumbernya.
Decimal	Nilai sumber, dibulatkan ke bawah ke yang terdekatInt.
Boolean	benar = 1,0, salah = 0,0.
String	Mencoba mengurai string sebagai. Decimal AWS IoT mencoba mengurai string yang cocok dengan regex: <code>^-? \d+ (\.\d+)? ((? i) E-? \d+)? \$</code> . "0", "-1.2", "5E-12" adalah semua contoh string yang dikonversi secara otomatis menjadi desimal. AWS IoT mencoba untuk mengubah string ke a Decimal dan membulatkan ke bawah ke yang terdekatInt.
Susunan	Undefined .
Objek	Undefined .
Null	Undefined .
Tidak terdefinisi	Undefined .

Melempar ke **Boolean**

Jenis Argumen	Hasil
Int	0 = Salah, any_nonzero_value = Benar.
Decimal	0 = Salah, any_nonzero_value = Benar.
Boolean	Nilai sumbernya.
String	“true” = True dan “false” = False (case insensitive). Nilai string lainnya =Undefined .
Susunan	Undefined .
Objek	Undefined .
Null	Undefined .
Tidak terdefinisi	Undefined .

Cast ke string

Jenis Argumen	Hasil
Int	Sebuah representasi string dariInt, dalam notasi standar.
Decimal	String yang mewakili Decimal nilai, mungkin dalam notasi ilmiah.
Boolean	“benar” atau “salah”, semua huruf kecil.
String	“True”=true dan “false”=false (case-insensitive). Nilai string lainnya =Undefined .
Array	Array diserialisasikan ke JSON. String hasil adalah daftar dipisahkan koma yang

Jenis Argumen	Hasil
	diapit tanda kurung siku. String diikuti. Decimal, Int, dan Boolean tidak.
Objek	Objek diserialisasikan ke JSON. String JSON adalah daftar pasangan kunci-nilai yang dipisahkan koma dan dimulai dan diakhiri dengan kurawal kurawal. String diikuti. Decimal, Int, Boolean, dan Null tidak.
Null	Undefined .
Tidak terdefinisi	Undefined .

ceil (Desimal)

Membulatkan yang diberikan Decimal ke yang terdekat Int. Didukung oleh SQL versi 2015-10-08 dan yang lebih baru.

Contoh:

`ceil(1.2)= 2`

`ceil(-1.2)= 1`

Jenis Argumen	Hasil
Int	Int, nilai argumen.
Decimal	Int, Decimal nilainya dibulatkan ke yang terdekat Int.
String	Int. String dikonversi ke Decimal dan dibulatkan ke yang terdekat Int. Jika string tidak dapat dikonversi ke aDecimal, hasilnya adalah Undefined .

Jenis Argumen	Hasil
Nilai lainnya	Undefined .

chr (Tali)

Mengembalikan karakter ASCII yang sesuai dengan argumen yang diberikan `Int`. Didukung oleh SQL versi 2015-10-08 dan yang lebih baru.

Contoh:

`chr(65)` = "A".

`chr(49)` = "1".

Jenis Argumen	Hasil
<code>Int</code>	Karakter yang sesuai dengan nilai ASCII yang ditentukan. Jika argumen tersebut bukan nilai ASCII yang valid, hasilnya adalah <code>Undefined</code>
<code>Decimal</code>	Karakter yang sesuai dengan nilai ASCII yang ditentukan. <code>DecimalArgumen</code> dibulatkan ke bawah ke yang terdekat <code>Int</code> . Jika argumen tersebut bukan nilai ASCII yang valid, hasilnya adalah <code>Undefined</code>
<code>Boolean</code>	<code>Undefined</code> .
<code>String</code>	Jika <code>String</code> dapat dikonversi ke <code>aDecimal</code> , itu dibulatkan ke bawah ke terdekat <code>Int</code> . Jika argumen tersebut bukan nilai ASCII yang valid, hasilnya adalah <code>Undefined</code>
<code>Susunan</code>	<code>Undefined</code> .
<code>Objek</code>	<code>Undefined</code> .

Jenis Argumen	Hasil
Null	Undefined .
Nilai lainnya	Undefined .

klien ()

Mengembalikan ID klien MQTT yang mengirim pesan, atau n/a jika pesan tidak dikirim melalui MQTT. Didukung oleh SQL versi 2015-10-08 dan yang lebih baru.

Contoh:

```
clientid() = "123456789012"
```

concat ()

Menggabung array atau string. Fungsi ini menerima sejumlah argumen dan mengembalikan a String atau Array. Didukung oleh SQL versi 2015-10-08 dan yang lebih baru.

Contoh:

```
concat() = Undefined.
```

```
concat(1) = "1".
```

```
concat([1, 2, 3], 4)=[1, 2, 3, 4].
```

```
concat([1, 2, 3], "hello")=[1, 2, 3, "halo"]
```

```
concat("con", "cat")= "concat"
```

```
concat(1, "hello")= "1halo"
```

```
concat("he", "is", "man")= "heisman"
```

```
concat([1, 2, 3], "hello", [4, 5, 6])=[1, 2, 3, "halo", 4, 5, 6]
```

Jumlah argumen	Hasil
0	Undefined .

Jumlah argumen	Hasil
1	Argumen dikembalikan tanpa dimodifikasi.
2+	Jika ada argumen <code>Array</code> , hasilnya adalah array tunggal yang berisi semua argumen. Jika tidak ada argumen yang merupakan array, dan setidaknya satu argumen adalah <code>aString</code> , hasilnya adalah gabungan dari <code>String</code> representasi semua argumen. Argumen dikonversi ke string menggunakan konversi standar yang sebelumnya terdaftar .

cos (Desimal)

Mengembalikan cosinus dari angka dalam radian. `Decimal` argumen dibulatkan ke presisi ganda sebelum aplikasi fungsi. Didukung oleh SQL versi 2015-10-08 dan yang lebih baru.

Contoh:

`cos(0)= 1.`

Jenis Argumen	Hasil
<code>Int</code>	<code>Decimal</code> (dengan presisi ganda), kosinus argumen. Hasil imajiner dikembalikan sebagai <code>Undefined</code> .
<code>Decimal</code>	<code>Decimal</code> (dengan presisi ganda), kosinus argumen. Hasil imajiner dikembalikan sebagai <code>Undefined</code> .
<code>Boolean</code>	<code>Undefined</code> .
<code>String</code>	<code>Decimal</code> (dengan presisi ganda), kosinus argumen. Jika string tidak dapat dikonversi ke <code>aDecimal</code> , hasilnya

Jenis Argumen	Hasil
	adalah <code>Undefined</code> . Hasil imajiner dikembalikan sebagai <code>Undefined</code> .
Susunan	<code>Undefined</code> .
Objek	<code>Undefined</code> .
Null	<code>Undefined</code> .
Tidak terdefinisi	<code>Undefined</code> .

cosh (Desimal)

Mengembalikan kosinus hiperbolik dari suatu angka dalam radian. `Decimal` argumen dibulatkan ke presisi ganda sebelum aplikasi fungsi. Didukung oleh SQL versi 2015-10-08 dan yang lebih baru.

Contoh: `cosh(2.3) = 5.037220649268761`.

Jenis Argumen	Hasil
Int	<code>Decimal</code> (dengan presisi ganda), kosinus hiperbolik argumen. Hasil imajiner dikembalikan sebagai <code>Undefined</code> .
Decimal	<code>Decimal</code> (dengan presisi ganda), kosinus hiperbolik argumen. Hasil imajiner dikembalikan sebagai <code>Undefined</code> .
Boolean	<code>Undefined</code> .
String	<code>Decimal</code> (dengan presisi ganda), kosinus hiperbolik argumen. Jika string tidak dapat dikonversi ke <code>aDecimal</code> , hasilnya adalah <code>Undefined</code> . Hasil imajiner dikembalikan sebagai <code>Undefined</code> .
Susunan	<code>Undefined</code> .

Jenis Argumen	Hasil
Objek	Undefined .
Null	Undefined .
Tidak terdefinisi	Undefined .

decode (nilai, decodingScheme)

Gunakan decode fungsi untuk memecahkan kode nilai yang dikodekan. Jika string yang diterjemahkan adalah dokumen JSON, objek yang dapat dialamatkan dikembalikan. Jika tidak, string yang diterjemahkan dikembalikan sebagai string. Fungsi mengembalikan NULL jika string tidak dapat diterjemahkan. Fungsi ini mendukung decoding string yang dikodekan base64 dan format pesan Protocol Buffer (protobuf).

Didukung oleh SQL versi 2016-03-23 dan yang lebih baru.

nilai

Nilai string atau salah satu ekspresi yang valid, seperti yang didefinisikan dalam [AWS IoT Referensi SQL](#), yang mengembalikan string.

DecodingScheme

String literal yang mewakili skema yang digunakan untuk memecahkan kode nilai. Saat ini, hanya 'base64' dan 'proto' didukung.

Decoding string yang dikodekan base64

Dalam contoh ini, payload pesan menyertakan nilai yang dikodekan.

```
{
  encoded_temp: "eyAidGVtcGVyYXR1cmUiOiAzMyB9Cg=="
}
```

decodeFungsi dalam pernyataan SQL ini menerjemahkan nilai dalam payload pesan.

```
SELECT decode(encoded_temp,"base64").temperature AS temp from 'topic/subtopic'
```

Decoding encoded_temp nilai menghasilkan dokumen JSON valid berikut, yang memungkinkan pernyataan SELECT membaca nilai suhu.

```
{ "temperature": 33 }
```

Hasil dari pernyataan SELECT dalam contoh ini ditampilkan di sini.

```
{ "temp": 33 }
```

Jika nilai decoded bukan dokumen JSON yang valid, nilai decoded akan dikembalikan sebagai string.

Decoding payload pesan protobuf

Anda dapat menggunakan fungsi decode SQL untuk mengkonfigurasi Aturan yang dapat memecahkan kode payload pesan protobuf Anda. Untuk informasi selengkapnya, lihat [Menguraikan muatan pesan protobuf](#).

Tanda tangan fungsi terlihat seperti berikut:

```
decode(<ENCODED DATA>, 'proto', '<S3 BUCKET NAME>', '<S3 OBJECT KEY>', '<PROTO NAME>',  
'<MESSAGE TYPE>')
```

ENCODED DATA

Menentukan data protobuf-dikodekan yang akan diterjemahkan. Jika seluruh pesan yang dikirim ke Aturan adalah data yang disandikan protobuf, Anda dapat mereferensikan payload masuk biner mentah menggunakan `*`. Jika tidak, bidang ini harus berupa string JSON yang dikodekan basis-64 dan referensi ke string dapat diteruskan secara langsung.

1) Untuk memecahkan kode muatan masuk protobuf biner mentah:

```
decode(*, 'proto', ...)
```

2) Untuk memecahkan kode pesan yang disandikan protobuf yang diwakili oleh string berkode base64 'a.b':

```
decode(a.b, 'proto', ...)
```

proto

Menentukan data yang akan diterjemahkan dalam format pesan protobuf. Jika Anda menentukan base64 alih-alih proto, fungsi ini akan memecahkan kode string yang dikodekan base64 sebagai JSON.

S3_BUCKET_NAME

Nama bucket Amazon S3 tempat Anda mengunggah file Anda. FileDescriptorSet

S3_OBJECT_KEY

Kunci objek yang menentukan FileDescriptorSet file dalam bucket Amazon S3.

PROTO_NAME

Nama .proto file (tidak termasuk ekstensi) dari mana FileDescriptorSet file itu dihasilkan.

MESSAGE_TYPE

Nama struktur pesan protobuf di dalam FileDescriptorSet file, yang data yang akan didekodekan harus sesuai.

Contoh ekspresi SQL menggunakan fungsi decode SQL dapat terlihat seperti berikut:

```
SELECT VALUE decode(*, 'proto', 's3-bucket', 'messageformat.desc', 'myproto',
'messagetype') FROM 'some/topic'
```

- *

Merupakan payload masuk biner, yang sesuai dengan jenis pesan protobuf yang disebut. mymessagetype

- messageformat.desc

FileDescriptorSetFile yang disimpan dalam ember Amazon S3 bernama. s3-bucket

- myproto

.protoFile asli yang digunakan untuk menghasilkan FileDescriptorSet file bernamamyproto.proto.

- messagetype

Jenis pesan yang disebut messagetype (bersama dengan dependensi yang diimpor) seperti yang didefinisikan dalam. myproto.proto

encode (nilai, EncodingScheme)

Gunakan encode fungsi untuk menyandikan payload, yang berpotensi menjadi data non-JSON, ke dalam representasi string berdasarkan skema pengkodean. Didukung oleh SQL versi 2016-03-23 dan yang lebih baru.

nilai

Salah satu ekspresi yang valid, sebagaimana didefinisikan dalam [AWS IoT Referensi SQL](#). Anda dapat menentukan * untuk menyandikan seluruh muatan, terlepas dari apakah itu dalam format JSON. Jika Anda memberikan ekspresi, hasil evaluasi dikonversi ke string sebelum dikodekan.

encodingScheme

String literal yang mewakili skema pengkodean yang ingin Anda gunakan. Saat ini, hanya 'base64' didukung.

endswith (String, String)

Mengembalikan Boolean menunjukkan apakah String argumen pertama berakhir dengan String argumen kedua. Jika salah satu argumen adalah Null atau Undefined, hasilnya adalah Undefined. Didukung oleh SQL versi 2015-10-08 dan yang lebih baru.

Contoh: `endswith("cat", "at") = benar`.

Tipe argumen 1	Argumen tipe 2	Hasil
String	String	Benar jika argumen pertama berakhir dengan argumen kedua. Kalau tidak, salah.
Nilai lainnya	Nilai lainnya	Kedua argumen dikonversi ke string menggunakan aturan konversi SQL. Benar jika argumen pertama berakhir dengan argumen kedua. Kalau tidak, salah. Jika salah satu argumen adalah Null atau Undefined, hasilnya adalah Undefined.

exp (Desimal)

Mengembalikan e diangkat ke Decimal argumen. Decimal argumen dibulatkan ke presisi ganda sebelum aplikasi fungsi. Didukung oleh SQL versi 2015-10-08 dan yang lebih baru.

Contoh: $\text{exp}(1) = e$.

Jenis Argumen	Hasil
Int	Decimal(dengan presisi ganda), e^{\wedge} argumen.
Decimal	Decimal(dengan presisi ganda), e^{\wedge} argumen.
String	Decimal(dengan presisi ganda), e^{\wedge} argumen. Jika String tidak dapat dikonversi ke aDecimal, hasilnya adalah Undefined .
Nilai lainnya	Undefined .

lantai (Desimal)

Membulatkan yang diberikan Decimal ke bawah ke yang terdekat Int. Didukung oleh SQL versi 2015-10-08 dan yang lebih baru.

Contoh:

$\text{floor}(1.2) = 1$

$\text{floor}(-1.2) = -2$

Jenis Argumen	Hasil
Int	Int, nilai argumen.
Decimal	Int, Decimal nilai dibulatkan ke bawah ke terdekat Int.

Jenis Argumen	Hasil
String	Int. String dikonversi ke Decimal dan dibulatkan ke bawah ke yang terdekatInt. Jika string tidak dapat dikonversi ke aDecimal, hasilnya adalahUndefined .
Nilai lainnya	Undefined .

memperoleh

Mengekstrak nilai dari tipe seperti koleksi (Array, String, Object). Tidak ada konversi yang diterapkan pada argumen pertama. Konversi berlaku seperti yang didokumentasikan dalam tabel ke argumen kedua. Didukung oleh SQL versi 2015-10-08 dan yang lebih baru.

Contoh:

```
get(["a", "b", "c"], 1) = "b"
```

```
get({"a":"b"}, "a")= "b"
```

```
get("abc", 0)= "a"
```

Tipe argumen 1	Argumen tipe 2	Hasil
Array	Tipe apa pun (dikonversi keInt)	Item pada indeks berbasis 0 dari Array disediakan oleh argumen kedua (dikonversi keInt). Jika konversi berhasil, hasilnya adalahUndefined. Jika indeks berada di luar batas Array (negatif atau >= array.length), hasilnya adalahUndefined.
String	Tipe apa pun (dikonversi keInt)	Karakter pada indeks 0 berbasis string disediakan oleh argumen kedua (dikonversi keInt). Jika konversi tidak berhasil, hasilnya adalahUndefined. Jika indeks berada di luar batas string (negatif atau >= string.length), hasilnya adalahUndefined.

Tipe argumen 1	Argumen tipe 2	Hasil
		string.length), hasilnya adalah. U
Objek	String(tidak ada konversi yang diterapkan)	Nilai yang disimpan dalam objek pertama yang sesuai dengan kunci disediakan sebagai argumen ke
Nilai lainnya	Nilai apa pun	Undefined .

`get_dynamodb (TableName,,,,, roLearn partitionKeyName) partitionKeyValue
sortKeyName sortKeyValue`

Mengambil data dari tabel DynamoDB. `get_dynamodb()` memungkinkan Anda untuk query tabel DynamoDB sementara aturan dievaluasi. Anda dapat memfilter atau menambah muatan pesan menggunakan data yang diambil dari DynamoDB. Didukung oleh SQL versi 2016-03-23 dan yang lebih baru.

`get_dynamodb()` mengambil parameter berikut:

`tableName`

Nama tabel DynamoDB untuk query.

`partitionKeyName`

Nama kunci partisi. Untuk informasi selengkapnya, lihat [DynamoDB Keys](#).

`partitionKeyValue`

Nilai kunci partisi yang digunakan untuk mengidentifikasi catatan. Untuk informasi selengkapnya, lihat [DynamoDB Keys](#).

`sortKeyName`

(Opsional) Nama kunci sortir. Parameter ini diperlukan hanya jika tabel DynamoDB query menggunakan kunci komposit. Untuk informasi selengkapnya, lihat [DynamoDB Keys](#).

`sortKeyValue`

(Opsional) Nilai kunci sortir. Parameter ini diperlukan hanya jika tabel DynamoDB query menggunakan kunci komposit. Untuk informasi selengkapnya, lihat [DynamoDB Keys](#).

roleArn

ARN dari peran IAM yang memberikan akses ke tabel DynamoDB. Mesin aturan mengasumsikan peran ini untuk mengakses tabel DynamoDB atas nama Anda. Hindari menggunakan peran yang terlalu permisif. Berikan peran hanya izin yang diperlukan oleh aturan. Berikut ini adalah contoh kebijakan yang memberikan akses ke satu tabel DynamoDB.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "dynamodb:GetItem",
      "Resource": "arn:aws:dynamodb:aws-region:account-id:table/table-name"
    }
  ]
}
```

Sebagai contoh cara menggunakan `get_dynamodb()`, katakanlah Anda memiliki tabel DynamoDB yang berisi ID perangkat dan informasi lokasi untuk semua perangkat yang terhubung. AWS IoT Pernyataan SELECT berikut menggunakan `get_dynamodb()` fungsi untuk mengambil lokasi untuk ID perangkat yang ditentukan:

```
SELECT *, get_dynamodb("InServiceDevices", "deviceId", id,
"arn:aws:iam::12345678910:role/getdynamo").location AS location FROM 'some/
topic'
```

Note

- Anda dapat memanggil maksimum `get_dynamodb()` satu kali per pernyataan SQL. Memanggil `get_dynamodb()` beberapa kali dalam satu pernyataan SQL menyebabkan aturan berakhir tanpa menjalankan tindakan apa pun.
- Jika `get_dynamodb()` mengembalikan lebih dari 8 KB data, tindakan aturan mungkin tidak dipanggil.

get_mqtt_property (nama)

Referensi salah satu header MQTT5 berikut: `contentType`, `payloadFormatIndicator` dan `responseTopic correlationData`. Fungsi ini mengambil salah satu string literal berikut sebagai argumen: `content_type`, `format_indicator` `response_topic`, dan `correlation_data`. Untuk informasi selengkapnya, lihat tabel Argumen fungsi berikut.

ContentType

String: String yang dikodekan UTF-8 yang menjelaskan isi pesan penerbitan.

payloadFormatIndikator

String: Nilai string Enum yang menunjukkan apakah payload diformat sebagai UTF-8. Nilai yang valid adalah `UNSPECIFIED_BYTES` dan `UTF8_DATA`.

ResponseTopik

String: String yang dikodekan UTF-8 yang digunakan sebagai nama topik untuk pesan respons. Topik respons digunakan untuk menggambarkan topik yang harus dipublikasikan oleh penerima sebagai bagian dari aliran permintaan-respons. Topik tidak boleh berisi karakter wildcard.

KorelasiData

String: Data biner berenkod base64 yang digunakan oleh pengirim Pesan Permintaan untuk mengidentifikasi permintaan Pesan Respons saat diterima.

Tabel berikut menunjukkan argumen fungsi yang dapat diterima dan jenis pengembalian terkait untuk `get_mqtt_property` fungsi tersebut:

Argumen fungsi

SQL	Tipe data yang dikembalikan (jika ada)	Tipe data yang dikembalikan (jika tidak ada)
<code>get_mqtt_property("format_indicator")</code>	String (<code>UNSPECIFIED_BYTES</code> atau <code>UTF8_DATA</code>)	String (<code>UNSPECIFIED_BYTES</code>)
<code>get_mqtt_property("content_type")</code>	String	Tidak terdefinisi

SQL	Tipe data yang dikembalikan (jika ada)	Tipe data yang dikembalikan (jika tidak ada)
<code>get_mqtt_property("response_topic")</code>	String	Tidak terdefinisi
<code>get_mqtt_property("correlation_data")</code>	base64 dikodekan String	Tidak terdefinisi
<code>get_mqtt_property("some_invalid_name")</code>	Tidak terdefinisi	Tidak terdefinisi

Contoh berikut Aturan SQL mereferensikan salah satu header MQTT5 berikut: `contentType`, dan `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_secret` (secretID, secretType, kunci, roLearn)

Mengambil nilai terenkripsi `SecretString` atau `SecretBinary` bidang versi rahasia saat ini di [AWS Secrets Manager](#) Untuk informasi lebih lanjut tentang membuat dan memelihara rahasia, lihat [CreateSecret](#), [UpdateSecret](#), dan [PutSecretValue](#).

`get_secret()` mengambil parameter berikut:

`secretId`

String: Nama Sumber Daya Amazon (ARN) atau nama ramah rahasia untuk diambil.

`SecretType`

String: Tipe rahasia. Nilai yang valid: `SecretString` | `SecretBinary`.

`SecretString`

- Untuk rahasia yang Anda buat sebagai objek JSON dengan menggunakan API, the AWS CLI, atau AWS Secrets Manager konsol:

- Jika Anda menentukan nilai untuk key parameter, fungsi ini mengembalikan nilai kunci yang ditentukan.
- Jika Anda tidak menentukan nilai untuk key parameter, fungsi ini mengembalikan seluruh objek JSON.
- Untuk rahasia yang Anda buat sebagai objek non-JSON dengan menggunakan API atau AWS CLI
 - Jika Anda menentukan nilai untuk key parameter, fungsi ini gagal dengan pengecualian.
 - Jika Anda tidak menentukan nilai untuk key parameter, fungsi ini mengembalikan isi rahasia.

SecretBinary

- Jika Anda menentukan nilai untuk key parameter, fungsi ini gagal dengan pengecualian.
- Jika Anda tidak menentukan nilai untuk key parameter, fungsi ini mengembalikan nilai rahasia sebagai string UTF-8 yang dikodekan base64.

kunci

(Opsional) String: Nama kunci di dalam objek JSON yang disimpan di `SecretString` bidang rahasia. Gunakan nilai ini ketika Anda ingin mengambil hanya nilai kunci yang disimpan dalam rahasia, bukan seluruh objek JSON.

Jika Anda menentukan nilai untuk parameter ini dan rahasia tidak berisi objek JSON di dalam `SecretString` bidangnya, fungsi ini gagal dengan pengecualian.

roleArn

String: Peran ARN dengan `secretsmanager:GetSecretValue` dan `secretsmanager:DescribeSecret` izin.

Note

Fungsi ini selalu mengembalikan versi rahasia saat ini (versi dengan `AWSCURRENT` tag). Mesin AWS IoT aturan menyimpan setiap rahasia hingga 15 menit. Akibatnya, mesin aturan dapat memakan waktu hingga 15 menit untuk memperbarui rahasia. Ini berarti bahwa jika Anda mengambil rahasia hingga 15 menit setelah pembaruan dengan AWS Secrets Manager, fungsi ini mungkin mengembalikan versi sebelumnya. Fungsi ini tidak diukur, tetapi AWS Secrets Manager dikenakan biaya. Karena mekanisme caching rahasia, mesin aturan kadang-kadang memanggil AWS Secrets Manager. Karena

mesin aturan adalah layanan terdistribusi penuh, Anda mungkin melihat beberapa panggilan API Secrets Manager dari mesin aturan selama jendela caching 15 menit.

Contoh:

Anda dapat menggunakan `get_secret` fungsi dalam header otentikasi dalam tindakan aturan HTTPS, seperti pada contoh otentikasi kunci API berikut.

```
"API_KEY": "${get_secret('API_KEY', 'SecretString', 'API_KEY_VALUE',  
'arn:aws:iam::12345678910:role/getsecret')}}"
```

Untuk informasi selengkapnya tentang tindakan aturan HTTPS, lihat [the section called “HTTP”](#).

`get_thing_shadow (ThingName, ShadowName, roleArn)`

Mengembalikan bayangan yang ditentukan dari hal yang ditentukan. Didukung oleh SQL versi 2016-03-23 dan yang lebih baru.

`thingName`

String: Nama benda yang bayangannya ingin Anda ambil.

`shadowName`

(Opsional) String: Nama bayangan. Parameter ini diperlukan hanya saat mereferensikan bayangan bernama.

`roleArn`

String: Peran ARN dengan `iot:GetThingShadow` izin.

Contoh:

Saat digunakan dengan bayangan bernama, berikan `shadowName` parameternya.

```
SELECT * from 'topic/subtopic'  
WHERE
```

```
get_thing_shadow("MyThing", "MyThingShadow", "arn:aws:iam::123456789012:role/
AllowsThingShadowAccess")
.state.reported.alarm = 'ON'
```

Saat digunakan dengan bayangan yang tidak disebutkan namanya, hilangkan parameternya.
shadowName

```
SELECT * from 'topic/subtopic'
WHERE
  get_thing_shadow("MyThing", "arn:aws:iam::123456789012:role/
AllowsThingShadowAccess")
.state.reported.alarm = 'ON'
```

get_user_properties () userPropertyKey

Referensi Properti Pengguna, yang merupakan salah satu jenis header properti yang didukung di MQTT5.

UserProperty

String: Properti pengguna adalah pasangan kunci-nilai. Fungsi ini mengambil kunci sebagai argumen dan mengembalikan array dari semua nilai yang cocok dengan kunci terkait.

Argumen fungsi

Untuk Properti Pengguna berikut di header pesan:

Kunci	Nilai
beberapa kunci	beberapa nilai
kunci yang berbeda	nilai yang berbeda
beberapa kunci	nilai dengan kunci duplikat

Tabel berikut menunjukkan perilaku SQL yang diharapkan:

SQL	Tipe data yang dikembalikan	Nilai data yang dikembalikan
<code>get_user_properties ('beberapa kunci')</code>	Array String	<code>['some value', 'value with duplicate key']</code>
<code>get_user_properties ('kunci lainnya')</code>	Array String	<code>['a different value']</code>
<code>get_user_properties ()</code>	Array pasangan kunci-nilai Objek	<code>[{"some key": "some value"}, {"other key": "a different value"}, {"some key": "value with duplicate key"}]</code>
<code>get_user_properties ('kunci tidak ada')</code>	Tidak terdefinisi	

Contoh berikut Aturan SQL referensi Properti Pengguna (jenis header properti MQTT5) ke payload:

```
SELECT *, get_user_properties('user defined property key') as userProperty
FROM 'some/topic'
```

Fungsi hashing

AWS IoT menyediakan fungsi hashing berikut:

- md2
- md5
- sha1
- sha224
- sha256
- sha384
- sha512

Semua fungsi hash mengharapkan satu argumen string. Hasilnya adalah nilai hash dari string itu. Konversi string standar berlaku untuk argumen non-string. Semua fungsi hash didukung oleh SQL versi 2015-10-08 dan yang lebih baru.

Contoh:

```
md2("hello")= "a9046c73e00331af68917d3804f70655"
```

```
md5("hello")= "5d41402abc4b2a76b9719d911017c592"
```

indexof (String, String)

Mengembalikan indeks pertama (0-based) dari argumen kedua sebagai substring dalam argumen pertama. Kedua argumen diharapkan sebagai string. Argumen yang bukan string dikenakan aturan konversi string standar. Fungsi ini tidak berlaku untuk array, hanya untuk string. Didukung oleh SQL versi 2016-03-23 dan yang lebih baru.

Contoh:

```
indexof("abcd", "bc") = 1
```

isNull ()

Mengembalikan nilai true jika argumen adalah Null nilai. Didukung oleh SQL versi 2015-10-08 dan yang lebih baru.

Contoh:

```
isNull(5) = salah.
```

```
isNull(Null) = benar.
```

Jenis Argumen	Hasil
Int	false
Decimal	false
Boolean	false

Jenis Argumen	Hasil
String	false
Array	false
Object	SALAH
Null	BETUL
Undefined	SALAH

isUndefined ()

Mengembalikan nilai true jika argumennya `Undefined`. Didukung oleh SQL versi 2016-03-23 dan yang lebih baru.

Contoh:

`isUndefined(5) = salah.`

`isUndefined(floor([1,2,3])) = benar.`

Jenis Argumen	Hasil
Int	false
Decimal	false
Boolean	false
String	false
Array	false
Object	false
Null	SALAH
Undefined	true

panjang (String)

Mengembalikan jumlah karakter dalam string yang disediakan. Aturan konversi standar berlaku untuk `String` non-argumen. Didukung oleh SQL versi 2016-03-23 dan yang lebih baru.

Contoh:

```
length("hi")= 2
```

```
length(false)= 5
```

ln (Desimal)

Mengembalikan logaritma natural dari argumen. `Decimal` argumen dibulatkan ke presisi ganda sebelum aplikasi fungsi. Didukung oleh SQL versi 2015-10-08 dan yang lebih baru.

Contoh: $\ln(e) = 1$.

Jenis Argumen	Hasil
Int	Decimal(dengan presisi ganda), log alami argumen.
Decimal	Decimal(dengan presisi ganda), log alami argumen.
Boolean	Undefined .
String	Decimal(dengan presisi ganda), log alami argumen. Jika string tidak dapat dikonversi ke <code>aDecimal</code> , hasilnya adalah <code>Undefined</code> .
Susunan	Undefined .
Objek	Undefined .
Null	Undefined .
Tidak terdefinisi	Undefined .

log (Desimal)

Mengembalikan basis 10 logaritma argumen. `Decimal` argumen dibulatkan ke presisi ganda sebelum aplikasi fungsi. Didukung oleh SQL versi 2015-10-08 dan yang lebih baru.

Contoh: `log(100) = 2.0`.

Jenis Argumen	Hasil
Int	Decimal(dengan presisi ganda), log dasar 10 argumen.
Decimal	Decimal(dengan presisi ganda), log dasar 10 argumen.
Boolean	Undefined .
String	Decimal(dengan presisi ganda), log dasar 10 argumen. Jika <code>String</code> tidak dapat dikonversi ke <code>aDecimal</code> , hasilnya adalah <code>Undefined</code> .
Susunan	Undefined .
Objek	Undefined .
Null	Undefined .
Tidak terdefinisi	Undefined .

lebih rendah (String)

Mengembalikan versi huruf kecil dari yang diberikan. `String` Argumen non-string dikonversi ke string menggunakan aturan konversi standar. Didukung oleh SQL versi 2015-10-08 dan yang lebih baru.

Contoh:

`lower("HELLO") = "halo"`.

```
lower(["HELLO"])= ["\ halo\ "].
```

lpad (Tali, Int)

Mengembalikan `String` argumen, empuk di sisi kiri dengan jumlah spasi yang ditentukan oleh argumen kedua. `IntArgumen` harus antara 0 dan 1000. Jika nilai yang diberikan berada di luar rentang yang valid ini, argumen diatur ke nilai valid terdekat (0 atau 1000). Didukung oleh SQL versi 2015-10-08 dan yang lebih baru.

Contoh:

```
lpad("hello", 2) = "  hello".
```

```
lpad(1, 3) = "  1"
```

Tipe argumen 1	Argumen tipe 2	Hasil
String	Int	String, yang disediakan <code>String</code> di sisi kiri dengan sejumlah spasi dengan yang disediakan <code>Int</code> .
String	Decimal	<code>DecimalArgumen</code> dibulatkan ke ke terdekat <code>Int</code> dan empuk di sisi kiri dengan jumlah spasi yang d <code>String</code>
String	String	Argumen kedua dikonversi ke a yang dibulatkan ke bawah ke te dan empuk dengan spasi nomor ditentukan di sebelah kiri. <code>String</code> argumen kedua tidak dapat diubah menjadi <code>Int</code> , hasilnya adalah <code>Undefined</code> .
Nilai lainnya	Int/Decimal/String	Nilai pertama dikonversi ke a <code>String</code> menggunakan konversi standar kemudian fungsi <code>LPAD</code> diterapkan itu. <code>String</code> Jika tidak dapat diubah hasilnya adalah <code>Undefined</code> .

Tipe argumen 1	Argumen tipe 2	Hasil
Nilai apa pun	Nilai lainnya	Undefined .

Ltrim (Tali)

Menghapus semua spasi putih terkemuka (tab dan spasi) dari yang disediakan `String`. Didukung oleh SQL versi 2015-10-08 dan yang lebih baru.

Contoh:

```
Ltrim(" h i ") = "hai".
```

Jenis Argumen	Hasil
Int	StringRepresentasi Int dengan semua ruang putih terkemuka dihapus.
Decimal	StringRepresentasi Decimal dengan semua ruang putih terkemuka dihapus.
Boolean	StringRepresentasi Boolean ("true" atau "false") dengan semua spasi putih terkemuka dihapus.
String	Argumen dengan semua ruang putih terkemuka dihapus.
Array	StringRepresentasi dari Array (menggunakan aturan konversi standar) dengan semua spasi putih terkemuka dihapus.
Objek	StringRepresentasi Object (menggunakan aturan konversi standar) dengan semua spasi putih terkemuka dihapus.
Null	Undefined .

Jenis Argumen	Hasil
Tidak terdefinisi	Undefined .

machinelearning_predict (modelID, roleArn, rekam)

Gunakan `machinelearning_predict` fungsi untuk membuat prediksi menggunakan data dari pesan MQTT berdasarkan model Amazon SageMaker. SageMaker didukung oleh SQL versi 2015-10-08 dan yang lebih baru. Argumen untuk `machinelearning_predict` fungsi tersebut adalah:

modelId

ID model yang digunakan untuk menjalankan prediksi. Titik akhir real-time model harus diaktifkan.

roleArn

Peran IAM yang memiliki kebijakan dengan `machinelearning:Predict` dan `machinelearning:GetMLModel` izin dan memungkinkan akses ke model tempat prediksi dijalankan.

catatan

Data yang akan diteruskan ke API SageMaker Predict. Ini harus direpresentasikan sebagai objek JSON lapisan tunggal. Jika catatan adalah objek JSON multi-level, catatan diratakan dengan membuat serial nilainya. Misalnya, JSON berikut:

```
{ "key1": {"innerKey1": "value1"}, "key2": 0 }
```

akan menjadi:

```
{ "key1": "{ \"innerKey1\": \"value1\" }", "key2": 0 }
```

Fungsi mengembalikan objek JSON dengan bidang-bidang berikut:

predictedLabel

Klasifikasi input berdasarkan model.

detail

Berisi atribut berikut:

PredictiveModelType

Jenis model. Nilai yang valid adalah REGRESSION, BINARY, MULTICLASS.

Algoritme

Algoritma yang digunakan SageMaker untuk membuat prediksi. Nilainya harus SGD.

predictedScores

Berisi skor klasifikasi mentah yang sesuai dengan setiap label.

predictedValue

Nilai yang diprediksi oleh SageMaker.

mod (Desimal, Desimal)

Mengembalikan sisa pembagian argumen pertama dengan argumen kedua. Setara dengan [sisanya \(Desimal, Desimal\)](#). Anda juga dapat menggunakan “%” sebagai operator infix untuk fungsionalitas modulo yang sama. Didukung oleh SQL versi 2015-10-08 dan yang lebih baru.

Contoh: `mod(8, 3) = 2`.

Operan kiri	Operan kanan	Output
Int	Int	Int, argumen pertama modulo kedua.
Int/Decimal	Int/Decimal	Decimal, argumen pertama modulo operan kedua.
String/Int/Decimal	String/Int/Decimal	Jika semua string dikonversi ke decimal, hasilnya adalah argumen pertama modulo argumen kedua. Atau, Undefined.
Nilai lainnya	Nilai lainnya	Undefined.

nanvl (,) AnyValue AnyValue

Mengembalikan argumen pertama jika itu adalah `validDecimal`. Jika tidak, argumen kedua dikembalikan. Didukung oleh SQL versi 2015-10-08 dan yang lebih baru.

Contoh: `Nanvl(8, 3) = 8.`

Tipe argumen 1	Argumen tipe 2	Output
Tidak terdefinisi	Nilai apa pun	Argumen kedua.
Null	Nilai apa pun	Argumen kedua.
<code>Decimal(NaN)</code>	Nilai apa pun	Argumen kedua.
<code>Decimal(bukan NaN)</code>	Nilai apa pun	Argumen pertama.
Nilai lainnya	Nilai apa pun	Argumen pertama.

`newuuid ()`

Mengembalikan UUID 16-byte acak. Didukung oleh SQL versi 2015-10-08 dan yang lebih baru.

Contoh: `newuuid() = 123a4567-b89c-12d3-e456-789012345000`

`numbytes (String)`

Mengembalikan jumlah byte dalam pengkodean UTF-8 dari string yang disediakan. Aturan konversi standar berlaku untuk `String` non-argumen. Didukung oleh SQL versi 2016-03-23 dan yang lebih baru.

Contoh:

`numbytes("hi")= 2`

`numbytes("€") = 3`

`parse_time (String, Panjang [, String])`

Gunakan `parse_time` fungsi untuk memformat stempel waktu menjadi format tanggal/waktu yang dapat dibaca manusia. Didukung oleh SQL versi 2016-03-23 dan yang lebih baru. Untuk mengubah string stempel waktu menjadi milidetik, lihat. [time_to_epoch \(String, String\)](#)

`parse_time` Fungsi mengharapkan argumen berikut:

pola

(String) Pola tanggal/waktu yang mengikuti format [Joda-Time](#).

timestamp

(Panjang) Waktu untuk diformat dalam milidetik sejak zaman Unix. Lihat fungsi [stempel waktu \(\)](#).

timezone

(String) Zona waktu tanggal/waktu yang diformat. Defaultnya adalah "UTC". Fungsi ini mendukung zona waktu [Joda-Time](#). Argumen ini opsional.

Contoh:

Ketika pesan ini dipublikasikan ke topik 'A/B', payload {"ts": "1970.01.01 AD at 21:46:40 CST"} dikirim ke bucket 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:role/ROLE_NAME",
          "bucketName": "BUCKET_NAME",
          "key": "KEY_NAME"
        }
      }
    ],
    "ruleName": "RULE_NAME"
  }
}
```

Saat pesan ini dipublikasikan ke topik 'A/B', muatan yang mirip dengan {"ts": "2017.06.09 AD at 17:19:46 UTC"} (tetapi dengan tanggal/waktu saat ini) dikirim ke bucket 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:role/ROLE_NAME",
        "bucketName": "BUCKET_NAME",
        "key": "KEY_NAME"
      }
    }
  ],
  "ruleName": "RULE_NAME"
}
}

```

`parse_time()` juga dapat digunakan sebagai template substitusi. Misalnya, ketika pesan ini dipublikasikan ke topik 'A/B', muatan dikirim ke bucket S3 dengan kunci = "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:role/ROLE_NAME",
        "bucketName": "BUCKET_NAME",
        "key": "${parse_time('yyyy', timestamp(), 'UTC')}}"
      }
    }],
    "ruleName": "RULE_NAME"
  }
}

```

kekuatan (Desimal, Desimal)

Mengembalikan argumen pertama yang diangkat ke argumen kedua. DecimaIargumen dibulatkan ke presisi ganda sebelum aplikasi fungsi. Didukung oleh SQL versi 2015-10-08 dan yang lebih baru. Didukung oleh SQL versi 2015-10-08 dan yang lebih baru.

Contoh: `power(2, 5) = 32.0`.

Tipe argumen 1	Argumen tipe 2	Output
Int/Decimal	Int/Decimal	A Decimal (dengan presisi ganda argumen pertama diangkat ke k argumen kedua.
Int/Decimal/String	Int/Decimal/String	A Decimal (dengan presisi ganda argumen pertama diangkat ke k argumen kedua. Setiap string di i ke desimal. Jika ada yang Str gagal dikonversiDecimal, hasil adalahUndefined .
Nilai lainnya	Nilai lainnya	Undefined .

prinsipal ()

Mengembalikan prinsipal yang digunakan perangkat untuk otentikasi, berdasarkan bagaimana pesan pemicu dipublikasikan. Tabel berikut menjelaskan prinsip yang dikembalikan untuk setiap metode penerbitan dan protokol.

Bagaimana pesan dipublikasikan	Protokol	Jenis kredensi
Klien MQTT	MQTT	Sertifikat perangkat X.509
AWS IoT konsol klien MQTT	MQTT	Pengguna atau peran IAM
AWS CLI	HTTP	Pengguna atau peran IAM

Bagaimana pesan dipublikasikan	Protokol	Jenis kredensi
AWS IoT Perangkat SDK	MQTT	Sertifikat perangkat X.509
AWS IoT Perangkat SDK	MQTT lebih WebSocket	Pengguna atau peran IAM

Contoh berikut menunjukkan berbagai jenis nilai yang `principal()` dapat dikembalikan:

- Sidik jari sertifikat X.509:
ba67293af50bf2506f5f93469686da660c7c844e7b3950bfb16813e0d31e9373
- ID peran IAM dan nama sesi: ABCD1EFG3HIJK2LMNOP5:my-session-name
- Mengembalikan ID pengguna: ABCD1EFG3HIJK2LMNOP5

rand ()

Mengembalikan pseudorandom, seragam didistribusikan ganda antara 0,0 dan 1,0. Didukung oleh SQL versi 2015-10-08 dan yang lebih baru.

Contoh:

```
rand() = 0.8231909191640703
```

regexp_matches (String, String)

Mengembalikan nilai true jika string (argumen pertama) berisi kecocokan untuk ekspresi reguler (argumen kedua). Jika Anda menggunakan `|` dalam ekspresi reguler, gunakan `dengan()`.

Contoh:

```
regexp_matches("aaaa", "a{2,}") = benar.
```

```
regexp_matches("aaaa", "b") = salah.
```

```
regexp_matches("aaa", "(aaa|bbb)") = benar.
```

```
regexp_matches("bbb", "(aaa|bbb)") = benar.
```

```
regexp_matches("ccc", "(aaa|bbb)") = salah.
```

Argumen pertama:

Jenis Argumen	Hasil
Int	StringRepresentasi dariInt.
Decimal	StringRepresentasi dariDecimal.
Boolean	StringRepresentasi dari Boolean (“true” atau “false”).
String	TheString.
Array	StringRepresentasi dari Array (menggunakan aturan konversi standar).
Objek	StringRepresentasi Object (menggunakan aturan konversi standar).
Null	Undefined .
Tidak terdefinisi	Undefined .

Argumen kedua:

Harus berupa ekspresi regex yang valid. Jenis non-string dikonversi untuk `String` menggunakan aturan konversi standar. Bergantung pada jenisnya, string yang dihasilkan mungkin bukan ekspresi reguler yang valid. Jika argumen (dikonversi) tidak valid regex, hasilnya adalah `Undefined`

`regexp_replace (String, String, String)`

Menggantikan semua kemunculan argumen kedua (ekspresi reguler) dalam argumen pertama dengan argumen ketiga. Grup tangkapan referensi dengan “\$”. Didukung oleh SQL versi 2015-10-08 dan yang lebih baru.

Contoh:

```
regexp_replace("abcd", "bc", "x")= "kapak".
```

```
regexp_replace("abcd", "b(.*)d", "$1")= "ac".
```

Argumen pertama:

Jenis Argumen	Hasil
Int	StringRepresentasi dariInt.
Decimal	StringRepresentasi dariDecimal.
Boolean	StringRepresentasi dari Boolean (“true” atau “false”).
String	Nilai sumbernya.
Array	StringRepresentasi dari Array (menggunakan aturan konversi standar).
Objek	StringRepresentasi Object (menggunakan aturan konversi standar).
Null	Undefined .
Tidak terdefinisi	Undefined .

Argumen kedua:

Harus berupa ekspresi regex yang valid. Jenis non-string dikonversi untuk `String` menggunakan aturan konversi standar. Bergantung pada jenisnya, string yang dihasilkan mungkin bukan ekspresi reguler yang valid. Jika argumen (dikonversi) bukan ekspresi regex yang valid, hasilnya adalah `Undefined`

Argumen ketiga:

Harus berupa string pengganti regex yang valid. (Dapat mereferensikan grup tangkapan.) Jenis non-string dikonversi untuk `String` menggunakan aturan konversi standar. Jika argumen (dikonversi) bukan string pengganti regex yang valid, hasilnya adalah `Undefined`

`regexp_substr (Tali, Tali)`

Menemukan kecocokan pertama dari parameter kedua (regex) di parameter pertama. Grup tangkapan referensi dengan “\$”. Didukung oleh SQL versi 2015-10-08 dan yang lebih baru.

Contoh:

```
regexp_substr("hihihello", "hi")= "hai"
```

```
regexp_substr("hihihello", "(hi)*")= "hihi"
```

Argumen pertama:

Jenis Argumen	Hasil
Int	StringRepresentasi dariInt.
Decimal	StringRepresentasi dariDecimal.
Boolean	StringRepresentasi Boolean ("true" atau "false").
String	StringArgumennya.
Array	StringRepresentasi dari Array (menggunakan aturan konversi standar).
Objek	StringRepresentasi Object (menggunakan aturan konversi standar).
Null	Undefined .
Tidak terdefinisi	Undefined .

Argumen kedua:

Harus berupa ekspresi regex yang valid. Jenis non-string dikonversi untuk `String` menggunakan aturan konversi standar. Bergantung pada jenisnya, string yang dihasilkan mungkin bukan ekspresi reguler yang valid. Jika argumen (dikonversi) bukan ekspresi regex yang valid, hasilnya adalah `Undefined`

sisanya (`Desimal, Desimal`)

Mengembalikan sisa pembagian argumen pertama dengan argumen kedua. Setara dengan [mod \(Desimal, Desimal\)](#). Anda juga dapat menggunakan "%" sebagai operator infix untuk fungsionalitas modulo yang sama. Didukung oleh SQL versi 2015-10-08 dan yang lebih baru.

Contoh: `remainder(8, 3) = 2`.

Operan kiri	Operan kanan	Output
Int	Int	Int, argumen pertama modulo kedua.
Int/Decimal	Int/Decimal	Decimal, argumen pertama modulo operan kedua.
String/Int/Decimal	String/Int/Decimal	Jika semua string dikonversi ke Int/Decimal, hasilnya adalah argumen pertama modulo argumen kedua. Atau, Undefined.
Nilai lainnya	Nilai lainnya	Undefined.

ganti (String, String, String)

Menggantikan semua kemunculan argumen kedua dalam argumen pertama dengan argumen ketiga. Didukung oleh SQL versi 2015-10-08 dan yang lebih baru.

Contoh:

`replace("abcd", "bc", "x") = "axd"`.

`replace("abcdabcd", "b", "x") = "axcdaxcd"`.

Semua argumen

Jenis Argumen	Hasil
Int	StringRepresentasi dariInt.
Decimal	StringRepresentasi dariDecimal.
Boolean	StringRepresentasi Boolean ("true" atau "false").
String	Nilai sumbernya.

Jenis Argumen	Hasil
Array	StringRepresentasi dari Array (menggunakan aturan konversi standar).
Objek	StringRepresentasi Object (menggunakan aturan konversi standar).
Null	Undefined .
Tidak terdefinisi	Undefined .

rpad (String, Int)

Mengembalikan argumen string, empuk di sisi kanan dengan jumlah spasi yang ditentukan dalam argumen kedua. IntArgumen harus antara 0 dan 1000. Jika nilai yang diberikan berada di luar rentang yang valid ini, argumen diatur ke nilai valid terdekat (0 atau 1000). Didukung oleh SQL versi 2015-10-08 dan yang lebih baru.

Contoh:

```
rpad("hello", 2) = "hello  ".
```

```
rpad(1, 3) = "1   ".
```

Tipe argumen 1	Argumen tipe 2	Hasil
String	Int	Empuk di sisi kanan dengan sejumlah spasi yang sama dengan yang disediakan Int. String
String	Decimal	DecimalArgumen dibulatkan ke bawah ke yang terdekat Int dan string empuk di sisi kanan dengan

Tipe argumen 1	Argumen tipe 2	Hasil
		sejumlah spasi sama dengan yang disediakanInt.
String	String	Argumen kedua diubah menjadi aDecimal, yang dibulatkan ke bawah ke yang terdekatInt. Empuk di sisi kanan dengan sejumlah spasi yang sama dengan Int nilainya. String
Nilai lainnya	Int/Decimal/String	Nilai pertama dikonversi ke a String menggunakan konversi standar, dan fungsi rpad diterapkan pada itu. String Jika tidak dapat dikonversi, hasilnya adalahUndefined .
Nilai apa pun	Nilai lainnya	Undefined .

bulat (Desimal)

Membulatkan yang diberikan Decimal ke yang terdekatInt. Jika Decimal berjarak sama dari dua Int nilai (misalnya, 0,5), Decimal dibulatkan ke atas. Didukung oleh SQL versi 2015-10-08 dan yang lebih baru.

Contoh: Round(1.2) = 1.

Round(1.5)= 2.

Round(1.7)= 2.

`Round(-1.1) = -1.`

`Round(-1.5) = -2.`

Jenis Argumen	Hasil
Int	Argumennya.
Decimal	Decimal dibulatkan ke bawah ke yang terdekat Int.
String	Decimal dibulatkan ke bawah ke yang terdekat Int. Jika string tidak dapat dikonversi ke aDecimal, hasilnya adalah Undefined .
Nilai lainnya	Undefined .

rtrim (Tali)

Menghapus semua spasi putih (tab dan spasi) dari yang disediakan. String Didukung oleh SQL versi 2015-10-08 dan yang lebih baru.

Contoh:

```
rtrim(" h i ") = "hi"
```

Jenis Argumen	Hasil
Int	StringRepresentasi dari Int.
Decimal	StringRepresentasi dari Decimal.
Boolean	StringRepresentasi Boolean ("true" atau "false").
Array	StringRepresentasi dari Array (menggunakan aturan konversi standar).

Jenis Argumen	Hasil
Objek	StringRepresentasi Object (menggunakan aturan konversi standar).
Null	Undefined .
Tidak terdefinisi	Undefined

tanda (Desimal)

Mengembalikan tanda nomor yang diberikan. Ketika tanda argumen positif, 1 dikembalikan. Ketika tanda argumen negatif, -1 dikembalikan. Jika argumen adalah 0, 0 dikembalikan. Didukung oleh SQL versi 2015-10-08 dan yang lebih baru.

Contoh:

`sign(-7) = -1.`

`sign(0) = 0.`

`sign(13) = 1.`

Jenis Argumen	Hasil
Int	Int, tanda Int nilai.
Decimal	Int, tanda Decimal nilai.
String	Int, tanda Decimal nilai. String dikonversi ke Decimal nilai, dan tanda Decimal nilai dikembalikan. Jika String tidak dapat dikonversi ke aDecimal, hasilnya adalah Undefined . Didukung oleh SQL versi 2015-10-08 dan yang lebih baru.
Nilai lainnya	Undefined .

sin (Desimal)

Mengembalikan sinus dari angka dalam radian. `Decimal` argumen dibulatkan ke presisi ganda sebelum aplikasi fungsi. Didukung oleh SQL versi 2015-10-08 dan yang lebih baru.

Contoh: $\sin(0) = 0.0$

Jenis Argumen	Hasil
Int	<code>Decimal</code> (dengan presisi ganda), sinus argumen.
Decimal	<code>Decimal</code> (dengan presisi ganda), sinus argumen.
Boolean	Undefined .
String	<code>Decimal</code> (dengan presisi ganda), sinus argumen. Jika string tidak dapat dikonversi ke <code>aDecimal</code> , hasilnya adalah <code>Undefined</code> .
Susunan	Undefined .
Objek	Undefined .
Null	Undefined .
Undefined	Undefined .

sinh (Desimal)

Mengembalikan sinus hiperbolik dari suatu angka. `Decimal` nilai dibulatkan ke presisi ganda sebelum aplikasi fungsi. Hasilnya adalah `Decimal` nilai presisi ganda. Didukung oleh SQL versi 2015-10-08 dan yang lebih baru.

Contoh: $\sinh(2.3) = 4.936961805545957$

Jenis Argumen	Hasil
Int	Decimal(dengan presisi ganda), sinus hiperbolik argumen.
Decimal	Decimal(dengan presisi ganda), sinus hiperbolik argumen.
Boolean	Undefined .
String	Decimal(dengan presisi ganda), sinus hiperbolik argumen. Jika string tidak dapat dikonversi ke aDecimal, hasilnya adalahUndefined .
Susunan	Undefined .
Objek	Undefined .
Null	Undefined .
Tidak terdefinisi	Undefined .

sourceip ()

Mengambil alamat IP perangkat atau router yang terhubung dengannya. Jika perangkat Anda terhubung ke internet secara langsung, fungsi tersebut akan mengembalikan alamat IP sumber perangkat. Jika perangkat Anda terhubung ke router yang terhubung ke internet, fungsi tersebut akan mengembalikan alamat IP sumber router. Didukung oleh SQL versi 2016-03-23. `sourceip()` tidak mengambil parameter apa pun.

Important

Alamat IP sumber publik perangkat sering kali merupakan alamat IP dari Network Address Translation (NAT) Gateway terakhir seperti router atau modem kabel penyedia layanan internet Anda.

Contoh:

```
sourceip()="192.158.1.38"
```

```
sourceip()="1.102.103.104"
```

```
sourceip()="2001:db8:ff00::12ab:34cd"
```

Contoh SQL:

```
SELECT *, sourceip() as deviceIp FROM 'some/topic'
```

Contoh cara menggunakan fungsi `sourceip()` dalam AWS IoT Core tindakan aturan:

Contoh 1

Contoh berikut menunjukkan bagaimana memanggil fungsi `()` sebagai [template substitusi](#) dalam tindakan [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"
        }
      }
    ]
  }
}
```

Contoh 2

Contoh berikut menunjukkan cara menambahkan `sourceip()` fungsi sebagai properti pengguna [MQTT menggunakan template substitusi](#).

```
{
  "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()}"
              }
            ]
          }
        }
      }
    ]
  }
}
```

Anda dapat mengambil alamat IP sumber dari pesan yang diteruskan ke AWS IoT Core aturan dari jalur Message Broker dan [Basic Ingest](#). Anda juga dapat mengambil IP sumber untuk pesan IPv4 dan IPv6. IP sumber akan ditampilkan seperti berikut:

IPv6: yyyy:yyyy:yyyy::yyyy:yyyy

IPv4: xxx.xxx.xxx.xxx

Note

IP sumber asli tidak akan diteruskan melalui [tindakan Republish](#).

substring (String, Int [, Int])

Mengharapkan `String` diikuti oleh satu atau dua `Int` nilai. Untuk `Int` argumen `String` dan satu, fungsi ini mengembalikan substring yang disediakan `String` dari `Int` indeks yang disediakan (berbasis 0, inklusif) ke akhir. `String` Untuk `Int` argumen `a String` dan dua, fungsi ini mengembalikan substring yang disediakan `String` dari argumen `Int` indeks pertama (berbasis 0, inklusif) ke argumen `Int` indeks kedua (berbasis 0, eksklusif). Indeks yang kurang dari nol ditetapkan ke nol. Indeks yang lebih besar dari `String` panjang diatur ke `String` panjang. Untuk versi tiga argumen, jika indeks pertama lebih besar dari (atau sama dengan) indeks kedua, hasilnya `kosongString`.

Jika argumen yang diberikan bukan (*String, Int*), atau (*String, Int, Int*), konversi standar diterapkan ke argumen untuk mencoba mengubahnya menjadi tipe yang benar. Jika tipe tidak dapat dikonversi, hasil fungsinya adalah `Undefined`. Didukung oleh SQL versi 2015-10-08 dan yang lebih baru.

Contoh:

```
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)= "rue".
```

```
substring(false, -2.411E247)= "salah".
```

```
substring("012345", 1, 3)= "12".
```

```
substring("012345", -50, 50)= "012345".
```

```
substring("012345", 3, 1) = "".
```

sql_version ()

Mengembalikan versi SQL yang ditentukan dalam aturan ini. Didukung oleh SQL versi 2015-10-08 dan yang lebih baru.

Contoh:

```
sql_version()= "2016-03-23"
```

sqrt (Desimal)

Mengembalikan akar kuadrat dari sebuah angka. DecimaIargumen dibulatkan ke presisi ganda sebelum aplikasi fungsi. Didukung oleh SQL versi 2015-10-08 dan yang lebih baru.

Contoh: `sqrt(9) = 3.0`.

Jenis Argumen	Hasil
Int	Akar kuadrat dari argumen.
Decimal	Akar kuadrat dari argumen.
Boolean	Undefined .
String	Akar kuadrat dari argumen. Jika string tidak dapat dikonversi ke aDecimal, hasilnya adalahUndefined .
Susunan	Undefined .
Objek	Undefined .
Null	Undefined .
Tidak terdefinisi	Undefined .

startswith (String, String)

PengembalianBoolean, apakah argumen string pertama dimulai dengan argumen string kedua. Jika salah satu argumen adalah Null atauUndefined, hasilnya adalahUndefined. Didukung oleh SQL versi 2015-10-08 dan yang lebih baru.

Contoh:

```
startswith("ranger", "ran")= benar
```

Tipe argumen 1	Argumen tipe 2	Hasil
String	String	Apakah string pertama dimulai dengan string kedua.
Nilai lainnya	Nilai lainnya	Kedua argumen dikonversi ke string menggunakan aturan konversi string. Mengembalikan nilai true jika string pertama dimulai dengan string kedua. Jika salah satu argumen adalah Null atau Undefined, hasilnya adalah Undefined.

tan (Desimal)

Mengembalikan garis singgung angka dalam radian. Nilai dibulatkan ke presisi ganda sebelum aplikasi fungsi. Didukung oleh SQL versi 2015-10-08 dan yang lebih baru.

Contoh: $\tan(3) = -0.1425465430742778$

Jenis Argumen	Hasil
Int	Decimal(dengan presisi ganda), garis singgung argumen.
Decimal	Decimal(dengan presisi ganda), garis singgung argumen.
Boolean	Undefined.
String	Decimal(dengan presisi ganda), garis singgung argumen. Jika string tidak dapat dikonversi ke aDecimal, hasilnya adalah Undefined.
Susunan	Undefined.
Objek	Undefined.

Jenis Argumen	Hasil
Null	Undefined .
Tidak terdefinisi	Undefined .

tanh (Desimal)

Mengembalikan tangen hiperbolik dari angka dalam radian. DecimaNilai dibulatkan ke presisi ganda sebelum aplikasi fungsi. Didukung oleh SQL versi 2015-10-08 dan yang lebih baru.

Contoh: $\tanh(2.3) = 0,9800963962661914$

Jenis Argumen	Hasil
Int	Decimal(dengan presisi ganda), garis singgung hiperbolik argumen.
Decimal	Decimal(dengan presisi ganda), garis singgung hiperbolik argumen.
Boolean	Undefined .
String	Decimal(dengan presisi ganda), garis singgung hiperbolik argumen. Jika string tidak dapat dikonversi ke aDecimal, hasilnya adalahUndefined .
Susunan	Undefined .
Objek	Undefined .
Null	Undefined .
Tidak terdefinisi	Undefined .

time_to_epoch (String, String)

Gunakan `time_to_epoch` fungsi untuk mengubah string timestamp menjadi beberapa milidetik dalam waktu epoch Unix. Didukung oleh SQL versi 2016-03-23 dan yang lebih baru. Untuk mengonversi milidetik ke string stempel waktu yang diformat, lihat. [parse_time \(String, Panjang \[, String\]\)](#)

`time_to_epoch` Fungsi mengharapkan argumen berikut:

timestamp

(String) String timestamp yang akan dikonversi menjadi milidetik sejak zaman Unix. Jika string stempel waktu tidak menentukan zona waktu, fungsi menggunakan zona waktu UTC.

pola

(String) Pola tanggal/waktu yang mengikuti Format Waktu [JDK11](#).

Contoh:

```
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
```

stempel waktu ()

Mengembalikan stempel waktu saat ini dalam milidetik dari 00:00:00 Coordinated Universal Time (UTC), Kamis, 1 Januari 1970, seperti yang diamati oleh mesin aturan. AWS IoT Didukung oleh SQL versi 2015-10-08 dan yang lebih baru.

Contoh: `timestamp() = 1481825251155`

topik (Desimal)

Mengembalikan topik ke mana pesan yang memicu aturan dikirim. Jika tidak ada parameter yang ditentukan, seluruh topik dikembalikan. `DecimalParameter` ini digunakan untuk menentukan segmen topik tertentu, dengan 1 menunjuk segmen pertama. Untuk `topikfoo/bar/baz`, topik (1)

kembalifoo, topik (2) kembalibar, dan sebagainya. Didukung oleh SQL versi 2015-10-08 dan yang lebih baru.

Contoh:

```
topic()= "Hal-hal/Mitos/Hal-hal"
```

```
topic(1)= "hal-hal"
```

Ketika [Basic Ingest](#) digunakan, awalan awal topic (`$aws/rules/rule-name`) tidak tersedia untuk fungsi topic (). Misalnya, mengingat topik:

```
$aws/rules/BuildingManager/Buildings/Building5/Floor2/Room201/Lights
```

```
topic()= "Bangunan/Bangunan/Lantai/Kamar201/Lampu"
```

```
topic(3)= "Lantai2"
```

traceid ()

Mengembalikan ID jejak (UUID) pesan MQTT, atau Undefined jika pesan tidak dikirim melalui MQTT. Didukung oleh SQL versi 2015-10-08 dan yang lebih baru.

Contoh:

```
traceid() = "12345678-1234-1234-1234-123456789012"
```

mengubah (String, Objek, Array)

Mengembalikan array objek yang berisi hasil transformasi tertentu dari Object parameter pada Array parameter.

Didukung oleh SQL versi 2016-03-23 dan yang lebih baru.

String

Mode transformasi yang akan digunakan. Lihat tabel berikut untuk mode transformasi yang didukung dan bagaimana mereka membuat Result dari Object dan Array parameter.

Objek

Objek yang berisi atribut untuk diterapkan ke setiap elemenArray.

Array

Sebuah array objek ke mana atribut Object diterapkan.

Setiap objek dalam Array ini sesuai dengan objek dalam respon fungsi. Setiap objek dalam respons fungsi berisi atribut yang ada di objek asli dan atribut yang disediakan oleh `Object` sebagaimana ditentukan oleh mode transformasi yang ditentukan dalam `String`.

String parameter	Object parameter	Array parameter	Hasil
<code>enrichArray</code>	Objek	Array objek	Array objek di mana setiap objek berisi atribut elemen dari <code>Array</code> parameter dan atribut <code>Object</code> parameter.
Nilai lainnya	Nilai apa pun	Nilai apa pun	Tidak terdefinisi

Note

Array yang dikembalikan oleh fungsi ini terbatas pada 128 KiB.

Contoh fungsi transformasi 1

Contoh ini menunjukkan bagaimana `transform()` fungsi menghasilkan array tunggal objek dari objek data dan array.

Dalam contoh ini, pesan berikut dipublikasikan ke topik MQTT. A/B

```
{
  "attributes": {
    "data1": 1,
    "data2": 2
  },
  "values": [
    {
      "a": 3
    },
    {
      "b": 4
    }
  ]
}
```

```
    },
    {
      "c": 5
    }
  ]
}
```

Pernyataan SQL ini untuk tindakan aturan topik menggunakan transform() fungsi dengan String nilai. enrichArray Dalam contoh ini, Object adalah attributes properti dari payload pesan dan Array values array, yang berisi tiga objek.

```
select value transform("enrichArray", attributes, values) from 'A/B'
```

Setelah menerima payload pesan, pernyataan SQL mengevaluasi respons berikut.

```
[
  {
    "a": 3,
    "data1": 1,
    "data2": 2
  },
  {
    "b": 4,
    "data1": 1,
    "data2": 2
  },
  {
    "c": 5,
    "data1": 1,
    "data2": 2
  }
]
```

Contoh fungsi transformasi 2

Contoh ini menunjukkan bagaimana transform() fungsi dapat menggunakan nilai literal untuk menyertakan dan mengganti nama atribut individual dari payload pesan.

Dalam contoh ini, pesan berikut dipublikasikan ke topik MQTT. A/B Ini adalah pesan yang sama yang digunakan dalam [the section called "Contoh fungsi transformasi 1"](#).

```
{
```



```
  "attributes": {
    "data1": 1,
    "data2": 2
  },
  "values": [
    {
      "a": 3
    },
    {
      "b": 4
    },
    {
      "c": 5
    }
  ]
}
```

Pernyataan SQL ini untuk tindakan aturan topik menggunakan transform() fungsi dengan String nilai. enrichArray Object Dalam transform() fungsi memiliki atribut tunggal bernama key dengan nilai attributes.data1 dalam payload pesan dan Array values array, yang berisi tiga objek yang sama yang digunakan dalam contoh sebelumnya.

```
select value transform("enrichArray", {"key": attributes.data1}, values) from 'A/B'
```

Setelah menerima payload pesan, pernyataan SQL ini mengevaluasi respons berikut. Perhatikan bagaimana data1 properti tersebut dinamai key dalam tanggapan.

```
[
  {
    "a": 3,
    "key": 1
  },
  {
    "b": 4,
    "key": 1
  },
  {
    "c": 5,
    "key": 1
  }
]
```

Contoh fungsi transformasi 3

Contoh ini menunjukkan bagaimana transform() fungsi dapat digunakan dalam klausa SELECT bersarang untuk memilih beberapa atribut dan membuat objek baru untuk pemrosesan selanjutnya.

Dalam contoh ini, pesan berikut dipublikasikan ke topik MQTT. A/B

```
{
  "data1": "example",
  "data2": {
    "a": "first attribute",
    "b": "second attribute",
    "c": [
      {
        "x": {
          "someInt": 5,
          "someString": "hello"
        },
        "y": true
      },
      {
        "x": {
          "someInt": 10,
          "someString": "world"
        },
        "y": false
      }
    ]
  }
}
```

Fungsi Object untuk transformasi ini adalah objek yang dikembalikan oleh pernyataan SELECT, yang berisi a dan b elemen data2 objek pesan. ArrayParameter terdiri dari dua objek dari data2.c array dalam pesan asli.

```
select value transform('enrichArray', (select a, b from data2), (select value c from data2)) from 'A/B'
```

Dengan pesan sebelumnya, pernyataan SQL mengevaluasi respons berikut.

```
[
```

```
{
  "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"
}
]
```

Array yang dikembalikan dalam respons ini dapat digunakan dengan tindakan aturan topik yang mendukung `batchMode`.

memangkas (Tali)

Menghapus semua spasi putih terdepan dan tertinggal dari yang disediakan `String`. Didukung oleh SQL versi 2015-10-08 dan yang lebih baru.

Contoh:

```
Trim(" hi ") = "hai"
```

Jenis Argumen	Hasil
Int	StringRepresentasi dari Int dengan semua ruang putih terkemuka dan tertinggal dihapus.
Decimal	StringRepresentasi dari Decimal dengan semua ruang putih terkemuka dan tertinggal dihapus.

Jenis Argumen	Hasil
Boolean	StringRepresentasi dari Boolean (“benar” atau “salah”) dengan semua spasi putih di depan dan belakang dihapus.
String	StringDengan semua ruang putih terkemuka dan tertinggal dihapus.
Array	StringRepresentasi dari Array menggunakan aturan konversi standar.
Objek	StringRepresentasi Object menggunakan aturan konversi standar.
Null	Undefined .
Tidak terdefinisi	Undefined .

batang (Desimal, Int)

Memangkas argumen pertama ke jumlah Decimal tempat yang ditentukan oleh argumen kedua. Jika argumen kedua kurang dari nol, itu diatur ke nol. Jika argumen kedua lebih besar dari 34, itu diatur ke 34. Trailing zeroes dilucuti dari hasilnya. Didukung oleh SQL versi 2015-10-08 dan yang lebih baru.

Contoh:

`trunc(2.3, 0)= 2.`

`trunc(2.3123, 2)= 2,31.`

`trunc(2.888, 2)= 2,88.`

`trunc(2.00, 5)= 2.`

Tipe argumen 1	Argumen tipe 2	Hasil
Int	Int	Nilai sumbernya.

Tipe argumen 1	Argumen tipe 2	Hasil
Int/Decimal	Int/Decimal	Argumen pertama dipotong dengan panjang yang dijelaskan oleh argumen kedua. Argumen kedua, jika bukan Int/Decimal, akan dibulatkan ke bawah ke yang terdekat.
Int/Decimal/String	Int/Decimal	Argumen pertama dipotong dengan panjang yang dijelaskan oleh argumen kedua. Argumen kedua, jika bukan Int/Decimal, akan dibulatkan ke bawah ke yang terdekat. Jika argumen kedua adalah String, akan dikonversi menjadi Double. Jika konversi string gagal, hasilnya adalah Undefined .
Nilai lainnya		Undefined .

atas (String)

Mengembalikan versi huruf besar dari yang diberikan. String StringNon-argumen dikonversi untuk String menggunakan aturan konversi standar. Didukung oleh SQL versi 2015-10-08 dan yang lebih baru.

Contoh:

```
upper("hello")= "HALO"
```

```
upper(["hello"])= [" HALO\ "]
```

Literal

Anda dapat langsung menentukan objek literal dalam klausa SELECT dan WHERE dari aturan SQL Anda, yang dapat berguna untuk meneruskan informasi.

Note

Literal hanya tersedia saat menggunakan versi SQL 2016-03-23 atau yang lebih baru.

Sintaks objek JSON digunakan (pasangan kunci-nilai, dipisahkan koma, di mana kunci adalah string dan nilai adalah nilai JSON, dibungkus dalam kurung kurawal {}). Sebagai contoh:

```
Payload masuk dipublikasikan pada topik: topic/subtopic {"lat_long":
[47.606, -122.332]}
```

```
Pernyataan SQL: SELECT {'latitude': get(lat_long,
0), 'longitude': get(lat_long, 1)} as lat_long FROM 'topic/subtopic'
```

```
Muatan keluar yang dihasilkan adalah: {"lat_long":
{"latitude":47.606, "longitude":-122.332}}
```

Anda juga dapat secara langsung menentukan array dalam klausa SELECT dan WHERE dari aturan SQL Anda, yang memungkinkan Anda untuk mengelompokkan informasi. Sintaks JSON digunakan (bungkus item yang dipisahkan koma dalam tanda kurung siku [] untuk membuat array literal). Sebagai contoh:

```
Payload masuk dipublikasikan pada topik: topic/subtopic {"lat": 47.696, "long":
-122.332}
```

```
Pernyataan SQL: SELECT [lat, long] as lat_long FROM 'topic/subtopic'
```

```
Muatan keluaran yang dihasilkan adalah: {"lat_long": [47.606, -122.332]}.
```

Pernyataan kasus

Pernyataan kasus dapat digunakan untuk eksekusi percabangan, seperti pernyataan switch.

Sintaksis:

```
CASE v WHEN t[1] THEN r[1]
  WHEN t[2] THEN r[2] ...
  WHEN t[n] THEN r[n]
  ELSE r[e] END
```

Eksresi *v* dievaluasi dan dicocokkan untuk kesetaraan terhadap *t[i]* nilai masing-masing klausa. **WHEN** Jika kecocokan ditemukan, *r[i]* ekspresi yang sesuai menjadi hasil CASE pernyataan. **WHEN** Klausa dievaluasi secara berurutan sehingga jika ada lebih dari satu klausa yang cocok, hasil klausa pencocokan pertama menjadi hasil pernyataan. **CASE** Jika tidak ada kecocokan, *r[e]* dari **ELSE** klausa adalah hasilnya. Jika tidak ada pertandingan dan tidak ada **ELSE** klausa, hasilnya adalah `Undefined`.

CASE pernyataan membutuhkan setidaknya satu WHEN klausa. Sebuah ELSE klausa adalah opsional.

Sebagai contoh:

Payload masuk dipublikasikan pada topik: `topic/subtopic`

```
{
  "color": "yellow"
}
```

Pernyataan 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'
```

Muatan keluaran yang dihasilkan adalah:

```
{
  "instructions": "caution"
}
```

Note

Jika `vyaUndefined`, hasil dari pernyataan kasus adalah `Undefined`.

Ekstensi JSON

Anda dapat menggunakan ekstensi berikut untuk sintaks ANSI SQL untuk memfasilitasi pekerjaan dengan objek JSON bersarang.

“.” Operator

Operator ini mengakses anggota dalam objek JSON tertanam dan fungsi identik dengan ANSI SQL dan JavaScript. Sebagai contoh:

```
SELECT foo.bar AS bar.baz FROM 'topic/subtopic'
```

memilih nilai bar properti dalam foo objek dari payload pesan berikut yang dikirim ke topik. topic/subtopic

```
{
  "foo": {
    "bar": "RED",
    "bar1": "GREEN",
    "bar2": "BLUE"
  }
}
```

Jika nama properti JSON menyertakan karakter tanda hubung atau karakter numerik, notasi 'titik' tidak akan berfungsi. Sebagai gantinya, Anda harus menggunakan [fungsi get](#) untuk mengekstrak nilai properti.

Dalam contoh ini, pesan berikut dikirim ke `iot/rules` topik.

```
{
  "mydata": {
    "item2": {
      "0": {
        "my-key": "myValue"
      }
    }
  }
}
```

Biasanya, nilai `my-key` akan diidentifikasi seperti dalam kueri ini.

```
SELECT * from iot/rules WHERE mydata.item2.0.my-key= "myValue"
```

Namun, karena nama properti `my-key` berisi tanda hubung dan `item2` berisi karakter numerik, [fungsi get](#) harus digunakan sebagai query berikut menunjukkan.

```
SELECT * from 'iot/rules' WHERE get(get(get(mydata,"item2"),"0"),"my-key") = "myValue"
```

*Operator

Ini berfungsi dengan cara yang sama seperti * wildcard di ANSI SQL. Ini digunakan dalam klausa SELECT saja dan membuat objek JSON baru yang berisi data pesan. Jika payload pesan tidak dalam format JSON, * mengembalikan seluruh payload pesan sebagai byte mentah. Sebagai contoh:

```
SELECT * FROM 'topic/subtopic'
```

Menerapkan Fungsi ke Nilai Atribut

Berikut ini adalah contoh payload JSON yang mungkin dipublikasikan oleh perangkat:

```
{
  "deviceid" : "iot123",
  "temp" : 54.98,
  "humidity" : 32.43,
  "coords" : {
    "latitude" : 47.615694,
    "longitude" : -122.3359976
  }
}
```

Contoh berikut menerapkan fungsi untuk nilai atribut dalam payload JSON:

```
SELECT temp, md5(deviceid) AS hashed_id FROM topic/#
```

Hasil dari query ini adalah objek JSON berikut:

```
{
  "temp": 54.98,
  "hashed_id": "e37f81fb397e595c4aeb5645b8cbbbd1"
}
```

Templat substitusi

Anda dapat menggunakan template substitusi untuk menambah data JSON yang dikembalikan saat aturan dipicu dan AWS IoT melakukan tindakan. Sintaks untuk template substitusi adalah `${ekspresi}`, di mana ekspresi dapat berupa ekspresi apa pun yang didukung oleh AWS IoT dalam klausa SELECT, klausa WHERE, dan [AWS IoT tindakan aturan](#). Ekspresi ini dapat dicolokkan ke bidang tindakan pada aturan, memungkinkan Anda mengonfigurasi tindakan secara dinamis. Akibatnya, fitur ini menggantikan sepotong informasi dalam suatu tindakan. Ini termasuk fungsi, operator, dan informasi yang ada dalam muatan pesan asli.

⚠ Important

Karena ekspresi dalam template substitusi dievaluasi secara terpisah dari pernyataan “SELECT...”, Anda tidak dapat mereferensikan alias yang dibuat menggunakan klausa AS. Anda hanya dapat mereferensikan informasi yang ada di muatan, [fungsi](#), dan [operator](#) asli.

Untuk informasi selengkapnya tentang ekspresi yang didukung, lihat [AWS IoT Referensi SQL](#).

Tindakan aturan berikut mendukung template substitusi. Setiap tindakan mendukung bidang yang berbeda yang dapat diganti.

- [Apache Kafka](#)
- [CloudWatch alarm](#)
- [CloudWatch Log](#)
- [CloudWatch metrik](#)
- [DynamoDB](#)
- [DynamoDBv2](#)
- [Elasticsearch](#)
- [HTTP](#)
- [IoT Analytics](#)
- [AWS IoT Events](#)
- [AWS IoT SiteWise](#)
- [Kinesis Data Streams](#)
- [Firehose](#)
- [Lambda](#)
- [Lokasi](#)
- [OpenSearch](#)
- [Publikasikan ulang](#)
- [S3](#)
- [SNS](#)
- [SQS](#)

- [Step Functions](#)
- [Timestream](#)

Templat substitusi muncul di parameter tindakan dalam aturan:

```
{
  "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"
    }
  ]
}
```

Jika aturan ini dipicu oleh JSON berikut yang diterbitkan kemy/iot/topic:

```
{
  "deviceid": "iot123",
  "temp": 54.98,
  "humidity": 32.43,
  "coords": {
    "latitude": 47.615694,
    "longitude": -122.3359976
  }
}
```

Kemudian aturan ini menerbitkan JSON berikut kemy/iot/topic/republish, yang AWS IoT menggantikan dari: `${topic()}/republish`

```
{
  "deviceid": "iot123",
  "temp": 54.98,
  "humidity": 32.43,
  "coords": {
    "latitude": 47.615694,
    "longitude": -122.3359976
  },
  "timestamp": 1579637878451
}
```

Kueri objek bersarang

Anda dapat menggunakan klausa SELECT bersarang untuk menanyakan atribut dalam array dan objek JSON bagian dalam. Didukung oleh SQL versi 2016-03-23 dan yang lebih baru.

Pertimbangkan pesan MQTT berikut:

```
{
  "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

Anda dapat mengonversi nilai ke array baru dengan aturan berikut.

```
SELECT (SELECT VALUE n FROM e) as sensors FROM 'my/topic'
```

Aturan menghasilkan output berikut.

```
{
  "sensors": [
    "temperature",
    "light",
    "acidity"
  ]
}
```

Example

Menggunakan pesan MQTT yang sama, Anda juga dapat menanyakan nilai tertentu dalam objek bersarang dengan aturan berikut.

```
SELECT (SELECT v FROM e WHERE n = 'temperature') as temperature FROM 'my/topic'
```

Aturan menghasilkan output berikut.

```
{
```

```
  "temperature": [  
    {  
      "v": 22.5  
    }  
  ]  
}
```

Example

Anda juga dapat meratakan output dengan aturan yang lebih rumit.

```
SELECT get((SELECT v FROM e WHERE n = 'temperature'), 0).v as temperature FROM 'topic'
```

Aturan menghasilkan output berikut.

```
{  
  "temperature": 22.5  
}
```

Bekerja dengan muatan biner

Untuk menangani payload pesan Anda sebagai data biner mentah (bukan objek JSON), Anda dapat menggunakan operator `*` untuk merujuknya dalam klausa `SELECT`.

Dalam topik ini:

- [Contoh muatan biner](#)
- [Decoding muatan pesan protobuf](#)

Contoh muatan biner

Saat Anda menggunakan `*` untuk merujuk ke payload pesan sebagai data biner mentah, Anda dapat menambahkan data ke aturan. Jika Anda memiliki muatan kosong atau JSON, payload yang dihasilkan dapat memiliki data yang ditambahkan menggunakan aturan. Berikut ini menunjukkan contoh `SELECT` klausa yang didukung.

- Anda dapat menggunakan `SELECT` klausa berikut hanya dengan `*` untuk muatan biner.

```
SELECT * FROM 'topic/subtopic'
```

- ```
SELECT * FROM 'topic/subtopic' WHERE timestamp() % 12 = 0
```
- Anda juga dapat menambahkan data dan menggunakan SELECT klausa berikut.
  - ```
SELECT *, principal() as principal, timestamp() as time FROM 'topic/subtopic'
```
 - ```
SELECT encode(*, 'base64') AS data, timestamp() AS ts FROM 'topic/subtopic'
```
- Anda juga dapat menggunakan SELECT klausa ini dengan muatan biner.
  - Berikut ini mengacu device\_type pada klausa WHERE.

```
SELECT * FROM 'topic/subtopic' WHERE device_type = 'thermostat'
```

- Berikut ini juga didukung.

```
{
 "sql": "SELECT * FROM 'topic/subtopic'",
 "actions": [
 {
 "republish": {
 "topic": "device/${device_id}"
 }
 }
]
}
```

Tindakan aturan berikut tidak mendukung muatan biner sehingga Anda harus mendekodekannya.

- Beberapa tindakan aturan tidak mendukung input payload biner, seperti tindakan [Lambda](#), jadi Anda harus memecahkan kode muatan biner. Tindakan aturan Lambda dapat menerima data biner, jika base64 dikodekan dan dalam muatan JSON. Anda dapat melakukan ini dengan mengubah aturan menjadi berikut.

```
SELECT encode(*, 'base64') AS data FROM 'my_topic'
```

- Pernyataan SQL tidak mendukung string sebagai input. Untuk mengonversi input string ke JSON, Anda dapat menjalankan perintah berikut.

```
SELECT decode(encode(*, 'base64'), 'base64') AS payload FROM 'topic'
```

## Decoding muatan pesan protobuf

[Protocol Buffers \(protobuf\)](#) adalah format data sumber terbuka yang digunakan untuk membuat serial data terstruktur dalam bentuk biner yang ringkas. Ini digunakan untuk mentransmisikan data melalui jaringan atau menyimpannya dalam file. Protobuf memungkinkan Anda untuk mengirim data dalam ukuran paket kecil dan pada tingkat yang lebih cepat daripada format pesan lainnya. AWS IoT Core Aturan mendukung protobuf dengan menyediakan fungsi SQL [decode \(value, decodingScheme\)](#), yang memungkinkan Anda untuk memecahkan kode muatan pesan yang disandikan protobuf ke format JSON dan merutekannya ke layanan hilir. Bagian ini merinci step-by-step proses untuk mengkonfigurasi decoding protobuf dalam Aturan. AWS IoT Core

Di bagian ini:

- [Prasyarat](#)
- [Buat file deskriptor](#)
- [Unggah file deskriptor ke bucket S3](#)
- [Konfigurasi decoding protobuf dalam Aturan](#)
- [Batasan](#)
- [Praktik terbaik](#)

### Prasyarat

- Pemahaman dasar tentang [Protocol Buffer \(protobuf\)](#)
- [.protoFile](#) yang menentukan jenis pesan dan dependensi terkait
- Menginstal [Protobuf Compiler \(protoc\)](#) pada sistem Anda

### Buat file deskriptor

Jika Anda sudah memiliki file deskriptor, Anda dapat melewati langkah ini. File deskriptor (.desc) adalah versi .proto file yang dikompilasi, yang merupakan file teks yang mendefinisikan struktur data dan jenis pesan yang akan digunakan dalam serialisasi protobuf. Untuk menghasilkan file deskriptor, Anda harus menentukan .proto file dan menggunakan kompiler [protoc](#) untuk mengompilasinya.

1. Buat .proto file yang menentukan jenis pesan. .protoFile contoh dapat terlihat seperti berikut:

```
syntax = "proto3";
```

```
message Person {
 optional string name = 1;
 optional int32 id = 2;
 optional string email = 3;
}
```

Dalam `.proto` file contoh ini, Anda menggunakan sintaks `proto3` dan menentukan jenis pesan. `Person` Definisi `Person` pesan menentukan tiga bidang (nama, id, dan email). Untuk informasi selengkapnya tentang format pesan `.proto` file, lihat [Panduan Bahasa \(proto3\)](#).

- Gunakan kompiler [protoc](#) untuk mengkompilasi `.proto` file dan menghasilkan file deskriptor. Contoh perintah untuk membuat file descriptor (`.desc`) dapat menjadi sebagai berikut:

```
protoc --descriptor_set_out=<FILENAME>.desc \
 --proto_path=<PATH_TO_IMPORTS_DIRECTORY> \
 --include_imports \
 <PROTO_FILENAME>.proto
```

Perintah contoh ini menghasilkan file deskriptor `<FILENAME>.desc`, yang dapat digunakan AWS IoT Core Aturan untuk memecahkan kode muatan `protobuf` yang sesuai dengan struktur data yang ditentukan. `<PROTO_FILENAME>.proto`

- `--descriptor_set_out`

Menentukan nama file deskriptor (`<FILENAME>.desc`) yang harus dihasilkan.

- `--proto_path`

Menentukan lokasi dari setiap `.proto` file impor yang direferensikan oleh file yang sedang dikompilasi. Anda dapat menentukan bendera beberapa kali jika Anda memiliki beberapa `.proto` file yang diimpor dengan lokasi yang berbeda.

- `--include_imports`

Menentukan bahwa setiap `.proto` file yang diimpor juga harus dikompilasi dan disertakan dalam file `<FILENAME>.desc` deskriptor.

- `<PROTO_FILENAME>.proto`

Menentukan nama `.proto` file yang ingin Anda kompilasi.



Untuk informasi selengkapnya tentang referensi protokol, lihat [Referensi API](#).

## Unggah file deskriptor ke bucket S3

Setelah membuat file deskriptor<FILENAME> .desc, unggah file deskriptor <FILENAME> .desc ke bucket Amazon S3, menggunakan AWS API, SDK AWS , atau file. AWS Management Console

### Pertimbangan penting

- Pastikan Anda mengunggah file deskriptor ke bucket Amazon S3 di Akun AWS tempat yang Wilayah AWS sama dengan tempat Anda ingin mengonfigurasi Aturan.
- Pastikan Anda memberikan AWS IoT Core akses untuk membaca FileDescriptorSet dari S3. Jika bucket S3 Anda menonaktifkan enkripsi sisi server (SSE) atau bucket S3 Anda dienkrpsi menggunakan kunci yang dikelola Amazon S3 (SSE-S3), konfigurasi kebijakan tambahan tidak diperlukan. Ini dapat dicapai dengan contoh kebijakan bucket:

```
{
 "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"
 }
]
}
```

- Jika bucket S3 Anda dienkrpsi menggunakan AWS Key Management Service kunci (SSE-KMS), pastikan Anda memberikan AWS IoT Core izin untuk menggunakan kunci saat mengakses bucket S3 Anda. Anda dapat melakukannya dengan menambahkan pernyataan ini ke kebijakan utama Anda:

```
{
 "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"
}

```

Konfigurasi decoding protobuf dalam Aturan

Setelah mengunggah file deskriptor ke bucket Amazon S3, konfigurasi Aturan yang dapat memecahkan kode format payload pesan protobuf Anda menggunakan fungsi SQL decode (value, decodingScheme). Tanda tangan dan contoh fungsi terperinci dapat ditemukan di fungsi SQL [decode \(value, decodingScheme\)](#) dari referensi SQL.AWS IoT

Berikut ini adalah contoh ekspresi SQL menggunakan fungsi [decode \(value, decodingScheme\)](#):

```

SELECT VALUE decode(*, 'proto', '<BUCKET NAME>', '<FILENAME>.desc', '<PROTO_FILENAME>',
'<PROTO_MESSAGE_TYPE>') FROM '<MY_TOPIC>'

```

Dalam contoh ekspresi ini:

- Anda menggunakan fungsi SQL [decode \(value, decodingScheme\)](#) untuk memecahkan kode payload pesan biner yang direferensikan oleh. \* Ini bisa berupa payload berkode protobuf biner atau string JSON yang mewakili payload protobuf yang dikodekan base64.
- Payload pesan yang disediakan dikodekan menggunakan jenis Person pesan yang ditentukan. PROTO\_FILENAME.proto
- Bucket Amazon S3 bernama BUCKET NAME berisi yang FILENAME.desc dihasilkan dari. PROTO\_FILENAME.proto

Setelah Anda menyelesaikan konfigurasi, publikasikan pesan ke AWS IoT Core topik yang menjadi aturan berlangganan.

## Batasan

AWS IoT Core Aturan mendukung protobuf dengan batasan berikut:

- Decoding payload pesan protobuf dalam template [substitusi](#) tidak didukung.
- Saat mendekode muatan pesan protobuf, Anda dapat menggunakan [fungsi SQL decode dalam satu ekspresi SQL hingga](#) dua kali.
- Ukuran muatan masuk maksimum adalah 128 KiB (1KiB = 1024 byte), ukuran muatan keluar maksimum adalah 128 KiB, dan ukuran maksimum untuk objek yang `FileDescriptorSet` disimpan dalam ember Amazon S3 adalah 32 KiB.
- Bucket Amazon S3 yang dienkripsi dengan enkripsi SSE-C tidak didukung.

## Praktik terbaik

Berikut adalah beberapa praktik terbaik dan kiat pemecahan masalah.

- Cadangkan file proto Anda di bucket Amazon S3.

Ini adalah praktik yang baik untuk membuat cadangan file proto Anda jika terjadi kesalahan. Misalnya, jika Anda salah memodifikasi file proto tanpa backup saat menjalankan protoc, ini dapat menyebabkan masalah pada tumpukan produksi Anda. Ada beberapa cara untuk mencadangkan file Anda di bucket Amazon S3. Misalnya, Anda dapat [menggunakan pembuatan versi di bucket S3](#). Untuk informasi selengkapnya tentang cara mencadangkan file di bucket Amazon S3, lihat Panduan Pengembang [Amazon S3](#).

- Konfigurasi AWS IoT logging untuk melihat entri log.

Ini adalah praktik yang baik untuk mengonfigurasi AWS IoT logging sehingga Anda dapat memeriksa AWS IoT log untuk akun Anda CloudWatch. Ketika kueri SQL aturan memanggil fungsi eksternal, AWS IoT Core Aturan menghasilkan entri log dengan `eventType` `FunctionExecution`, yang berisi bidang alasan yang akan membantu Anda memecahkan masalah kegagalan. Kemungkinan kesalahan termasuk objek Amazon S3 tidak ditemukan, atau deskriptor file protobuf yang tidak valid. Untuk informasi selengkapnya tentang cara mengonfigurasi AWS IoT logging dan melihat entri log, lihat [Mengkonfigurasi entri log mesin AWS IoT logging dan Aturan](#).

- Perbarui `FileDescriptorSet` menggunakan kunci objek baru dan perbarui kunci objek di Aturan Anda.

Anda dapat memperbarui `FileDescriptorSet` dengan mengunggah file deskriptor yang diperbarui ke bucket Amazon S3 Anda. Pembaruan Anda `FileDescriptorSet` dapat memakan waktu hingga 15 menit untuk tercermin. Untuk menghindari penundaan ini, adalah praktik yang baik untuk mengunggah pembaruan Anda `FileDescriptorSet` menggunakan kunci objek baru, dan memperbarui kunci objek di Aturan Anda.

## Versi SQL

Mesin AWS IoT aturan menggunakan sintaks seperti SQL untuk memilih data dari pesan MQTT. Pernyataan SQL ditafsirkan berdasarkan versi SQL yang ditentukan dengan `awsIotSqlVersion` properti dalam dokumen JSON yang menjelaskan aturan. Untuk informasi selengkapnya tentang struktur dokumen aturan JSON, lihat [Membuat Aturan](#). `awsIotSqlVersion` properti ini memungkinkan Anda menentukan versi mesin aturan AWS IoT SQL yang ingin Anda gunakan. Saat versi baru diterapkan, Anda dapat terus menggunakan versi yang lebih lama atau mengubah aturan Anda untuk menggunakan versi baru. Aturan Anda saat ini terus menggunakan versi yang dengannya mereka dibuat.

Contoh JSON berikut menunjukkan cara menentukan versi SQL menggunakan properti.

`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 saat ini mendukung versi SQL berikut:

- 2016-03-23- Versi SQL dibangun pada 2016-03-23 (disarankan).
- 2015-10-08- Versi SQL asli dibangun pada 2015-10-08.
- beta— Versi beta SQL terbaru. Versi ini dapat memperkenalkan perubahan yang melanggar aturan Anda.

## Apa yang baru di versi mesin aturan SQL 2016-03-23

- Perbaikan untuk memilih objek JSON bersarang.
- Perbaikan untuk query array.
- Dukungan kueri intra-objek. Untuk informasi selengkapnya, lihat [Kueri objek bersarang](#).
- Support untuk menampilkan array sebagai objek tingkat atas.
- Penambahan `encode(value, encodingScheme)` fungsi, yang dapat diterapkan pada data format JSON dan non-JSON. Untuk informasi selengkapnya, lihat [fungsi encode](#).

### Output **Array** sebagai objek tingkat atas

Fitur ini memungkinkan aturan untuk mengembalikan array sebagai objek tingkat atas. Misalnya, diberikan pesan MQTT berikut:

```
{
 "a": {"b":"c"},
 "arr":[1,2,3,4]
}
```

Dan aturan berikut:

```
SELECT VALUE arr FROM 'topic'
```

Aturan menghasilkan output berikut.

```
[1,2,3,4]
```

# AWS IoT Layanan Device Shadow

Layanan AWS IoT Device Shadow menambahkan bayangan ke objek AWS IoT benda. Bayangan dapat membuat status perangkat tersedia untuk aplikasi dan layanan lain apakah perangkat terhubung AWS IoT atau tidak. AWS IoT benda benda dapat memiliki beberapa bayangan bernama sehingga solusi IoT Anda memiliki lebih banyak opsi untuk menghubungkan perangkat Anda ke aplikasi dan layanan lain.

AWS IoT benda benda tidak memiliki bayangan sampai mereka dibuat secara eksplisit. Bayangan dapat dibuat, diperbarui, dan dihapus dengan menggunakan AWS IoT konsol. [Perangkat, klien web lainnya, dan layanan dapat membuat, memperbarui, dan menghapus bayangan dengan menggunakan MQTT dan topik MQTT yang dicadangkan, HTTP menggunakan Device Shadow REST API, dan for. AWS CLI AWS IoT](#) Karena bayangan disimpan oleh AWS di cloud, mereka dapat mengumpulkan dan melaporkan data status perangkat dari aplikasi dan layanan cloud lainnya apakah perangkat terhubung atau tidak.

## Menggunakan bayangan

Shadows menyediakan penyimpanan data yang andal untuk perangkat, aplikasi, dan layanan cloud lainnya untuk berbagi data. Mereka memungkinkan perangkat, aplikasi, dan layanan cloud lainnya untuk terhubung dan memutuskan sambungan tanpa kehilangan status perangkat.

Sementara perangkat, aplikasi, dan layanan cloud lainnya terhubung AWS IoT, mereka dapat mengakses dan mengontrol keadaan perangkat saat ini melalui bayangannya. Misalnya, aplikasi dapat meminta perubahan dalam status perangkat dengan memperbarui bayangan. AWS IoT menerbitkan pesan yang menunjukkan perubahan pada perangkat. Perangkat menerima pesan ini, memperbarui statusnya agar cocok, dan menerbitkan pesan dengan statusnya yang diperbarui. Layanan Device Shadow mencerminkan status yang diperbarui ini dalam bayangan yang sesuai. Aplikasi ini dapat berlangganan pembaruan bayangan atau dapat menanyakan bayangan untuk statusnya saat ini.

Saat perangkat offline, aplikasi masih dapat berkomunikasi dengan AWS IoT dan bayangan perangkat. Ketika perangkat terhubung kembali, ia menerima keadaan bayangannya saat ini sehingga dapat memperbarui statusnya agar sesuai dengan bayangannya, dan kemudian menerbitkan pesan dengan status yang diperbarui. Demikian juga, ketika aplikasi offline dan status perangkat berubah saat offline, perangkat terus memperbarui bayangan sehingga aplikasi dapat menanyakan bayangan untuk statusnya saat ini saat terhubung kembali.

Jika perangkat Anda sering offline dan Anda ingin mengonfigurasi perangkat Anda untuk menerima pesan delta setelah terhubung kembali, Anda dapat menggunakan fitur sesi persisten. Untuk informasi lebih lanjut tentang periode kedaluwarsa sesi persisten, lihat [Periode kedaluwarsa sesi persisten](#).

## Memilih untuk menggunakan bayangan bernama atau tidak disebutkan namanya

Layanan Device Shadow mendukung bayangan bernama dan tidak disebutkan namanya, atau klasik. Objek benda dapat memiliki beberapa bayangan bernama, dan tidak lebih dari satu bayangan yang tidak disebutkan namanya. Objek benda juga dapat memiliki bayangan bernama cadangan, yang beroperasi mirip dengan bayangan bernama kecuali Anda tidak dapat memperbarui namanya. Untuk informasi selengkapnya, lihat [Cadangan bernama bayangan](#).

Objek benda dapat memiliki bayangan bernama dan tidak disebutkan namanya secara bersamaan; namun, API yang digunakan untuk mengakses masing-masing sedikit berbeda, jadi mungkin lebih efisien untuk memutuskan jenis bayangan mana yang paling cocok untuk solusi Anda dan hanya menggunakan jenis itu. Untuk informasi selengkapnya tentang API untuk mengakses bayangan, lihat [Topik bayangan](#).

Dengan bayangan bernama, Anda dapat membuat tampilan yang berbeda dari keadaan objek benda. Misalnya, Anda dapat membagi objek benda dengan banyak properti menjadi bayangan dengan kelompok properti logis, masing-masing diidentifikasi dengan nama bayangannya. Anda juga dapat membatasi akses ke properti dengan mengelompokkannya ke dalam bayangan yang berbeda dan menggunakan kebijakan untuk mengontrol akses. Untuk informasi selengkapnya tentang kebijakan yang akan digunakan dengan bayangan perangkat, lihat [Tindakan, sumber daya, dan kunci kondisi untuk AWS IoT](#) dan [AWS IoT Core kebijakan](#).

Bayangan klasik yang tidak disebutkan namanya lebih sederhana, tetapi agak lebih terbatas daripada bayangan bernama. Setiap AWS IoT benda hanya dapat memiliki satu bayangan yang tidak disebutkan namanya. Jika Anda mengharapkan solusi IoT Anda memiliki kebutuhan terbatas untuk data bayangan, ini mungkin cara Anda ingin mulai menggunakan bayangan. Namun, jika Anda berpikir Anda mungkin ingin menambahkan bayangan tambahan di masa depan, pertimbangkan untuk menggunakan bayangan bernama sejak awal.

Pengindeksan armada mendukung bayangan yang tidak disebutkan namanya dan bayangan bernama berbeda. Untuk informasi selengkapnya, lihat [Mengelola pengindeksan armada](#).

## Mengakses bayangan

Setiap bayangan memiliki [topik MQTT](#) yang dicadangkan dan [URL HTTP](#) yang mendukung `get`, `update`, dan `delete` tindakan pada bayangan.

Bayangan menggunakan [dokumen bayangan JSON](#) untuk menyimpan dan mengambil data. Dokumen bayangan berisi properti state yang menjelaskan aspek-aspek status perangkat ini:

- `desired`

Aplikasi menentukan status properti perangkat yang diinginkan dengan memperbarui `desired` objek.

- `reported`

Perangkat melaporkan keadaan mereka saat ini di `reported` objek.

- `delta`

AWS IoT melaporkan perbedaan antara keadaan yang diinginkan dan yang dilaporkan dalam `delta` objek.

Data yang disimpan dalam bayangan ditentukan oleh properti status badan pesan tindakan pembaruan. Tindakan pembaruan selanjutnya dapat memodifikasi nilai objek data yang ada, dan juga menambahkan dan menghapus kunci dan elemen lain dalam objek status bayangan. Untuk informasi selengkapnya tentang mengakses bayangan, lihat [Menggunakan bayangan di perangkat](#) dan [Menggunakan bayangan di aplikasi dan layanan](#).

### Important

Izin untuk membuat permintaan pembaruan harus dibatasi pada aplikasi dan perangkat tepercaya. Ini mencegah properti status bayangan diubah secara tidak terduga; jika tidak, perangkat dan aplikasi yang menggunakan bayangan harus dirancang untuk mengharapkan kunci di properti status berubah.

## Menggunakan bayangan di perangkat, aplikasi, dan layanan cloud lainnya

Menggunakan bayangan di perangkat, aplikasi, dan layanan cloud lainnya membutuhkan konsistensi dan koordinasi di antara semua ini. Layanan AWS IoT Device Shadow menyimpan status bayangan,



mengirim pesan saat status bayangan berubah, dan merespons pesan yang mengubah statusnya. Perangkat, aplikasi, dan layanan cloud lainnya dalam solusi IoT Anda harus mengelola statusnya dan menjaganya tetap konsisten dengan status bayangan perangkat.

Data status bayangan bersifat dinamis dan dapat diubah oleh perangkat, aplikasi, dan layanan cloud lainnya dengan izin untuk mengakses bayangan. Untuk alasan ini, penting untuk mempertimbangkan bagaimana setiap perangkat, aplikasi, dan layanan cloud lainnya akan berinteraksi dengan bayangan. Sebagai contoh:

- Perangkat harus menulis hanya ke `reported` properti status bayangan saat mengkomunikasikan data status ke bayangan.
- Aplikasi dan layanan cloud lainnya harus menulis hanya ke `desired` properti saat mengkomunikasikan permintaan perubahan status ke perangkat melalui bayangan.

#### Important

Data yang terkandung dalam objek data bayangan independen dari bayangan lain dan properti objek benda lain, seperti atribut benda dan konten pesan MQTT yang mungkin dipublikasikan oleh perangkat objek benda. Namun, perangkat dapat melaporkan data yang sama dalam topik dan bayangan MQTT yang berbeda jika perlu.

Perangkat yang mendukung banyak bayangan harus menjaga konsistensi data yang dilaporkan dalam bayangan yang berbeda.

## Pesanan pesan

Tidak ada jaminan bahwa pesan dari AWS IoT layanan akan tiba di perangkat dalam urutan tertentu. Skenario berikut menunjukkan apa yang terjadi dalam kasus ini.

Dokumen negara awal:

```
{
 "state": {
 "reported": {
 "color": "blue"
 }
 },
 "version": 9,
 "timestamp": 123456776
}
```

```
}
```

### Pembaruan 1:

```
{
 "state": {
 "desired": {
 "color": "RED"
 }
 },
 "version": 10,
 "timestamp": 123456777
}
```

### Perbarui 2:

```
{
 "state": {
 "desired": {
 "color": "GREEN"
 }
 },
 "version": 11,
 "timestamp": 123456778
}
```

### Dokumen negara bagian akhir:

```
{
 "state": {
 "reported": {
 "color": "GREEN"
 }
 },
 "version": 12,
 "timestamp": 123456779
}
```

### Ini menghasilkan dua pesan delta:

```
{
```

```
"state": {
 "color": "RED"
},
"version": 11,
"timestamp": 123456778
}
```

```
{
 "state": {
 "color": "GREEN"
 },
 "version": 12,
 "timestamp": 123456779
}
```

Perangkat mungkin menerima pesan-pesan ini rusak. Karena status dalam pesan ini bersifat kumulatif, perangkat dapat dengan aman membuang pesan apa pun yang berisi nomor versi yang lebih lama dari yang dilacaknya. Jika perangkat menerima delta untuk versi 12 sebelum versi 11, perangkat dapat dengan aman membuang pesan versi 11.

## Pangkas pesan bayangan

Untuk mengurangi ukuran pesan bayangan yang dikirim ke perangkat Anda, tentukan aturan yang hanya memilih bidang yang dibutuhkan perangkat Anda, lalu menerbitkan kembali pesan pada topik MQTT yang mendengarkan perangkat Anda.

Aturan ditentukan dalam JSON dan akan terlihat seperti berikut:

```
{
 "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"
 }
 }
]
}
```

Pernyataan `SELECT` menentukan bidang mana dari pesan yang akan dipublikasikan ulang ke topik yang ditentukan. Kartu liar “+” digunakan untuk mencocokkan semua nama bayangan. Aturan menetapkan bahwa semua pesan yang cocok harus dipublikasikan ulang ke topik yang ditentukan. Dalam hal ini, "`topic()`" fungsi ini digunakan untuk menentukan topik yang akan diterbitkan ulang. `topic(3)` mengevaluasi nama benda dalam topik asli. Untuk informasi selengkapnya tentang membuat aturan, lihat [Aturan untuk AWS IoT](#).

## Menggunakan bayangan di perangkat

Bagian ini menjelaskan komunikasi perangkat dengan bayangan menggunakan pesan MQTT, metode yang disukai perangkat untuk berkomunikasi dengan layanan AWS IoT Device Shadow.

Komunikasi bayangan meniru model permintaan/respons menggunakan model komunikasi terbitkan/berlangganan MQTT. Setiap tindakan bayangan terdiri dari topik permintaan, topik respons yang berhasil (`accepted`), dan topik respons kesalahan (`rejected`).

Jika Anda ingin aplikasi dan layanan dapat menentukan apakah perangkat terhubung, lihat [Mendeteksi perangkat terhubung](#).

### Important

Karena MQTT menggunakan model komunikasi terbitkan/berlangganan, Anda harus berlangganan topik respons sebelum mempublikasikan topik permintaan. Jika tidak, Anda mungkin tidak menerima tanggapan atas permintaan yang Anda publikasikan. Jika Anda menggunakan a [AWS IoT Device SDK](#) untuk memanggil API layanan Device Shadow, ini akan ditangani untuk Anda.

Contoh di bagian ini menggunakan bentuk singkat dari topik di mana *ShadowTopicPrefix* dapat merujuk ke bayangan bernama atau tidak disebutkan namanya, seperti yang dijelaskan dalam tabel ini.

Bayangan dapat dinamai atau tidak disebutkan namanya (klasik). Topik yang digunakan oleh masing-masing hanya berbeda dalam awalan topik. Tabel ini menunjukkan awalan topik yang digunakan oleh setiap jenis bayangan.

| Nilai <i>ShadowTopicPrefix</i>                                 | Jenis bayangan               |
|----------------------------------------------------------------|------------------------------|
| \$aws/things/ <i>thingName</i> /shadow                         | Bayangan tanpa nama (klasik) |
| \$aws/things/ <i>thingName</i> /shadow/name/ <i>shadowName</i> | Bernama bayangan             |

### ⚠ Important

Pastikan penggunaan bayangan aplikasi atau layanan Anda konsisten dan didukung oleh implementasi yang sesuai di perangkat Anda. Misalnya, pertimbangkan bagaimana bayangan dibuat, diperbarui, dan dihapus. Pertimbangkan juga bagaimana pembaruan ditangani di perangkat dan aplikasi atau layanan yang mengakses perangkat melalui bayangan. Desain Anda harus jelas tentang bagaimana status perangkat diperbarui dan dilaporkan serta bagaimana aplikasi dan layanan Anda berinteraksi dengan perangkat dan bayangannya.

Untuk membuat topik lengkap, pilih jenis bayangan yang ingin Anda rujuk, ganti, dan *shadowName* jika berlaku *thingName*, dengan nilai yang sesuai, lalu tambahkan dengan rintisan topik seperti yang ditunjukkan pada tabel berikut. *ShadowTopicPrefix* Ingatlah bahwa topik peka huruf besar/kecil.

Lihat [Topik bayangan](#) untuk informasi lebih lanjut tentang topik yang dicadangkan untuk bayangan.

## Menginisialisasi perangkat pada koneksi pertama ke AWS IoT

Setelah perangkat mendaftar AWS IoT, perangkat harus berlangganan pesan MQTT ini untuk bayangan yang didukungnya.

| Topik                                      | Arti                                                          | Tindakan yang harus dilakukan perangkat saat topik ini diterima                                             |
|--------------------------------------------|---------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------|
| <i>ShadowTopicPrefix</i> / delete/accepted | deletePermintaan itu diterima dan AWS IoT menghapus bayangan. | Tindakan yang diperlukan untuk mengakomodasi bayangan yang dihapus, seperti berhenti menerbitkan pembaruan. |

| Topik                                       | Arti                                                                                                      | Tindakan yang harus dilakukan perangkat saat topik ini diterima                     |
|---------------------------------------------|-----------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|
| <i>ShadowTopicPrefix</i> / delete/rejected  | deletePermintaan ditolak oleh AWS IoT dan bayangan tidak dihapus. Badan pesan berisi informasi kesalahan. | Menanggapi pesan kesalahan di badan pesan.                                          |
| <i>ShadowTopicPrefix</i> / get/accepted     | getPermintaan diterima oleh AWS IoT, dan badan pesan berisi dokumen bayangan saat ini.                    | Tindakan yang diperlukan untuk memproses dokumen negara di badan pesan.             |
| <i>ShadowTopicPrefix</i> / get/rejected     | getPermintaan ditolak oleh AWS IoT, dan badan pesan berisi informasi kesalahan.                           | Menanggapi pesan kesalahan di badan pesan.                                          |
| <i>ShadowTopicPrefix</i> / update/accepted  | updatePermintaan diterima oleh AWS IoT, dan badan pesan berisi dokumen bayangan saat ini.                 | Konfirmasikan data yang diperbarui di badan pesan cocok dengan status perangkat.    |
| <i>ShadowTopicPrefix</i> / update/rejected  | updatePermintaan ditolak oleh AWS IoT, dan badan pesan berisi informasi kesalahan.                        | Menanggapi pesan kesalahan di badan pesan.                                          |
| <i>ShadowTopicPrefix</i> / update/delta     | Dokumen bayangan diperbarui oleh permintaan ke AWS IoT, dan badan pesan berisi perubahan yang diminta.    | Perbarui status perangkat agar sesuai dengan status yang diinginkan di badan pesan. |
| <i>ShadowTopicPrefix</i> / update/documents | Pembaruan bayangan baru-baru ini selesai, dan badan pesan berisi dokumen bayangan saat ini.               | Konfirmasikan status yang diperbarui di badan pesan cocok dengan status perangkat.  |

Setelah berlangganan pesan di tabel sebelumnya untuk setiap bayangan, perangkat harus menguji untuk melihat apakah bayangan yang didukungnya telah dibuat dengan menerbitkan `/get` topik ke setiap bayangan. Jika `/get/accepted` pesan diterima, badan pesan berisi dokumen bayangan, yang dapat digunakan perangkat untuk menginisialisasi statusnya. Jika `/get/rejected` pesan diterima, bayangan harus dibuat dengan menerbitkan `/update` pesan dengan status perangkat saat ini.

Misalnya, Anda memiliki sesuatu `My_IoT_Thing` yang tidak memiliki bayangan klasik atau bernama. Jika Anda sekarang mempublikasikan `/get` permintaan pada topik yang dicadangkan `$aws/things/My_IoT_Thing/shadow/get`, itu mengembalikan kesalahan pada `$aws/things/My_IoT_Thing/shadow/get/rejected` topik karena benda tersebut tidak memiliki bayangan. Untuk mengatasi kesalahan ini, pertama-tama publikasikan `/update` pesan dengan menggunakan `$aws/things/My_IoT_Thing/shadow/update` topik dengan status perangkat saat ini seperti payload berikut.

```
{
 "state": {
 "reported": {
 "welcome": "aws-iot",
 "color": "yellow"
 }
 }
}
```

Bayangan klasik sekarang dibuat untuk benda itu dan pesan dipublikasikan ke `$aws/things/My_IoT_Thing/shadow/update/accepted` topik. Jika Anda memublikasikan ke topik `$aws/things/My_IoT_Thing/shadow/get`, ia mengembalikan respons ke `$aws/things/My_IoT_Thing/shadow/get/accepted` topik dengan status perangkat.

Untuk bayangan bernama, Anda harus terlebih dahulu membuat bayangan bernama atau mempublikasikan pembaruan dengan nama bayangan sebelum menggunakan permintaan `get`. Misalnya, untuk membuat bayangan bernama `namedShadow1`, pertama-tama publikasikan informasi status perangkat ke topik `$aws/things/My_IoT_Thing/shadow/name/namedShadow1/update`. Untuk mengambil informasi status, gunakan `/get` permintaan untuk bayangan bernama, `$aws/things/My_IoT_Thing/shadow/name/namedShadow1/get`.

## Memproses pesan saat perangkat terhubung AWS IoT

Saat perangkat terhubung AWS IoT, perangkat dapat menerima pesan `/update/delta` dan harus menjaga status perangkat tetap sesuai dengan perubahan bayangannya dengan:

1. Membaca semua pesan `/update/delta` yang diterima dan menyinkronkan status perangkat agar sesuai.
2. Menerbitkan pesan `/update` dengan badan `reported` pesan yang memiliki status perangkat saat ini, setiap kali status perangkat berubah.

Saat perangkat terhubung, perangkat harus mempublikasikan pesan-pesan ini saat ditunjukkan.

| Indikasi                                                                                                                                 | Topik                                | Muatan                                                  |
|------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------|---------------------------------------------------------|
| Status perangkat telah berubah.                                                                                                          | <i>ShadowTopicPrefix</i> /<br>update | Dokumen bayangan dengan <code>reported</code> properti. |
| Perangkat mungkin tidak disinkronkan dengan bayangan.                                                                                    | <i>ShadowTopicPrefix</i> /get        | (kosong)                                                |
| Tindakan pada perangkat menunjukkan bahwa bayangan tidak akan lagi didukung oleh perangkat, seperti saat perangkat dilepas atau diganti. | <i>ShadowTopicPrefix</i> /<br>delete | (kosong)                                                |

## Memproses pesan saat perangkat tersambung kembali AWS IoT

Ketika perangkat dengan satu atau lebih bayangan terhubung AWS IoT, itu harus menyinkronkan statusnya dengan semua bayangan yang didukungnya dengan:

1. Membaca semua pesan `/update/delta` yang diterima dan menyinkronkan status perangkat agar sesuai.
2. Menerbitkan pesan `/update` dengan badan `reported` pesan yang memiliki status perangkat saat ini.



## Menggunakan bayangan di aplikasi dan layanan

Bagian ini menjelaskan cara aplikasi atau layanan berinteraksi dengan layanan AWS IoT Device Shadow. Contoh ini mengasumsikan aplikasi atau layanan hanya berinteraksi dengan bayangan dan, melalui bayangan, perangkat. Contoh ini tidak menyertakan tindakan manajemen apa pun, seperti membuat atau menghapus bayangan.

Contoh ini menggunakan REST API layanan AWS IoT Device Shadow untuk berinteraksi dengan bayangan. Berbeda dengan contoh yang digunakan dalam [Menggunakan bayangan di perangkat](#), yang menggunakan model komunikasi terbitkan/berlangganan, contoh ini menggunakan model komunikasi permintaan/respons dari REST API. Ini berarti aplikasi atau layanan harus membuat permintaan sebelum dapat menerima tanggapan dari AWS IoT. Kerugian dari model ini, bagaimanapun, adalah tidak mendukung notifikasi. Jika aplikasi atau layanan Anda memerlukan pemberitahuan perubahan status perangkat secara tepat waktu, pertimbangkan protokol MQTT atau MQTT melalui WSS, yang mendukung model komunikasi terbitkan/berlangganan, seperti yang dijelaskan dalam [Menggunakan bayangan di perangkat](#)

### Important

Pastikan penggunaan bayangan aplikasi atau layanan Anda konsisten dan didukung oleh implementasi yang sesuai di perangkat Anda. Pertimbangkan, misalnya, bagaimana bayangan dibuat, diperbarui, dan dihapus, dan bagaimana pembaruan ditangani di perangkat dan aplikasi atau layanan yang mengakses bayangan. Desain Anda harus secara jelas menentukan bagaimana status perangkat diperbarui dan dilaporkan, dan bagaimana aplikasi dan layanan Anda berinteraksi dengan perangkat dan bayangannya.

URL REST API untuk bayangan bernama adalah:

```
https://endpoint/things/thingName/shadow?name=shadowName
```

dan untuk bayangan yang tidak disebutkan namanya:

```
https://endpoint/things/thingName/shadow
```

di mana:

## titik akhir

Titik akhir yang dikembalikan oleh perintah CLI:

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

## thingName

Nama benda benda yang menjadi milik bayangan

## shadowName

Nama bayangan bernama. Parameter ini tidak digunakan dengan bayangan yang tidak disebutkan namanya.

## Menginisialisasi aplikasi atau layanan pada koneksi ke AWS IoT

Ketika aplikasi pertama kali terhubung AWS IoT, itu harus mengirim permintaan HTTP GET ke URL bayangan yang digunakannya untuk mendapatkan status bayangan saat ini yang digunakannya. Ini memungkinkannya untuk menyinkronkan aplikasi atau layanan ke bayangan.

## Status pemrosesan berubah saat aplikasi atau layanan terhubung AWS IoT

Saat aplikasi atau layanan terhubung AWS IoT, aplikasi dapat menanyakan status saat ini secara berkala dengan mengirimkan permintaan HTTP GET pada URL bayangan yang digunakannya.

Saat pengguna akhir berinteraksi dengan aplikasi atau layanan untuk mengubah status perangkat, aplikasi atau layanan dapat mengirim permintaan HTTP POST ke URL bayangan yang digunakan untuk memperbarui `desired` status bayangan. Permintaan ini mengembalikan perubahan yang diterima, tetapi Anda mungkin harus melakukan polling bayangan dengan membuat permintaan HTTP GET hingga perangkat memperbarui bayangan dengan status barunya.

## Mendeteksi perangkat terhubung

Untuk menentukan apakah perangkat saat ini terhubung, sertakan `connected` properti dalam dokumen bayangan dan gunakan pesan MQTT Last Will and Testament (LWT) untuk menyetel `connected` properti `false` jika perangkat terputus karena kesalahan.

**Note**

Pesan MQTT LWT yang dikirim ke topik yang AWS IoT pesan (topik yang dimulai dengan \$) diabaikan oleh layanan Device Shadow. AWS IoT Namun, mereka diproses oleh klien berlangganan dan oleh mesin AWS IoT aturan, jadi Anda perlu membuat pesan LWT yang dikirim ke topik yang tidak pesan dan aturan yang menerbitkan kembali pesan MQTT LWT sebagai pesan pembaruan bayangan ke topik pembaruan bayangan yang dicadangkan, *ShadowTopicPrefix/update*

Untuk mengirim pesan LWT ke layanan Device Shadow

1. Buat aturan yang menerbitkan kembali pesan MQTT LWT pada topik yang dicadangkan. Contoh berikut adalah aturan yang mendengarkan pesan tentang `my/things/myLightBulb/update` topik tersebut dan menerbitkannya kembali. `$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. Saat perangkat terhubung AWS IoT, perangkat akan mendaftarkan pesan LWT ke topik yang tidak dicadangkan agar aturan penerbitan ulang dapat dikenali. Dalam contoh ini, topik itu `my/things/myLightBulb/update` dan menetapkan properti yang terhubung ke `false`.

```
{
 "state": {
 "reported": {
 "connected": "false"
 }
 }
}
```

```
}
}
```

3. Setelah terhubung, perangkat menerbitkan pesan pada topik pembaruan bayangannya, `$aws/things/myLightBulb/shadow/update`, untuk melaporkan statusnya saat ini, yang mencakup pengaturan `connected` properti `true`.

```
{
 "state": {
 "reported": {
 "connected": "true"
 }
 }
}
```

4. Sebelum perangkat terputus dengan anggun, ia menerbitkan pesan tentang topik pembaruan bayangannya, `$aws/things/myLightBulb/shadow/update`, untuk melaporkan status terbarunya, yang mencakup pengaturan propertinya. `connected false`

```
{
 "state": {
 "reported": {
 "connected": "false"
 }
 }
}
```

5. Jika perangkat terputus karena kesalahan, broker AWS IoT pesan menerbitkan pesan LWT perangkat atas nama perangkat. Aturan republish mendeteksi pesan ini dan menerbitkan pesan pembaruan bayangan untuk memperbarui `connected` properti bayangan perangkat.

## Simulasi komunikasi layanan Device Shadow

Topik ini menunjukkan bagaimana layanan Device Shadow bertindak sebagai perantara dan memungkinkan perangkat dan aplikasi menggunakan bayangan untuk memperbarui, menyimpan, dan mengambil status perangkat.

Untuk mendemonstrasikan interaksi yang dijelaskan dalam topik ini, dan untuk menjelajahnya lebih lanjut, Anda memerlukan Akun AWS dan sistem tempat Anda dapat menjalankannya AWS CLI. Jika Anda tidak memilikinya, Anda masih dapat melihat interaksi dalam contoh kode.

Dalam contoh ini, AWS IoT konsol mewakili perangkat. AWS CLI Ini mewakili aplikasi atau layanan yang mengakses perangkat melalui bayangan. AWS CLI Antarmukanya sangat mirip dengan API yang mungkin digunakan aplikasi untuk berkomunikasi AWS IoT. Perangkat dalam contoh ini adalah bola lampu pintar dan aplikasi menampilkan status bola lampu dan dapat mengubah status bola lampu.

## Menyiapkan simulasi

Prosedur ini menginisialisasi simulasi dengan membuka [AWS IoT konsol](#), yang mensimulasikan perangkat Anda, dan jendela baris perintah yang mensimulasikan aplikasi Anda.

Untuk mengatur lingkungan simulasi Anda

1. Anda Akun AWS perlu menjalankan contoh dari topik ini sendiri. Jika Anda tidak memiliki Akun AWS, buat satu, seperti yang dijelaskan dalam [Siapkan Anda Akun AWS](#).
2. Buka [AWS IoT konsol](#), dan di menu sebelah kiri, pilih Uji untuk membuka klien MQTT.
3. Di jendela lain, buka jendela terminal pada sistem yang telah AWS CLI diinstal di atasnya.

Anda harus membuka dua jendela: satu dengan AWS IoT konsol di halaman Uji, dan satu dengan prompt baris perintah.

## Inisialisasi perangkat

Dalam simulasi ini, kita akan bekerja dengan benda bernama objek,, dan bayangannya bernama mySimulatedThing, SimShadow1.

Buat objek benda dan kebijakan IoT-nya

Untuk membuat objek benda, di AWS IoT Konsol:

1. Pilih Kelola dan kemudian pilih Things.
2. Klik tombol Create jika hal-hal tercantum jika tidak, klik Daftarkan satu hal untuk membuat satu AWS IoT hal.
3. Masukkan namamySimulatedThing, biarkan pengaturan lain menjadi default, lalu klik Berikutnya.
4. Gunakan pembuatan sertifikat sekali klik untuk menghasilkan sertifikat yang akan mengautentikasi koneksi perangkat. AWS IoT Klik Aktifkan untuk mengaktifkan sertifikat.

5. Anda dapat melampirkan kebijakan `My_IoT_Policy` yang akan memberikan izin perangkat untuk mempublikasikan dan berlangganan topik yang dicadangkan MQTT. Untuk langkah-langkah lebih rinci tentang cara membuat AWS IoT sesuatu dan cara membuat kebijakan ini, lihat [Buat objek benda](#).

Buat bayangan bernama untuk objek benda

Anda dapat membuat bayangan bernama untuk sesuatu dengan menerbitkan permintaan pembaruan ke topik `$aws/things/mySimulatedThing/shadow/name/simShadow1/update` seperti yang dijelaskan di bawah ini.

Atau, untuk membuat bayangan bernama:

1. Di AWS IoT Console, pilih objek benda Anda dalam daftar hal-hal yang ditampilkan dan kemudian pilih Shadows.
2. Pilih Tambahkan bayangan, masukkan `simShadow1`, lalu pilih Buat untuk menambahkan bayangan bernama.

Berlangganan dan publikasikan ke topik MQTT yang dipesan

Di konsol, berlangganan topik bayangan MQTT yang dipesan. Topik-topik ini adalah tanggapan terhadap `get`, `update`, dan `delete` tindakan sehingga perangkat Anda akan siap menerima tanggapan setelah menerbitkan tindakan.

Untuk berlangganan topik MQTT di klien MQTT

1. Di klien MQTT, pilih Berlangganan topik.
2. Masukkan `get`, `update`, dan `delete` topik untuk berlangganan. Salin satu topik pada satu waktu dari daftar berikut, tempel di bidang Filter topik, lalu klik Berlangganan. Anda akan melihat topik muncul di bawah Langganan.
  - `$aws/things/mySimulatedThing/shadow/name/simShadow1/delete/accepted`
  - `$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`

Pada titik ini, perangkat simulasi Anda siap menerima topik saat dipublikasikan oleh AWS IoT.

Untuk mempublikasikan ke topik MQTT di klien MQTT

Setelah perangkat menginisialisasi dirinya sendiri dan berlangganan topik respons, perangkat harus meminta bayangan yang didukungnya. Simulasi ini hanya mendukung satu bayangan, bayangan yang mendukung objek benda bernama, bernama `mySimulatedThing`, `SimShadow1`.

Untuk mendapatkan status bayangan saat ini dari klien MQTT

1. Di klien MQTT, pilih Publikasikan ke topik.
2. Di bawah Publikasikan, masukkan topik berikut dan hapus konten apa pun dari jendela isi pesan di bawah tempat Anda memasukkan topik yang akan didapatkan. Anda kemudian dapat memilih Publikasikan ke topik untuk mempublikasikan permintaan. `$aws/things/mySimulatedThing/shadow/name/simShadow1/get`.

Jika Anda belum membuat bayangan bernama `simShadow1`, Anda menerima pesan dalam `$aws/things/mySimulatedThing/shadow/name/simShadow1/get/rejected` topik dan code adalah `404`, seperti dalam contoh ini sebagai bayangan belum dibuat, jadi kita akan membuatnya berikutnya.

```
{
 "code": 404,
 "message": "No shadow exists with name: 'simShadow1'"
}
```

Untuk membuat bayangan dengan status perangkat saat ini

1. Di klien MQTT, pilih Publikasikan ke topik dan masukkan topik ini:

```
$aws/things/mySimulatedThing/shadow/name/simShadow1/update
```

2. Di jendela isi pesan di bawah tempat Anda memasukkan topik, masukkan dokumen bayangan ini untuk menunjukkan perangkat melaporkan ID dan warnanya saat ini dalam nilai RGB. Pilih Publikasikan untuk mempublikasikan permintaan.

```
{
 "state": {
 "reported": {
 "ID": "SmartLamp21",
 "ColorRGB": [
 128,
 128,
 128
]
 }
 },
 "clientToken": "426bfd96-e720-46d3-95cd-014e3ef12bb6"
}
```

Jika Anda menerima pesan dalam topik:

- `$aws/things/mySimulatedThing/shadow/name/simShadow1/update/accepted`: Ini berarti bayangan telah dibuat dan badan pesan berisi dokumen bayangan saat ini.
- `$aws/things/mySimulatedThing/shadow/name/simShadow1/update/rejected`: Tinjau kesalahan di badan pesan.
- `$aws/things/mySimulatedThing/shadow/name/simShadow1/get/accepted`: Bayangan sudah ada dan badan pesan memiliki status bayangan saat ini, seperti dalam contoh ini. Dengan ini, Anda dapat mengatur perangkat Anda atau mengonfirmasi bahwa itu cocok dengan status bayangan.

```
{
 "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"
}
```

## Kirim pembaruan dari aplikasi

Bagian ini menggunakan AWS CLI untuk menunjukkan bagaimana aplikasi dapat berinteraksi dengan bayangan.

Untuk mendapatkan status bayangan saat ini menggunakan AWS CLI

Dari baris perintah, masukkan perintah ini.

```
aws iot-data get-thing-shadow --thing-name mySimulatedThing --shadow-name simShadow1 /dev/stdout
```

Pada platform Windows, Anda dapat menggunakan con alih-alih/dev/stdout.

```
aws iot-data get-thing-shadow --thing-name mySimulatedThing --shadow-name simShadow1 con
```

Karena bayangan ada dan telah diinisialisasi oleh perangkat untuk mencerminkan keadaan saat ini, itu harus mengembalikan dokumen bayangan berikut.

```
{
 "state": {
 "reported": {
 "ID": "SmartLamp21",
 "ColorRGB": [
 128,
 128,
 128
]
 }
 },
 "metadata": {
 "reported": {
 "ID": {
 "timestamp": 1591140517
 },
 "ColorRGB": [
 {
 "timestamp": 1591140517
 },
 {
 "timestamp": 1591140517
 },
 {
 "timestamp": 1591140517
 }
]
 }
 },
 "version": 3,
 "timestamp": 1591141111
}
```

Aplikasi dapat menggunakan respons ini untuk menginisialisasi representasi status perangkat.

Jika aplikasi memperbarui status, seperti ketika pengguna akhir mengubah warna bola lampu pintar kami menjadi kuning, aplikasi akan mengirim `update-thing-shadow` perintah. Perintah ini sesuai dengan `UpdateThingShadow` REST API.

Untuk memperbarui bayangan dari aplikasi

Dari baris perintah, masukkan perintah ini.

### 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
```

Jika berhasil, perintah ini harus mengembalikan dokumen bayangan berikut.

```
{
 "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"
}
```

## Menanggapi pembaruan di perangkat

Kembali ke klien MQTT di AWS konsol, Anda akan melihat pesan yang AWS IoT diterbitkan untuk mencerminkan perintah pembaruan yang dikeluarkan di bagian sebelumnya.

Untuk melihat pesan pembaruan di klien MQTT

Di klien MQTT, pilih `$aws/things/ /shadow/name/simshadow1/update/delta mySimulatedThing` di kolom Langganan. Jika nama topik terpotong, Anda dapat menjeda untuk melihat topik lengkapnya. Dalam log topik topik ini, Anda akan melihat `/delta` pesan yang mirip dengan yang ini.

```
{
 "version": 4,
 "timestamp": 1591141596,
 "state": {
 "ColorRGB": [
 255,
 255,
 0
]
 },
 "metadata": {
 "ColorRGB": [
 {
 "timestamp": 1591141596
 },
 {
 "timestamp": 1591141596
 },
 {
 "timestamp": 1591141596
 }
]
 },
 "clientToken": "21b21b21-bfd2-4279-8c65-e2f697ff4fab"
}
```

```
}
```

Perangkat Anda akan memproses konten pesan ini untuk menyetel status perangkat agar sesuai dengan `desired` status dalam pesan.

Setelah perangkat memperbarui status agar sesuai dengan `desired` status dalam pesan, perangkat harus mengirim status baru yang dilaporkan kembali AWS IoT dengan menerbitkan pesan pembaruan. Prosedur ini mensimulasikan ini di klien MQTT.

Untuk memperbarui bayangan dari perangkat

1. Di klien MQTT, pilih Publikasikan ke topik.
2. Di jendela isi pesan, di bidang topik di atas jendela isi pesan, masukkan topik bayangan diikuti dengan `/update` tindakan: `$aws/things/mySimulatedThing/shadow/name/simShadow1/update` dan di badan pesan, masukkan dokumen bayangan yang diperbarui ini, yang menjelaskan keadaan perangkat saat ini. Klik Publikasikan untuk mempublikasikan status perangkat yang diperbarui.

```
{
 "state": {
 "reported": {
 "ColorRGB": [255,255,0]
 }
 },
 "clientToken": "a4dc2227-9213-4c6a-a6a5-053304f60258"
}
```

Jika pesan berhasil diterima oleh AWS IoT, Anda akan melihat respons baru di log pesan `$aws/things/mySimulatedThing/shadow/name/simshadow1/update/accepted` di klien MQTT dengan status bayangan saat ini, seperti contoh ini.

```
{
 "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"
}
```

Pembaruan yang berhasil ke status perangkat yang dilaporkan juga AWS IoT menyebabkan pengiriman deskripsi komprehensif tentang status bayangan dalam pesan ke topik, seperti badan pesan ini yang dihasilkan dari pembaruan bayangan yang dilakukan oleh perangkat dalam prosedur sebelumnya.

```
{
 "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"
}
```

## Amati pembaruan di aplikasi

Aplikasi sekarang dapat menanyakan bayangan untuk status saat ini seperti yang dilaporkan oleh perangkat.

Untuk mendapatkan status bayangan saat ini menggunakan AWS CLI

1. Dari baris perintah, masukkan perintah ini.

```
aws iot-data get-thing-shadow --thing-name mySimulatedThing --shadow-name
simShadow1 /dev/stdout
```

Pada platform Windows, Anda dapat menggunakan `con` alih-alih `/dev/stdout`.

```
aws iot-data get-thing-shadow --thing-name mySimulatedThing --shadow-name
simShadow1 con
```

2. Karena bayangan baru saja diperbarui oleh perangkat untuk mencerminkan keadaan saat ini, itu harus mengembalikan dokumen bayangan berikut.

```
{
 "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
}
```

## Melampaui simulasi

Bereksperimenlah dengan interaksi antara AWS CLI (mewakili aplikasi) dan konsol (mewakili perangkat) untuk memodelkan solusi IoT Anda.

## Berinteraksi dengan bayangan

Topik ini menjelaskan pesan yang terkait dengan masing-masing dari tiga metode yang AWS IoT menyediakan untuk bekerja dengan bayangan. Metode-metode ini meliputi:

### UPDATE

Membuat bayangan jika tidak ada, atau memperbarui konten bayangan yang ada dengan informasi status yang disediakan di badan pesan. AWS IoT merekam stempel waktu dengan setiap pembaruan untuk menunjukkan kapan status terakhir diperbarui. Saat status bayangan berubah, AWS IoT kirim `/delta` pesan ke semua pelanggan MQTT dengan perbedaan antara status `desired` dan status `reported`. Perangkat atau aplikasi yang menerima `/delta` pesan dapat melakukan tindakan berdasarkan perbedaannya. Misalnya, perangkat dapat memperbarui statusnya ke status yang diinginkan, atau aplikasi dapat memperbarui UI untuk mencerminkan perubahan status perangkat.

### GET

Mengambil dokumen bayangan saat ini yang berisi status lengkap bayangan, termasuk metadata.

### DELETE

Menghapus bayangan perangkat dan isinya.

Anda tidak dapat memulihkan dokumen bayangan perangkat yang dihapus, tetapi Anda dapat membuat bayangan perangkat baru dengan nama dokumen bayangan perangkat yang dihapus. Jika Anda membuat dokumen bayangan perangkat yang memiliki nama yang sama dengan yang dihapus dalam 48 jam terakhir, nomor versi dokumen bayangan perangkat baru akan mengikuti yang dihapus. Jika dokumen bayangan perangkat telah dihapus selama lebih dari 48 jam, nomor versi dokumen bayangan perangkat baru dengan nama yang sama adalah 0.

## Dukungan protokol

AWS IoT mendukung [MQTT](#) dan REST API melalui protokol HTTPS untuk berinteraksi dengan bayangan. AWS IoT menyediakan serangkaian topik permintaan dan respons yang dicadangkan untuk tindakan publikasi dan berlangganan MQTT. Perangkat dan aplikasi harus berlangganan topik respons sebelum memublikasikan topik permintaan untuk informasi tentang cara AWS IoT menangani permintaan tersebut. Lihat informasi yang lebih lengkap di [Topik Device Shadow MQTT](#) dan [Device Shadow REST API](#).

## Meminta dan melaporkan status

Saat merancang solusi IoT Anda menggunakan AWS IoT dan bayangan, Anda harus menentukan aplikasi atau perangkat yang akan meminta perubahan dan yang akan menerapkannya. Biasanya, perangkat mengimplementasikan dan melaporkan perubahan kembali ke bayangan dan aplikasi serta layanan merespons dan meminta perubahan dalam bayangan. Solusi Anda mungkin berbeda, tetapi contoh dalam topik ini mengasumsikan bahwa aplikasi klien atau layanan meminta perubahan dalam bayangan dan perangkat melakukan perubahan dan melaporkannya kembali ke bayangan.

## Memperbarui bayangan

Aplikasi atau layanan Anda dapat memperbarui status bayangan dengan menggunakan [UpdateThingShadow](#) API atau dengan memublikasikan ke [/perbarui](#) topik. Pembaruan hanya memengaruhi bidang yang ditentukan dalam permintaan.

## Memperbarui bayangan saat klien meminta perubahan status

Ketika klien meminta perubahan status dalam bayangan dengan menggunakan protokol MQTT

1. Klien harus memiliki dokumen bayangan saat ini sehingga dapat mengidentifikasi properti yang akan diubah. Lihat tindakan `/get` untuk cara mendapatkan dokumen bayangan saat ini.
2. Klien berlangganan topik MQTT ini:
  - `$aws/things/thingName/shadow/name/shadowName/update/accepted`
  - `$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. Klien menerbitkan topik `$aws/things/thingName/shadow/name/shadowName/update` permintaan dengan dokumen negara yang berisi status bayangan yang diinginkan. Hanya properti yang akan diubah yang perlu dimasukkan dalam dokumen. Ini adalah contoh dokumen dengan keadaan yang diinginkan.

```
{
 "state": {
 "desired": {
 "color": {
 "r": 10
 }
 },
 },
}
```

```
 "engine": "ON"
 }
}
}
```

4. Jika permintaan pembaruan valid, AWS IoT perbarui status yang diinginkan dalam bayangan dan menerbitkan pesan tentang topik ini:
  - `$aws/things/thingName/shadow/name/shadowName/update/accepted`
  - `$aws/things/thingName/shadow/name/shadowName/update/delta`

`/update/accepted` Pesan berisi dokumen [/dokumen status respons yang diterima](#) bayangan, dan `/update/delta` pesan berisi dokumen [/dokumen status respons delta](#) bayangan.
5. Jika permintaan pembaruan tidak valid, AWS IoT menerbitkan pesan dengan `$aws/things/thingName/shadow/name/shadowName/update/rejected` topik dengan dokumen [Dokumen respons kesalahan](#) bayangan yang menjelaskan kesalahan.

Saat klien meminta perubahan status dalam bayangan dengan menggunakan API

1. Klien memanggil [UpdateThingShadow](#) API dengan dokumen [Minta dokumen negara](#) status sebagai badan pesannya.
2. Jika permintaan itu valid, AWS IoT mengembalikan kode respons sukses HTTP dan dokumen [/dokumen status respons yang diterima](#) bayangan sebagai badan pesan responsnya.

AWS IoT juga akan menerbitkan pesan MQTT ke `$aws/things/thingName/shadow/name/shadowName/update/delta` topik dengan dokumen [/dokumen status respons delta](#) bayangan untuk perangkat atau klien apa pun yang berlangganan.

3. Jika permintaan tidak valid, AWS IoT mengembalikan kode respons kesalahan HTTP [Dokumen respons kesalahan](#) sebagai badan pesan responsnya.

Ketika perangkat menerima `/desired` status pada `/update/delta` topik, itu membuat perubahan yang diinginkan pada perangkat. Kemudian mengirim pesan ke `/update` topik untuk melaporkan keadaan saat ini ke bayangan.

## Memperbarui bayangan saat perangkat melaporkan statusnya saat ini

Saat perangkat melaporkan statusnya saat ini ke bayangan dengan menggunakan protokol MQTT

1. Perangkat harus berlangganan topik MQTT ini sebelum memperbarui bayangan:
  - `$aws/things/thingName/shadow/name/shadowName/update/accepted`
  - `$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. Perangkat melaporkan statusnya saat ini dengan menerbitkan pesan ke `$aws/things/thingName/shadow/name/shadowName/update` topik yang melaporkan status saat ini, seperti dalam contoh ini.

```
{
 "state": {
 "reported" : {
 "color" : { "r" : 10 },
 "engine" : "ON"
 }
 }
}
```

3. Jika AWS IoT menerima pembaruan, ia menerbitkan pesan ke `$aws/things/thingName/shadow/name/shadowName/update/accepted` topik dengan dokumen [/dokumen status respons yang diterima](#) bayangan.
4. Jika permintaan pembaruan tidak valid, AWS IoT menerbitkan pesan dengan `$aws/things/thingName/shadow/name/shadowName/update/rejected` topik dengan dokumen [Dokumen respons kesalahan](#) bayangan yang menjelaskan kesalahan.

Saat perangkat melaporkan statusnya saat ini ke bayangan dengan menggunakan API

1. Perangkat memanggil [UpdateThingShadow](#) API dengan dokumen [Minta dokumen negara status](#) sebagai badan pesannya.
2. Jika permintaan itu valid, AWS IoT perbarui bayangan dan mengembalikan kode respons sukses HTTP dengan dokumen [/dokumen status respons yang diterima](#) bayangan sebagai badan pesan responsnya.

AWS IoT juga akan menerbitkan pesan MQTT ke `$aws/things/thingName/shadow/name/shadowName/update/delta` topik dengan dokumen [/dokumen status respons delta](#) bayangan untuk perangkat atau klien apa pun yang berlangganan.

3. Jika permintaan tidak valid, AWS IoT mengembalikan kode respons kesalahan HTTP [Dokumen respons kesalahan](#) sebagai badan pesan responsnya.

## Penguncian optimis

Anda dapat menggunakan versi dokumen negara untuk memastikan Anda memperbarui versi terbaru dari dokumen bayangan perangkat. Saat Anda menyediakan versi dengan permintaan pembaruan, layanan menolak permintaan dengan kode respons konflik HTTP 409 jika versi dokumen status saat ini tidak cocok dengan versi yang disediakan. Kode respons konflik juga dapat terjadi pada API apa pun yang memodifikasi `ThingShadow`, termasuk `DeleteThingShadow`.

Sebagai contoh:

Dokumen awal:

```
{
 "state": {
 "desired": {
 "colors": [
 "RED",
 "GREEN",
 "BLUE"
]
 }
 },
 "version": 10
}
```

Pembaruan: (versi tidak cocok; permintaan ini akan ditolak)

```
{
 "state": {
 "desired": {
 "colors": [
 "BLUE"
]
 }
 }
}
```

```
},
"version": 9
}
```

Hasil:

```
{
 "code": 409,
 "message": "Version conflict",
 "clientToken": "426bfd96-e720-46d3-95cd-014e3ef12bb6"
}
```

Pembaruan: (versi cocok; permintaan ini akan diterima)

```
{
 "state": {
 "desired": {
 "colors": [
 "BLUE"
]
 }
 },
 "version": 10
}
```

Keadaan akhir:

```
{
 "state": {
 "desired": {
 "colors": [
 "BLUE"
]
 }
 },
 "version": 11
}
```

## Mengambil dokumen bayangan

Anda dapat mengambil dokumen bayangan dengan menggunakan [GetThingShadow](#) API atau dengan berlangganan dan menerbitkan topik. [/dapatkan](#) Ini mengambil dokumen bayangan lengkap,



termasuk delta apa pun antara status `desired` dan `reported`. Prosedur untuk tugas ini sama apakah perangkat atau klien membuat permintaan.

Untuk mengambil dokumen bayangan dengan menggunakan protokol MQTT

1. Perangkat atau klien harus berlangganan topik MQTT ini sebelum memperbarui bayangan:
  - `$aws/things/thingName/shadow/name/shadowName/get/accepted`
  - `$aws/things/thingName/shadow/name/shadowName/get/rejected`
2. Perangkat atau klien menerbitkan pesan ke `$aws/things/thingName/shadow/name/shadowName/get` topik dengan badan pesan kosong.
3. Jika permintaan berhasil, AWS IoT menerbitkan pesan ke `$aws/things/thingName/shadow/name/shadowName/get/accepted` topik dengan [/dokumen status respons yang diterima](#) di badan pesan.
4. Jika permintaan tidak valid, AWS IoT menerbitkan pesan ke `$aws/things/thingName/shadow/name/shadowName/get/rejected` topik dengan [Dokumen respons kesalahan](#) di badan pesan.

Untuk mengambil dokumen bayangan dengan menggunakan REST API

1. Perangkat atau klien memanggil [GetThingShadow](#) API dengan badan pesan kosong.
2. Jika permintaan valid, AWS IoT mengembalikan kode respons sukses HTTP dengan dokumen [/dokumen status respons yang diterima](#) bayangan sebagai badan pesan responsnya.
3. Jika permintaan tidak valid, AWS IoT mengembalikan kode respons kesalahan HTTP [Dokumen respons kesalahan](#) sebagai badan pesan responsnya.

## Menghapus data bayangan

Ada dua cara untuk menghapus data bayangan: Anda dapat menghapus properti tertentu dalam dokumen bayangan dan Anda dapat menghapus bayangan sepenuhnya.

- Untuk menghapus properti tertentu dari bayangan, perbarui bayangan; namun tetapkan nilai properti yang ingin Anda hapus `null`. Bidang dengan nilai `null` dihapus dari dokumen bayangan.
- Untuk menghapus seluruh bayangan, gunakan [DeleteThingShadow](#) API atau publikasikan ke [/delete](#) topik.

**Note**

Menghapus bayangan tidak mengatur ulang nomor versinya ke nol sekaligus. Ini akan diatur ulang ke nol setelah 48 jam.

## Menghapus properti dari dokumen bayangan

Untuk menghapus properti dari bayangan dengan menggunakan protokol MQTT

1. Perangkat atau klien harus memiliki dokumen bayangan saat ini sehingga dapat mengidentifikasi properti yang akan diubah. Lihat [Mengambil dokumen bayangan](#) untuk informasi tentang cara mendapatkan dokumen bayangan saat ini.
2. Perangkat atau klien berlangganan topik MQTT ini:
  - `$aws/things/thingName/shadow/name/shadowName/update/accepted`
  - `$aws/things/thingName/shadow/name/shadowName/update/rejected`
3. Perangkat atau klien menerbitkan topik `$aws/things/thingName/shadow/name/shadowName/update` permintaan dengan dokumen status yang menetapkan `null` nilai ke properti bayangan yang akan dihapus. Hanya properti yang akan diubah yang perlu dimasukkan dalam dokumen. Ini adalah contoh dokumen yang menghapus engine properti.

```
{
 "state": {
 "desired": {
 "engine": null
 }
 }
}
```

4. Jika permintaan pembaruan valid, AWS IoT hapus properti yang ditentukan dalam bayangan dan menerbitkan pesan dengan `$aws/things/thingName/shadow/name/shadowName/update/accepted` topik dengan dokumen [dokumen status respons yang diterima](#) bayangan di badan pesan.
5. Jika permintaan pembaruan tidak valid, AWS IoT menerbitkan pesan dengan `$aws/things/thingName/shadow/name/shadowName/update/rejected` topik dengan dokumen [Dokumen respons kesalahan](#) bayangan yang menjelaskan kesalahan.

Untuk menghapus properti dari bayangan dengan menggunakan REST API

1. Perangkat atau klien memanggil [UpdateThingShadow](#) API dengan [Minta dokumen negara](#) yang menetapkan null nilai ke properti bayangan untuk dihapus. Sertakan hanya properti yang ingin Anda hapus dalam dokumen. Ini adalah contoh dokumen yang menghapus engine properti.

```
{
 "state": {
 "desired": {
 "engine": null
 }
 }
}
```

2. Jika permintaan itu valid, AWS IoT mengembalikan kode respons sukses HTTP dan dokumen [/ dokumen status respons yang diterima](#) bayangan sebagai badan pesan responsnya.
3. Jika permintaan tidak valid, AWS IoT mengembalikan kode respons kesalahan HTTP [Dokumen respons kesalahan](#) sebagai badan pesan responsnya.

## Menghapus bayangan

Berikut ini adalah beberapa pertimbangan saat menghapus bayangan perangkat.

- Menyetel status bayangan perangkat ke null tidak menghapus bayangan. Versi bayangan akan bertambah pada pembaruan berikutnya.
- Menghapus bayangan perangkat tidak menghapus objek benda. Menghapus objek benda tidak menghapus bayangan perangkat yang sesuai.
- Menghapus bayangan tidak mengatur ulang nomor versinya ke nol sekaligus. Ini akan diatur ulang ke nol setelah 48 jam.

Untuk menghapus bayangan dengan menggunakan protokol MQTT

1. Perangkat atau klien berlangganan topik MQTT ini:
  - \$aws/things/*thingName*/shadow/name/*shadowName*/delete/accepted
  - \$aws/things/*thingName*/shadow/name/*shadowName*/delete/rejected

2. Perangkat atau klien menerbitkan buffer pesan `$aws/things/thingName/shadow/name/shadowName/delete` dengan kosong.
3. Jika permintaan hapus valid, AWS IoT hapus bayangan dan publikasikan pesan dengan `$aws/things/thingName/shadow/name/shadowName/delete/accepted` topik dan dokumen [/dokumen status respons yang diterima](#) bayangan yang disingkat di badan pesan. Ini adalah contoh pesan hapus yang diterima:

```
{
 "version": 4,
 "timestamp": 1591057529
}
```

4. Jika permintaan pembaruan tidak valid, AWS IoT menerbitkan pesan dengan `$aws/things/thingName/shadow/name/shadowName/delete/rejected` topik dengan dokumen [Dokumen respons kesalahan](#) bayangan yang menjelaskan kesalahan.

Untuk menghapus bayangan dengan menggunakan REST API

1. Perangkat atau klien memanggil [DeleteThingShadow](#) API dengan buffer pesan kosong.
2. Jika permintaan itu valid, AWS IoT mengembalikan kode respons sukses HTTP dan [/dokumen status respons yang diterima](#) dan dokumen [/dokumen status respons yang diterima](#) bayangan disingkat di badan pesan. Ini adalah contoh pesan hapus yang diterima:

```
{
 "version": 4,
 "timestamp": 1591057529
}
```

3. Jika permintaan tidak valid, AWS IoT mengembalikan kode respons kesalahan HTTP [Dokumen respons kesalahan](#) sebagai badan pesan responsnya.

## Device Shadow REST API

Bayangan mengekspos URI berikut untuk memperbarui informasi status:

```
https://account-specific-prefix-ats.iot.region.amazonaws.com/things/thingName/shadow
```

Titik akhir khusus untuk Anda Akun AWS. Untuk menemukan titik akhir Anda, Anda dapat:

- Gunakan perintah [describe-endpoint](#) dari AWS CLI
- Gunakan pengaturan AWS IoT konsol. Di Pengaturan, titik akhir tercantum di bawah titik akhir Kustom
- Gunakan halaman detail hal AWS IoT konsol. Di konsol:
  1. Buka Kelola dan di bawah Kelola, pilih Things.
  2. Dalam daftar hal-hal, pilih hal yang ingin Anda dapatkan URI endpoint.
  3. Pilih tab Device Shadows dan pilih bayangan Anda. Anda dapat melihat URI titik akhir di bagian URL Device Shadow pada halaman detail Device Shadow.

Format titik akhir adalah sebagai berikut:

```
identifier.iot.region.amazonaws.com
```

Shadow REST API mengikuti protokol HTTPS/pemetaan port yang sama seperti yang dijelaskan dalam [Protokol komunikasi perangkat](#)

#### Note

Untuk menggunakan API, Anda harus menggunakan `iotdevicegateway` sebagai nama layanan untuk otentikasi. Untuk informasi selengkapnya, lihat [IoT DataPlane](#).

#### Tindakan API

- [GetThingShadow](#)
- [UpdateThingShadow](#)
- [DeleteThingShadow](#)
- [ListNamedShadowsForThing](#)

Anda juga dapat menggunakan API untuk membuat bayangan bernama dengan menyediakan `name=shadowName` sebagai bagian dari parameter kueri API.

## GetThingShadow

Mendapat bayangan untuk hal yang ditentukan.

Dokumen status respons mencakup delta antara negara `reported` bagian `desired` dan negara bagian.

## Permintaan

Permintaan termasuk header HTTP standar ditambah URI berikut:

```
HTTP GET https://endpoint/things/thingName/shadow?name=shadowName
Request body: (none)
```

Parameter name kueri tidak diperlukan untuk bayangan (klasik) yang tidak disebutkan namanya.

## Respons

Setelah berhasil, respon mencakup header HTTP standar ditambah kode dan tubuh berikut:

```
HTTP 200
Response Body: response state document
```

Untuk informasi selengkapnya, lihat [Contoh Dokumen Status Respons](#).

## Otorisasi

Mengambil bayangan memerlukan kebijakan yang memungkinkan pemanggil untuk melakukan tindakan. `iot:GetThingShadow` Layanan Device Shadow menerima dua bentuk otentikasi: Signature Version 4 dengan kredensial IAM atau otentikasi timbal balik TLS dengan sertifikat klien.

Berikut ini adalah contoh kebijakan yang memungkinkan pemanggil untuk mengambil bayangan perangkat:

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": "iot:GetThingShadow",
 "Resource": [
 "arn:aws:iot:region:account:thing/thing"
]
 }
]
}
```

```
}
```

## UpdateThingShadow

Memperbarui bayangan untuk hal yang ditentukan.

Pembaruan hanya memengaruhi bidang yang ditentukan dalam dokumen status permintaan. Bidang apa pun dengan nilai `null` dihapus dari bayangan perangkat.

### Permintaan

Permintaan mencakup header HTTP standar ditambah URI dan isi berikut:

```
HTTP POST https://endpoint/things/thingName/shadow?name=shadowName
Request body: request state document
```

Parameter name kueri tidak diperlukan untuk bayangan (klasik) yang tidak disebutkan namanya.

Untuk informasi selengkapnya, lihat [Contoh Permintaan Dokumen Negara](#).

### Respons

Setelah berhasil, respon mencakup header HTTP standar ditambah kode dan tubuh berikut:

```
HTTP 200
Response body: response state document
```

Untuk informasi selengkapnya, lihat [Contoh Dokumen Status Respons](#).

### Otorisasi

Memperbarui bayangan memerlukan kebijakan yang memungkinkan pemanggil untuk melakukan `iot:UpdateThingShadow` tindakan. Layanan Device Shadow menerima dua bentuk otentikasi: Signature Version 4 dengan kredensial IAM atau otentikasi timbal balik TLS dengan sertifikat klien.

Berikut ini adalah contoh kebijakan yang memungkinkan pemanggil memperbarui bayangan perangkat:

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

```

```
{
 "Effect": "Allow",
 "Action": "iot:UpdateThingShadow",
 "Resource": [
 "arn:aws:iot:region:account:thing/thing"
]
}
```

## DeleteThingShadow

Menghapus bayangan untuk objek yang ditentukan.

### Permintaan

Permintaan termasuk header HTTP standar ditambah URI berikut:

```
HTTP DELETE https://endpoint/things/thingName/shadow?name=shadowName
Request body: (none)
```

Parameter name kueri tidak diperlukan untuk bayangan (klasik) yang tidak disebutkan namanya.

### Respons

Setelah berhasil, respon mencakup header HTTP standar ditambah kode dan tubuh berikut:

```
HTTP 200
Response body: Empty response state document
```

Perhatikan bahwa menghapus bayangan tidak mengatur ulang nomor versinya ke 0.

### Otorisasi

Menghapus bayangan perangkat memerlukan kebijakan yang memungkinkan pemanggil melakukan tindakan. `iot:DeleteThingShadow` Layanan Device Shadow menerima dua bentuk otentikasi: Signature Version 4 dengan kredensial IAM atau otentikasi timbal balik TLS dengan sertifikat klien.

Berikut ini adalah contoh kebijakan yang memungkinkan pemanggil menghapus bayangan perangkat:

```
{
```



```
"Version": "2012-10-17",
"Statement": [
 {
 "Effect": "Allow",
 "Action": "iot:DeleteThingShadow",
 "Resource": [
 "arn:aws:iot:region:account:thing/thing"
]
 }
]
```

## ListNamedShadowsForThing

Daftar bayangan untuk hal yang ditentukan.

### Permintaan

Permintaan termasuk header HTTP standar ditambah URI berikut:

```
HTTP GET /api/things/shadow/ListNamedShadowsForThing/thingName?
nextToken=nextToken&pageSize=pageSize
Request body: (none)
```

### nextToken

Token untuk mengambil set hasil berikutnya.

Nilai ini dikembalikan pada hasil berhalaman dan digunakan dalam panggilan yang mengembalikan halaman berikutnya.

### pageSize

Jumlah nama bayangan yang akan dikembalikan di setiap panggilan. Lihat juga nextToken.

### thingName

Nama objek perangkat yang bayangan bernama didaftar untuknya.

### Respons

Setelah berhasil, responsnya mencakup header HTTP standar ditambah kode respons berikut dan [aDokumen respons daftar nama bayangan](#).

**Note**

Bayangan yang tidak disebutkan namanya (klasik) tidak muncul dalam daftar ini. Responsnya adalah daftar kosong jika Anda hanya memiliki bayangan klasik atau jika yang `thingName` Anda tentukan tidak ada.

HTTP 200

Response body: *Shadow name list document*

## Otorisasi

Membuat daftar bayangan perangkat memerlukan kebijakan yang memungkinkan pemanggil untuk melakukan `iot:ListNamedShadowsForThing` tindakan. Layanan Device Shadow menerima dua bentuk otentikasi: Signature Version 4 dengan kredensial IAM atau otentikasi timbal balik TLS dengan sertifikat klien.

Berikut ini adalah contoh kebijakan yang memungkinkan pemanggil untuk membuat daftar bayangan bernama sesuatu:

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": "iot:ListNamedShadowsForThing",
 "Resource": [
 "arn:aws:iot:region:account:thing/thing"
]
 }
]
}
```

## Topik Device Shadow MQTT

Layanan Device Shadow menggunakan topik MQTT yang dicadangkan untuk memungkinkan perangkat dan aplikasi mendapatkan, memperbarui, atau menghapus informasi status untuk perangkat (bayangan).

Menerbitkan dan berlangganan topik bayangan memerlukan otorisasi berbasis topik. AWS IoT berhak untuk menambahkan topik baru ke struktur topik yang ada. Untuk alasan ini, kami menyarankan Anda menghindari langganan wild card ke topik bayangan. Misalnya, hindari berlangganan filter topik seperti `$aws/things/thingName/shadow/#` karena jumlah topik yang cocok dengan filter topik ini mungkin meningkat saat AWS IoT memperkenalkan topik bayangan baru. Untuk contoh pesan yang dipublikasikan pada topik ini lihat [Berinteraksi dengan bayangan](#).

Bayangan dapat dinamai atau tidak disebutkan namanya (klasik). Topik yang digunakan oleh masing-masing hanya berbeda dalam awalan topik. Tabel ini menunjukkan awalan topik yang digunakan oleh setiap jenis bayangan.

| Nilai <i>ShadowTopicPrefix</i>                                              | Jenis bayangan               |
|-----------------------------------------------------------------------------|------------------------------|
| <code>\$aws/things/ <i>thingName</i> /shadow</code>                         | Bayangan tanpa nama (klasik) |
| <code>\$aws/things/ <i>thingName</i> /shadow/name/ <i>shadowName</i></code> | Bernama bayangan             |

Untuk membuat topik lengkap, pilih jenis bayangan yang ingin Anda rujuk, ganti, dan *shadowName* jika berlaku *thingName*, dengan nilai yang sesuai, lalu tambahkan dengan rintisan topik seperti yang ditunjukkan pada bagian berikut. *ShadowTopicPrefix*

Berikut ini adalah topik MQTT yang digunakan untuk berinteraksi dengan bayangan.

#### Topik

- [/dapatkan](#)
- [/dapatkan/diterima](#)
- [/dapatkan/ditolak](#)
- [/perbarui](#)
- [/perbarui/delta](#)
- [/perbarui/diterima](#)
- [/pembaruan/dokumen](#)
- [/perbarui/ditolak](#)
- [/delete](#)
- [/hapus/diterima](#)

- [/hapus/ditolak](#)

## /dapatkan

Publikasikan pesan kosong ke topik ini untuk mendapatkan bayangan perangkat:

```
ShadowTopicPrefix/get
```

AWS IoT merespons dengan menerbitkan salah satu [/dapatkan/diterima](#) atau [/dapatkan/ditolak](#).

### Contoh kebijakan

Berikut ini adalah contoh kebijakan yang diperlukan:

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "iot:Publish"
],
 "Resource": [
 "arn:aws:iot:region:account:topic/$aws/things/thingName/shadow/get"
]
 }
]
}
```

## /dapatkan/diterima

AWS IoT menerbitkan dokumen bayangan respons ke topik ini saat mengembalikan bayangan perangkat:

```
ShadowTopicPrefix/get/accepted
```

Untuk informasi selengkapnya, lihat [Dokumen negara respons](#).

### Contoh kebijakan

Berikut ini adalah contoh kebijakan yang diperlukan:

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "iot:Subscribe"
],
 "Resource": [
 "arn:aws:iot:region:account:topicfilter/$aws/things/thingName/shadow/get/
accepted"
]
 },
 {
 "Effect": "Allow",
 "Action": [
 "iot:Receive"
],
 "Resource": [
 "arn:aws:iot:region:account:topic/$aws/things/thingName/shadow/get/accepted"
]
 }
]
}
```

## /dapatkan/ditolak

AWS IoT menerbitkan dokumen respons kesalahan ke topik ini jika tidak dapat mengembalikan bayangan perangkat:

```
ShadowTopicPrefix/get/rejected
```

Untuk informasi selengkapnya, lihat [Dokumen respons kesalahan](#).

## Contoh kebijakan

Berikut ini adalah contoh kebijakan yang diperlukan:

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

```

 "Effect": "Allow",
 "Action": [
 "iot:Subscribe"
],
 "Resource": [
 "arn:aws:iot:region:account:topicfilter/$aws/things/thingName/shadow/get/
rejected"
]
 },
 {
 "Action": [
 "iot:Receive"
],
 "Resource": [
 "arn:aws:iot:region:account:topic/$aws/things/thingName/shadow/get/rejected"
]
 }
]
}

```

## /perbarui

Publikasikan dokumen status permintaan ke topik ini untuk memperbarui bayangan perangkat:

```
ShadowTopicPrefix/update
```

Badan pesan berisi [dokumen status permintaan sebagian](#).

Klien yang mencoba memperbarui status perangkat akan mengirim dokumen status permintaan JSON dengan `desired` properti seperti ini:

```

{
 "state": {
 "desired": {
 "color": "red",
 "power": "on"
 }
 }
}

```

Perangkat yang memperbarui bayangannya akan mengirim dokumen status permintaan JSON dengan `reported` properti, seperti ini:

```
{
 "state": {
 "reported": {
 "color": "red",
 "power": "on"
 }
 }
}
```

AWS IoT merespons dengan menerbitkan salah satu [/perbarui/diterima](#) atau [/perbarui/ditolak](#).

## Contoh kebijakan

Berikut ini adalah contoh kebijakan yang diperlukan:

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "iot:Publish"
],
 "Resource": [
 "arn:aws:iot:region:account:topic/$aws/things/thingName/shadow/update"
]
 }
]
}
```

## /perbarui/delta

AWS IoT menerbitkan dokumen status respons ke topik ini saat menerima perubahan untuk bayangan perangkat, dan dokumen status permintaan berisi nilai `desired` dan `reported` status yang berbeda:

```
ShadowTopicPrefix/update/delta
```

Buffer pesan berisi file. [/dokumen status respons delta](#)

## Detail isi pesan

- Pesan yang dipublikasikan hanya update/delta mencakup atribut yang diinginkan yang berbeda antara bagian `desired` dan `reported` bagian. Ini berisi semua atribut ini, terlepas dari apakah atribut ini terkandung dalam pesan pembaruan saat ini atau sudah disimpan di AWS IoT. Atribut yang tidak berbeda antara `reported` bagian `desired` dan bagian tidak termasuk.
- Jika atribut ada di `reported` bagian tetapi tidak memiliki padanan di `desired` bagian, itu tidak termasuk.
- Jika atribut ada di `desired` bagian tetapi tidak memiliki padanan di `reported` bagian, itu disertakan.
- Jika atribut dihapus dari `reported` bagian tetapi masih ada di `desired` bagian, itu disertakan.

## Contoh kebijakan

Berikut ini adalah contoh kebijakan yang diperlukan:

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "iot:Subscribe"
],
 "Resource": [
 "arn:aws:iot:region:account:topicfilter/$aws/things/thingName/shadow/update/
delta"
]
 },
 {
 "Effect": "Allow",
 "Action": [
 "iot:Receive"
],
 "Resource": [
 "arn:aws:iot:region:account:topic/$aws/things/thingName/shadow/update/delta"
]
 }
]
}
```



## /perbarui/diterima

AWS IoT menerbitkan dokumen status respons ke topik ini saat menerima perubahan untuk bayangan perangkat:

```
ShadowTopicPrefix/update/accepted
```

Buffer pesan berisi file. [/dokumen status respons yang diterima](#)

## Contoh kebijakan

Berikut ini adalah contoh kebijakan yang diperlukan:

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "iot:Subscribe"
],
 "Resource": [
 "arn:aws:iot:region:account:topicfilter/$aws/things/thingName/shadow/update/
accepted"
]
 },
 {
 "Effect": "Allow",
 "Action": [
 "iot:Receive"
],
 "Resource": [
 "arn:aws:iot:region:account:topic/$aws/things/thingName/shadow/update/accepted"
]
 }
]
}
```

## /pembaruan/dokumen

AWS IoT menerbitkan dokumen status ke topik ini setiap kali pembaruan bayangan berhasil dilakukan:

```
ShadowTopicPrefix/update/documents
```

Badan pesan berisi [a/dokumen dokumen status respons](#).

## Contoh kebijakan

Berikut ini adalah contoh kebijakan yang diperlukan:

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "iot:Subscribe"
],
 "Resource": [
 "arn:aws:iot:region:account:topicfilter/$aws/things/thingName/shadow/update/
documents"
]
 },
 {
 "Effect": "Allow",
 "Action": [
 "iot:Receive"
],
 "Resource": [
 "arn:aws:iot:region:account:topic/$aws/things/thingName/shadow/update/
documents"
]
 }
]
}
```

## /perbarui/ditolak

AWS IoT menerbitkan dokumen respons kesalahan ke topik ini saat menolak perubahan bayangan perangkat:

```
ShadowTopicPrefix/update/rejected
```

Badan pesan berisi [Dokumen respons kesalahan](#).

## Contoh kebijakan

Berikut ini adalah contoh kebijakan yang diperlukan:

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "iot:Subscribe"
],
 "Resource": [
 "arn:aws:iot:region:account:topicfilter/$aws/things/thingName/shadow/update/
rejected"
]
 },
 {
 "Effect": "Allow",
 "Action": [
 "iot:Receive"
],
 "Resource": [
 "arn:aws:iot:region:account:topic/$aws/things/thingName/shadow/update/rejected"
]
 }
]
}
```

## /delete

Untuk menghapus bayangan perangkat, publikasikan pesan kosong ke topik hapus:

```
ShadowTopicPrefix/delete
```

Isi pesan diabaikan.

Perhatikan bahwa menghapus bayangan tidak mengatur ulang nomor versinya ke 0.

AWS IoT merespons dengan menerbitkan salah satu [/hapus/diterima](#) atau [/hapus/ditolak](#).

## Contoh kebijakan

Berikut ini adalah contoh kebijakan yang diperlukan:

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "iot:Publish"
],
 "Resource": [
 "arn:aws:iot:region:account:topic/$aws/things/thingName/shadow/delete"
]
 }
]
}
```

## /hapus/diterima

AWS IoT menerbitkan pesan ke topik ini saat bayangan perangkat dihapus:

```
ShadowTopicPrefix/delete/accepted
```

## Contoh kebijakan

Berikut ini adalah contoh kebijakan yang diperlukan:

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "iot:Subscribe"
],
 "Resource": [
 "arn:aws:iot:region:account:topicfilter/$aws/things/thingName/shadow/delete/accepted"
]
 },
],
}
```

```

{
 "Effect": "Allow",
 "Action": [
 "iot:Receive"
],
 "Resource": [
 "arn:aws:iot:region:account:topic/$aws/things/thingName/shadow/delete/accepted"
]
}
]
}

```

## /hapus/ditolak

AWS IoT menerbitkan dokumen respons kesalahan ke topik ini jika tidak dapat menghapus bayangan perangkat:

*ShadowTopicPrefix*/delete/rejected

Badan pesan berisi [Dokumen respons kesalahan](#).

## Contoh kebijakan

Berikut ini adalah contoh kebijakan yang diperlukan:

```

{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "iot:Subscribe"
],
 "Resource": [
 "arn:aws:iot:region:account:topicfilter/$aws/things/thingName/shadow/delete/rejected"
]
 },
 {
 "Effect": "Allow",
 "Action": [
 "iot:Receive"
]
 }
]
}

```

```
],
 "Resource": [
 "arn:aws:iot:region:account:topic/$aws/things/thingName/shadow/delete/rejected"
]
 }
]
```

## Dokumen layanan Device Shadow

Layanan Device Shadow menghormati semua aturan spesifikasi JSON. Nilai, objek, dan array disimpan dalam dokumen bayangan perangkat.

### Daftar Isi

- [Contoh dokumen bayangan](#)
- [Properti dokumen](#)
- [Negara bagian Delta](#)
- [Dokumen bayangan versi](#)
- [Token klien dalam dokumen bayangan](#)
- [Properti dokumen bayangan kosong](#)
- [Nilai array dalam dokumen bayangan](#)

## Contoh dokumen bayangan

Layanan Device Shadow menggunakan dokumen-dokumen ini dalam operasi UPDATE, GET, dan DELETE menggunakan [REST API](#) atau [MQTT Pub/Sub Messages](#).

### Contoh-contoh

- [Minta dokumen negara](#)
- [Dokumen negara respons](#)
- [Dokumen respons kesalahan](#)
- [Dokumen respons daftar nama bayangan](#)

## Minta dokumen negara

Dokumen status permintaan memiliki format berikut:

```
{
 "state": {
 "desired": {
 "attribute1": integer2,
 "attribute2": "string2",
 ...
 "attributeN": boolean2
 },
 "reported": {
 "attribute1": integer1,
 "attribute2": "string1",
 ...
 "attributeN": boolean1
 }
 },
 "clientToken": "token",
 "version": version
}
```

- **state**— Pembaruan hanya memengaruhi bidang yang ditentukan. Biasanya, Anda akan menggunakan properti `desired` atau `reported` properti, tetapi tidak keduanya dalam permintaan yang sama.
  - **desired**— Properti dan nilai status yang diminta untuk diperbarui di perangkat.
  - **reported**— Properti dan nilai status yang dilaporkan oleh perangkat.
- **clientToken**— Jika digunakan, Anda dapat mencocokkan permintaan dan respons yang sesuai dengan token klien.
- **version**— Jika digunakan, layanan Device Shadow memproses pembaruan hanya jika versi yang ditentukan cocok dengan versi terbaru yang dimilikinya.

## Dokumen negara respons

Dokumen status respons memiliki format berikut tergantung pada jenis respons.

/dokumen status respons yang diterima

```
{
 "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
}

```

### /dokumen status respons delta

```

{
 "state": {
 "attribute1": integer2,
 "attribute2": "string2",
 ...
 "attributeN": boolean2
 },
 "metadata": {
 "attribute1": {
 "timestamp": timestamp
 },
 "attribute2": {
 "timestamp": timestamp
 },
 ...
 "attributeN": {
 "timestamp": timestamp
 }
 }
}

```



```
 }
 },
 "timestamp": timestamp,
 "clientToken": "token",
 "version": version
}
```

## /dokumen dokumen status respons

```
{
 "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"
}

```

### Properti dokumen status respons

- **previous**— Setelah pembaruan berhasil, berisi objek sebelum pembaruan. `state`
- **current**— Setelah pembaruan berhasil, berisi objek setelah pembaruan. `state`
- **state**
  - **reported**— Hadir hanya jika sesuatu melaporkan data apa pun di `reported` bagian dan hanya berisi bidang yang ada di dokumen negara permintaan.
  - **desired**— Hadir hanya jika perangkat melaporkan data apa pun di `desired` bagian dan hanya berisi bidang yang ada di dokumen negara permintaan.
  - **delta**— Hadir hanya jika `desired` data berbeda dari `reported` data bayangan saat ini.
- **metadata**— Berisi stempel waktu untuk setiap atribut di `reported` bagian `desired` dan sehingga Anda dapat menentukan kapan status diperbarui.
- **timestamp**— Tanggal dan waktu Epoch respons dihasilkan oleh AWS IoT.
- **clientToken**— Hadir hanya jika token klien digunakan saat menerbitkan JSON yang valid ke /`update` topik tersebut.
- **version**— Versi dokumen saat ini untuk bayangan perangkat yang dibagikan AWS IoT. Ini meningkat satu kali lipat dari versi dokumen sebelumnya.

### Dokumen respons kesalahan

Dokumen respons kesalahan memiliki format berikut:

```

{
 "code": error-code,

```

```
"message": "error-message",
"timestamp": timestamp,
"clientToken": "token"
}
```

- `code`— Kode respons HTTP yang menunjukkan jenis kesalahan.
- `message`— Pesan teks yang memberikan informasi tambahan.
- `timestamp`— Tanggal dan waktu respons dihasilkan oleh AWS IoT. Properti ini tidak ada di semua dokumen respons kesalahan.
- `clientToken`— Hadir hanya jika token klien digunakan dalam pesan yang diterbitkan.

Untuk informasi selengkapnya, lihat [Pesan kesalahan Device Shadow](#).

## Dokumen respons daftar nama bayangan

Dokumen respons daftar nama bayangan memiliki format berikut:

```
{
 "results": [
 "shadowName-1",
 "shadowName-2",
 "shadowName-3",
 "shadowName-n"
],
 "nextToken": "nextToken",
 "timestamp": timestamp
}
```

- `results`— Array nama bayangan.
- `nextToken`— Nilai token yang digunakan dalam permintaan halaman untuk mendapatkan halaman berikutnya dalam urutan. Properti ini tidak ada ketika tidak ada lagi nama bayangan untuk dikembalikan.
- `timestamp`— Tanggal dan waktu respons dihasilkan oleh AWS IoT.

## Properti dokumen

Dokumen bayangan perangkat memiliki properti berikut:

## state

### desired

Keadaan perangkat yang diinginkan. Aplikasi dapat menulis ke bagian dokumen ini untuk memperbarui status perangkat secara langsung tanpa harus terhubung ke sana.

### reported

Status perangkat yang dilaporkan. Perangkat menulis ke bagian dokumen ini untuk melaporkan keadaan baru mereka. Aplikasi membaca bagian dokumen ini untuk menentukan status perangkat yang dilaporkan terakhir.

## metadata

Informasi tentang data yang disimpan di `state` bagian dokumen. Ini termasuk stempel waktu, dalam waktu Epoch, untuk setiap atribut di `state` bagian, yang memungkinkan Anda untuk menentukan kapan mereka diperbarui.

### Note

Metadata tidak berkontribusi pada ukuran dokumen untuk batas layanan atau harga. Untuk informasi selengkapnya, lihat [Batas AWS IoT Layanan](#).

## timestamp

Menunjukkan kapan pesan dikirim oleh AWS IoT. Dengan menggunakan stempel waktu dalam pesan dan stempel waktu untuk atribut individual di `reported` bagian `desired` atau, perangkat dapat menentukan usia properti, meskipun perangkat tidak memiliki jam internal.

## clientToken

String unik untuk perangkat yang memungkinkan Anda mengaitkan respons dengan permintaan di lingkungan MQTT.

## version

Versi dokumen. Setiap kali dokumen diperbarui, nomor versi ini bertambah. Digunakan untuk memastikan versi dokumen yang diperbarui adalah yang terbaru.

Untuk informasi selengkapnya, lihat [Contoh dokumen bayangan](#).

## Negara bagian Delta

Delta state adalah jenis virtual state yang berisi perbedaan antara negara bagian `desired` dan `reported` negara bagian. Bidang di `desired` bagian yang tidak ada di `reported` bagian termasuk dalam delta. Bidang yang ada di `reported` bagian dan bukan di `desired` bagian tidak termasuk dalam delta. Delta berisi metadata, dan nilainya sama dengan metadata di lapangan. `desired` Sebagai contoh:

```
{
 "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
 }
}
```

Ketika objek bersarang berbeda, delta berisi jalur sampai ke root.

```
{
 "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
}
```

Layanan Device Shadow menghitung delta dengan mengulangi setiap bidang dalam `desired` status dan membandingkannya dengan status. `reported`

Array diperlakukan seperti nilai. Jika array di `desired` bagian tidak cocok dengan array di `reported` bagian, maka seluruh array yang diinginkan disalin ke delta.

## Dokumen bayangan versi

Layanan Device Shadow mendukung pembuatan versi pada setiap pesan pembaruan, baik permintaan maupun respons. Ini berarti bahwa dengan setiap pembaruan bayangan, versi dokumen JSON bertambah. Ini memastikan dua hal:

- Klien dapat menerima kesalahan jika mencoba menimpa bayangan menggunakan nomor versi yang lebih lama. Klien diberi tahu bahwa ia harus melakukan sinkronisasi ulang sebelum dapat memperbarui bayangan perangkat.
- Klien dapat memutuskan untuk tidak bertindak atas pesan yang diterima jika pesan memiliki versi yang lebih rendah dari versi yang disimpan oleh klien.

Klien dapat melewati pencocokan versi dengan tidak menyertakan versi dalam dokumen bayangan.

## Token klien dalam dokumen bayangan

Anda dapat menggunakan token klien dengan pesan berbasis MQTT untuk memverifikasi token klien yang sama terkandung dalam permintaan dan respons permintaan. Ini memastikan respons dan permintaan terkait.

### Note

Token klien tidak boleh lebih dari 64 byte. Token klien yang lebih panjang dari 64 byte menyebabkan respons 400 (Permintaan Buruk) dan pesan kesalahan ClientToken Tidak Valid.

## Properti dokumen bayangan kosong

`desired` Properti `reported` dan dalam dokumen bayangan dapat kosong atau dihilangkan ketika tidak berlaku untuk status bayangan saat ini. Misalnya, dokumen bayangan berisi `desired` properti



hanya jika memiliki status yang diinginkan. Berikut ini adalah contoh valid dari dokumen negara tanpa `desired` properti:

```
{
 "reported" : { "temp": 55 }
}
```

`reportedProperti` juga bisa kosong, seperti jika bayangan belum diperbarui oleh perangkat:

```
{
 "desired" : { "color" : "RED" }
}
```

Jika pembaruan menyebabkan `reported` properti `desired` atau menjadi nol, itu dihapus dari dokumen. Berikut ini menunjukkan cara menghapus `desired` properti dengan menyetelnya ke `null`. Anda dapat melakukan ini ketika perangkat memperbarui statusnya, misalnya.

```
{
 "state": {
 "reported": {
 "color": "red"
 },
 "desired": null
 }
}
```

Dokumen bayangan juga dapat memiliki keduanya `desired` atau `reported` properti, membuat dokumen bayangan kosong. Ini adalah contoh dokumen bayangan kosong namun valid.

```
{
}
```

## Nilai array dalam dokumen bayangan

Shadows mendukung array, tetapi memperlakukannya sebagai nilai normal karena pembaruan ke array menggantikan seluruh array. Hal ini tidak mungkin untuk memperbarui bagian dari array.

Keadaan awal:

```
{
```

```
"desired" : { "colors" : ["RED", "GREEN", "BLUE"] }
}
```

Memperbarui:

```
{
 "desired" : { "colors" : ["RED"] }
}
```

Keadaan akhir:

```
{
 "desired" : { "colors" : ["RED"] }
}
```


Array tidak dapat memiliki nilai null. Misalnya, array berikut tidak valid dan akan ditolak.

```
{
 "desired" : {
 "colors" : [null, "RED", "GREEN"]
 }
}
```

## Pesan kesalahan Device Shadow

Layanan Device Shadow menerbitkan pesan tentang topik kesalahan (melalui MQTT) ketika upaya untuk mengubah dokumen status gagal. Pesan ini hanya dipancarkan sebagai tanggapan atas permintaan publikasi pada salah satu topik yang dicadangkan\$aws. Jika klien memperbarui dokumen menggunakan REST API, maka ia menerima kode kesalahan HTTP sebagai bagian dari responsnya, dan tidak ada pesan kesalahan MQTT yang dipancarkan.

| Kode kesalahan HTTP    | Pesan kesalahan                                                                                                                                                                                                      |
|------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 400 (Permintaan Buruk) | <ul style="list-style-type: none"> <li>• JSON tidak valid</li> <li>• Node yang diperlukan tidak ada: status</li> <li>• Node negara harus menjadi objek</li> <li>• Node yang diinginkan harus berupa objek</li> </ul> |

| Kode kesalahan HTTP                  | Pesan kesalahan                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|--------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                      | <ul style="list-style-type: none"> <li>• Node yang dilaporkan harus berupa objek</li> <li>• Versi tidak valid</li> <li>• ClientToken tidak valid</li> </ul> <div data-bbox="716 401 1508 621" style="border: 1px solid #add8e6; border-radius: 10px; padding: 10px; margin: 10px 0;"> <p> <b>Note</b><br/>Token klien yang lebih panjang dari 64 byte akan menyebabkan respons ini.</p> </div> <ul style="list-style-type: none"> <li>• JSON mengandung terlalu banyak tingkat bersarang; maksimum adalah 6</li> <li>• Status berisi node yang tidak valid</li> </ul> |
| 401 (Tidak Sah)                      | <ul style="list-style-type: none"> <li>• Tidak sah</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 403 (Terlarang)                      | <ul style="list-style-type: none"> <li>• Dilarang</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 404 (Tidak Ditemukan)                | <ul style="list-style-type: none"> <li>• Hal yang tidak ditemukan</li> <li>• Tidak ada bayangan dengan nama: <i>ShadoName</i></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 409 (Konflik)                        | <ul style="list-style-type: none"> <li>• Konflik versi</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 413 (Muatan Terlalu Besar)           | <ul style="list-style-type: none"> <li>• Muatan melebihi ukuran maksimum yang diizinkan</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 415 (Tipe Media yang Tidak Didukung) | <ul style="list-style-type: none"> <li>• Pengkodean terdokumentasi yang tidak didukung; pengkodean yang didukung adalah UTF-8</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 429 (Terlalu Banyak Permintaan)      | <ul style="list-style-type: none"> <li>• Layanan Device Shadow akan menghasilkan pesan kesalahan ini ketika ada lebih dari 10 permintaan dalam penerbangan pada satu koneksi. Permintaan dalam penerbangan adalah permintaan yang sedang berlangsung yang telah dimulai tetapi belum selesai.</li> </ul>                                                                                                                                                                                                                                                                                                                                               |
| 500 (Kesalahan Server Internal)      | <ul style="list-style-type: none"> <li>• Kegagalan layanan internal</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |

# Tugas

Gunakan AWS IoT Jobs untuk menentukan satu set operasi jarak jauh yang dapat dikirim dan dijalankan pada satu atau beberapa perangkat yang terhubung AWS IoT. Misalnya, Anda dapat menentukan pekerjaan yang menginstruksikan satu set perangkat untuk mengunduh dan menginstal aplikasi, menjalankan pembaruan firmware, reboot, memutar sertifikat, atau melakukan operasi pemecahan masalah jarak jauh.

## Mengakses pekerjaan AWS IoT

Anda dapat memulai dengan AWS IoT Jobs dengan menggunakan konsol atau AWS IoT Core API.

Menggunakan konsol

Masuk ke AWS Management Console, dan pergi ke AWS IoT konsol. Di panel navigasi, pilih Kelola, lalu pilih Pekerjaan. Anda dapat membuat dan mengelola pekerjaan dari bagian ini. Jika Anda ingin membuat dan mengelola template pekerjaan, di panel navigasi, pilih Templat Job. Untuk informasi selengkapnya, lihat [Buat dan kelola pekerjaan dengan menggunakan AWS Management Console](#).

Menggunakan API atau CLI

Anda dapat memulai dengan menggunakan operasi AWS IoT Core API. Untuk informasi lebih lanjut, lihat [Referensi API AWS IoT](#). AWS IoT Core API tempat AWS IoT pekerjaan dibangun didukung oleh AWS SDK. Untuk informasi selengkapnya, lihat [AWS SDK dan Kit Alat](#).

Anda dapat menggunakan perintah AWS CLI untuk menjalankan untuk membuat dan mengelola pekerjaan dan templat pekerjaan. Untuk informasi lebih lanjut, lihat [AWS IoT referensi CLI](#).

## AWS IoT Lowongan Kerja Wilayah dan titik akhir

AWS IoT Jobs mendukung titik akhir API bidang kontrol dan bidang data yang khusus untuk Anda Wilayah AWS. Titik akhir API bidang data khusus untuk Anda Akun AWS dan Wilayah AWS. Untuk informasi selengkapnya tentang titik akhir AWS IoT Pekerjaan, lihat [AWS IoT Device Management - titik akhir data pekerjaan di Referensi AWS](#) Umum.

## Apa itu operasi jarak jauh?

Operasi jarak jauh adalah setiap pembaruan atau tindakan yang dapat Anda lakukan pada perangkat fisik, perangkat virtual, atau titik akhir yang dapat dilakukan dari jarak jauh tanpa perlu kehadiran

fisik operator atau teknisi. Operasi jarak jauh dilakukan menggunakan pembaruan over-the-air (OTA) sehingga perangkat Anda tidak harus hadir secara fisik. Mengelola armada perangkat Anda di AWS Cloud memungkinkan Anda untuk melakukan operasi jarak jauh pada perangkat Anda ketika mereka terdaftar AWS IoT Core.

AWS IoT Device Management Jobs menawarkan pendekatan skalabel untuk melakukan tindakan jarak jauh pada perangkat Anda yang AWS IoT Core terdaftar. Pekerjaan dibuat di AWS Cloud dan didorong ke semua perangkat yang ditargetkan menggunakan pembaruan OTA melalui protokol MQTT atau HTTP.

AWS IoT Device Management Jobs memberi Anda kemampuan untuk melakukan operasi jarak jauh seperti reset pabrik, reboot perangkat, dan pembaruan OTA perangkat lunak dengan cara yang aman, terukur, dan lebih hemat biaya.

Untuk informasi lebih lanjut tentang AWS IoT Core, lihat [Apa itu AWS IoT?](#).

Untuk informasi lebih lanjut tentang AWS IoT Device Management Pekerjaan, lihat [Apa itu AWS IoT Jobs?](#).

## Manfaat menggunakan AWS IoT Device Management Jobs untuk operasi jarak jauh

Menggunakan AWS IoT Device Management Jobs untuk melakukan operasi jarak jauh Anda menyederhanakan pengelolaan armada perangkat Anda. Daftar berikut menyoroti beberapa manfaat utama menggunakan AWS IoT Device Management Jobs untuk melakukan operasi jarak jauh Anda:

- Integrasi yang mulus dengan yang lain Layanan AWS
  - AWS IoT Device Management Jobs terintegrasi erat dengan nilai tambah dan fitur Layanan AWS berikut:
    - Amazon S3: Simpan petunjuk pengoperasian jarak jauh di bucket Amazon S3 yang aman tempat Anda mengontrol izin akses untuk konten tersebut. Menggunakan bucket Amazon S3 memberikan solusi penyimpanan yang dapat diskalakan dan tahan lama yang terintegrasi secara native dengan Katalog Paket Perangkat Lunak AWS IoT Manajemen Perangkat Lunak yang memungkinkan AWS IoT Device Management Pekerjaan untuk mereferensikan dan mengganti petunjuk pembaruan. Untuk informasi selengkapnya, lihat [Apa itu Amazon S3?](#) .
    - Amazon CloudWatch: Memantau dan mencatat status implementasi operasi jarak jauh dari pelaksanaan pekerjaan untuk setiap perangkat selain aktivitas perangkat lain untuk melacak dan menganalisis kinerja pekerjaan secara keseluruhan di AWS IoT Device Management

Pekerjaan. Untuk informasi selengkapnya, lihat [Apa itu Amazon CloudWatch?](#) Memantau log pekerjaan dan menangkap data historis untuk pemecahan masalah. Cara kerjanya dengan pekerjaan.

- Layanan AWS IoT Device Shadow: Pertahankan representasi digital dari AWS IoT benda Anda melalui bayangan perangkat menggunakan AWS IoT Device Management Jobs sehingga status perangkat Anda tersedia untuk aplikasi dan layanan lain terlepas dari konektivitas perangkat. Untuk informasi selengkapnya, lihat [Layanan AWS IoT Device Shadow](#).
- Fleet Hub for AWS IoT Device Management: Bangun aplikasi web mandiri untuk memantau kesehatan armada perangkat Anda. Untuk informasi selengkapnya, lihat [Apa itu Fleet Hub for AWS IoT Device Management?](#) .
- Praktik terbaik keamanan
  - Kontrol izin: Kontrol izin akses ke petunjuk pengoperasian jarak jauh menggunakan Amazon S3 dan tentukan pengguna IAM mana yang dapat menerapkan instruksi pengoperasian jarak jauh ke armada perangkat AWS IoT menggunakan kebijakan dan peran pengguna IAM.
  - Untuk informasi selengkapnya tentang AWS IoT kebijakan, lihat [Buat AWS IoT kebijakan](#).
  - Untuk informasi selengkapnya tentang peran pengguna IAM, lihat [Identitas dan manajemen akses untuk AWS IoT](#).
- Skalabilitas
  - Penerapan pekerjaan yang ditargetkan: Kontrol perangkat mana yang menerima dokumen pekerjaan dari pekerjaan dengan penerapan pekerjaan yang ditargetkan menggunakan kriteria pengelompokan perangkat tertentu yang dimasukkan dalam dokumen pekerjaan Anda saat membuat pekerjaan. Membuat AWS IoT sesuatu untuk setiap perangkat dan menyimpan informasi itu di AWS IoT registri memungkinkan Anda melakukan pencarian yang ditargetkan menggunakan pengindeksan armada. Anda dapat membuat grup kustom berdasarkan hasil pencarian pengindeksan armada untuk mendukung penerapan pekerjaan target Anda. Untuk informasi selengkapnya, lihat [Mengelola perangkat dengan AWS IoT](#). Gunakan pekerjaan untuk melakukan snapshot vs pekerjaan berkelanjutan.
  - Status pekerjaan: Lacak status peluncuran dokumen pekerjaan ke armada perangkat Anda dan status pekerjaan keseluruhan dari tingkat armada perangkat selain status implementasi individual dokumen pekerjaan di setiap perangkat. Untuk informasi selengkapnya, lihat [Pekerjaan dan status eksekusi pekerjaan](#).
  - Skalabilitas perangkat baru: Menerapkan dokumen pekerjaan Anda dengan mudah ke perangkat baru dengan menambahkannya ke grup kustom yang sudah ada yang dibuat menggunakan

pengindeksan armada melalui pekerjaan berkelanjutan. Ini akan menghemat waktu Anda karena harus menyebarkan dokumen pekerjaan ke setiap perangkat baru secara terpisah. Atau, Anda dapat menggunakan pendekatan yang lebih bertarget dengan snapshot dengan menerapkan dokumen pekerjaan ke grup perangkat yang telah ditentukan sebelumnya sekali dan kemudian pekerjaan selesai.

- **Fleksibilitas**
  - **Konfigurasi Job:** Sesuaikan dokumen pekerjaan dan pekerjaan Anda dengan peluncuran konfigurasi pekerjaan opsional, penjadwalan, pembatalan, batas waktu, dan coba lagi untuk memenuhi kebutuhan spesifik Anda. Untuk informasi selengkapnya, lihat [Konfigurasi Job](#).
- **Hemat biaya**
  - Perkenalkan struktur biaya yang lebih efisien untuk mempertahankan armada perangkat Anda dengan memanfaatkan AWS IoT Device Management Jobs untuk menerapkan pembaruan penting dan melakukan tugas pemeliharaan rutin. Solusi do-it-yourself (DIY) untuk memelihara armada perangkat Anda mencakup biaya variabel berulang seperti infrastruktur yang diperlukan untuk meng-host dan mengelola solusi DIY, biaya tenaga kerja untuk mengembangkan, memelihara, dan menskalakan solusi DIY, dan biaya transmisi data. Manfaatkan struktur AWS IoT Device Management Jobs yang transparan dan biaya tetap, Anda tahu persis berapa biaya pelaksanaan pekerjaan untuk perangkat selain biaya transmisi data yang diperlukan untuk memfasilitasi peluncuran dokumen pekerjaan ke armada perangkat Anda dan melacak status pelaksanaan pekerjaan untuk setiap perangkat. Untuk informasi selengkapnya, lihat [harga AWS IoT Core](#).

## Apa itu AWS IoT Jobs?

Gunakan AWS IoT Jobs untuk menentukan satu set operasi jarak jauh yang dapat dikirim dan dijalankan pada satu atau beberapa perangkat yang terhubung AWS IoT.

Untuk membuat pekerjaan, pertama-tama tentukan dokumen pekerjaan yang berisi daftar instruksi yang menjelaskan operasi yang harus dilakukan perangkat dari jarak jauh. Untuk melakukan operasi ini, tentukan daftar target, yang merupakan hal-hal individual, [kelompok benda](#), atau keduanya. Dokumen pekerjaan dan target bersama-sama merupakan penyebaran.

Setiap penerapan dapat memiliki konfigurasi tambahan:

- **Peluncuran:** Konfigurasi ini menentukan berapa banyak perangkat yang menerima dokumen pekerjaan setiap menit.

- **Batalan:** Jika sejumlah perangkat tidak menerima pemberitahuan pekerjaan, gunakan konfigurasi ini untuk membatalkan pekerjaan. Ini menghindari pengiriman pembaruan yang buruk ke seluruh armada.
- **Timeout:** Jika respons tidak diterima dari target pekerjaan Anda dalam durasi tertentu, pekerjaan itu bisa gagal. Anda dapat melacak pekerjaan yang berjalan di perangkat ini.
- **Coba lagi:** Jika perangkat melaporkan kegagalan atau waktu kerja habis, Anda dapat menggunakan AWS IoT Pekerjaan untuk mengirim ulang dokumen pekerjaan ke perangkat secara otomatis.
- **Penjadwalan:** Konfigurasi ini memungkinkan Anda menjadwalkan pekerjaan untuk tanggal dan waktu yang akan datang. Ini juga memungkinkan Anda untuk membuat jendela pemeliharaan berulang yang memperbarui perangkat selama periode lalu lintas rendah yang telah ditentukan sebelumnya.

AWS IoT Jobs mengirim pesan untuk menginformasikan target bahwa pekerjaan tersedia. Target memulai pelaksanaan pekerjaan dengan mengunduh dokumen pekerjaan, melakukan operasi yang ditentukannya, dan melaporkan kemajuannya. AWS IoT Anda dapat melacak kemajuan pekerjaan untuk target tertentu atau untuk semua target dengan menjalankan perintah yang disediakan oleh AWS IoT Jobs. Ketika pekerjaan dimulai, ia memiliki status Sedang berlangsung. Perangkat kemudian melaporkan pembaruan tambahan saat menampilkan status ini hingga pekerjaan berhasil, gagal, atau habis waktu.

Topik berikut menjelaskan beberapa konsep kunci pekerjaan dan siklus hidup pekerjaan dan eksekusi pekerjaan.

Topik

- [Konsep kunci pekerjaan](#)
- [Pekerjaan dan status eksekusi pekerjaan](#)

## Konsep kunci pekerjaan

Konsep berikut memberikan rincian tentang AWS IoT Jobs dan cara membuat dan menerapkan pekerjaan untuk menjalankan operasi jarak jauh di perangkat Anda.

## Konsep dasar

Berikut ini adalah konsep dasar yang harus Anda ketahui saat menggunakan AWS IoT Jobs.



## Job

Pekerjaan adalah operasi jarak jauh yang dikirim ke dan dijalankan pada satu atau lebih perangkat yang terhubung AWS IoT. Misalnya, Anda dapat menentukan pekerjaan yang menginstruksikan satu set perangkat untuk mengunduh dan menginstal aplikasi atau menjalankan pembaruan firmware, reboot, memutar sertifikat, atau melakukan operasi pemecahan masalah jarak jauh.

### Dokumen Job

Untuk membuat pekerjaan, Anda harus terlebih dahulu membuat dokumen pekerjaan yang merupakan deskripsi operasi jarak jauh yang akan dilakukan oleh perangkat.

Dokumen Job adalah dokumen JSON yang dikodekan UTF-8 dan berisi informasi yang diperlukan perangkat Anda untuk melakukan pekerjaan. Dokumen pekerjaan berisi satu atau beberapa URL tempat perangkat dapat mengunduh pembaruan atau data lainnya. Dokumen pekerjaan dapat disimpan dalam bucket Amazon S3, atau disertakan sebaris dengan perintah yang membuat pekerjaan.

#### Tip

Untuk contoh dokumen pekerjaan, lihat contoh [jobs-agent.js](#) di AWS IoT SDK untuk JavaScript.

### Target

Saat Anda membuat pekerjaan, Anda menentukan daftar target yang merupakan perangkat yang harus melakukan operasi. Targetnya bisa berupa benda atau [kelompok benda](#) atau keduanya. Layanan AWS IoT Jobs mengirimkan pesan ke setiap target untuk menginformasikan bahwa pekerjaan tersedia.

### Deployment

Setelah Anda membuat pekerjaan dengan menyediakan dokumen pekerjaan dan menentukan daftar target Anda, dokumen pekerjaan kemudian disebar ke perangkat target jarak jauh yang ingin Anda lakukan pembaruannya. Untuk pekerjaan snapshot, pekerjaan akan selesai setelah menerapkan ke perangkat target. Untuk pekerjaan berkelanjutan, pekerjaan dikerahkan ke sekelompok perangkat saat ditambahkan ke grup.

## Eksekusi Job

Eksekusi pekerjaan adalah contoh pekerjaan pada perangkat target. Target memulai eksekusi pekerjaan dengan mengunduh dokumen pekerjaan. Kemudian melakukan operasi yang ditentukan dalam dokumen, dan melaporkan kemajuannya ke AWS IoT. Nomor eksekusi adalah pengidentifikasi unik dari eksekusi pekerjaan pada target tertentu. Layanan AWS IoT Jobs menyediakan perintah untuk melacak kemajuan pelaksanaan pekerjaan pada target dan kemajuan pekerjaan di semua target.

## Konsep tipe pekerjaan

Konsep-konsep berikut dapat membantu Anda memahami lebih banyak tentang berbagai jenis pekerjaan yang dapat Anda buat dengan AWS IoT Jobs.

### Pekerjaan snapshot

Secara default, pekerjaan dikirim ke semua target yang Anda tentukan saat Anda membuat pekerjaan. Setelah target tersebut menyelesaikan pekerjaan (atau melaporkan bahwa mereka tidak dapat melakukannya), pekerjaan selesai.

### Pekerjaan berkelanjutan

Pekerjaan berkelanjutan dikirim ke semua target yang Anda tentukan saat Anda membuat pekerjaan. Ini terus berjalan dan dikirim ke perangkat baru (hal-hal) yang ditambahkan ke grup target. Misalnya, pekerjaan berkelanjutan dapat digunakan untuk onboard atau upgrade perangkat saat ditambahkan ke grup. Anda dapat membuat pekerjaan berkelanjutan dengan menetapkan parameter opsional saat Anda membuat pekerjaan.

#### Note

Saat menargetkan armada IoT Anda menggunakan grup benda dinamis, kami sarankan Anda menggunakan pekerjaan berkelanjutan alih-alih pekerjaan snapshot. Dengan menggunakan pekerjaan berkelanjutan, perangkat yang bergabung dengan grup menerima eksekusi pekerjaan bahkan setelah pekerjaan dibuat.

## URL yang ditandatangani sebelumnya

Untuk akses data yang aman dan terbatas waktu yang tidak disertakan dalam dokumen pekerjaan, Anda dapat menggunakan URL Amazon S3 yang telah ditetapkan sebelumnya. Tempatkan data

Anda di bucket Amazon S3 dan tambahkan tautan placeholder ke data dalam dokumen pekerjaan. Ketika AWS IoT Jobs menerima permintaan untuk dokumen pekerjaan, itu mem-parsing dokumen pekerjaan dengan mencari tautan placeholder, dan kemudian mengganti tautan dengan URL Amazon S3 yang telah ditetapkan sebelumnya.

Tautan placeholder dalam format berikut:

```
${aws:iot:s3-presigned-url:https://s3.amazonaws.com/bucket/key}
```

di mana *bucket* adalah nama bucket Anda dan *kuncinya* adalah objek dalam ember yang Anda tautkan.

Di Wilayah Beijing dan Ningxia, URL presigned hanya berfungsi jika pemilik sumber daya memiliki lisensi ICP (Internet Content Provider). Untuk informasi selengkapnya, lihat [Amazon Simple Storage Service](#) di dokumentasi Memulai AWS Layanan di China.

## Konsep konfigurasi Job

Konsep-konsep berikut dapat membantu Anda memahami cara mengkonfigurasi pekerjaan.

### Peluncuran

Anda dapat menentukan seberapa cepat target diberi tahu tentang eksekusi pekerjaan yang tertunda. Ini memungkinkan Anda membuat peluncuran bertahap untuk mengelola pembaruan, reboot, dan operasi lainnya dengan lebih baik. Anda dapat membuat konfigurasi peluncuran dengan menggunakan tingkat peluncuran statis atau tingkat peluncuran eksponensial. Untuk menentukan jumlah maksimum target pekerjaan yang akan diinformasikan per menit, gunakan tingkat peluncuran statis.

Untuk contoh pengaturan tarif peluncuran dan untuk informasi selengkapnya tentang mengonfigurasi peluncuran pekerjaan, lihat [Peluncuran pekerjaan, penjadwalan, dan membatalkan konfigurasi](#)

### Penjadwalan

Penjadwalan pekerjaan memungkinkan Anda menjadwalkan jangka waktu peluncuran dokumen pekerjaan ke semua perangkat dalam kelompok sasaran untuk pekerjaan berkelanjutan dan snapshot. Selain itu, Anda dapat membuat jendela pemeliharaan opsional yang berisi tanggal dan waktu tertentu pekerjaan akan meluncurkan dokumen pekerjaan ke semua perangkat dalam grup target. Jendela pemeliharaan adalah contoh berulang dengan frekuensi tanggal dan waktu harian,

mingguan, bulanan, atau kustom yang dipilih selama pekerjaan awal atau pembuatan templat pekerjaan. Hanya pekerjaan berkelanjutan yang dapat dijadwalkan untuk melakukan peluncuran selama jendela pemeliharaan.

Penjadwalan Pekerjaan khusus untuk pekerjaan Anda. Eksekusi Job Individual tidak dapat dijadwalkan. Untuk informasi selengkapnya, lihat [Peluncuran pekerjaan, penjadwalan, dan membatalkan konfigurasi](#).

## Batalkan

Anda dapat membuat serangkaian kondisi untuk membatalkan peluncuran ketika kriteria yang Anda tentukan telah terpenuhi. Untuk informasi selengkapnya, lihat [Peluncuran pekerjaan, penjadwalan, dan membatalkan konfigurasi](#).

## Timeout

Batas waktu kerja memberi tahu Anda setiap kali penerapan pekerjaan macet di IN\_PROGRESS negara bagian untuk jangka waktu yang sangat lama. Ada dua jenis pengatur waktu: pengatur waktu dalam proses dan pengatur waktu langkah. Ketika pekerjaan itu IN\_PROGRESS, Anda dapat memantau dan melacak kemajuan penyebaran pekerjaan Anda.

Konfigurasi peluncuran dan pembatalan khusus untuk pekerjaan Anda, sedangkan konfigurasi batas waktu khusus untuk penerapan pekerjaan. Untuk informasi selengkapnya, lihat [Konfigurasi batas waktu eksekusi pekerjaan dan coba lagi](#).

## Mencoba lagi

Job retries memungkinkan untuk mencoba kembali eksekusi pekerjaan ketika pekerjaan gagal, waktu habis, atau keduanya. Anda dapat memiliki hingga 10 percobaan ulang untuk melaksanakan pekerjaan itu. Anda dapat memantau dan melacak kemajuan upaya percobaan ulang Anda dan apakah eksekusi pekerjaan berhasil.

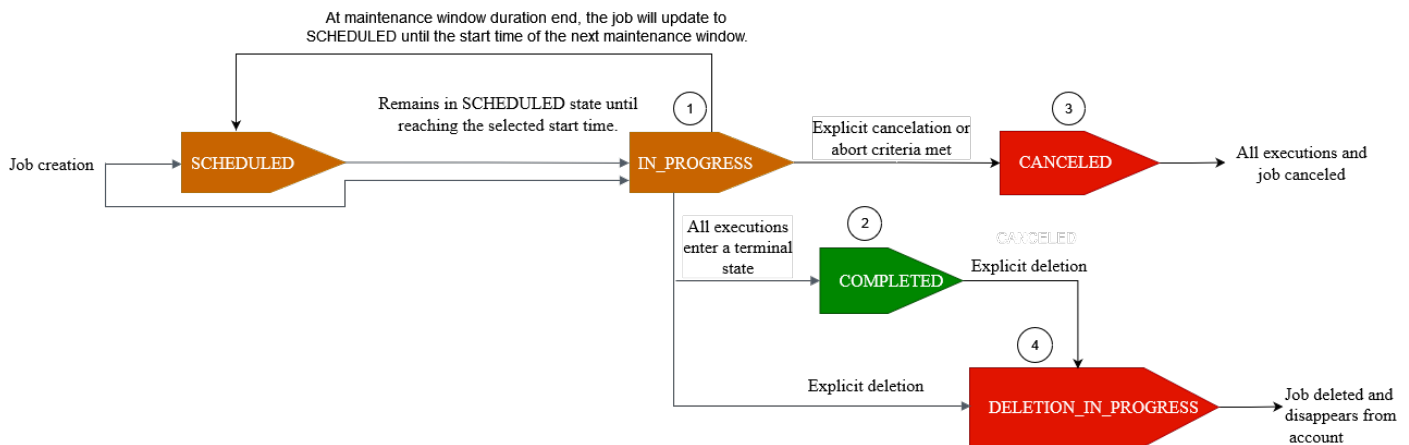
Konfigurasi peluncuran dan pembatalan khusus untuk pekerjaan Anda, sedangkan konfigurasi batas waktu dan coba lagi khusus untuk eksekusi pekerjaan. Untuk informasi selengkapnya, lihat [Konfigurasi batas waktu eksekusi pekerjaan dan coba lagi](#).

## Pekerjaan dan status eksekusi pekerjaan

Bagian berikut menjelaskan siklus hidup AWS IoT pekerjaan dan siklus hidup pelaksanaan pekerjaan.

## Status Job

Diagram berikut menunjukkan keadaan yang berbeda dari suatu AWS IoT pekerjaan.



Pekerjaan yang Anda buat menggunakan AWS IoT Jobs dapat berada di salah satu status berikut:

- **DIJADWALKAN**

Selama pembuatan templat pekerjaan atau pekerjaan awal menggunakan AWS IoT konsol, [CreateJob](#) API, atau [CreateJobTemplate](#) API, Anda dapat memilih konfigurasi penjadwalan opsional di AWS IoT konsol atau `SchedulingConfig` di [CreateJob](#) API atau [CreateJobTemplate](#) API. Saat Anda memulai pekerjaan terjadwal yang berisi status pekerjaan tertentu `startTime` `endTime` `endBehavior`, dan, pembaruan status pekerjaan `SCHEDULED`. Ketika pekerjaan mencapai yang Anda pilih `startTime` atau jendela pemeliharaan berikutnya (jika Anda memilih peluncuran pekerjaan selama jendela pemeliharaan), status akan diperbarui dari `SCHEDULED` ke `IN_PROGRESS` dan mulai peluncuran dokumen pekerjaan ke semua perangkat dalam grup target. `startTime`

- **IN\_PROGRESS**

Saat Anda membuat pekerjaan menggunakan AWS IoT konsol atau [CreateJob](#) API, status pekerjaan akan diperbarui ke `IN_PROGRESS`. Selama penciptaan lapangan kerja, AWS IoT Jobs mulai meluncurkan eksekusi pekerjaan ke perangkat di grup target Anda. Setelah semua eksekusi pekerjaan diluncurkan, AWS IoT Jobs menunggu perangkat untuk menyelesaikan tindakan jarak jauh.

Untuk informasi tentang konkurensi dan batasan yang berlaku untuk pekerjaan yang sedang berlangsung, lihat. [Batas Job](#)

**Note**

Ketika IN\_PROGRESS pekerjaan mencapai akhir jendela pemeliharaan saat ini, peluncuran dokumen pekerjaan akan berhenti. Pekerjaan akan diperbarui SCHEDULED hingga jendela `startTime` pemeliharaan berikutnya.

**• DISELESAIKAN**

Pekerjaan berkelanjutan ditangani dengan salah satu cara berikut:

- Untuk pekerjaan berkelanjutan tanpa konfigurasi penjadwalan opsional yang dipilih, selalu dalam proses dan terus berjalan untuk perangkat baru yang ditambahkan ke grup target. Itu tidak akan pernah mencapai status `statusCOMPLETED`.
- Untuk pekerjaan berkelanjutan dengan konfigurasi penjadwalan opsional yang dipilih, berikut ini benar:
  - Jika `endTime` diberikan, pekerjaan berkelanjutan akan mencapai `COMPLETED` status ketika `endTime` telah berlalu dan semua eksekusi pekerjaan telah mencapai status terminal.
  - Jika tidak disediakan dalam konfigurasi penjadwalan opsional, pekerjaan berkelanjutan akan terus melakukan peluncuran dokumen pekerjaan. `endTime`

Untuk pekerjaan snapshot, status pekerjaan berubah menjadi `COMPLETED` saat semua eksekusi pekerjaannya memasuki status terminal, seperti, `SUCCEEDED`, `FAILED`, `TIMED_OUT`, `REMOVED`, atau `CANCELED`.

**• MEMBATALKAN**

Saat Anda membatalkan pekerjaan menggunakan AWS IoT konsol, [CancelJobAPI](#), atau [Konfigurasi pembatalan pekerjaan](#), status pekerjaan akan berubah menjadi `CANCELED`. Selama pembatalan pekerjaan, AWS IoT Jobs mulai membatalkan eksekusi pekerjaan yang dibuat sebelumnya.

Untuk informasi tentang konkurensi dan batasan yang berlaku untuk pekerjaan yang dibatalkan, lihat [Batas Job](#).

**• DELETION\_IN\_PROGRESS**

Saat Anda menghapus pekerjaan menggunakan AWS IoT konsol atau [DeleteJobAPI](#), status pekerjaan akan berubah menjadi `DELETION_IN_PROGRESS`. Selama penghapusan pekerjaan,

AWS IoT Jobs mulai menghapus eksekusi pekerjaan yang dibuat sebelumnya. Setelah semua eksekusi pekerjaan telah dihapus, pekerjaan menghilang dari AWS akun Anda.

## Status eksekusi Job

Tabel berikut menunjukkan status yang berbeda dari eksekusi AWS IoT pekerjaan dan apakah perubahan status diprakarsai oleh perangkat atau oleh AWS IoT Jobs.

### Status dan sumber eksekusi Job

| Status eksekusi Job | Dimulai oleh perangkat? | Diprakarsai oleh AWS IoT Jobs? | Status terminal? | Bisa dicoba lagi? |
|---------------------|-------------------------|--------------------------------|------------------|-------------------|
| QUEUED              | Tidak                   | Ya                             | Tidak            | Tidak berlaku     |
| IN_PROGRESS         | Ya                      | Tidak                          | Tidak            | Tidak berlaku     |
| SUCCEEDED           | Ya                      | Tidak                          | Ya               | Tidak berlaku     |
| FAILED              | Ya                      | Tidak                          | Ya               | Ya                |
| TIMED_OUT           | Tidak                   | Ya                             | Ya               | Ya                |
| REJECTED            | Ya                      | Tidak                          | Ya               | Tidak             |
| REMOVED             | Tidak                   | Ya                             | Ya               | Tidak             |
| CANCELED            | Tidak                   | Ya                             | Ya               | Tidak             |

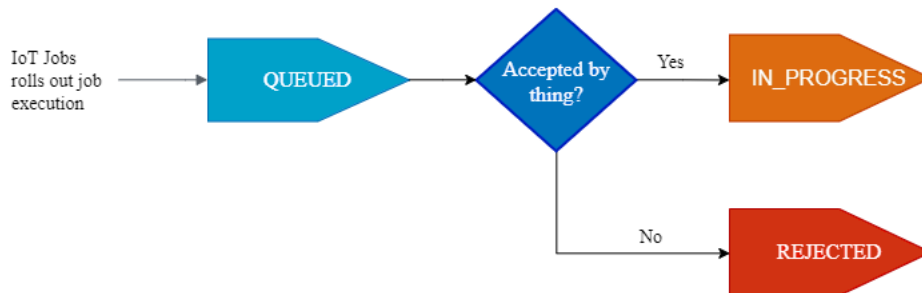
Bagian berikut menjelaskan lebih lanjut tentang status eksekusi pekerjaan yang diluncurkan saat Anda membuat pekerjaan dengan AWS IoT Jobs.

- **DIANTREKAN**

Saat AWS IoT Jobs meluncurkan eksekusi pekerjaan untuk perangkat target, status eksekusi pekerjaan disetel ke `QUEUED`. Eksekusi pekerjaan tetap di `QUEUED` negara bagian sampai:

- Perangkat Anda menerima eksekusi pekerjaan dan memanggil operasi API Pekerjaan dan melaporkan statusnya sebagai `IN_PROGRESS`.

- Anda membatalkan pelaksanaan pekerjaan atau pekerjaan, atau ketika kriteria pembatalan yang Anda tentukan terpenuhi, dan statusnya berubah menjadi CANCELED.
- Perangkat Anda dihapus dari grup target dan statusnya berubah menjadi REMOVED.



## • IN\_PROGRESS

Jika perangkat IoT Anda berlangganan ke yang dicadangkan [Topik Job](#) \$notify dan \$notify-next, dan perangkat Anda memanggil StartNextPendingJobExecution API atau UpdateJobExecution API dengan status IN\_PROGRESS, AWS IoT Jobs akan menyetel status eksekusi pekerjaan. IN\_PROGRESS

UpdateJobExecution API dapat dipanggil beberapa kali dengan status. IN\_PROGRESS Anda dapat menentukan detail tambahan tentang langkah-langkah eksekusi menggunakan statusDetails objek.

### Note

Jika Anda membuat beberapa pekerjaan untuk setiap perangkat, AWS IoT Jobs dan protokol MQTT tidak menjamin pesanan pengiriman.

## • SUKSES

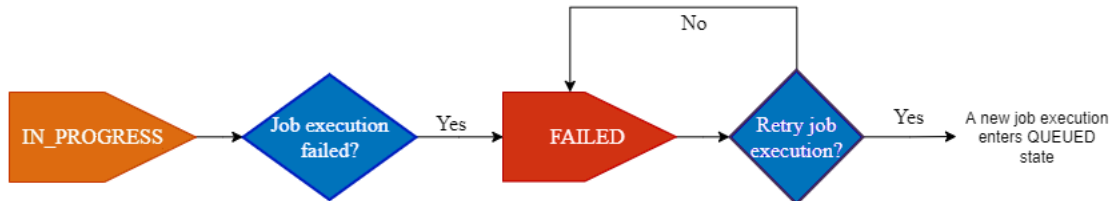
Ketika perangkat Anda berhasil menyelesaikan operasi jarak jauh, perangkat harus menjalankan UpdateJobExecution API dengan status SUCCEEDED untuk menunjukkan bahwa eksekusi pekerjaan berhasil. AWS IoT Jobs kemudian memperbarui dan mengembalikan status eksekusi pekerjaan sebagai SUCCEEDED.



## • Failed

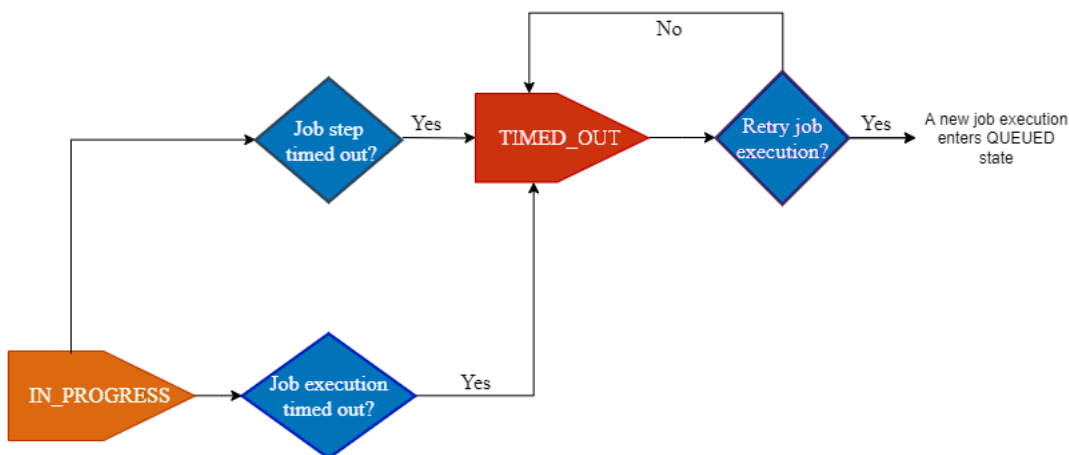


Ketika perangkat Anda gagal menyelesaikan operasi jarak jauh, perangkat harus menjalankan UpdateJobExecution API dengan status Failed untuk menunjukkan bahwa eksekusi pekerjaan gagal. AWS IoT Jobs kemudian memperbarui dan mengembalikan status eksekusi pekerjaan sebagai Failed. Anda dapat mencoba lagi eksekusi pekerjaan ini untuk perangkat menggunakan file. [Konfigurasi coba lagi eksekusi Job](#)



- HABIS\_WAKTU

Saat perangkat Anda gagal menyelesaikan langkah pekerjaan saat statusnya IN\_PROGRESS, atau saat gagal menyelesaikan operasi jarak jauh dalam durasi waktu tunggu pengatur waktu yang sedang berlangsung, AWS IoT Jobs akan menetapkan status eksekusi pekerjaan. TIMED\_OUT Anda juga memiliki pengatur waktu langkah untuk setiap langkah pekerjaan dari pekerjaan yang sedang berlangsung dan hanya berlaku untuk pelaksanaan pekerjaan. Durasi pengatur waktu yang sedang berlangsung ditentukan menggunakan inProgressTimeoutInMinutes properti. [Konfigurasi batas waktu eksekusi Job](#) Anda dapat mencoba lagi eksekusi pekerjaan ini untuk perangkat menggunakan file. [Konfigurasi coba lagi eksekusi Job](#)



- MENOLAK

Ketika perangkat Anda menerima permintaan yang tidak valid atau tidak kompatibel, perangkat harus menjalankan `UpdateJobExecution` API dengan status `REJECTED`. AWS IoT Jobs kemudian memperbarui dan mengembalikan status eksekusi pekerjaan sebagai `REJECTED`.

- **DIKELUARKAN**

Jika perangkat Anda tidak lagi menjadi target yang valid untuk eksekusi pekerjaan, seperti saat perangkat terlepas dari grup hal dinamis, AWS IoT Jobs akan menetapkan status eksekusi pekerjaan. `REMOVED` Anda dapat melampirkan kembali benda itu ke grup target Anda dan memulai ulang eksekusi pekerjaan untuk perangkat.

- **MEMBATALKAN**

Saat Anda membatalkan pekerjaan atau membatalkan eksekusi pekerjaan menggunakan konsol atau `CancelJobExecution` API, `CancelJob` atau saat kriteria pembatalan yang ditentukan menggunakan tugas [Konfigurasi pembatalan pekerjaan](#) terpenuhi, AWS IoT Pekerjaan akan membatalkan pekerjaan tersebut dan menetapkan status eksekusi pekerjaan. `CANCELED`

## Mengelola tugas

Gunakan pekerjaan untuk memberi tahu perangkat tentang pembaruan perangkat lunak atau firmware. Anda dapat menggunakan [AWS IoT konsol](#), yang [Manajemen pekerjaan dan pengendalian operasi API](#), yang [AWS Command Line Interface](#), atau [AWS SDK](#) untuk membuat dan mengelola pekerjaan.

## Penandatanganan kode untuk pekerjaan

Saat mengirim kode ke perangkat, agar perangkat dapat mendeteksi apakah kode telah dimodifikasi saat transit, kami sarankan Anda menandatangani file kode dengan menggunakan AWS CLI. Untuk instruksi, lihat [Buat dan kelola pekerjaan dengan menggunakan AWS CLI](#).

Untuk informasi lebih lanjut, lihat [Untuk Apa Penandatanganan Kode AWS IoT?](#).

## Dokumen pekerjaan

Sebelum Anda membuat pekerjaan, Anda harus membuat dokumen pekerjaan. Jika Anda menggunakan penandatanganan kode untuk AWS IoT, Anda harus mengunggah dokumen pekerjaan Anda ke bucket Amazon S3 berseri. Untuk informasi selengkapnya tentang membuat bucket

Amazon S3 dan mengunggah file ke dalamnya, lihat [Memulai dengan Amazon Simple Storage Service](#) di dalam Panduan Memulai Amazon S3.

### Tip

Untuk contoh dokumen pekerjaan, lihat [jobs-agent.js](#) contoh di AWS IoT SDK untuk JavaScript.

## URLs yang ditandatangani sebelumnya

Dokumen pekerjaan Anda dapat berisi URL Amazon S3 yang telah ditandatangani sebelumnya yang menunjuk ke file kode Anda (atau file lain). URL Amazon S3 yang telah ditandatangani sebelumnya hanya berlaku untuk jangka waktu terbatas dan dihasilkan saat perangkat meminta dokumen pekerjaan. Karena URL yang telah ditandatangani sebelumnya tidak dibuat saat Anda membuat dokumen pekerjaan, gunakan URL placeholder di dokumen pekerjaan Anda. URL placeholder terlihat seperti berikut ini:

```
${aws:iot:s3-presigned-url:https://s3.region.amazonaws.com/<bucket>/<code file>}
```

di mana:

- *ember* adalah bucket Amazon S3 yang berisi file kode.
- *berkas kode* adalah kunci Amazon S3 dari file kode.

Saat perangkat meminta dokumen pekerjaan, AWS IoT menghasilkan URL yang telah ditandatangani sebelumnya dan menggantikan URL placeholder dengan URL yang telah ditandatangani sebelumnya. Dokumen pekerjaan Anda kemudian dikirim ke perangkat.

### Peran IAM untuk memberikan izin untuk mengunduh file dari S3

Saat Anda membuat pekerjaan yang menggunakan URL Amazon S3 yang telah ditandatangani sebelumnya, Anda harus menyediakan peran IAM. Peran harus memberikan izin untuk mengunduh file dari bucket Amazon S3 tempat data atau pembaruan disimpan. Peran juga harus memberikan izin untuk AWS IoT untuk mengasumsikan peran.

Anda dapat menentukan batas waktu opsional untuk URL yang telah ditandatangani sebelumnya. Untuk informasi lebih lanjut, lihat [CreateJob](#).

## Hibah AWS IoT Izin pekerjaan untuk mengambil peran Anda

1. Pergi ke [Hub peran konsol IAM](#) dan pilih peran Anda.
2. Pada [Hubungan Kepercayaan](#) tab, pilih [Edit Hubungan Kepercayaan](#) dan ganti dokumen kebijakan dengan JSON berikut. Pilih [Perbarui Kebijakan Kepercayaan](#).

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Sid": "",
 "Effect": "Allow",
 "Principal": {
 "Service": [
 "iot.amazonaws.com"
]
 },
 "Action": "sts:AssumeRole"
 }
]
}
```

3. Untuk melindungi dari masalah wakil yang bingung, tambahkan kunci konteks kondisi global [aws:SourceArn](#) dan [aws:SourceAccount](#) untuk kebijakan.

### Important

Anda `aws:SourceArn` harus sesuai dengan format: `arn:aws:iot:region:account-id:*`. Pastikan bahwa `daerah` cocok dengan Anda AWS IoT Wilayah dan `akun-id` cocok dengan ID akun pelanggan Anda. Untuk informasi lebih lanjut, lihat [Pencegahan wakil bingung lintas layanan](#).

```
{
 "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. Jika pekerjaan Anda menggunakan dokumen pekerjaan yang merupakan objek Amazon S3, pilih lzin dan gunakan JSON berikut. Ini menambahkan kebijakan yang memberikan izin untuk mengunduh file dari bucket Amazon S3 Anda:

```

{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": "s3:GetObject",
 "Resource": "arn:aws:s3:::your_s3_bucket/*"
 }
]
}

```

## Topik


- [Buat dan kelola pekerjaan dengan menggunakan AWS Management Console](#)
- [Buat dan kelola pekerjaan dengan menggunakan AWS CLI](#)

## Buat dan kelola pekerjaan dengan menggunakan AWS Management Console

Untuk membuat pekerjaan

1. Masuk ke AWS Management Console dan masuk ke AWS IoT konsol.

2. Di panel navigasi kiri, di bawah bagian Kelola, pilih Tindakan Jarak Jauh, dan kemudian pilih Lowongan.
3. Pada Lowongan halaman di Lowongan kotak dialog, pilih Buat pekerjaan.
4. Bergantung pada perangkat yang Anda gunakan, Anda dapat membuat pekerjaan khusus, pekerjaan pembaruan FreeRTOS OTA, atau AWS IoT Greengrass pekerjaan. Untuk contoh ini, pilih Buat pekerjaan khusus. Pilih Selanjutnya.
5. Pada Properti pekerjaan kustom halaman, di Properti pekerjaan kotak dialog, masukkan informasi Anda untuk bidang berikut:
  - Nama: Masukkan nama pekerjaan alfanumerik yang unik.
  - Deskripsi - opsional: Masukkan deskripsi opsional tentang Pekerjaan Anda.
  - Tag - opsional:

 Note

Kami menyarankan Anda untuk tidak menggunakan informasi identitas pribadi dalam ID pekerjaan dan deskripsi Anda.

Pilih Selanjutnya.

6. Pada Konfigurasi file halaman di Target pekerjaan kotak dialog, pilih Hal-hal atau Kelompok hal bahwa Anda ingin menjalankan pekerjaan ini.

Dalam Dokumen pekerjaan kotak dialog, pilih salah satu dari pilihan berikut:

- Dari file: File pekerjaan JSON yang sebelumnya Anda unggah ke bucket Amazon S3
  - Penandatanganan kode

Dalam dokumen pekerjaan yang terdapat di URL Amazon S3 Anda, `{aws:iot:code-sign-signature:s3://region.bucket/code-file@code-file-version-id}` diperlukan sebagai placeholder sampai diganti dengan path file kode yang ditandatangani menggunakan Profil penandatanganan kode. File kode yang ditandatangani baru awalnya akan muncul di Signed Images folder di bucket sumber Amazon S3 Anda. Dokumen pekerjaan baru yang berisi `Codesigned_awalan` akan dibuat dengan jalur file kode yang ditandatangani menggantikan placeholder tanda kode dan ditempatkan di URL Amazon S3 Anda untuk membuat pekerjaan baru.

- URL sumber daya pra-tanda tangan

Dalam Peran pra-penandatanganan tarik-turun, pilih peran IAM yang Anda buat [URL yang telah ditandatangani sebelumnya](#). Menggunakan `${aws:iot:s3-presigned-url}` untuk menetapkan URL untuk objek yang terletak di Amazon S3 adalah praktik keamanan terbaik untuk perangkat yang mengunduh objek dari Amazon S3.

Jika Anda ingin menggunakan URL yang telah ditandatangani sebelumnya untuk placeholder penandatanganan kode, gunakan contoh template berikut:

```
${aws:iot:s3-presigned-url:${aws:iot:code-sign-signature:<S3 URL>}
```

- Dari template: Template pekerjaan yang berisi dokumen pekerjaan dan konfigurasi pekerjaan. Template pekerjaan dapat menjadi template pekerjaan kustom yang Anda buat atau AWS template yang dikelola.

Jika Anda membuat pekerjaan untuk melakukan tindakan jarak jauh yang sering digunakan seperti mem-boot ulang perangkat, Anda dapat menggunakan AWS template yang dikelola. Template ini telah dikonfigurasi sebelumnya untuk digunakan. Untuk informasi selengkapnya, lihat [Buat template pekerjaan khusus](#) dan [Buat templat pekerjaan khusus dari templat dikelola](#).

7. Pada Konfigurasi pekerjaan halaman di Konfigurasi pekerjaan kotak dialog, pilih salah satu dari jenis pekerjaan berikut:
  - Pekerjaan snapshot: Pekerjaan snapshot selesai setelah selesai dijalankan pada perangkat dan grup target.
  - Pekerjaan berkelanjutan: Pekerjaan berkelanjutan berlaku untuk grup benda dan berjalan di perangkat apa pun yang nantinya Anda tambahkan ke grup target tertentu.
8. Dalam Konfigurasi tambahan - opsional kotak dialog, tinjau konfigurasi Job opsional berikut dan buat pilihan Anda sesuai:
  - Konfigurasi peluncuran
  - Konfigurasi penjadwalan
  - Konfigurasi batas waktu eksekusi pekerjaan
  - Eksekusi pekerjaan coba lagi konfigurasi - baru
  - Batalkan konfigurasi

Lihat bagian berikut untuk informasi tambahan tentang konfigurasi Pekerjaan:

- [Peluncuran pekerjaan, penjadwalan, dan membatalkan konfigurasi](#)
- [Konfigurasi batas waktu eksekusi pekerjaan dan coba lagi](#)

Tinjau semua pilihan pekerjaan Anda dan kemudian pilih **Menyerahkan** untuk menciptakan pekerjaan Anda.

Setelah Anda membuat pekerjaan, konsol menghasilkan tanda tangan JSON dan menempatkannya di dokumen pekerjaan Anda. Anda dapat menggunakan [AWS IoT konsol](#) untuk melihat status, membatalkan, atau menghapus pekerjaan. Untuk mengelola pekerjaan, pergi ke [Hub pekerjaan konsol](#).

## Buat dan kelola pekerjaan dengan menggunakan AWS CLI

Bagian ini menjelaskan cara membuat dan mengelola pekerjaan.

### Ciptakan lowongan kerja

Untuk membuat AWS IoT pekerjaan, gunakan `CreateJob` perintah. Pekerjaan ini antri untuk eksekusi pada target (hal-hal atau kelompok hal) yang Anda tentukan. Untuk membuat AWS IoT pekerjaan, Anda memerlukan dokumen pekerjaan yang dapat dimasukkan dalam badan permintaan atau sebagai tautan ke dokumen Amazon S3. Jika tugas termasuk mengunduh file menggunakan URL Amazon S3 yang telah ditandatangani sebelumnya, Anda memerlukan peran IAM Amazon Resource Name (ARN) yang memiliki izin untuk mengunduh file dan memberikan izin kepada AWS IoT Jobs layanan untuk mengambil peran.

Untuk informasi lebih lanjut tentang sintaks saat memasukkan tanggal dan waktu menggunakan perintah API atau AWS CLI, lihat [Stempel waktu](#).

### Penandatanganan kode dengan pekerjaan

Jika Anda menggunakan penandatanganan kode untuk AWS IoT, Anda harus memulai pekerjaan penandatanganan kode dan menyertakan output dalam dokumen pekerjaan Anda. Ini akan menggantikan placeholder tanda tangan kode dalam dokumen pekerjaan Anda, yang diperlukan sebagai placeholder sampai diganti dengan path file kode yang ditandatangani



menggunakan `AndaProfil` penandatanganan kode. Tanda kode tanda tangan placeholder akan terlihat seperti berikut:

```
${aws:iot:code-sign-signature:s3://region.bucket/code-file@code-file-version-id}
```

Gunakan [start-signing-job](#) perintah untuk membuat pekerjaan kode penandatanganan. `start-signing-job` mengembalikan ID pekerjaan. Untuk mendapatkan lokasi Amazon S3 tempat tanda tangan disimpan, gunakan `describe-signing-job` perintah. Anda kemudian dapat mengunduh tanda tangan dari Amazon S3. Untuk informasi selengkapnya tentang lowongan penandatanganan kode, lihat [Penandatanganan kode untuk AWS IoT](#).

Dokumen pekerjaan Anda harus berisi placeholder URL yang telah ditandatangani sebelumnya untuk file kode Anda dan keluaran tanda tangan JSON yang ditempatkan di bucket Amazon S3 menggunakan `start-signing-job` perintah:

```
{
 "presign": "${aws:iot:s3-presigned-url:https://s3.region.amazonaws.com/bucket/
image}",
}
```

Buat pekerjaan dengan dokumen pekerjaan

Perintah berikut menunjukkan cara membuat pekerjaan menggunakan dokumen pekerjaan (*job-document.json*) disimpan dalam bucket Amazon S3 (*JobBucket*), dan peran dengan izin untuk mengunduh file dari 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/my-s3-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}"
```

Pekerjaan dijalankan pada *ThingOne*.

Opsional `timeout-config` parameter menentukan jumlah waktu setiap perangkat harus menyelesaikan pelaksanaannya dari pekerjaan. Timer dimulai ketika status eksekusi pekerjaan diatur ke `IN_PROGRESS`. Jika status eksekusi pekerjaan tidak disetel ke status terminal lain sebelum waktu berakhir, maka akan diatur ke `TIMED_OUT`.

Timer dalam proses tidak dapat diperbarui dan berlaku untuk semua eksekusi pekerjaan untuk pekerjaan tersebut. Setiap kali eksekusi pekerjaan tetap di `IN_PROGRESS` negara lebih lama dari interval ini, gagal dan beralih ke terminal `TIMED_OUT` status. AWS IoT juga menerbitkan pemberitahuan MQTT.

Untuk informasi selengkapnya tentang membuat konfigurasi untuk peluncuran pekerjaan dan pembatalan, lihat [Peluncuran Pekerjaan dan Batalkan Konfigurasi](#).

#### Note

Dokumen pekerjaan yang ditetapkan sebagai file Amazon S3 diambil pada saat Anda membuat pekerjaan. Jika Anda mengubah konten file Amazon S3 yang Anda gunakan sebagai sumber dokumen pekerjaan setelah Anda membuat dokumen pekerjaan, maka apa yang dikirim ke target pekerjaan tidak berubah.

## Memperbarui pekerjaan

Untuk memperbarui pekerjaan, gunakan `UpdateJob` perintah. Anda dapat memperbarui `description`, `presignedUrlConfig`, `jobExecutionsRolloutConfig`, `abortConfig`, dan `timeoutConfig` bidang pekerjaan.

```
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}"
```

Untuk informasi lebih lanjut, lihat [Peluncuran Pekerjaan dan Batalkan Konfigurasi](#).

## Membatalkan tugas

Untuk membatalkan pekerjaan, gunakan `CancelJob` perintah. Membatalkan pemberhentian pekerjaan AWS IoT dari meluncurkan eksekusi pekerjaan baru untuk pekerjaan itu. Ini juga membatalkan eksekusi pekerjaan apa pun yang ada di `QUEUED` negara. AWS IoT membuat eksekusi pekerjaan apa pun dalam status terminal tidak tersentuh karena perangkat telah menyelesaikan pekerjaan. Jika status eksekusi pekerjaan `IN_PROGRESS`, itu juga tetap tidak tersentuh kecuali Anda menggunakan opsional `--force` parameter.

Perintah berikut menunjukkan cara membatalkan pekerjaan dengan ID 010.

```
aws iot cancel-job --job-id 010
```

Perintah menampilkan output sebagai berikut:

```
{
 "jobArn": "string",
 "jobId": "string",
 "description": "string"
}
```

Ketika Anda membatalkan pekerjaan, eksekusi pekerjaan yang berada di `QUEUED` negara dibatalkan. Eksekusi pekerjaan yang ada di `IN_PROGRESS` negara dibatalkan, tetapi hanya jika Anda menentukan opsional `--force` parameter. Eksekusi pekerjaan dalam status terminal tidak dibatalkan.

### Warning

Membatalkan pekerjaan yang ada di `IN_PROGRESS` negara (dengan mengatur `--force` parameter) membatalkan eksekusi pekerjaan apa pun yang sedang berlangsung dan menyebabkan perangkat yang menjalankan tugas tidak dapat memperbarui status eksekusi pekerjaan. Berhati-hatilah dan pastikan bahwa setiap perangkat yang menjalankan tugas yang dibatalkan dapat pulih ke status yang valid.

Status pekerjaan yang dibatalkan atau salah satu eksekusi pekerjaannya pada akhirnya konsisten. AWS IoT berhenti menjadwalkan eksekusi pekerjaan baru dan `QUEUE` eksekusi pekerjaan untuk pekerjaan itu ke perangkat sesegera mungkin. Mengubah status eksekusi pekerjaan `CANCELED` mungkin memakan waktu, tergantung pada jumlah perangkat dan faktor lainnya.

Jika pekerjaan dibatalkan karena memenuhi kriteria yang ditentukan oleh `AbortConfig` objek, layanan menambahkan nilai auto-populated untuk `comment` dan `reasonCode` bidang. Anda dapat membuat nilai Anda sendiri untuk `reasonCode` saat pembatalan pekerjaan digerakkan oleh pengguna.

## Membatalkan eksekusi pekerjaan

Untuk membatalkan eksekusi pekerjaan di perangkat, gunakan `CancelJobExecution` perintah. Ini membatalkan eksekusi pekerjaan yang ada di `QUEUE` negara. Jika Anda ingin membatalkan eksekusi pekerjaan yang sedang berlangsung, Anda harus menggunakan `--force` parameter.

Perintah berikut menunjukkan bagaimana untuk membatalkan eksekusi pekerjaan dari pekerjaan 010 berjalan pada `myThing`.

```
aws iot cancel-job-execution --job-id 010 --thing-name myThing
```

Perintah tidak menampilkan output.

Eksekusi pekerjaan yang ada di `QUEUE` negara dibatalkan. Eksekusi pekerjaan yang ada di `IN_PROGRESS` negara dibatalkan, tetapi hanya jika Anda menentukan opsional `--force` parameter. Eksekusi pekerjaan dalam status terminal tidak dapat dibatalkan.

### Warning

Saat Anda membatalkan eksekusi pekerjaan yang ada di `IN_PROGRESS` negara, perangkat tidak dapat memperbarui status eksekusi pekerjaan. Berhati-hatilah dan pastikan perangkat dapat pulih ke keadaan yang valid.

Jika eksekusi pekerjaan dalam keadaan terminal, atau jika eksekusi pekerjaan dalam `IN_PROGRESS` negara dan `--force` parameter tidak disetel ke `true`, perintah ini menyebabkan `InvalidStateTransitionException`.

Status eksekusi pekerjaan yang dibatalkan pada akhirnya konsisten. Mengubah status eksekusi pekerjaan `CANCELED` mungkin memakan waktu, tergantung pada berbagai faktor.

## Menghapus pekerjaan

Untuk menghapus pekerjaan dan eksekusi pekerjaannya, gunakan `DeleteJob` perintah. Secara default, Anda hanya dapat menghapus pekerjaan yang berada dalam status terminal (`SUCCEEDED` atau `CANCELED`). Jika tidak, pengecualian terjadi. Anda dapat menghapus pekerjaan di `IN_PROGRESS` negara, bagaimanapun, hanya jika `force` parameter diatur ke `true`.

Untuk menghapus pekerjaan, jalankan perintah berikut:

```
aws iot delete-job --job-id 010 --force|--no-force
```

Perintah tidak menampilkan output.

### Warning

Ketika Anda menghapus pekerjaan yang ada di `IN_PROGRESS` state, perangkat yang menerapkan pekerjaan tidak dapat mengakses informasi pekerjaan atau memperbarui status eksekusi pekerjaan. Berhati-hatilah dan pastikan bahwa setiap perangkat yang menerapkan pekerjaan yang telah dihapus dapat pulih ke status yang valid.

Diperlukan beberapa waktu untuk menghapus pekerjaan, tergantung pada jumlah eksekusi pekerjaan yang dibuat untuk pekerjaan dan faktor lainnya. Sementara pekerjaan sedang dihapus, `DELETION_IN_PROGRESS` muncul sebagai status pekerjaan. Hasil galat jika Anda mencoba menghapus atau membatalkan pekerjaan dengan status yang sudah `DELETION_IN_PROGRESS`.

Hanya 10 pekerjaan yang dapat memiliki status `DELETION_IN_PROGRESS` pada saat yang sama. Jika tidak, `LimitExceededException` terjadi.

## Dapatkan dokumen pekerjaan

Untuk mengambil dokumen pekerjaan untuk suatu pekerjaan, gunakan `GetJobDocument` perintah. Dokumen pekerjaan adalah deskripsi operasi jarak jauh yang akan dilakukan oleh perangkat.

Untuk mendapatkan dokumen pekerjaan, jalankan perintah berikut:

```
aws iot get-job-document --job-id 010
```

Perintah mengembalikan dokumen pekerjaan untuk pekerjaan yang ditentukan:

```
{
 "document": "{\n\t\"operation\": \"install\",\n\t\"url\": \"http://amazon.com/firmWareUdate-01\",\n\t\"data\": \"${aws:iot:s3-presigned-url:https://s3.amazonaws.com/job-test-bucket/datafile}\"\n}"
}
```

### Note

Saat Anda menggunakan perintah ini untuk mengambil dokumen pekerjaan, URL placeholder tidak diganti dengan URL Amazon S3 yang telah ditandatangani sebelumnya. Saat perangkat memanggil [GetPendingJobExecutions](#) Operasi API, URL placeholder diganti dengan URL Amazon S3 yang telah ditandatangani sebelumnya dalam dokumen pekerjaan.

## Lowongan kerja List

Untuk mendapatkan daftar semua pekerjaan di Akun AWS, gunakan `ListJobs` perintah. Data pekerjaan dan data eksekusi pekerjaan disimpan untuk [waktu terbatas](#). Jalankan perintah berikut untuk mencantumkan semua pekerjaan di Akun AWS:

```
aws iot list-jobs
```

Perintah mengembalikan semua pekerjaan di akun Anda, diurutkan berdasarkan status pekerjaan:

```
{
 "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"
 }
]
}

```

## Jelaskan pekerjaan

Untuk mendapatkan status pekerjaan, jalankan `DescribeJob` perintah. Perintah berikut menunjukkan bagaimana untuk menggambarkan pekerjaan:

```
$ aws iot describe-job --job-id 010
```

Perintah mengembalikan status pekerjaan yang ditentukan. Misalnya:

```

{
 "documentSource": "https://s3.amazonaws.com/job-test-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
}
}
```



## Eksekusi daftar untuk pekerjaan

Pekerjaan yang berjalan pada perangkat tertentu diwakili oleh objek eksekusi pekerjaan. Jalankan `ListJobExecutionsForJob` perintah untuk daftar semua eksekusi pekerjaan untuk pekerjaan. Berikut ini menunjukkan bagaimana daftar eksekusi untuk pekerjaan:

```
aws iot list-job-executions-for-job --job-id 010
```

Perintah mengembalikan daftar eksekusi pekerjaan:

```
{
 "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
 }
 }
]
}
```

## Mencantumkan eksekusi pekerjaan untuk suatu hal

Jalankan `ListJobExecutionsForThing` perintah untuk daftar semua eksekusi pekerjaan yang berjalan pada suatu hal. Berikut ini menunjukkan cara membuat daftar eksekusi pekerjaan untuk suatu hal:

```
aws iot list-job-executions-for-thing --thing-name thingOne
```

Perintah mengembalikan daftar eksekusi pekerjaan yang sedang berjalan atau telah berjalan pada hal yang ditentukan:

```
{
 "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"
 }
]
}
```

```
}
]
}
```

## Jelaskan eksekusi pekerjaan

Jalankan `DescribeJobExecution` perintah untuk mendapatkan status eksekusi pekerjaan. Anda harus menentukan ID pekerjaan dan nama benda dan, opsional, nomor eksekusi untuk mengidentifikasi eksekusi pekerjaan. Berikut ini menunjukkan cara menggambarkan eksekusi pekerjaan:

```
aws iot describe-job-execution --job-id 017 --thing-name thingOne
```

Perintah mengembalikan [JobExecution](#). Misalnya:

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

## Hapus eksekusi pekerjaan

Jalankan `DeleteJobExecution` perintah untuk menghapus eksekusi pekerjaan. Anda harus menentukan ID pekerjaan, nama hal, dan nomor eksekusi untuk mengidentifikasi eksekusi pekerjaan. Berikut ini menunjukkan cara menghapus eksekusi pekerjaan:

```
aws iot delete-job-execution --job-id 017 --thing-name thingOne --execution-number
1234567890 --force|--no-force
```

Perintah tidak menampilkan output.

Secara default, status eksekusi pekerjaan harus `QUEUED` atau dalam keadaan terminal (`SUCCEEDED`, `FAILED`, `REJECTED`, `TIMED_OUT`, `REMOVED`, atau `CANCELED`). Jika tidak, muncul kesalahan. Untuk menghapus eksekusi pekerjaan dengan status `IN_PROGRESS`, Anda dapat mengatur `force` parameter ke `true`.

#### Warning

Ketika Anda menghapus eksekusi pekerjaan dengan status `IN_PROGRESS`, perangkat yang menjalankan tugas tidak dapat mengakses informasi pekerjaan atau memperbarui status eksekusi pekerjaan. Berhati-hatilah dan pastikan perangkat dapat pulih ke keadaan yang valid.

## Templat Job

Gunakan templat pekerjaan untuk mengkonfigurasi pekerjaan yang dapat Anda terapkan ke beberapa set perangkat target. Untuk menerapkan tindakan jarak jauh yang sering dilakukan ke perangkat Anda, seperti me-reboot atau menginstal aplikasi, Anda dapat menggunakan template untuk menentukan konfigurasi standar. Untuk melakukan operasi seperti menerapkan patch keamanan dan perbaikan bug, Anda dapat membuat template dari pekerjaan yang ada.

Saat membuat templat pekerjaan, tentukan konfigurasi dan sumber daya tambahan berikut.

- Properti Job
- Dokumen dan target pekerjaan
- Kriteria peluncuran, penjadwalan, dan pembatalan
- Kriteria batas waktu dan coba lagi

## Template khusus dan AWS terkelola

Bergantung pada tindakan jarak jauh yang ingin Anda lakukan, Anda dapat membuat templat pekerjaan khusus atau menggunakan templat AWS terkelola. Gunakan templat pekerjaan khusus

untuk menyediakan dokumen pekerjaan kustom Anda sendiri dan buat pekerjaan yang dapat digunakan kembali untuk diterapkan ke perangkat Anda. AWS template terkelola adalah template pekerjaan yang disediakan oleh AWS IoT Jobs untuk tindakan yang umum dilakukan. Template ini memiliki dokumen pekerjaan yang telah ditentukan untuk beberapa tindakan jarak jauh sehingga Anda tidak perlu membuat dokumen pekerjaan Anda sendiri. Template terkelola membantu Anda membuat pekerjaan yang dapat digunakan kembali untuk peluncuran yang lebih cepat ke perangkat Anda.

## Topik

- [Gunakan templat AWS terkelola untuk menerapkan operasi jarak jauh yang umum](#)
- [Buat templat pekerjaan khusus](#)

## Gunakan templat AWS terkelola untuk menerapkan operasi jarak jauh yang umum

AWS template terkelola adalah template pekerjaan yang disediakan oleh AWS. Mereka digunakan untuk tindakan jarak jauh yang sering dilakukan seperti me-reboot, mengunduh file, atau menginstal aplikasi di perangkat Anda. Template ini memiliki dokumen pekerjaan yang telah ditentukan untuk setiap tindakan jarak jauh sehingga Anda tidak perlu membuat dokumen pekerjaan Anda sendiri.

Anda dapat memilih dari satu set konfigurasi yang telah ditentukan dan membuat pekerjaan menggunakan template ini tanpa menulis kode tambahan apa pun. Dengan menggunakan templat terkelola, Anda dapat melihat dokumen pekerjaan yang disebarkan ke armada Anda. Anda dapat membuat pekerjaan menggunakan template ini dan membuat template pekerjaan khusus yang dapat Anda gunakan kembali untuk operasi jarak jauh Anda.

## Apa isi template terkelola?

Setiap template yang AWS dikelola berisi:

- Lingkungan untuk menjalankan perintah dalam dokumen pekerjaan.
- Dokumen pekerjaan yang menentukan nama operasi dan parameternya. Misalnya, jika Anda menggunakan templat file Unduh, nama operasinya adalah Unduh file dan parameternya dapat berupa:
  - URL file yang ingin Anda unduh ke perangkat Anda. Ini bisa berupa sumber daya internet atau URL Amazon Simple Storage Service ((Amazon S3) publik atau yang telah ditandatangani sebelumnya.

- Jalur file lokal pada perangkat untuk menyimpan file yang diunduh.

Untuk informasi lebih lanjut tentang dokumen pekerjaan dan parameternya, lihat [Tindakan jarak jauh template terkelola dan dokumen pekerjaan](#).

## Prasyarat

Agar perangkat dapat menjalankan tindakan jarak jauh yang ditentukan oleh dokumen pekerjaan templat terkelola, Anda harus:

- Instal perangkat lunak tertentu di perangkat Anda

Gunakan perangkat lunak perangkat Anda sendiri dan penanganan pekerjaan, atau Klien AWS IoT Perangkat. Tergantung pada kasus bisnis Anda, Anda juga dapat menjalankan keduanya sehingga mereka melakukan fungsi yang berbeda.

- Gunakan perangkat lunak perangkat Anda sendiri dan penanganan pekerjaan

Anda dapat menulis kode Anda sendiri untuk perangkat dengan menggunakan AWS IoT Device SDK dan perpustakaan penanganan yang mendukung operasi jarak jauh. Untuk menerapkan dan menjalankan pekerjaan, verifikasi bahwa pustaka agen perangkat telah diinstal dengan benar dan berjalan di perangkat.

Anda juga dapat memilih untuk menggunakan penanganan Anda sendiri yang mendukung operasi jarak jauh. Untuk informasi selengkapnya, lihat [Contoh penanganan pekerjaan](#) di GitHub repositori Klien AWS IoT Perangkat.

- Gunakan Klien AWS IoT Perangkat

Atau, Anda dapat menginstal dan menjalankan AWS IoT Device Client di perangkat Anda karena mendukung penggunaan semua template terkelola langsung dari konsol secara default.

Device Client adalah perangkat lunak sumber terbuka yang ditulis dalam C++ yang dapat Anda kompilasi dan instal pada perangkat IoT berbasis Linux yang disematkan. Klien Perangkat memiliki klien dasar dan fitur sisi klien diskrit. Klien dasar membangun konektivitas dengan AWS IoT lebih dari protokol MQTT dan dapat terhubung dengan fitur sisi klien yang berbeda.

Untuk melakukan operasi jarak jauh pada perangkat Anda, gunakan fitur Pekerjaan sisi klien dari Klien Perangkat. Fitur ini berisi parser untuk menerima dokumen pekerjaan dan penanganan pekerjaan yang menerapkan tindakan jarak jauh yang ditentukan dalam dokumen pekerjaan.

Untuk informasi selengkapnya tentang Klien Perangkat dan fitur-fiturnya, lihat [Klien AWS IoT Perangkat](#).

Saat berjalan di perangkat, Klien Perangkat menerima dokumen pekerjaan dan memiliki implementasi khusus platform yang digunakan untuk menjalankan perintah dalam dokumen. Untuk informasi selengkapnya tentang pengaturan Device Client dan menggunakan fitur Jobs, lihat [AWS IoT tutorial](#).

- Gunakan lingkungan yang didukung

Untuk setiap template terkelola, Anda akan menemukan informasi tentang lingkungan yang dapat Anda gunakan untuk menjalankan tindakan jarak jauh. Kami menyarankan Anda menggunakan template dengan lingkungan Linux yang didukung seperti yang ditentukan dalam template. Gunakan AWS IoT Device Client untuk menjalankan tindakan jarak jauh template terkelola karena mendukung mikroprosesor umum dan lingkungan Linux, seperti Debian dan Ubuntu.

## Tindakan jarak jauh template terkelola dan dokumen pekerjaan

Bagian berikut mencantumkan berbagai templat AWS terkelola untuk AWS IoT Pekerjaan, dan menjelaskan tindakan jarak jauh yang dapat dilakukan pada perangkat. Bagian berikut memiliki informasi tentang dokumen pekerjaan dan deskripsi parameter dokumen pekerjaan untuk setiap tindakan jarak jauh. Perangkat lunak sisi perangkat Anda menggunakan nama templat dan parameter untuk melakukan tindakan jarak jauh.

AWS template terkelola menerima parameter input yang Anda tentukan nilainya saat membuat pekerjaan menggunakan templat. Semua template yang dikelola memiliki dua parameter input opsional yang sama: `runAsUser` dan `pathToHandler`. Kecuali untuk AWS-`Reboot` template, template memerlukan parameter input tambahan yang harus Anda tentukan nilainya saat membuat pekerjaan menggunakan templat. Parameter input yang diperlukan ini bervariasi tergantung pada template yang Anda pilih. Misalnya, jika Anda memilih AWS-`Download-File` template, Anda harus menentukan daftar paket untuk menginstal, dan URL untuk men-download file dari.

Tentukan nilai untuk parameter input saat menggunakan AWS IoT konsol atau AWS Command Line Interface (AWS CLI) untuk membuat pekerjaan yang menggunakan templat terkelola. Saat menggunakan CLI, berikan nilai-nilai ini dengan menggunakan objek `document-parameters`. Untuk informasi selengkapnya, lihat [DocumentParameters](#).

**Note**

Gunakan `document-parameters` hanya saat membuat pekerjaan dari template AWS terkelola. Parameter ini tidak dapat digunakan dengan template pekerjaan khusus atau untuk membuat pekerjaan dari mereka.

Berikut ini menunjukkan deskripsi parameter input opsional umum. Anda akan melihat deskripsi parameter input lain yang diperlukan oleh setiap template terkelola di bagian berikutnya.

**runAsUser**

Parameter ini menentukan apakah akan menjalankan job handler sebagai pengguna lain. Jika tidak ditentukan selama pembuatan pekerjaan, penanganan pekerjaan dijalankan sebagai pengguna yang sama dengan Klien Perangkat. Saat Anda menjalankan job handler sebagai pengguna lain, tentukan nilai string yang tidak lebih dari 256 karakter.

**pathToHandler**

Jalur ke penanganan pekerjaan yang berjalan di perangkat. Jika tidak ditentukan selama pembuatan pekerjaan, Klien Perangkat menggunakan direktori kerja saat ini.

Berikut ini menunjukkan berbagai tindakan jarak jauh, dokumen pekerjaan mereka, dan parameter yang mereka terima. Semua template ini mendukung lingkungan Linux untuk menjalankan operasi jarak jauh pada perangkat.

**AWS—Unduh—File****Nama templat****AWS-Download-File****Deskripsi templat**

Template terkelola yang disediakan oleh AWS untuk mengunduh file.

**Parameter input**

Template ini memiliki parameter yang diperlukan berikut. Anda juga dapat menentukan parameter opsional `runAsUser` dan `pathToHandler`.



## downloadUrl

URL untuk mengunduh file dari. Ini bisa berupa sumber daya internet, objek di Amazon S3 yang dapat diakses publik, atau objek di Amazon S3 yang hanya dapat diakses oleh perangkat Anda menggunakan URL yang telah ditetapkan sebelumnya. Untuk informasi selengkapnya tentang penggunaan URL yang telah ditetapkan sebelumnya dan pemberian izin, lihat. [URLs yang ditandatangani sebelumnya](#)

## filePath

Jalur file lokal yang menunjukkan lokasi di perangkat untuk menyimpan file yang diunduh.

## Perilaku perangkat

Perangkat mengunduh file dari lokasi yang ditentukan, memverifikasi bahwa unduhan selesai, dan menyimpannya secara lokal.

## Dokumen Job

Berikut ini menunjukkan dokumen pekerjaan dan versi terbarunya. Template menunjukkan path ke job handler dan shell script, `download-file.sh`, yang harus dijalankan oleh job handler untuk men-download file. Ini juga menunjukkan parameter yang diperlukan `downloadUrl` dan `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—Instal—Aplikasi

### Nama templat

AWS-Install-Application

### Deskripsi templat

Template terkelola yang disediakan oleh AWS untuk menginstal satu atau lebih aplikasi.

### Parameter input

Template ini memiliki parameter yang diperlukan sebagai berikut, `packages`. Anda juga dapat menentukan parameter opsional `runAsUser` dan `pathToHandler`.

### `packages`

Daftar terpisah spasi dari satu atau lebih aplikasi yang akan diinstal.

### Perilaku perangkat

Perangkat menginstal aplikasi seperti yang ditentukan dalam dokumen pekerjaan.

### Dokumen Job

Berikut ini menunjukkan dokumen pekerjaan dan versi terbarunya. Template menunjukkan path ke job handler dan shell script, `install-packages.sh`, yang harus dijalankan oleh job handler untuk men-download file. Ini juga menunjukkan parameter yang diperlukan `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

### Nama templat

AWS-**Reboot**

### Deskripsi templat

Template terkelola yang disediakan oleh AWS untuk me-reboot perangkat Anda.

### Parameter input

Template ini tidak memiliki parameter yang diperlukan. Anda dapat menentukan parameter opsional `runAsUser` dan `pathToHandler`.

### Perilaku perangkat

Perangkat berhasil reboot.

### Dokumen Job

Berikut ini menunjukkan dokumen pekerjaan dan versi terbarunya. Template menunjukkan path ke job handler dan shell script, `reboot.sh`, yang harus dijalankan oleh job handler untuk me-reboot perangkat.

```
{
 "version": "1.0",
 "steps": [
 {
 "action": {
 "name": "Reboot",
 "type": "runHandler",
 "input": {
 "handler": "reboot.sh",
 "path": "${aws:iot:parameter:pathToHandler}"
 }
 },
 "runAsUser": "${aws:iot:parameter:runAsUser}"
 }
]
}
```

```
 }
 }
]
}
```

## AWS—Hapus—Aplikasi

### Nama templat

AWS-Remove-Application

### Deskripsi templat

Template terkelola yang disediakan oleh AWS untuk menghapus satu atau beberapa aplikasi.

### Parameter input

Template ini memiliki parameter yang diperlukan sebagai berikut, `packages`. Anda juga dapat menentukan parameter opsional `runAsUser` dan `pathToHandler`.

### `packages`

Daftar terpisah spasi dari satu atau lebih aplikasi yang akan dihapus instalasinya.

### Perilaku perangkat

Perangkat menghapus instalasi aplikasi seperti yang ditentukan dalam dokumen pekerjaan.

### Dokumen Job

Berikut ini menunjukkan dokumen pekerjaan dan versi terbarunya. Template menunjukkan path ke job handler dan shell script, `remove-packages.sh`, yang harus dijalankan oleh job handler untuk men-download file. Ini juga menunjukkan parameter yang diperlukan `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 — Aplikasi

### Nama templat

AWS-Restart-Application

### Deskripsi templat

Template terkelola yang disediakan oleh AWS untuk menghentikan dan memulai ulang satu atau beberapa layanan.

### Parameter input

Template ini memiliki parameter yang diperlukan sebagai berikut, `services`. Anda juga dapat menentukan parameter opsional `runAsUser` dan `pathToHandler`.

### Layanan

Daftar terpisah spasi dari satu atau lebih aplikasi yang akan dimulai ulang.

### Perilaku perangkat

Aplikasi yang ditentukan dihentikan dan kemudian dimulai ulang pada perangkat.

### Dokumen Job

Berikut ini menunjukkan dokumen pekerjaan dan versi terbarunya. Template menunjukkan path ke job handler dan shell script `restart-services.sh`, yang harus dijalankan oleh job handler untuk me-restart layanan sistem. Ini juga menunjukkan parameter yang diperlukan `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—Mulai—Aplikasi

### Nama templat

AWS-Start-Application

### Deskripsi templat

Template terkelola yang disediakan oleh AWS untuk memulai satu atau lebih layanan.

### Parameter input

Template ini memiliki parameter yang diperlukan sebagai berikut, `services`. Anda juga dapat menentukan parameter opsional `runAsUser` dan `pathToHandler`.

#### `services`

Daftar terpisah spasi dari satu atau lebih aplikasi yang akan dimulai.

### Perilaku perangkat

Aplikasi yang ditentukan mulai berjalan pada perangkat.

### Dokumen Job

Berikut ini menunjukkan dokumen pekerjaan dan versi terbarunya. Template menunjukkan path ke job handler dan shell script `start-services.sh`, yang harus dijalankan oleh job handler untuk memulai layanan sistem. Ini juga menunjukkan parameter yang diperlukan `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—Berhenti—Aplikasi

### Nama templat

### AWS-Stop-Application

### Deskripsi templat

Template terkelola yang disediakan oleh AWS untuk menghentikan satu atau beberapa layanan.

### Parameter input

Template ini memiliki parameter yang diperlukan sebagai berikut, `services`. Anda juga dapat menentukan parameter opsional `runAsUser` dan `pathToHandler`.

### `services`

Daftar terpisah spasi dari satu atau lebih aplikasi yang akan dihentikan.

### Perilaku perangkat

Aplikasi yang ditentukan berhenti berjalan pada perangkat.

### Dokumen Job

Berikut ini menunjukkan dokumen pekerjaan dan versi terbarunya. Template menunjukkan path ke job handler dan shell script `stop-services.sh`, yang harus dijalankan oleh job handler untuk menghentikan layanan sistem. Ini juga menunjukkan parameter yang diperlukan `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—Jalankan — Perintah

Nama templat

AWS-Run-Command

Deskripsi templat

Template terkelola yang disediakan oleh AWS untuk menjalankan perintah shell.

Parameter input

Template ini memiliki parameter yang diperlukan sebagai berikut, `command`. Anda juga dapat menentukan parameter opsional `runAsUser`.

`command`

Sebuah string perintah yang dipisahkan koma. Setiap koma yang terkandung dalam perintah itu sendiri harus lolos.



## Perilaku perangkat

Perangkat menjalankan perintah shell seperti yang ditentukan dalam dokumen pekerjaan.

## Dokumen Job

Berikut ini menunjukkan dokumen pekerjaan dan versi terbarunya. Template menunjukkan jalur ke perintah pekerjaan dan perintah yang Anda berikan yang akan dijalankan perangkat.

```
{
 "version": "1.0",
 "steps": [
 {
 "action": {
 "name": "Run-Command",
 "type": "runCommand",
 "input": {
 "command": "${aws:iot:parameter:command}"
 },
 "runAsUser": "${aws:iot:parameter:runAsUser}"
 }
 }
]
}
```

## Topik

- [Buat pekerjaan dari template AWS terkelola dengan menggunakan AWS Management Console](#)
- [Buat pekerjaan dari template AWS terkelola dengan menggunakan AWS CLI](#)

## Buat pekerjaan dari template AWS terkelola dengan menggunakan AWS Management Console

Gunakan AWS Management Console untuk mendapatkan informasi tentang templat AWS terkelola dan buat pekerjaan dengan menggunakan templat ini. Anda kemudian dapat menyimpan pekerjaan yang Anda buat sebagai template kustom Anda sendiri.

## Dapatkan detail tentang templat terkelola

Anda bisa mendapatkan informasi tentang berbagai templat terkelola yang tersedia untuk digunakan dari AWS IoT konsol.

1. Untuk melihat templat terkelola yang tersedia, buka [hub Job templates di AWS IoT konsol](#) dan pilih tab Managed templates.
2. Untuk melihat detail, pilih templat terkelola.

Halaman detail berisi informasi berikut:

- Nama, deskripsi, dan Nama Sumber Daya Amazon (ARN) dari template yang dikelola.
- Lingkungan di mana operasi jarak jauh dapat dilakukan, seperti Linux.
- Dokumen pekerjaan JSON yang menentukan jalur ke penanganan pekerjaan dan perintah untuk dijalankan pada perangkat. Misalnya, berikut ini menunjukkan contoh dokumen pekerjaan untuk template AWS-reboot. Template menunjukkan path ke job handler dan shell script, `reboot.sh`, yang harus dijalankan oleh job handler untuk me-reboot perangkat.

```
{
 "version": "1.0",
 "steps": [
 {
 "action": {
 "name": "Reboot",
 "type": "runHandler",
 "input": {
 "handler": "reboot.sh",
 "path": "${aws:iot:parameter:pathToHandler}"
 },
 "runAsUser": "${aws:iot:parameter:runAsUser}"
 }
 }
]
}
```

Untuk informasi selengkapnya tentang dokumen pekerjaan dan parameternya untuk berbagai tindakan jarak jauh, lihat [Tindakan jarak jauh template terkelola dan dokumen pekerjaan](#).

- Versi terbaru dari dokumen pekerjaan.

### Buat pekerjaan menggunakan templat terkelola

Anda dapat menggunakan konsol AWS Manajemen untuk memilih templat AWS terkelola yang akan digunakan untuk membuat pekerjaan. Dalam bagian ini, akan ditunjukkan caranya.

Anda juga dapat memulai alur kerja pembuatan pekerjaan dan kemudian memilih template AWS terkelola yang ingin Anda gunakan saat membuat pekerjaan. Untuk informasi selengkapnya tentang alur kerja ini, lihat [Buat dan kelola pekerjaan dengan menggunakan AWS Management Console](#).

1. Pilih template AWS terkelola Anda

Buka [hub Job templates di AWS IoT konsol](#), pilih tab Managed templates, lalu pilih template Anda.

2. Buat pekerjaan menggunakan template terkelola Anda

1. Di halaman detail template Anda, pilih Buat pekerjaan.

Konsol beralih ke langkah Properti pekerjaan kustom dari alur kerja Buat pekerjaan di mana konfigurasi template Anda telah ditambahkan.

2. Masukkan nama pekerjaan alfanumerik yang unik, serta deskripsi dan tag opsional, lalu pilih Berikutnya.

3. Pilih hal-hal atau kelompok benda sebagai target pekerjaan yang ingin Anda jalankan dalam pekerjaan ini.

4. Di bagian Job document, template Anda ditampilkan dengan pengaturan konfigurasi dan parameter input. Masukkan nilai untuk parameter input template yang Anda pilih. Misalnya, jika Anda memilih template AWS-Download-File:

- Untuk `DownloadUrl`, masukkan URL file yang akan diunduh, misalnya: `https://example.com/index.html`.
- Untuk `FilePath`, masukkan jalur pada perangkat untuk menyimpan file yang diunduh, misalnya: `path/to/file`

Anda juga dapat secara opsional memasukkan nilai untuk parameter `runAsUser` dan `pathToHandler`. Untuk informasi selengkapnya tentang parameter masukan dari setiap template, lihat [Tindakan jarak jauh template terkelola dan dokumen pekerjaan](#).

5. Pada halaman konfigurasi Job, pilih jenis pekerjaan sebagai pekerjaan berkelanjutan atau snapshot. Pekerjaan snapshot selesai ketika selesai dijalankan pada perangkat dan grup target. Pekerjaan berkelanjutan berlaku untuk grup benda dan berjalan di perangkat apa pun yang Anda tambahkan ke grup target tertentu.

6. Terus tambahkan konfigurasi tambahan untuk pekerjaan Anda dan kemudian tinjau dan buat pekerjaan Anda. Untuk informasi tentang konfigurasi tambahan, lihat:

- [Peluncuran pekerjaan, penjadwalan, dan membatalkan konfigurasi](#)

- [Konfigurasi batas waktu eksekusi pekerjaan dan coba lagi](#)

Buat templat pekerjaan khusus dari templat terkelola

Anda dapat menggunakan template AWS terkelola dan pekerjaan khusus sebagai titik awal untuk membuat template pekerjaan kustom Anda sendiri. Untuk membuat templat pekerjaan khusus, pertama-tama buat pekerjaan dari templat AWS terkelola Anda seperti yang dijelaskan di bagian sebelumnya.

Anda kemudian dapat menyimpan pekerjaan kustom sebagai template untuk membuat template pekerjaan kustom Anda sendiri. Untuk menyimpan sebagai template:

1. Buka [hub Job AWS IoT konsol](#) dan pilih job yang berisi template terkelola Anda.
2. Pilih Simpan sebagai templat pekerjaan dan kemudian buat templat pekerjaan khusus Anda. Untuk informasi selengkapnya tentang membuat templat pekerjaan kustom, lihat [Buat template pekerjaan dari pekerjaan yang ada](#).

Buat pekerjaan dari template AWS terkelola dengan menggunakan AWS CLI

Gunakan AWS CLI untuk mendapatkan informasi tentang templat AWS terkelola dan buat pekerjaan dengan menggunakan templat ini. Anda kemudian dapat menyimpan pekerjaan sebagai template dan kemudian membuat template kustom Anda sendiri.

Daftar template terkelola

[list-managed-job-templates](#) AWS CLI Perintah ini mencantumkan semua template pekerjaan di Anda Akun AWS.

```
aws iot list-managed-job-templates
```

Secara default, menjalankan perintah ini menampilkan semua template AWS terkelola yang tersedia dan detailnya.

```
{
 "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"
 }
]

```

```
}
```

Untuk informasi lebih lanjut, lihat [ListManagedJobTemplates](#).

Dapatkan detail tentang template terkelola

[describe-managed-job-template](#) AWS CLI Perintah mendapatkan rincian tentang template pekerjaan tertentu. Tentukan nama template pekerjaan dan versi templat opsional. Jika versi template tidak ditentukan, versi default yang telah ditentukan akan dikembalikan. Berikut ini menunjukkan contoh menjalankan perintah untuk mendapatkan rincian tentang AWS-Download-File template.

```
aws iot describe-managed-job-template \
 --template-name AWS-Download-File
```

Perintah menampilkan detail template dan ARN, dokumen pekerjaannya, dan `documentParameters` parameternya, yang merupakan daftar pasangan nilai kunci dari parameter input template. Untuk informasi tentang berbagai templat dan parameter input, lihat [Tindakan jarak jauh template terkelola dan dokumen pekerjaan](#).

#### Note

`documentParameters` Objek yang dikembalikan saat Anda menggunakan API ini hanya boleh digunakan saat membuat pekerjaan dari templat AWS terkelola. Objek tidak boleh digunakan untuk template pekerjaan khusus. Untuk contoh yang menunjukkan cara menggunakan parameter ini, lihat [Buat pekerjaan dengan menggunakan templat terkelola](#).

```
{
 "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}\" } }] } }"
}

```

Untuk informasi lebih lanjut, lihat [DescribeManagedJobTemplate](#).

Buat pekerjaan dengan menggunakan templat terkelola

[create-job](#) AWS CLI Perintah ini dapat digunakan untuk membuat pekerjaan dari template pekerjaan. Ini menargetkan perangkat bernama thingOne dan menentukan Nama Sumber Daya

Amazon (ARN) dari template terkelola untuk digunakan sebagai dasar untuk pekerjaan itu. Anda dapat mengganti konfigurasi lanjutan, seperti batas waktu dan membatalkan konfigurasi, dengan meneruskan parameter terkait perintah. `create-job`

Contoh menunjukkan cara membuat pekerjaan yang menggunakan `AWS-Download-File` template. Ini juga menunjukkan cara menentukan parameter input template dengan menggunakan `document-parameters` parameter.

### Note

Gunakan `document-parameters` objek hanya dengan template AWS terkelola. Objek ini tidak boleh digunakan dengan template pekerjaan khusus.

```
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
```

di mana:

- *wilayah* adalah Wilayah AWS.
- *Account-ID* adalah nomor unik Akun AWS .
- *thingOne* adalah nama hal IoT yang menjadi sasaran pekerjaan.
- *AWS-Download-File:1.0* adalah nama template yang dikelola.
- `https://example.com/index.html` adalah URL untuk mengunduh file dari.
- `https://path/to/file/index` adalah jalur pada perangkat untuk menyimpan file yang diunduh.

Jalankan perintah berikut untuk membuat pekerjaan untuk template, *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."
}
```



## Buat templat pekerjaan khusus dari templat terkelola

1. Buat pekerjaan menggunakan template terkelola seperti yang dijelaskan di bagian sebelumnya.
2. Buat template pekerjaan khusus dengan menggunakan ARN dari pekerjaan yang Anda buat.  
Untuk informasi selengkapnya, lihat [Buat template pekerjaan dari pekerjaan yang ada](#).

## Buat templat pekerjaan khusus

Anda dapat membuat template pekerjaan dengan menggunakan AWS CLI dan AWS IoT konsol. Anda juga dapat membuat pekerjaan dari templat pekerjaan dengan menggunakan aplikasi web AWS CLI, AWS IoT konsol, dan Fleet Hub for AWS IoT Device Management. Untuk informasi selengkapnya tentang bekerja dengan templat pekerjaan di aplikasi Fleet Hub, lihat [Bekerja dengan templat pekerjaan di Fleet Hub for AWS IoT Device Management](#).

### Note

Jumlah total pola substitusi dalam dokumen pekerjaan harus kurang dari atau sama dengan sepuluh.

### Topik

- [Buat templat pekerjaan khusus dengan menggunakan AWS Management Console](#)
- [Buat templat pekerjaan khusus dengan menggunakan AWS CLI](#)

## Buat templat pekerjaan khusus dengan menggunakan AWS Management Console

Topik ini menjelaskan cara membuat, menghapus, dan melihat detail tentang templat pekerjaan menggunakan AWS IoT konsol.

### Buat template pekerjaan khusus

Anda dapat membuat template pekerjaan kustom asli atau membuat template pekerjaan dari pekerjaan yang ada. Anda juga dapat membuat template pekerjaan kustom dari pekerjaan yang ada yang dibuat menggunakan template AWS terkelola. Untuk informasi selengkapnya, lihat [Buat templat pekerjaan khusus dari templat terkelola](#).

## Buat template pekerjaan asli

### 1. Mulai membuat template pekerjaan Anda

1. Buka [hub Job templates di AWS IoT konsol](#) dan pilih tab Custom templates.
2. Pilih Buat template pekerjaan.

#### Note

Anda juga dapat menavigasi ke halaman Templat Job dari halaman Layanan terkait di Fleet Hub.

### 2. Tentukan properti template pekerjaan

Di halaman Buat templat pekerjaan, masukkan pengidentifikasi alfanumerik untuk nama pekerjaan Anda dan deskripsi alfanumerik untuk memberikan detail tambahan tentang templat tersebut.

#### Note

Kami tidak menyarankan untuk menggunakan informasi identitas pribadi di ID atau deskripsi pekerjaan Anda.

### 3. Berikan dokumen pekerjaan

Berikan file pekerjaan JSON yang disimpan dalam bucket S3 atau sebagai dokumen pekerjaan inline yang ditentukan dalam pekerjaan. File pekerjaan ini akan menjadi dokumen pekerjaan saat Anda membuat pekerjaan menggunakan template ini.

Jika file pekerjaan disimpan dalam bucket S3, masukkan URL S3 atau pilih Browse S3, lalu navigasikan ke dokumen pekerjaan Anda dan pilih.

#### Note

Anda hanya dapat memilih bucket S3 di Wilayah Anda saat ini.

### 4. Terus tambahkan konfigurasi tambahan untuk pekerjaan Anda dan kemudian tinjau dan buat pekerjaan Anda. Untuk informasi tentang konfigurasi tambahan dan opsional, lihat tautan berikut:

- [Peluncuran pekerjaan, penjadwalan, dan membatalkan konfigurasi](#)
- [Konfigurasi batas waktu eksekusi pekerjaan dan coba lagi](#)

Buat template pekerjaan dari pekerjaan yang ada

## 1. Pilih pekerjaan Anda

1. Pergi ke [hub Job AWS IoT konsol](#) dan pilih pekerjaan yang ingin Anda gunakan sebagai dasar untuk template pekerjaan Anda.
2. Pilih Simpan sebagai templat pekerjaan.

### Note

Secara opsional, Anda dapat memilih dokumen pekerjaan yang berbeda atau mengedit konfigurasi lanjutan dari pekerjaan asli, lalu memilih Buat templat pekerjaan. Template pekerjaan baru Anda muncul di halaman Job templates.

## 2. Tentukan properti template pekerjaan

Di halaman Buat templat pekerjaan, masukkan pengidentifikasi alfanumerik untuk nama pekerjaan Anda dan deskripsi alfanumerik untuk memberikan detail tambahan tentang templat tersebut.

### Note

Dokumen pekerjaan adalah file pekerjaan yang Anda tentukan saat membuat template. Jika dokumen pekerjaan ditentukan dalam pekerjaan alih-alih lokasi S3, Anda dapat melihat dokumen pekerjaan di halaman detail pekerjaan ini.

## 3. Terus tambahkan konfigurasi tambahan untuk pekerjaan Anda dan kemudian tinjau dan buat pekerjaan Anda. Untuk informasi tentang konfigurasi tambahan, lihat:

- [Peluncuran pekerjaan, penjadwalan, dan membatalkan konfigurasi](#)
- [Konfigurasi batas waktu eksekusi pekerjaan dan coba lagi](#)

## Buat pekerjaan dari template pekerjaan khusus

Anda dapat membuat pekerjaan dari template pekerjaan khusus dengan membuka halaman detail template pekerjaan Anda seperti yang dijelaskan dalam topik ini. Anda juga dapat membuat pekerjaan atau dengan memilih template pekerjaan yang ingin Anda gunakan saat menjalankan alur kerja pembuatan pekerjaan. Untuk informasi selengkapnya, lihat [Buat dan kelola pekerjaan dengan menggunakan AWS Management Console](#).

Topik ini menunjukkan cara membuat pekerjaan dari halaman detail template pekerjaan khusus. Anda juga dapat membuat pekerjaan dari template yang AWS dikelola. Untuk informasi selengkapnya, lihat [Buat pekerjaan menggunakan templat terkelola](#).

### 1. Pilih templat pekerjaan khusus Anda

Buka [hub Job templates di AWS IoT konsol](#) dan pilih tab Custom templates, lalu pilih template Anda.

### 2. Buat pekerjaan menggunakan template kustom Anda

Untuk membuat pekerjaan:

#### 1. Di halaman detail template Anda, pilih Buat pekerjaan.

Konsol beralih ke langkah Properti pekerjaan kustom dari alur kerja Buat pekerjaan di mana konfigurasi template Anda telah ditambahkan.

#### 2. Masukkan nama pekerjaan alfanumerik yang unik, serta deskripsi dan tag opsional, lalu pilih Berikutnya.

#### 3. Pilih hal-hal atau kelompok benda sebagai target pekerjaan yang ingin Anda jalankan dalam pekerjaan ini.

Di bagian Dokumen Job, template Anda ditampilkan dengan pengaturan konfigurasinya. Jika Anda ingin menggunakan dokumen pekerjaan yang berbeda, pilih Browse dan pilih bucket dan dokumen yang berbeda. Pilih Berikutnya.

#### 4. Pada halaman konfigurasi Job, pilih jenis pekerjaan sebagai pekerjaan berkelanjutan atau snapshot. Pekerjaan snapshot selesai ketika selesai dijalankan pada perangkat dan grup target. Pekerjaan berkelanjutan berlaku untuk grup benda dan berjalan di perangkat apa pun yang Anda tambahkan ke grup target tertentu.

#### 5. Terus tambahkan konfigurasi tambahan untuk pekerjaan Anda dan kemudian tinjau dan buat pekerjaan Anda. Untuk informasi tentang konfigurasi tambahan, lihat:

- [Peluncuran pekerjaan, penjadwalan, dan membatalkan konfigurasi](#)
- [Konfigurasi batas waktu eksekusi pekerjaan dan coba lagi](#)

#### Note

Saat pekerjaan yang dibuat dari templat pekerjaan memperbarui parameter yang ada yang disediakan oleh templat pekerjaan, parameter yang diperbarui tersebut akan mengganti parameter yang ada yang disediakan oleh templat pekerjaan untuk pekerjaan itu.

Anda juga dapat membuat pekerjaan dari templat pekerjaan dengan aplikasi web Fleet Hub. Untuk informasi tentang membuat lowongan kerja di Fleet Hub, lihat [Bekerja dengan templat pekerjaan di Fleet Hub for AWS IoT Device Management](#).

### Hapus template pekerjaan

Untuk menghapus template pekerjaan, pertama-tama buka [hub Job templates di AWS IoT konsol](#) dan pilih tab Custom templates. Kemudian, pilih template pekerjaan yang ingin Anda hapus dan pilih Berikutnya.

#### Note

Penghapusan bersifat permanen dan templat pekerjaan tidak lagi muncul di tab Template kustom.

### Buat templat pekerjaan khusus dengan menggunakan AWS CLI

Topik ini menjelaskan cara membuat, menghapus, dan mengambil detail tentang template pekerjaan dengan menggunakan AWS CLI

#### Buat template pekerjaan dari awal

AWS CLI *Perintah berikut menunjukkan cara membuat pekerjaan menggunakan dokumen pekerjaan (job-document.json) yang disimpan dalam bucket S3 dan peran dengan izin untuk mengunduh file dari Amazon S3 (S3). DownloadRole*

```
aws iot create-job-template \
```

```

--job-template-id 010 \
--document-source https://s3.amazonaws.com/my-s3-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}"

```

`timeout-configParameter` opsional menentukan jumlah waktu yang dimiliki setiap perangkat untuk menyelesaikan menjalankan pekerjaan. Timer dimulai ketika status eksekusi pekerjaan diatur ke `IN_PROGRESS`. Jika status eksekusi pekerjaan tidak disetel ke status terminal lain sebelum waktu kedaluwarsa, status tersebut disetel ke `TIMED_OUT`.

Timer yang sedang berlangsung tidak dapat diperbarui dan berlaku untuk semua peluncuran pekerjaan untuk pekerjaan tersebut. Setiap kali peluncuran pekerjaan tetap dalam `IN_PROGRESS` status lebih lama dari interval ini, peluncuran pekerjaan gagal dan beralih ke `TIMED_OUT` status terminal. AWS IoT juga menerbitkan pemberitahuan MQTT.

Untuk informasi selengkapnya tentang membuat konfigurasi tentang peluncuran dan pembatalan pekerjaan, lihat [Peluncuran pekerjaan](#) dan membatalkan konfigurasi.

#### Note

Dokumen Job yang ditetapkan sebagai file Amazon S3 diambil pada saat Anda membuat pekerjaan. Jika Anda mengubah konten file Amazon S3 yang Anda gunakan sebagai sumber dokumen pekerjaan Anda setelah Anda membuat pekerjaan, apa yang dikirim ke target pekerjaan tidak berubah.

Buat template pekerjaan dari pekerjaan yang ada

AWS CLI Perintah berikut membuat template pekerjaan dengan menentukan Amazon Resource Name (ARN) dari pekerjaan yang ada. Template pekerjaan baru menggunakan semua konfigurasi yang ditentukan dalam pekerjaan. Secara opsional, Anda dapat mengubah konfigurasi apa pun di pekerjaan yang ada dengan menggunakan salah satu parameter opsional.

```
aws iot create-job-template \
 --job-arn arn:aws:iot:region:123456789012:job/job-name \
 --timeout-config inProgressTimeoutInMinutes=100
```

Dapatkan detail tentang template pekerjaan

AWS CLI Perintah berikut mendapatkan rincian tentang template pekerjaan tertentu.

```
aws iot describe-job-template \
 --job-template-id template-id
```

Perintah menampilkan output berikut.

```
{
 "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
 }
}
```

## Daftar templat pekerjaan

AWS CLI Perintah berikut mencantumkan semua template pekerjaan di Anda Akun AWS.

```
aws iot list-job-templates
```

Perintah menampilkan output berikut.

```
{
 "jobTemplates": [
 {
 "createdAt": number,
 "description": "string",
 "jobTemplateArn": "string",
 "jobTemplateId": "string"
 }
],
 "nextToken": "string"
}
```

Untuk mengambil halaman hasil tambahan, gunakan nilai nextToken bidang.

## Hapus template pekerjaan

AWS CLI Perintah berikut menghapus template pekerjaan tertentu.



```
aws iot delete-job-template \
 --job-template-id template-id
```

Perintah tidak menampilkan output.

Buat pekerjaan dari template pekerjaan khusus

AWS CLI Perintah berikut membuat pekerjaan dari template pekerjaan kustom. Ini menargetkan perangkat bernama `thingOne` dan menentukan Amazon Resource Name (ARN) dari template pekerjaan untuk digunakan sebagai dasar untuk pekerjaan itu. Anda dapat mengganti konfigurasi lanjutan, seperti batas waktu dan membatalkan konfigurasi, dengan meneruskan parameter terkait perintah. `create-job`

#### Warning

`document-parameters` Objek harus digunakan dengan `create-job` perintah hanya saat membuat pekerjaan dari template yang AWS dikelola. Objek ini tidak boleh digunakan dengan template pekerjaan khusus. Untuk contoh yang menunjukkan cara membuat pekerjaan menggunakan parameter ini, lihat [Buat pekerjaan dengan menggunakan templat terkelola](#).

```
aws iot create-job \
 --targets arn:aws:iot:region:123456789012:thing/thingOne \
 --job-template-arn arn:aws:iot:region:123456789012:jobtemplate/template-id
```

## Konfigurasi Job

Anda dapat memiliki konfigurasi tambahan berikut untuk setiap pekerjaan yang Anda terapkan ke target yang ditentukan.

- Peluncuran: Mendefinisikan berapa banyak perangkat yang menerima dokumen pekerjaan setiap menit.
- Penjadwalan: Menjadwalkan pekerjaan untuk tanggal dan waktu masa depan selain menggunakan jendela pemeliharaan berulang.

- **Batalkan:** Membatalkan pekerjaan jika beberapa perangkat tidak menerima pemberitahuan pekerjaan, atau perangkat Anda melaporkan kegagalan eksekusi pekerjaannya.
- **Timeout:** Jika tidak ada respons dari target pekerjaan Anda dalam durasi tertentu setelah eksekusi pekerjaan mereka dimulai, pekerjaan itu bisa gagal.
- **Coba lagi:** Mencoba ulang eksekusi pekerjaan jika perangkat Anda melaporkan kegagalan saat mencoba menyelesaikan eksekusi pekerjaan, atau jika waktu eksekusi pekerjaan Anda habis.

Dengan menggunakan konfigurasi ini, Anda dapat memantau status eksekusi pekerjaan Anda dan menghindari pembaruan yang buruk dikirim ke seluruh armada.

Topik

- [Cara kerja konfigurasi pekerjaan](#)
- [Tentukan konfigurasi tambahan](#)

## Cara kerja konfigurasi pekerjaan

Anda menggunakan konfigurasi peluncuran dan pembatalan saat menerapkan pekerjaan, serta konfigurasi batas waktu dan coba lagi untuk eksekusi pekerjaan. Bagian berikut menunjukkan informasi selengkapnya tentang cara kerja konfigurasi ini.

Topik

- [Peluncuran pekerjaan, penjadwalan, dan membatalkan konfigurasi](#)
- [Konfigurasi batas waktu eksekusi pekerjaan dan coba lagi](#)

## Peluncuran pekerjaan, penjadwalan, dan membatalkan konfigurasi

Anda dapat menggunakan konfigurasi peluncuran pekerjaan, penjadwalan, dan membatalkan untuk menentukan berapa banyak perangkat yang menerima dokumen pekerjaan, menjadwalkan peluncuran pekerjaan, dan menentukan kriteria untuk membatalkan pekerjaan.

### Konfigurasi peluncuran Job

Anda dapat menentukan seberapa cepat target diberi tahu tentang eksekusi pekerjaan yang tertunda. Anda juga dapat membuat peluncuran bertahap untuk mengelola pembaruan, reboot, dan operasi lainnya. Untuk menentukan bagaimana target Anda diberi tahu, gunakan tarif peluncuran pekerjaan.

## Tarif peluncuran Job

Anda dapat membuat konfigurasi peluncuran dengan menggunakan tingkat peluncuran konstan atau tingkat peluncuran eksponensial. Untuk menentukan jumlah maksimum target pekerjaan yang akan diinformasikan per menit, gunakan tingkat peluncuran konstan.

AWS IoT pekerjaan dapat digunakan menggunakan tingkat peluncuran eksponensial karena berbagai kriteria dan ambang batas terpenuhi. Jika jumlah pekerjaan yang gagal cocok dengan serangkaian kriteria yang Anda tentukan, maka Anda dapat membatalkan peluncuran pekerjaan. Anda menetapkan kriteria tingkat peluncuran pekerjaan saat Anda membuat pekerjaan dengan menggunakan objek. [JobExecutionsRolloutConfig](#) Anda juga menetapkan kriteria pembatalan pekerjaan pada penciptaan pekerjaan dengan menggunakan [AbortConfig](#) objek.

Contoh berikut menunjukkan cara kerja tingkat peluncuran. Misalnya, peluncuran pekerjaan dengan tingkat dasar 50 per menit, faktor kenaikan 2, dan jumlah perangkat yang diberitahukan dan berhasil masing-masing sebagai 1.000, akan berfungsi sebagai berikut: Pekerjaan akan dimulai pada tingkat 50 eksekusi pekerjaan per menit dan berlanjut pada tingkat itu sampai 1.000 hal telah menerima pemberitahuan pelaksanaan pekerjaan, atau 1.000 eksekusi pekerjaan yang berhasil telah terjadi.

Tabel berikut menggambarkan bagaimana peluncuran akan dilanjutkan selama empat kenaikan pertama.

| Tingkat peluncuran per menit                                                                            | 50    | 100   | 200   | 400   |
|---------------------------------------------------------------------------------------------------------|-------|-------|-------|-------|
| Jumlah perangkat yang diberitahukan atau eksekusi pekerjaan yang berhasil untuk memenuhi kenaikan tarif | 1.000 | 2.000 | 3.000 | 4.000 |

### Note

Jika Anda berada pada batas maksimum bersamaan 500 Jobs (`isConcurrent = True`), maka semua pekerjaan aktif akan tetap dengan status IN-PROGRESS dan tidak meluncurkan eksekusi pekerjaan baru sampai jumlah pekerjaan bersamaan adalah 499 atau kurang (`isConcurrent = False`) Ini berlaku untuk snapshot dan pekerjaan berkelanjutan. Jika `isConcurrent = True`, pekerjaan saat ini meluncurkan eksekusi pekerjaan ke semua perangkat di grup target Anda. Jika `isConcurrent = False`, pekerjaan telah menyelesaikan peluncuran semua eksekusi pekerjaan ke semua perangkat di grup target Anda. Ini akan memperbarui status statusnya setelah semua perangkat dalam grup target

Anda mencapai status terminal, atau persentase ambang batas grup target Anda jika Anda memilih konfigurasi pembatalan pekerjaan. Status tingkat Job menyatakan untuk `isConcurrent = True` dan `isConcurrent = False` keduanya `IN_PROGRESS`. Untuk informasi selengkapnya tentang batas pekerjaan aktif dan bersamaan, lihat [Batas pekerjaan aktif dan bersamaan](#).

Tarif peluncuran Job untuk pekerjaan berkelanjutan menggunakan grup benda dinamis

Ketika Anda menggunakan pekerjaan berkelanjutan untuk meluncurkan operasi jarak jauh pada armada Anda, AWS IoT Jobs meluncurkan eksekusi pekerjaan untuk perangkat dalam kelompok hal target Anda. Untuk perangkat baru yang ditambahkan ke grup benda dinamis, eksekusi pekerjaan ini terus diluncurkan ke perangkat tersebut bahkan setelah pekerjaan dibuat.

Konfigurasi peluncuran dapat mengontrol tingkat peluncuran hanya untuk perangkat yang ditambahkan ke grup hingga pembuatan lapangan kerja. Setelah pekerjaan dibuat, untuk perangkat baru apa pun, eksekusi pekerjaan dibuat dalam waktu dekat segera setelah perangkat bergabung dengan grup sasaran.

Konfigurasi penjadwalan pekerjaan

Anda dapat menjadwalkan pekerjaan berkelanjutan atau snapshot hingga satu tahun sebelumnya menggunakan waktu mulai yang telah ditentukan, waktu akhir, dan perilaku akhir untuk apa yang akan terjadi pada setiap eksekusi pekerjaan setelah mencapai waktu akhir. Selain itu, Anda dapat membuat jendela pemeliharaan berulang opsional dengan frekuensi fleksibel, waktu mulai, dan durasi untuk pekerjaan berkelanjutan untuk meluncurkan dokumen pekerjaan ke semua perangkat dalam grup target.

Konfigurasi penjadwalan pekerjaan

Waktu mulai

Waktu mulai dari pekerjaan terjadwal adalah tanggal dan waktu masa depan bahwa pekerjaan akan memulai peluncuran dokumen pekerjaan ke semua perangkat dalam kelompok sasaran. Waktu mulai untuk pekerjaan terjadwal berlaku untuk pekerjaan berkelanjutan dan pekerjaan snapshot. Ketika pekerjaan terjadwal awalnya dibuat, ia mempertahankan status `statusSCHEDULED`. Setelah tiba di `startTime` yang Anda pilih, itu memperbarui `IN_PROGRESS` dan memulai peluncuran dokumen pekerjaan. `startTime` harus kurang dari atau sama dengan satu tahun dari tanggal dan waktu awal Anda membuat pekerjaan yang dijadwalkan.

Untuk informasi selengkapnya tentang sintaks `startTime` saat menggunakan perintah API atau perintah AWS CLI, lihat [Timestamp](#).

Untuk pekerjaan dengan konfigurasi penjadwalan opsional yang berlangsung selama jendela pemeliharaan berulang di lokasi yang mengamati waktu musim panas (DST), waktu akan berubah satu jam saat beralih dari DST ke waktu standar dan dari waktu standar ke DST.

#### Note

Zona waktu yang ditampilkan di AWS Management Console adalah zona waktu sistem Anda saat ini. Namun, zona waktu ini akan diubah menjadi UTC dalam sistem.

### Waktu akhir

Waktu akhir dari pekerjaan terjadwal adalah tanggal dan waktu masa depan bahwa pekerjaan akan menghentikan peluncuran dokumen pekerjaan ke perangkat yang tersisa dalam kelompok sasaran. Waktu akhir untuk pekerjaan terjadwal berlaku untuk pekerjaan berkelanjutan dan pekerjaan snapshot. Setelah pekerjaan terjadwal tiba di yang dipilih `endTime`, dan semua eksekusi pekerjaan telah mencapai status terminal, itu memperbarui status statusnya dari `IN_PROGRESS` ke `COMPLETED`. `endTime` harus kurang dari atau sama dengan dua tahun dari tanggal dan waktu awal Anda membuat pekerjaan yang dijadwalkan. Durasi minimum antara `startTime` dan `endTime` 30 menit. Upaya percobaan ulang eksekusi Job akan terjadi sampai pekerjaan mencapai `endTime`, kemudian `endBehavior` akan menentukan bagaimana untuk melanjutkan.

Untuk informasi selengkapnya tentang sintaks `endTime` saat menggunakan perintah API atau perintah AWS CLI, lihat [Timestamp](#).

Untuk pekerjaan dengan konfigurasi penjadwalan opsional yang berlangsung selama jendela pemeliharaan berulang di lokasi yang mengamati waktu musim panas (DST), waktu akan berubah satu jam saat beralih dari DST ke waktu standar dan dari waktu standar ke DST.

#### Note


Zona waktu yang ditampilkan di AWS Management Console adalah zona waktu sistem Anda saat ini. Namun, zona waktu ini akan diubah menjadi UTC dalam sistem.

### Akhiri perilaku

Perilaku akhir dari pekerjaan terjadwal menentukan apa yang terjadi pada pekerjaan dan semua eksekusi pekerjaan yang belum selesai ketika pekerjaan mencapai yang dipilih. `endTime`

Berikut ini mencantumkan perilaku akhir yang dapat Anda pilih saat membuat templat pekerjaan atau pekerjaan:

- **STOP\_ROLLOUT**
  - **STOP\_ROLLOUT** menghentikan peluncuran dokumen pekerjaan ke semua perangkat yang tersisa dalam kelompok sasaran untuk pekerjaan itu. Selain itu, semua **QUEUED** dan eksekusi **IN\_PROGRESS** pekerjaan akan berlanjut sampai mereka mencapai status terminal. Ini adalah perilaku akhir default kecuali Anda memilih **CANCEL** atau **FORCE\_CANCEL**.
- **CANCEL**
  - **CANCEL** menghentikan peluncuran dokumen pekerjaan ke semua perangkat yang tersisa dalam kelompok sasaran untuk pekerjaan itu. Selain itu, semua eksekusi **QUEUED** pekerjaan akan dibatalkan sementara semua eksekusi **IN\_PROGRESS** pekerjaan akan berlanjut hingga mencapai status terminal.
- **FORCE\_CANCEL**
  - **FORCE\_CANCEL** menghentikan peluncuran dokumen pekerjaan ke semua perangkat yang tersisa dalam kelompok sasaran untuk pekerjaan itu. Selain itu, semua eksekusi **QUEUED** dan **IN\_PROGRESS** pekerjaan akan dibatalkan.

 Note

Untuk memilih `behavior`, Anda harus memilih `endtime`

## Durasi maks

Durasi maksimum pekerjaan terjadwal harus kurang dari atau sama dengan dua tahun terlepas dari `startTime` dan `endTime`.

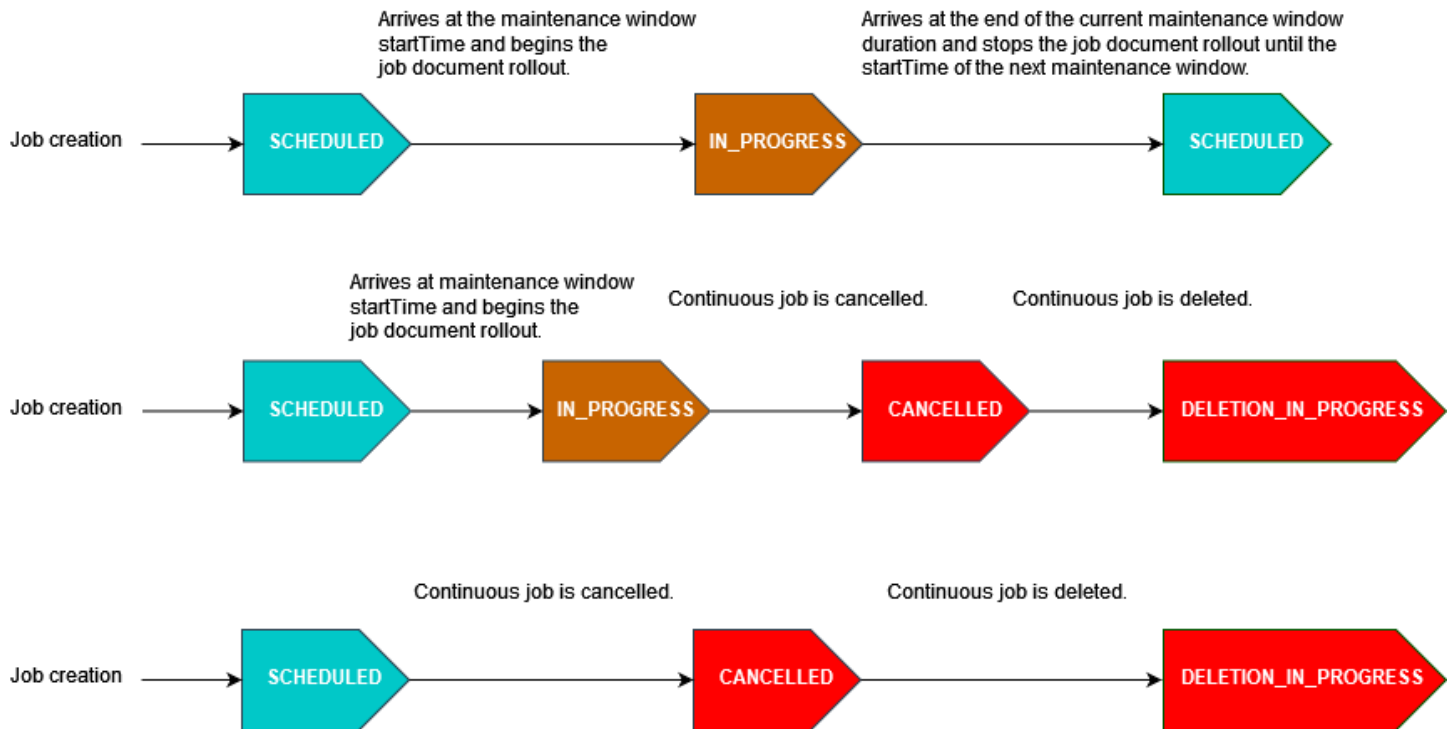
Tabel berikut mencantumkan skenario durasi umum dari pekerjaan terjadwal:

| Nomor contoh Job Terjadwal | startTime                                          | EndTime                                            | Durasi maks |
|----------------------------|----------------------------------------------------|----------------------------------------------------|-------------|
| 1                          | Segera setelah penciptaan lapangan kerja awal.     | Satu tahun setelah penciptaan lapangan kerja awal. | Satu tahun  |
| 2                          | Satu bulan setelah penciptaan lapangan kerja awal. | 13 bulan setelah penciptaan lapangan kerja awal.   | Satu tahun  |
| 3                          | Satu tahun setelah penciptaan lapangan kerja awal. | Dua tahun setelah penciptaan lapangan kerja awal.  | Satu tahun  |
| 4                          | Segera setelah penciptaan lapangan kerja awal.     | Dua tahun setelah penciptaan lapangan kerja awal.  | Dua tahun   |

### Jendela pemeliharaan berulang

Jendela pemeliharaan adalah konfigurasi opsional dalam konfigurasi penjadwalan AWS Management Console dan di `SchedulingConfig` dalam `CreateJob` dan `CreateJobTemplate` API. Anda dapat mengatur jendela pemeliharaan berulang dengan waktu mulai, durasi, dan frekuensi yang telah ditentukan sebelumnya (harian, mingguan, atau bulanan) bahwa jendela pemeliharaan terjadi. Jendela pemeliharaan hanya berlaku untuk pekerjaan berkelanjutan. Durasi maksimum jendela pemeliharaan berulang adalah 23 jam, 50 menit.

Diagram berikut menggambarkan status status pekerjaan untuk berbagai skenario pekerjaan terjadwal dengan jendela pemeliharaan opsional:



Untuk informasi selengkapnya tentang status status pekerjaan, lihat [Pekerjaan dan status eksekusi pekerjaan](#).

#### Note

Jika pekerjaan tiba di `endTime` selama jendela pemeliharaan, itu akan diperbarui dari **IN\_PROGRESS** ke **COMPLETED**. Selain itu, setiap eksekusi pekerjaan yang tersisa akan mengikuti `endBehavior` untuk pekerjaan itu.

## Eksresi cron

Untuk pekerjaan terjadwal yang meluncurkan dokumen pekerjaan selama jendela pemeliharaan dengan frekuensi khusus, frekuensi kustom dimasukkan menggunakan ekspresi cron. Ekspresi cron memiliki enam bidang wajib, yang dipisahkan oleh spasi putih.

## Sintaksis

```
cron(fields)
```



| Bidang       | Nilai-nilai       | Wildcard      |
|--------------|-------------------|---------------|
| Menit        | 0-59              | , - * /       |
| Jam          | 0-23              | , - * /       |
| Day-of-month | 1-31              | , - * ? / L W |
| Bulan        | 1-12 atau JAN-DES | , - * /       |
| Day-of-week  | 1-7 atau MGG-SBT  | , - * ? L #   |
| Tahun        | 1970-2199         | , - * /       |

## Wildcard

- Wildcard , (koma) mencakup nilai tambahan. Di field Bulan, JAN, FEB, MAR akan mencakup Januari, Februari, dan Maret.
- Wildcard - (tanda hubung) menentukan rentang. Di field Tanggal, 1-15 akan mencakup tanggal 1 hingga 15 pada bulan yang ditentukan.
- Wildcard \* (bintang) mencakup semua nilai di bidang. Di field Jam, \* akan mencakup setiap jam. Anda tidak dapat menggunakan\* di ay-of-week bidang D ay-of-month dan D. Jika Anda menggunakannya di satu bidang, Anda harus menggunakan ? di bidang lain.
- Wildcard / (garis miring) menentukan tambahan. Di bidang menit, Anda bisa memasukkan 1/10 untuk menentukan setiap menit kesepuluh, mulai dari menit pertama jam (sebagai contoh, menit ke-11, 21, dan 31, dan seterusnya).
- Wildcard ? (tanda tanya) menentukan satu atau yang lain. Di ay-of-month bidang D, Anda bisa memasukkan 7 dan jika Anda tidak peduli hari apa dalam minggu ke-7, Anda bisa masuk? di ay-of-week bidang D.
- Wildcard L di ay-of-week bidang D ay-of-month atau D menentukan hari terakhir bulan atau minggu.
- Wildcard di ay-of-month bidang D menentukan hari kerja. Di ay-of-month bidang D, **3W** tentukan hari kerja yang paling dekat dengan hari ketiga bulan itu.
- Wildcard # di ay-of-week bidang D menentukan contoh tertentu dari hari yang ditentukan dalam seminggu dalam sebulan. Sebagai contoh, **3#2** akan menjadi hari Selasa kedua setiap bulan: 3

mengacu pada hari Selasa karena itu adalah hari ketiga setiap minggu, dan 2 mengacu pada hari kedua dari jenis tersebut dalam bulan tersebut.

### Note

Jika Anda menggunakan karakter '#', Anda hanya dapat menentukan satu ekspresi di day-of-week bidang. Misalnya, "3#1,6#3" tidak valid karena ditafsirkan sebagai dua ekspresi.

### Pembatasan

- Anda tidak dapat menentukan ay-of-week bidang D ay-of-month dan D dalam ekspresi cron yang sama. Jika Anda menentukan nilai (atau \*) di salah satu bidang, Anda harus menggunakan ? di sisi lain.

### Contoh

Lihat contoh string cron berikut saat menggunakan ekspresi cron untuk jendela `startTime` pemeliharaan berulang.

| Menit | Jam | Hari dalam sebulan | Bulan | Hari dalam seminggu | Tahun | Arti                                             |
|-------|-----|--------------------|-------|---------------------|-------|--------------------------------------------------|
| 0     | 10  | *                  | *     | ?                   | *     | Jalankan pada pukul 10:00 pagi (UTC) setiap hari |
| 15    | 12  | *                  | *     | ?                   | *     | Jalankan pada pukul 12.15 (UTC) setiap hari      |
| 0     | 18  | ?                  | *     | MON-FRI             | *     | Jalankan pada pukul 18.00                        |

| Menit | Jam | Hari dalam sebulan | Bulan | Hari dalam seminggu | Tahun | Arti                                                                             |
|-------|-----|--------------------|-------|---------------------|-------|----------------------------------------------------------------------------------|
|       |     |                    |       |                     |       | (UTC)<br>setiap<br>Senin<br>hingga<br>Jumat                                      |
| 0     | 8   | 1                  | *     | ?                   | *     | Jalankan<br>pada pukul<br>8:00 (UTC)<br>setiap<br>tanggal<br>satu di<br>bulannya |

Logika akhir durasi jendela pemeliharaan berulang

Ketika peluncuran pekerjaan selama jendela pemeliharaan mencapai akhir durasi kejadian jendela pemeliharaan saat ini, tindakan berikut akan terjadi:

- Job akan menghentikan semua peluncuran dokumen pekerjaan ke perangkat yang tersisa di grup target Anda. Ini akan dilanjutkan `startTime` di jendela pemeliharaan berikutnya.
- Semua eksekusi pekerjaan dengan status `QUEUED` akan tetap ada `QUEUED` sampai jendela `startTime` pemeliharaan berikutnya terjadi. Di jendela berikutnya, mereka dapat beralih ke `IN_PROGRESS` saat perangkat siap untuk mulai melakukan tindakan yang ditentukan dalam dokumen pekerjaan.
- Semua eksekusi pekerjaan dengan status `IN_PROGRESS` akan terus melakukan tindakan yang ditentukan dalam dokumen pekerjaan sampai mencapai status terminal. Setiap upaya coba lagi seperti yang ditentukan dalam `JobExecutionsRetryConfig` akan dilakukan di `startTime` jendela pemeliharaan berikutnya.

## Konfigurasi pembatalan pekerjaan

Gunakan konfigurasi ini untuk membuat kriteria untuk membatalkan pekerjaan ketika persentase ambang perangkat memenuhi kriteria tersebut. Misalnya, Anda dapat menggunakan konfigurasi ini untuk membatalkan pekerjaan dalam kasus berikut:

- Ketika persentase ambang batas perangkat tidak menerima pemberitahuan eksekusi pekerjaan, seperti saat perangkat Anda tidak kompatibel untuk pembaruan Over-The-Air (OTA). Dalam hal ini, perangkat Anda dapat melaporkan REJECTED status.
- Ketika persentase ambang batas perangkat melaporkan kegagalan untuk eksekusi pekerjaannya, seperti saat perangkat Anda mengalami pemutusan saat mencoba mengunduh dokumen pekerjaan dari URL Amazon S3. Dalam kasus seperti itu, perangkat Anda harus diprogram untuk melaporkan FAILURE statusnya. AWS IoT
- Ketika TIMED\_OUT status dilaporkan karena waktu eksekusi pekerjaan habis untuk persentase ambang perangkat setelah eksekusi pekerjaan dimulai.
- Ketika ada beberapa kegagalan coba lagi. Saat Anda menambahkan konfigurasi coba lagi, setiap upaya coba lagi dapat dikenakan biaya tambahan untuk Anda. Akun AWS Dalam kasus seperti itu, membatalkan pekerjaan dapat membatalkan eksekusi pekerjaan yang diantrian dan menghindari upaya coba lagi untuk eksekusi ini. Untuk informasi selengkapnya tentang konfigurasi coba lagi dan menggunakannya dengan konfigurasi batalkan, lihat. [Konfigurasi batas waktu eksekusi pekerjaan dan coba lagi](#)

Anda dapat mengatur kondisi pembatalan pekerjaan dengan menggunakan AWS IoT konsol atau AWS IoT Jobs API.

## Konfigurasi batas waktu eksekusi pekerjaan dan coba lagi

Gunakan konfigurasi batas waktu eksekusi pekerjaan untuk mengirim Anda [Pemberitahuan pekerjaan](#) ketika eksekusi pekerjaan telah berlangsung lebih lama dari durasi yang ditetapkan. Gunakan konfigurasi coba ulang eksekusi pekerjaan untuk mencoba kembali eksekusi saat pekerjaan gagal atau habis waktu.

## Konfigurasi batas waktu eksekusi Job

Gunakan konfigurasi batas waktu eksekusi pekerjaan untuk memberi tahu Anda setiap kali eksekusi pekerjaan macet di IN\_PROGRESS status untuk jangka waktu yang sangat lama. Ketika pekerjaan itu IN\_PROGRESS, Anda dapat memantau kemajuan pelaksanaan pekerjaan Anda.

## Timer untuk batas waktu kerja

Ada dua jenis pengatur waktu: pengatur waktu dalam proses dan pengatur waktu langkah.

### Pengatur waktu dalam proses

Saat Anda membuat pekerjaan atau templat pekerjaan, Anda dapat menentukan nilai untuk pengatur waktu yang sedang berlangsung antara 1 menit dan 7 hari. Anda dapat memperbarui nilai timer ini hingga dimulainya eksekusi pekerjaan Anda. Setelah timer Anda dimulai, itu tidak dapat diperbarui, dan nilai timer berlaku untuk semua eksekusi pekerjaan untuk pekerjaan tersebut. Setiap kali eksekusi pekerjaan tetap dalam `IN_PROGRESS` status lebih lama dari interval ini, eksekusi pekerjaan gagal dan beralih ke `TIMED_OUT` status terminal. AWS IoT juga menerbitkan pemberitahuan MQTT.

### Pengatur waktu langkah

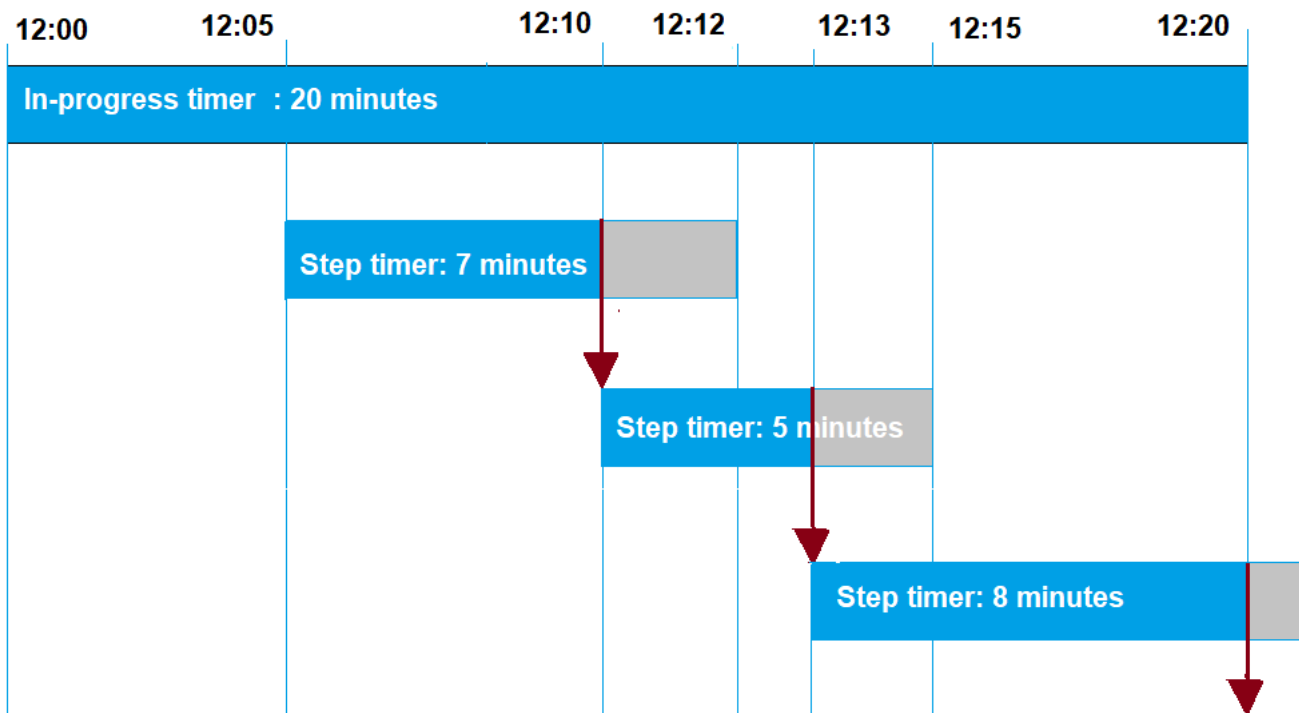
Anda juga dapat mengatur pengatur waktu langkah yang hanya berlaku untuk eksekusi pekerjaan yang ingin Anda perbarui. Timer ini tidak berpengaruh pada timer yang sedang berlangsung. Setiap kali Anda memperbarui eksekusi pekerjaan, Anda dapat menetapkan nilai baru untuk pengatur waktu langkah. Anda juga dapat membuat pengatur waktu langkah baru saat memulai eksekusi pekerjaan tertunda berikutnya untuk suatu hal. Jika eksekusi pekerjaan tetap dalam `IN_PROGRESS` status lebih lama dari interval pengatur waktu langkah, itu gagal dan beralih ke `TIMED_OUT` status terminal.

#### Note

Anda dapat mengatur timer yang sedang berlangsung dengan menggunakan AWS IoT konsol atau AWS IoT Jobs API. Untuk menentukan pengatur waktu langkah, gunakan API.

## Bagaimana pengatur waktu bekerja untuk batas waktu kerja

Berikut ini menggambarkan cara-cara di mana batas waktu yang sedang berlangsung dan batas waktu langkah berinteraksi satu sama lain dalam periode batas waktu 20 menit.



Berikut ini menunjukkan langkah-langkah yang berbeda:

1. 12:00

Pekerjaan baru dibuat dan timer dalam proses selama dua puluh menit dimulai saat membuat pekerjaan. Timer yang sedang berlangsung mulai berjalan dan eksekusi pekerjaan beralih ke IN\_PROGRESS status.

2. 12:05PM

Timer langkah baru dengan nilai 7 menit dibuat. Pelaksanaan pekerjaan sekarang akan habis pada 12:12 PM.

3. 12:10PM

Timer langkah baru dengan nilai 5 menit dibuat. Ketika pengatur waktu langkah baru dibuat, pengatur waktu langkah sebelumnya dibuang, dan eksekusi pekerjaan sekarang akan habis pada pukul 12:15.

4. 12:13PM

Timer langkah baru dengan nilai 9 menit dibuat. Timer langkah sebelumnya dibuang dan eksekusi pekerjaan sekarang akan habis pada pukul 12:20 karena waktu yang sedang berlangsung habis

pada pukul 12:20. Pengatur waktu langkah tidak dapat melebihi batas absolut pengatur waktu yang sedang berlangsung.

## Konfigurasi coba lagi eksekusi Job

Anda dapat menggunakan konfigurasi coba lagi untuk mencoba kembali eksekusi pekerjaan ketika serangkaian kriteria tertentu terpenuhi. Coba lagi dapat dicoba ketika waktu kerja habis atau ketika perangkat gagal. Untuk mencoba kembali eksekusi karena kegagalan batas waktu, Anda harus mengaktifkan konfigurasi batas waktu.

## Cara menggunakan konfigurasi coba lagi

Gunakan langkah-langkah berikut untuk mencoba kembali konfigurasi:

1. Tentukan apakah akan menggunakan konfigurasi coba lagi untuk `FAILED`, `TIMED_OUT`, atau kedua kriteria kegagalan. Untuk `TIMED_OUT` status, setelah status dilaporkan, AWS IoT Jobs secara otomatis mencoba ulang eksekusi pekerjaan untuk perangkat.
2. Untuk `FAILED` status, periksa apakah kegagalan eksekusi pekerjaan Anda dapat dicoba ulang. Jika dapat dicoba ulang, program perangkat Anda untuk melaporkan `FAILURE` statusnya. AWS IoT Bagian berikut menjelaskan lebih lanjut tentang kegagalan yang dapat dicoba ulang dan tidak dapat dicoba kembali.
3. Tentukan jumlah percobaan ulang yang akan digunakan untuk setiap jenis kegagalan dengan menggunakan informasi sebelumnya. Untuk satu perangkat, Anda dapat menentukan hingga 10 percobaan ulang untuk kedua jenis kegagalan yang digabungkan. Upaya coba lagi berhenti secara otomatis ketika eksekusi berhasil atau ketika mencapai jumlah upaya yang ditentukan.
4. Tambahkan konfigurasi batal untuk membatalkan pekerjaan jika ada kegagalan coba lagi berulang untuk menghindari biaya tambahan yang dikeluarkan dengan sejumlah besar upaya coba lagi.

### Note

Ketika pekerjaan mencapai akhir dari kejadian jendela pemeliharaan berulang, semua eksekusi `IN_PROGRESS` pekerjaan akan terus melakukan tindakan yang diidentifikasi dalam dokumen pekerjaan sampai mereka mencapai status terminal. Jika eksekusi pekerjaan mencapai status terminal `FAILED` atau `TIMED_OUT` di luar jendela pemeliharaan, upaya coba lagi akan terjadi di jendela berikutnya jika upaya tidak habis. Pada `startTime` kemunculan

jendela pemeliharaan berikutnya, eksekusi pekerjaan baru akan dibuat dan memasukkan status status QUEUED hingga perangkat siap untuk memulai.

Coba lagi dan batalkan konfigurasi

Setiap upaya coba lagi menimbulkan biaya tambahan untuk Anda. Akun AWS Untuk menghindari biaya tambahan dari kegagalan percobaan ulang berulang, kami sarankan menambahkan konfigurasi batal. Untuk informasi selengkapnya tentang harga, silakan lihat [harga AWS IoT Device Management](#).

Anda mungkin mengalami beberapa kegagalan coba lagi ketika persentase ambang batas tinggi perangkat Anda baik waktu habis atau melaporkan kegagalan. Dalam hal ini, Anda dapat menggunakan konfigurasi batal untuk membatalkan pekerjaan dan menghindari eksekusi pekerjaan antrian atau upaya coba lagi lebih lanjut.

#### Note

Ketika kriteria pembatalan terpenuhi untuk membatalkan eksekusi pekerjaan, hanya eksekusi QUEUED pekerjaan yang dibatalkan. Setiap percobaan ulang antrian untuk perangkat tidak akan dicoba. Namun, eksekusi pekerjaan saat ini yang memiliki IN\_PROGRESS status tidak akan dibatalkan.

Sebelum mencoba kembali eksekusi pekerjaan yang gagal, kami juga menyarankan Anda memeriksa apakah kegagalan eksekusi pekerjaan Anda dapat dicoba kembali, seperti yang dijelaskan di bagian berikut.

Coba lagi untuk jenis kegagalan **FAILED**

Untuk mencoba mencoba ulang untuk jenis kegagalan FAILED, perangkat Anda harus diprogram untuk melaporkan FAILURE status eksekusi pekerjaan yang gagal. AWS IoT Tetapkan konfigurasi coba lagi dengan kriteria untuk mencoba lagi eksekusi FAILED pekerjaan dan tentukan jumlah percobaan ulang yang akan dilakukan. Ketika AWS IoT Jobs mendeteksi FAILURE status, maka secara otomatis akan mencoba untuk mencoba kembali eksekusi pekerjaan untuk perangkat. Percobaan ulang berlanjut sampai eksekusi pekerjaan berhasil atau ketika mencapai jumlah maksimum upaya coba lagi.



Anda dapat melacak setiap percobaan ulang dan pekerjaan yang berjalan pada perangkat ini. Dengan melacak status eksekusi, setelah jumlah percobaan ulang yang ditentukan telah dicoba, Anda dapat menggunakan perangkat Anda untuk melaporkan kegagalan dan memulai percobaan ulang lainnya.

Kegagalan yang dapat dicoba ulang dan tidak dapat dicoba kembali

Kegagalan eksekusi pekerjaan Anda dapat dicoba kembali atau tidak dapat dicoba kembali. Setiap upaya coba lagi dapat menimbulkan biaya untuk Anda. Akun AWS Untuk menghindari biaya tambahan dari beberapa upaya coba lagi, pertama-tama pertimbangkan untuk memeriksa apakah kegagalan eksekusi pekerjaan Anda dapat dicoba kembali. Contoh kegagalan yang dapat dicoba ulang mencakup kesalahan koneksi yang ditemui perangkat Anda saat mencoba mengunduh dokumen pekerjaan dari URL Amazon S3. Jika kegagalan eksekusi pekerjaan Anda dapat dicoba ulang, program perangkat Anda untuk melaporkan FAILURE status jika eksekusi pekerjaan gagal. Kemudian, atur konfigurasi coba lagi untuk mencoba lagi FAILED eksekusi.

Jika eksekusi tidak dapat dicoba lagi, untuk menghindari percobaan ulang dan berpotensi menimbulkan biaya tambahan ke akun Anda, kami sarankan Anda memprogram perangkat untuk melaporkan statusnya. REJECTED AWS IoT Contoh kegagalan yang tidak dapat dicoba ulang termasuk saat perangkat Anda tidak kompatibel dalam menerima pembaruan pekerjaan, atau saat mengalami kesalahan memori saat menjalankan pekerjaan. Dalam kasus ini, AWS IoT Jobs tidak akan mencoba lagi eksekusi pekerjaan karena mencoba kembali eksekusi pekerjaan hanya ketika mendeteksi status atau FAILED. TIMED\_OUT

Setelah Anda menentukan bahwa kegagalan eksekusi pekerjaan dapat dicoba ulang, jika upaya coba lagi masih gagal, pertimbangkan untuk memeriksa log perangkat.

#### Note

Ketika pekerjaan dengan konfigurasi penjadwalan opsional mencapai `endTime`, yang dipilih `endBehavior` akan menghentikan peluncuran dokumen pekerjaan ke semua perangkat yang tersisa di grup target dan menentukan cara melanjutkan eksekusi pekerjaan yang tersisa. Upaya dicoba lagi jika dipilih melalui konfigurasi coba lagi.

Coba lagi untuk jenis kegagalan **TIMEOUT**

Jika Anda mengaktifkan batas waktu saat membuat pekerjaan, AWS IoT Jobs akan mencoba kembali eksekusi pekerjaan untuk perangkat saat status berubah dari `IN_PROGRESS` ke `TIMED_OUT`

Perubahan status ini dapat terjadi saat waktu pengatur waktu yang sedang berlangsung habis, atau saat pengatur waktu langkah yang Anda tentukan masuk IN\_PROGRESS dan kemudian habis waktu. Percobaan ulang berlanjut sampai eksekusi pekerjaan berhasil, atau ketika mencapai jumlah maksimum upaya coba lagi untuk jenis kegagalan ini.

Pekerjaan berkelanjutan dan pembaruan keanggotaan grup

Untuk pekerjaan berkelanjutan yang memiliki status pekerjaan sebagai IN\_PROGRESS, jumlah upaya coba lagi diatur ulang ke nol ketika ada pembaruan untuk keanggotaan grup sesuatu. Misalnya, pertimbangkan bahwa Anda menentukan lima upaya coba lagi dan tiga percobaan ulang telah dilakukan. Jika sesuatu sekarang dihapus dari grup benda dan kemudian bergabung kembali dengan grup, seperti dengan grup benda dinamis, jumlah upaya coba lagi diatur ulang ke nol. Anda sekarang dapat melakukan lima upaya coba lagi untuk kelompok hal Anda alih-alih dua upaya yang tersisa. Selain itu, ketika sesuatu dihapus dari grup benda, upaya coba lagi tambahan dibatalkan.

## Tentukan konfigurasi tambahan

Saat Anda membuat templat pekerjaan atau pekerjaan, Anda dapat menentukan konfigurasi tambahan ini. Berikut ini menunjukkan kapan Anda dapat menentukan konfigurasi ini.

- Saat membuat template pekerjaan khusus. Pengaturan konfigurasi tambahan yang Anda tentukan akan disimpan saat Anda membuat pekerjaan dari template.
- Saat membuat pekerjaan khusus dengan menggunakan file pekerjaan. File pekerjaan dapat berupa file JSON yang diunggah dalam bucket S3.
- Saat membuat pekerjaan khusus dengan menggunakan template pekerjaan khusus. Jika templat sudah memiliki pengaturan ini yang ditentukan, Anda dapat menggunakannya kembali atau menggantinya dengan menentukan pengaturan konfigurasi baru.
- Saat membuat pekerjaan khusus dengan menggunakan templat AWS terkelola.

Topik

- [Tentukan konfigurasi pekerjaan dengan menggunakan AWS Management Console](#)
- [Menentukan konfigurasi pekerjaan dengan menggunakan AWS IoT Jobs API](#)

## Tentukan konfigurasi pekerjaan dengan menggunakan AWS Management Console

Anda dapat menambahkan konfigurasi yang berbeda untuk pekerjaan Anda dengan menggunakan AWS IoT konsol. Setelah membuat pekerjaan, Anda dapat melihat detail status konfigurasi pekerjaan

Anda di halaman detail pekerjaan. Untuk informasi selengkapnya tentang konfigurasi yang berbeda dan cara kerjanya, lihat [Cara kerja konfigurasi pekerjaan](#).

Tambahkan konfigurasi pekerjaan saat Anda membuat pekerjaan atau templat pekerjaan.

Saat membuat template pekerjaan khusus

Untuk menentukan konfigurasi peluncuran saat membuat template pekerjaan khusus

1. Buka [hub Job templates di AWS IoT konsol](#) dan pilih Create job template.
2. Tentukan properti template pekerjaan, berikan dokumen pekerjaan, perluas konfigurasi yang ingin Anda tambahkan, lalu tentukan parameter konfigurasi.

Saat membuat pekerjaan khusus

Untuk menentukan konfigurasi peluncuran saat membuat pekerjaan khusus

1. Buka [hub Job AWS IoT konsol](#) dan pilih Create job.
2. Pilih Buat pekerjaan khusus dan tentukan properti pekerjaan, target, dan apakah akan menggunakan file pekerjaan atau templat untuk dokumen pekerjaan. Anda dapat menggunakan templat khusus atau templat AWS terkelola.
3. Pilih konfigurasi pekerjaan dan kemudian perluas konfigurasi Peluncuran untuk menentukan apakah akan menggunakan tingkat Konstan atau Tingkat eksponensial. Kemudian, tentukan parameter konfigurasi.

Bagian selanjutnya menunjukkan parameter yang dapat Anda tentukan untuk setiap konfigurasi.

Konfigurasi peluncuran

Anda dapat menentukan apakah akan menggunakan tingkat peluncuran konstan atau tingkat eksponensial.

- Tetapkan tingkat peluncuran konstan

Untuk menetapkan tingkat konstan untuk eksekusi pekerjaan, pilih Tingkat konstan, lalu tentukan Maksimum per menit untuk batas atas tarif. Nilai ini opsional dan berkisar dari 1 hingga 1000. Jika Anda tidak mengaturnya, ia menggunakan 1000 sebagai nilai default.

- Tetapkan tingkat peluncuran eksponensial

Untuk menetapkan laju eksponensial, pilih Tingkat eksponensial dan kemudian tentukan parameter ini:

- Tarif dasar per menit

Tingkat di mana pekerjaan dijalankan hingga Jumlah perangkat yang diberitahukan atau ambang batas Jumlah perangkat yang berhasil terpenuhi untuk kriteria kenaikan tarif.

- Faktor kenaikan

Faktor eksponensial dimana tingkat peluncuran meningkat setelah Jumlah perangkat yang diberitahukan atau ambang batas Jumlah perangkat yang berhasil terpenuhi untuk kriteria kenaikan tarif.

- Kriteria kenaikan tarif

Ambang batas untuk Jumlah perangkat yang diberitahukan atau Jumlah perangkat yang berhasil.

## Batalkan konfigurasi

Pilih Tambahkan konfigurasi baru dan tentukan parameter berikut untuk setiap konfigurasi:

- Jenis kegagalan

Menentukan jenis kegagalan yang memulai pekerjaan abort. Ini termasuk GAGAL, DITOLAK, TIMED\_OUT, atau SEMUA.

- Faktor kenaikan

Menentukan jumlah eksekusi pekerjaan selesai yang harus terjadi sebelum kriteria pembatalan pekerjaan telah dipenuhi.

- Persentase ambang

Menentukan jumlah total hal-hal yang dieksekusi yang memulai pekerjaan abort.

## Konfigurasi penjadwalan

Setiap pekerjaan dapat dimulai segera setelah pembuatan awal, dijadwalkan untuk dimulai di kemudian hari dan waktu, atau berlangsung selama jendela pemeliharaan berulang.

Pilih Tambahkan konfigurasi baru dan tentukan parameter berikut untuk setiap konfigurasi:

- Job dimulai

Tentukan tanggal dan waktu kapan pekerjaan akan dimulai.

- Jendela pemeliharaan berulang


Jendela pemeliharaan berulang menentukan tanggal dan waktu tertentu pekerjaan dapat menyebarkan dokumen pekerjaan ke perangkat target dalam pekerjaan. Jendela pemeliharaan dapat diulang setiap hari, mingguan, bulanan, atau pengulangan hari dan waktu khusus.

- Job berakhir

Tentukan tanggal dan waktu kapan pekerjaan akan berakhir.

- Perilaku akhir pekerjaan

Pilih perilaku akhir untuk semua eksekusi pekerjaan yang belum selesai saat pekerjaan selesai.


 Note

Ketika pekerjaan dengan konfigurasi penjadwalan opsional dan waktu akhir yang dipilih mencapai waktu akhir, pekerjaan menghentikan peluncuran ke semua perangkat yang tersisa di grup target. Ini juga memanfaatkan perilaku akhir yang dipilih tentang cara melanjutkan eksekusi pekerjaan yang tersisa dan upaya coba lagi mereka per konfigurasi coba lagi.

## Konfigurasi waktu habis

Secara default, tidak ada batas waktu dan pekerjaan Anda dibatalkan atau dihapus. Untuk menggunakan batas waktu, pilih Aktifkan batas waktu, lalu tentukan nilai batas waktu antara 1 menit dan 7 hari.

## Coba lagi konfigurasi

 Note

Setelah pekerjaan dibuat, jumlah percobaan ulang tidak dapat diperbarui. Anda hanya dapat menghapus konfigurasi coba lagi untuk semua jenis kegagalan. Saat Anda membuat pekerjaan, pertimbangkan jumlah percobaan ulang yang sesuai untuk digunakan untuk konfigurasi Anda. Untuk menghindari timbulnya biaya berlebih karena potensi kegagalan coba lagi, tambahkan konfigurasi batal.

Pilih Tambahkan konfigurasi baru dan tentukan parameter berikut untuk setiap konfigurasi:

- Jenis kegagalan

Menentukan jenis kegagalan yang harus memicu percobaan ulang eksekusi pekerjaan. Ini termasuk Gagal, Timeout, dan Semua.

- Jumlah percobaan

Menentukan jumlah percobaan ulang untuk jenis Kegagalan yang dipilih. Untuk kedua jenis kegagalan yang digabungkan, hingga 10 percobaan ulang dapat dicoba.

## Menentukan konfigurasi pekerjaan dengan menggunakan AWS IoT Jobs API

Anda dapat menggunakan [CreateJob](#) atau [CreateJobTemplate](#) API untuk menentukan konfigurasi pekerjaan yang berbeda. Bagian berikut menjelaskan cara menambahkan konfigurasi ini. Setelah menambahkan konfigurasi, Anda dapat menggunakan [JobExecutionSummary](#) dan [JobExecutionSummaryForJob](#) melihat statusnya.

Untuk informasi selengkapnya tentang konfigurasi yang berbeda dan cara kerjanya, lihat [Cara kerja konfigurasi pekerjaan](#).

### Konfigurasi peluncuran

Anda dapat menentukan tingkat peluncuran konstan atau tingkat peluncuran eksponensial untuk konfigurasi peluncuran Anda.

- Tetapkan tingkat peluncuran konstan

Untuk menetapkan tingkat peluncuran konstan, gunakan [JobExecutionsRolloutConfig](#) objek untuk menambahkan `maximumPerMinute` parameter ke permintaan. `CreateJob` Parameter ini menentukan batas atas tingkat di mana eksekusi pekerjaan dapat terjadi. Nilai ini opsional dan berkisar dari 1 hingga 1000. Jika Anda tidak menetapkan nilai, itu menggunakan 1000 sebagai nilai default.

```
"jobExecutionsRolloutConfig": {
 "maximumPerMinute": 1000
}
```

- Tetapkan tingkat peluncuran eksponensial

Untuk menetapkan tingkat peluncuran pekerjaan variabel, gunakan objek.

[JobExecutionsRolloutConfig](#) Anda dapat mengonfigurasi `ExponentialRolloutRate` properti saat menjalankan operasi `CreateJob` API. Contoh berikut menetapkan tingkat peluncuran eksponensial dengan menggunakan parameter. `exponentialRate` Untuk informasi tentang parameter, lihat [ExponentialRolloutRate](#).

```
{
 ...
 "jobExecutionsRolloutConfig": {
 "exponentialRate": {
 "baseRatePerMinute": 50,
 "incrementFactor": 2,
 "rateIncreaseCriteria": {
 "numberOfNotifiedThings": 1000,
 "numberOfSucceededThings": 1000
 },
 "maximumPerMinute": 1000
 }
 }
 ...
}
```

Dimana parameternya:

`baseRatePerMenit`

Menentukan tingkat di mana pekerjaan dieksekusi sampai `numberOfSucceededThings` ambang batas `numberOfNotifiedThings` atau telah terpenuhi.

`IncrementFactor`

Menentukan faktor eksponensial dimana tingkat peluncuran meningkat setelah ambang batas `numberOfNotifiedThings` atau `numberOfSucceededThings` terpenuhi.

`rateIncreaseCriteria`

Menentukan baik `numberOfNotifiedThings` atau `numberOfSucceededThings` ambang batas.

## Batalkan konfigurasi

Untuk menambahkan konfigurasi ini dengan menggunakan API, tentukan [AbortConfig](#) parameter saat Anda menjalankan [CreateJob](#), atau operasi [CreateJobTemplate](#) API. Contoh berikut menunjukkan konfigurasi abort untuk peluncuran pekerjaan yang mengalami beberapa eksekusi gagal, seperti yang ditentukan dengan operasi API. [CreateJob](#)

### Note

Menghapus eksekusi pekerjaan memengaruhi nilai komputasi dari total eksekusi yang diselesaikan. Ketika pekerjaan dibatalkan, layanan membuat otomatis comment dan `reasonCode` untuk membedakan pembatalan yang digerakkan pengguna dari pembatalan pembatalan pekerjaan.

```
"abortConfig": {
 "criteriaList": [
 {
 "action": "CANCEL",
 "failureType": "FAILED",
 "minNumberOfExecutedThings": 100,
 "thresholdPercentage": 20
 },
 {
 "action": "CANCEL",
 "failureType": "TIMED_OUT",
 "minNumberOfExecutedThings": 200,
 "thresholdPercentage": 50
 }
]
}
```

Dimana parameternya:

aksi

Menentukan tindakan yang harus diambil ketika kriteria abort telah dipenuhi. Parameter ini diperlukan, dan CANCEL merupakan satu-satunya nilai yang valid.



## FailureType

Menentukan jenis kegagalan yang harus memulai pekerjaan abort. Nilai yang valid adalah FAILED, REJECTED, TIMED\_OUT, dan ALL.

## minNumberOfExecutedThings

Menentukan jumlah eksekusi pekerjaan selesai yang harus terjadi sebelum kriteria pembatalan pekerjaan telah dipenuhi. Dalam contoh ini, AWS IoT tidak memeriksa untuk melihat apakah pembatalan pekerjaan harus terjadi sampai setidaknya 100 perangkat telah menyelesaikan eksekusi pekerjaan.

## ThresholdPercentage

Menentukan jumlah total hal-hal yang pekerjaan dieksekusi yang dapat memulai pekerjaan abort. Dalam contoh ini, AWS IoT memeriksa secara berurutan dan memulai pembatalan pekerjaan jika persentase ambang terpenuhi. Jika setidaknya 20% dari eksekusi lengkap gagal setelah 100 eksekusi selesai, itu membatalkan peluncuran pekerjaan. Jika kriteria ini tidak terpenuhi, AWS IoT maka periksa apakah setidaknya 50% dari eksekusi selesai habis setelah 200 eksekusi selesai. Jika ini masalahnya, itu membatalkan peluncuran pekerjaan.

## Konfigurasi penjadwalan

Untuk menambahkan konfigurasi ini dengan menggunakan API, tentukan opsional [SchedulingConfig](#) saat Anda menjalankan [CreateJob](#), atau operasi [CreateJobTemplateAPI](#).

```
"SchedulingConfig": {
 "endBehavior": string
 "endTime": string
 "maintenanceWindows": string
 "startTime": string
}
```

Dimana parameternya:

### startTime

Menentukan tanggal dan waktu ketika pekerjaan akan dimulai.

### EndTime

Menentukan tanggal dan waktu ketika pekerjaan akan berakhir.

## PemeliharaanWindows

Menentukan apakah jendela pemeliharaan opsional dipilih untuk pekerjaan terjadwal untuk meluncurkan dokumen pekerjaan ke semua perangkat dalam grup target. Format string untuk `maintenanceWindow` adalah YYYY/MM/DD untuk tanggal dan hh: mm untuk waktu.

## EndBehavior

Menentukan perilaku pekerjaan untuk pekerjaan terjadwal setelah mencapai. `endTime`

### Note

Opsional `SchedulingConfig` untuk pekerjaan dapat dilihat di [DescribeJobTemplateAPI](#) [DescribeJob](#) dan.

## Konfigurasi waktu habis

Untuk menambahkan konfigurasi ini dengan menggunakan API, tentukan [TimeoutConfig](#) parameter saat Anda menjalankan [CreateJob](#), atau operasi [CreateJobTemplateAPI](#).

Untuk menggunakan konfigurasi batas waktu

1. Untuk mengatur timer yang sedang berlangsung saat Anda membuat templat pekerjaan atau pekerjaan, tetapkan nilai untuk `inProgressTimeoutInMinutes` properti [TimeoutConfig](#) objek opsional.

```
"timeoutConfig": {
 "inProgressTimeoutInMinutes": number
}
```

2. Untuk menentukan pengatur waktu langkah untuk eksekusi pekerjaan, tetapkan nilai `stepTimeoutInMinutes` saat Anda menelepon [UpdateJobExecution](#). Pengatur waktu langkah hanya berlaku untuk eksekusi pekerjaan yang Anda perbarui. Anda dapat menetapkan nilai baru untuk timer ini setiap kali Anda memperbarui eksekusi pekerjaan.

**Note**

`UpdateJobExecution` dapat membuang pengatur waktu langkah yang sudah dibuat dengan membuat pengatur waktu langkah baru dengan nilai `-1`.

```
{
 ...
 "statusDetails": {
 "string" : "string"
 },
 "stepTimeoutInMinutes": number
}
```

- Untuk membuat pengatur waktu langkah baru, Anda juga dapat memanggil operasi [StartNextPendingJobExecution](#) API.

Coba lagi konfigurasi

**Note**

Saat Anda membuat pekerjaan, pertimbangkan jumlah percobaan ulang yang sesuai untuk digunakan untuk konfigurasi Anda. Untuk menghindari timbulnya biaya berlebih karena potensi kegagalan coba lagi, tambahkan konfigurasi batal. Setelah pekerjaan dibuat, jumlah percobaan ulang tidak dapat diperbarui. Anda hanya dapat mengatur jumlah percobaan ulang ke 0 dengan menggunakan operasi [UpdateJob](#) API.

Untuk menambahkan konfigurasi ini dengan menggunakan API, tentukan [jobExecutionsRetryConfig](#) parameter saat Anda menjalankan [CreateJob](#), atau operasi [CreateJobTemplate](#) API.

```
{
 ...
 "jobExecutionsRetryConfig": {
 "criteriaList": [
 {
 "failureType": "string",
```

```
 "numberOfRetries": number
 }
]
 }
 ...
}
```

Di mana `Criterialist` adalah array yang menentukan daftar kriteria yang menentukan jumlah percobaan ulang yang diizinkan untuk setiap jenis kegagalan untuk pekerjaan.

## Perangkat dan pekerjaan

Perangkat dapat berkomunikasi dengan AWS IoT Jobs menggunakan MQTT, HTTP Signature Version 4, atau HTTP TLS. Untuk menentukan titik akhir yang akan digunakan saat perangkat Anda berkomunikasi dengan AWS IoT Jobs, jalankan perintah. `DescribeEndpoint` Misalnya, jika Anda menjalankan perintah ini:

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

Anda mendapatkan hasil yang mirip dengan yang berikut ini:

```
{
 "endpointAddress": "a1b2c3d4e5f6g7-ats.iot.us-west-2.amazonaws.com"
}
```

## Menggunakan protokol MQTT

Perangkat dapat berkomunikasi dengan AWS IoT Jobs menggunakan protokol MQTT. Perangkat berlangganan topik MQTT untuk diberitahu tentang pekerjaan baru dan menerima tanggapan dari layanan Pekerjaan. AWS IoT Perangkat mempublikasikan topik MQTT untuk menanyakan atau memperbarui status peluncuran pekerjaan. Setiap perangkat memiliki topik MQTT umum sendiri. Untuk informasi selengkapnya tentang menerbitkan dan berlangganan topik MQTT, lihat [Protokol komunikasi perangkat](#)

Dengan metode komunikasi ini, perangkat Anda menggunakan sertifikat khusus perangkat dan kunci privat untuk mengautentikasi dengan Jobs. AWS IoT

Perangkat Anda dapat berlangganan topik berikut. `thing-name` adalah nama benda yang terkait dengan perangkat.

- **`$aws/things/thing-name/jobs/notify`**

Berlangganan topik ini untuk memberi tahu Anda saat peluncuran pekerjaan ditambahkan atau dihapus dari daftar peluncuran pekerjaan yang tertunda.

- **`$aws/things/thing-name/jobs/notify-next`**

Berlangganan topik ini untuk memberi tahu Anda ketika eksekusi pekerjaan tertunda berikutnya telah berubah.

- **`$aws/things/thing-name/jobs/request-name/accepted`**

Layanan AWS IoT Jobs menerbitkan pesan sukses dan gagal pada topik MQTT. Topik dibentuk dengan menambahkan `accepted` atau `rejected` ke topik yang digunakan untuk membuat permintaan. Di sini, `request-name` adalah nama permintaan seperti `Get` dan topiknya dapat berupa `$aws/things/myThing/jobs/get`. AWS IoT Jobs kemudian menerbitkan pesan sukses tentang `$aws/things/myThing/jobs/get/accepted` topik tersebut.

- **`$aws/things/thing-name/jobs/request-name/rejected`**

Di sini, `request-name` adalah nama permintaan seperti `Get`. Jika permintaan gagal, AWS IoT Jobs menerbitkan pesan kegagalan pada `$aws/things/myThing/jobs/get/rejected` topik tersebut.

Anda juga dapat menggunakan operasi HTTPS API berikut:

- Perbarui status eksekusi pekerjaan dengan memanggil [UpdateJobExecutionAPI](#).
- Kueri status eksekusi pekerjaan dengan memanggil [DescribeJobExecutionAPI](#).
- Ambil daftar eksekusi pekerjaan yang tertunda dengan memanggil API [GetPendingJobExecutions](#)
- Ambil eksekusi pekerjaan tertunda berikutnya dengan memanggil [DescribeJobExecutionAPI](#) dengan `jobId as$next`.
- Dapatkan dan mulai eksekusi pekerjaan tertunda berikutnya dengan memanggil [StartNextPendingJobExecutionAPI](#).

## Menggunakan HTTP Signature Versi 4

Perangkat dapat berkomunikasi dengan AWS IoT Jobs menggunakan HTTP Signature Version 4 pada port 443. Ini adalah metode yang digunakan oleh AWS SDK dan CLI. Untuk informasi

selengkapnya tentang alat tersebut, lihat [Referensi AWS CLI Perintah: iot-jobs-data](#) atau [AWS SDK dan Alat](#) dan lihat `lotJobsDataPlane` bagian untuk bahasa pilihan Anda.

Dengan metode komunikasi ini, perangkat Anda menggunakan kredensial IAM untuk mengautentikasi dengan Jobs. AWS IoT

Perintah berikut tersedia menggunakan metode ini:

- `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 ...
```

## Menggunakan HTTP TLS

Perangkat dapat berkomunikasi dengan AWS IoT Jobs menggunakan HTTP TLS pada port 8443 menggunakan klien perangkat lunak pihak ketiga yang mendukung protokol ini.

Dengan metode ini, perangkat Anda menggunakan otentikasi berbasis sertifikat X.509 (misalnya, sertifikat khusus perangkat dan kunci privat).

Perintah berikut tersedia menggunakan metode ini:

- `DescribeJobExecution`

- `GetPendingJobExecutions`

- `StartNextPendingJobExecution`

- `UpdateJobExecution`

## Perangkat pemrograman untuk bekerja dengan pekerjaan

Contoh di bagian ini menggunakan MQTT untuk mengilustrasikan cara kerja perangkat dengan layanan Jobs. AWS IoT Atau, Anda dapat menggunakan perintah API atau CLI yang sesuai. Untuk contoh ini, kami mengasumsikan perangkat bernama *MyThing* yang berlangganan topik MQTT berikut:

- `$aws/things/MyThing/jobs/notify` (atau `$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`

Jika Anda menggunakan penandatanganan kode AWS IoT, kode perangkat Anda harus memverifikasi tanda tangan file kode Anda. Tanda tangan ada di dokumen pekerjaan di `codesign` properti. Untuk informasi selengkapnya tentang memverifikasi tanda tangan file kode, lihat [Contoh Agen Perangkat](#).

### Topik

- [Alur kerja perangkat](#)
- [Alur kerja pekerjaan](#)
- [Pemberitahuan pekerjaan](#)

## Alur kerja perangkat

Perangkat dapat menangani pekerjaan yang dijalankannya menggunakan salah satu cara berikut.

- Dapatkan pekerjaan berikutnya
  1. Saat perangkat pertama kali online, perangkat harus berlangganan `notify-next` topik perangkat.
  2. Hubungi [DescribeJobExecution](#) MQTT API dengan `JobId $next` untuk mendapatkan pekerjaan berikutnya, dokumen pekerjaannya, dan detail lainnya, termasuk status apa pun yang disimpan. `statusDetails` Jika dokumen pekerjaan memiliki tanda tangan file kode, Anda harus memverifikasi tanda tangan sebelum melanjutkan dengan memproses permintaan pekerjaan.

3. Panggil API [UpdateJobExecution](#) MQTT untuk memperbarui status pekerjaan. Atau, untuk menggabungkan ini dan langkah sebelumnya dalam satu panggilan, perangkat dapat menelepon [StartNextPendingJobExecution](#).
4. (Opsional) Anda dapat menambahkan pengatur waktu langkah dengan menetapkan nilai `stepTimeoutInMinutes` saat Anda memanggil salah satu [UpdateJobExecution](#) atau [StartNextPendingJobExecution](#).
5. Lakukan tindakan yang ditentukan oleh dokumen pekerjaan menggunakan [UpdateJobExecution](#) MQTT API untuk melaporkan kemajuan pekerjaan.
6. Lanjutkan untuk memantau eksekusi pekerjaan dengan memanggil [DescribeJobExecution](#) MQTT API dengan `JobId` ini. Jika eksekusi pekerjaan dihapus, [DescribeJobExecution](#) mengembalikan `fileResourceNotFoundException`.

Perangkat harus dapat memulihkan ke keadaan yang valid jika eksekusi pekerjaan dibatalkan atau dihapus saat perangkat menjalankan pekerjaan.

7. Hubungi [UpdateJobExecution](#) MQTT API setelah selesai dengan pekerjaan untuk memperbarui status pekerjaan dan melaporkan keberhasilan atau kegagalan.
8. Karena status eksekusi pekerjaan ini telah diubah menjadi status terminal, pekerjaan berikutnya yang tersedia untuk eksekusi (jika ada) berubah. Perangkat diberi tahu bahwa eksekusi pekerjaan tertunda berikutnya telah berubah. Pada titik ini, perangkat harus melanjutkan seperti yang dijelaskan pada langkah 2.

Jika perangkat tetap online, ia terus menerima pemberitahuan tentang eksekusi pekerjaan tertunda berikutnya. Ini termasuk data eksekusi pekerjaannya, ketika menyelesaikan pekerjaan atau eksekusi pekerjaan baru yang tertunda ditambahkan. Ketika ini terjadi, perangkat berlanjut seperti yang dijelaskan pada langkah 2.

- Pilih dari pekerjaan yang tersedia
  1. Ketika perangkat pertama kali online, itu harus berlangganan `notify` topik hal itu.
  2. Hubungi [GetPendingJobExecutions](#) MQTT API untuk mendapatkan daftar eksekusi pekerjaan yang tertunda.
  3. Jika daftar berisi satu atau lebih eksekusi pekerjaan, pilih salah satu.
  4. Panggil API [DescribeJobExecution](#) MQTT untuk mendapatkan dokumen pekerjaan dan detail lainnya, termasuk status apa pun yang disimpan. `statusDetails`



5. Panggil API [UpdateJobExecution](#) MQTT untuk memperbarui status pekerjaan. Jika `includeJobDocument` bidang diatur ke `true` dalam perintah ini, perangkat dapat melewati langkah sebelumnya dan mengambil dokumen pekerjaan pada saat ini.
6. Secara opsional, Anda dapat menambahkan pengatur waktu langkah dengan menyetel nilai `stepTimeoutInMinutes` saat Anda menelepon [UpdateJobExecution](#).
7. Lakukan tindakan yang ditentukan oleh dokumen pekerjaan menggunakan [UpdateJobExecution](#) MQTT API untuk melaporkan kemajuan pekerjaan.
8. Lanjutkan untuk memantau eksekusi pekerjaan dengan memanggil [DescribeJobExecution](#) MQTT API dengan `JobId` ini. Jika eksekusi pekerjaan dibatalkan atau dihapus saat perangkat menjalankan pekerjaan, perangkat harus dapat memulihkan ke status yang valid.
9. Hubungi [UpdateJobExecution](#) MQTT API setelah selesai dengan pekerjaan untuk memperbarui status pekerjaan dan melaporkan keberhasilan atau kegagalan.

Jika perangkat tetap online, itu akan diberitahu tentang semua eksekusi pekerjaan yang tertunda ketika eksekusi pekerjaan baru yang tertunda tersedia. Ketika ini terjadi, perangkat dapat melanjutkan seperti yang dijelaskan pada langkah 2.

Jika perangkat tidak dapat melakukan pekerjaan, perangkat harus memanggil [UpdateJobExecution](#) MQTT API untuk memperbarui status pekerjaan ke `REJECTED`

## Alur kerja pekerjaan

Berikut ini menunjukkan langkah-langkah yang berbeda dalam alur kerja pekerjaan mulai dari memulai pekerjaan baru hingga melaporkan status penyelesaian pelaksanaan pekerjaan.

### Memulai pekerjaan baru

Saat pekerjaan baru dibuat, AWS IoT Jobs menerbitkan pesan tentang `$aws/things/thing-name/jobs/notify` topik untuk setiap perangkat target.

Pesan tersebut berisi informasi berikut:

```
{
 "timestamp":1476214217017,
 "jobs":{
 "QUEUED":[{"
 "jobId":"0001",
 "queuedAt":1476214216981,
```

```
 "lastUpdatedAt":1476214216981,
 "versionNumber" : 1
]}
}
```

Perangkat menerima pesan ini tentang '\$aws/things/*thingName*/jobs/notify' topik saat eksekusi pekerjaan diantrian.

### Note

Untuk pekerjaan dengan opsionalSchedulingConfig, pekerjaan akan mempertahankan status status awalSCHEDULED. Ketika pekerjaan mencapai yang dipilihstartTime, hal berikut akan terjadi:

- Status status pekerjaan akan diperbarui keIN\_PROGRESS.
- Pekerjaan akan memulai peluncuran dokumen pekerjaan ke semua perangkat dalam kelompok sasaran.

## Dapatkan informasi pekerjaan

Untuk mendapatkan informasi selengkapnya tentang eksekusi pekerjaan, perangkat memanggil API [DescribeJobExecution](#) MQTT dengan includeJobDocument bidang yang disetel ke true (default).

Jika permintaan berhasil, layanan AWS IoT Jobs menerbitkan pesan tentang \$aws/things/MyThing/jobs/0023/get/accepted topik:

```
{
 "clientToken" : "client-001",
 "timestamp" : 1489097434407,
 "execution" : {
 "approximateSecondsBeforeTimedOut": number,
 "jobId" : "023",
 "status" : "QUEUED",
 "queuedAt" : 1489097374841,
 "lastUpdatedAt" : 1489097374841,
 "versionNumber" : 1,
 "jobDocument" : {
 < contents of job document >
 }
 }
}
```

```
}
}
```

Jika permintaan gagal, layanan AWS IoT Jobs akan menerbitkan pesan tentang `$aws/things/MyThing/jobs/0023/get/rejected` topik tersebut.

Perangkat sekarang memiliki dokumen pekerjaan yang dapat digunakan untuk melakukan operasi jarak jauh untuk pekerjaan itu. Jika dokumen pekerjaan berisi URL presigned Amazon S3, perangkat dapat menggunakan URL tersebut untuk mengunduh file apa pun yang diperlukan untuk pekerjaan tersebut.

## Laporkan status eksekusi pekerjaan

Saat perangkat menjalankan pekerjaan, perangkat dapat memanggil [UpdateJobExecution](#) MQTT API untuk memperbarui status eksekusi pekerjaan.

Misalnya, perangkat dapat memperbarui status eksekusi pekerjaan `IN_PROGRESS` dengan menerbitkan pesan berikut tentang `$aws/things/MyThing/jobs/0023/update` topik:

```
{
 "status": "IN_PROGRESS",
 "statusDetails": {
 "progress": "50%"
 },
 "expectedVersion": "1",
 "clientToken": "client001"
}
```

Pekerjaan merespons dengan menerbitkan pesan ke `$aws/things/MyThing/jobs/0023/update/accepted` atau `$aws/things/MyThing/jobs/0023/update/rejected` topik:

```
{
 "clientToken": "client001",
 "timestamp": 1476289222841
}
```

Perangkat dapat menggabungkan dua permintaan sebelumnya dengan menelepon [StartNextPendingJobExecution](#). Itu mendapatkan dan memulai eksekusi pekerjaan tertunda berikutnya dan memungkinkan perangkat untuk memperbarui status eksekusi pekerjaan. Permintaan ini juga mengembalikan dokumen pekerjaan ketika ada eksekusi pekerjaan yang tertunda.

Jika pekerjaan berisi a [TimeoutConfig](#), pengatur waktu yang sedang berlangsung mulai berjalan. Anda juga dapat mengatur pengatur waktu langkah untuk eksekusi pekerjaan dengan menetapkan nilai `stepTimeoutInMinutes` saat Anda menelepon [UpdateJobExecution](#). Pengatur waktu langkah hanya berlaku untuk eksekusi pekerjaan yang Anda perbarui. Anda dapat menetapkan nilai baru untuk timer ini setiap kali Anda memperbarui eksekusi pekerjaan. Anda juga dapat membuat pengatur waktu langkah saat menelepon [StartNextPendingJobExecution](#). Jika eksekusi pekerjaan tetap dalam `IN_PROGRESS` status lebih lama dari interval pengatur waktu langkah, itu gagal dan beralih ke `TIMED_OUT` status terminal. Pengatur waktu langkah tidak berpengaruh pada pengatur waktu yang sedang berlangsung yang Anda atur saat membuat pekerjaan.

`statusBidang` dapat diatur ke `IN_PROGRESS`, `SUCCEEDED`, atau `FAILED`. Anda tidak dapat memperbarui status eksekusi pekerjaan yang sudah dalam status terminal.

## Eksekusi laporan selesai

Ketika perangkat selesai menjalankan pekerjaan, ia memanggil [UpdateJobExecution](#) MQTT API. Jika pekerjaan berhasil, atur status ke `SUCCEEDED` dan, di payload pesan `statusDetails`, tambahkan informasi lain tentang pekerjaan sebagai pasangan nama-nilai. Pengatur waktu dalam proses dan langkah berakhir ketika eksekusi pekerjaan selesai.

Sebagai contoh:

```
{
 "status": "SUCCEEDED",
 "statusDetails": {
 "progress": "100%"
 },
 "expectedVersion": "2",
 "clientToken": "client-001"
}
```

Jika pekerjaan tidak berhasil, atur status ke `FAILED` dan, di `statusDetails`, tambahkan informasi tentang kesalahan yang terjadi:

```
{
 "status": "FAILED",
 "statusDetails": {
 "errorCode": "101",
 "errorMsg": "Unable to install update"
 },
}
```

```
"expectedVersion":"2",
"clientToken":"client-001"
}
```

### Note

`statusDetailsAtribut` dapat berisi sejumlah pasangan nama-nilai.

Ketika layanan AWS IoT Jobs menerima pembaruan ini, ia menerbitkan pesan tentang `$aws/things/MyThing/jobs/notify` topik tersebut untuk menunjukkan bahwa pelaksanaan pekerjaan telah selesai:

```
{
 "timestamp":1476290692776,
 "jobs":{}
}
```

## Pekerjaan tambahan

Jika ada eksekusi pekerjaan lain yang tertunda untuk perangkat, mereka disertakan dalam pesan yang dipublikasikan ke `$aws/things/MyThing/jobs/notify`.

Sebagai contoh:

```
{
 "timestamp":1476290692776,
 "jobs":{
 "QUEUED":[{
 "jobId":"0002",
 "queuedAt":1476290646230,
 "lastUpdatedAt":1476290646230
 }],
 "IN_PROGRESS":[{
 "jobId":"0003",
 "queuedAt":1476290646230,
 "lastUpdatedAt":1476290646230
 }
]
}
```

## Pemberitahuan pekerjaan

Layanan AWS IoT Pekerjaan menerbitkan pesan MQTT ke topik yang dipesan saat lowongan tertunda atau saat eksekusi pekerjaan pertama dalam daftar berubah. Perangkat dapat melacak pekerjaan yang tertunda dengan berlangganan topik ini.

### Jenis pemberitahuan Job

Notifikasi pekerjaan dipublikasikan ke topik MQTT sebagai muatan JSON. Ada dua jenis notifikasi:

#### ListNotification

A `ListNotification` berisi daftar tidak lebih dari 15 eksekusi pekerjaan yang tertunda. Mereka diurutkan berdasarkan status (eksekusi `IN_PROGRESS` pekerjaan sebelum eksekusi `QUEUED` pekerjaan) dan kemudian pada saat mereka antri.

A `ListNotification` diterbitkan setiap kali salah satu kriteria berikut terpenuhi.

- Eksekusi pekerjaan baru diantrian atau diubah ke status non-terminal (`IN_PROGRESS` atau `QUEUED`).
- Eksekusi pekerjaan lama berubah menjadi status terminal (`FAILED`, `SUCCEEDED`, `CANCELED`, `TIMED_OUT`, `REJECTED`, atau `REMOVED`).

#### Pemberitahuan Daftar (Hingga 15 eksekusi pekerjaan yang tertunda di **QUEUED** atau **IN\_PROGRESS** )

Tanpa Konfigurasi Penjadwalan Opsional dan Jendela Pemeliharaan Berulang

(Hingga 10 eksekusi pekerjaan)

Selalu muncul di `ListNotification`.

Dengan Konfigurasi Penjadwalan Opsional dan Jendela Pemeliharaan Berulang

(Hingga 5 eksekusi pekerjaan)

Hanya muncul di `ListNotification` selama jendela pemeliharaan.

#### NextNotification

- A `NextNotification` berisi informasi ringkasan tentang eksekusi pekerjaan yang berikutnya dalam antrian.

A `NextNotification` diterbitkan setiap kali eksekusi pekerjaan pertama dalam daftar berubah.

- Eksekusi pekerjaan baru ditambahkan ke daftar sebagai `QUEUED`, dan itu yang pertama dalam daftar.
- Status eksekusi pekerjaan yang ada yang bukan yang pertama dalam daftar berubah dari `QUEUED` ke `IN_PROGRESS`, dan menjadi yang pertama dalam daftar. (Ini terjadi ketika tidak ada eksekusi `IN_PROGRESS` pekerjaan lain dalam daftar atau ketika eksekusi pekerjaan yang statusnya berubah dari `QUEUED` ke `IN_PROGRESS` antri lebih awal dari eksekusi `IN_PROGRESS` pekerjaan lainnya dalam daftar.)
- Status eksekusi pekerjaan yang pertama dalam daftar berubah menjadi status terminal dan dihapus dari daftar.

Untuk informasi selengkapnya tentang menerbitkan dan berlangganan topik MQTT, lihat [the section called "Protokol komunikasi perangkat"](#)

#### Note

Pemberitahuan tidak tersedia saat Anda menggunakan Tanda Tangan HTTP Versi 4 atau HTTP TLS untuk berkomunikasi dengan pekerjaan.

## Job tertunda

Layanan AWS IoT Pekerjaan menerbitkan pesan tentang topik MQTT saat pekerjaan ditambahkan atau dihapus dari daftar eksekusi pekerjaan yang tertunda untuk suatu hal atau eksekusi pekerjaan pertama dalam daftar berubah:

- `$aws/things/thingName/jobs/notify`
- `$aws/things/thingName/jobs/notify-next`

Pesan berisi contoh payload berikut:

`$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
 }]
 }
}

```

Jika eksekusi pekerjaan yang dipanggil `this-job` berasal dari pekerjaan dengan konfigurasi penjadwalan opsional yang dipilih dan peluncuran dokumen pekerjaan dijadwalkan berlangsung selama jendela pemeliharaan, itu hanya akan muncul selama jendela pemeliharaan berulang. Di luar jendela pemeliharaan, pekerjaan yang dipanggil `this-job` akan dikeluarkan dari daftar eksekusi pekerjaan yang tertunda seperti yang ditunjukkan pada contoh berikut.

```

{
 "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"}
 }
}

```

Jika eksekusi pekerjaan yang dipanggil `other-job` berasal dari pekerjaan dengan konfigurasi penjadwalan opsional yang dipilih dan peluncuran dokumen pekerjaan dijadwalkan berlangsung selama jendela pemeliharaan, itu hanya akan muncul selama jendela pemeliharaan berulang. Di luar jendela pemeliharaan, pekerjaan yang dipanggil `other-job` tidak akan terdaftar sebagai eksekusi pekerjaan berikutnya seperti yang ditunjukkan pada contoh berikut.

```
{ } //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"

```

Nilai status eksekusi pekerjaan yang mungkin adalah `QUEUED`, `IN_PROGRESS`, `FAILED`, `SUCCEEDED`, `CANCELED`, `TIMED_OUT`, `REJECTED`, dan `REMOVED`.

Rangkaian contoh berikut menunjukkan pemberitahuan yang dipublikasikan untuk setiap topik saat eksekusi pekerjaan dibuat dan diubah dari satu status ke negara bagian lainnya.

Pertama, satu pekerjaan, disebut `job1`, dibuat. Pemberitahuan ini dipublikasikan ke `jobs/notify` topik:

```

{
 "timestamp": 1517016948,

```

```
"jobs": {
 "QUEUED": [
 {
 "jobId": "job1",
 "queuedAt": 1517016947,
 "lastUpdatedAt": 1517016947,
 "executionNumber": 1,
 "versionNumber": 1
 }
]
}
```

Pemberitahuan ini dipublikasikan ke `jobs/notify-next` topik:

```
{
 "timestamp": 1517016948,
 "execution": {
 "jobId": "job1",
 "status": "QUEUED",
 "queuedAt": 1517016947,
 "lastUpdatedAt": 1517016947,
 "versionNumber": 1,
 "executionNumber": 1,
 "jobDocument": {
 "operation": "test"
 }
 }
}
```

Ketika pekerjaan lain dibuat (`job2`), pemberitahuan ini dipublikasikan ke `jobs/notify` topik:

```
{
 "timestamp": 1517017192,
 "jobs": {
 "QUEUED": [
 {
 "jobId": "job1",
 "queuedAt": 1517016947,
 "lastUpdatedAt": 1517016947,
 "executionNumber": 1,
 "versionNumber": 1
 },

```

```
{
 "jobId": "job2",
 "queuedAt": 1517017191,
 "lastUpdatedAt": 1517017191,
 "executionNumber": 1,
 "versionNumber": 1
}
]
```

Notifikasi tidak dipublikasikan ke `jobs/notify-next` topik karena pekerjaan berikutnya dalam antrian (`job1`) tidak berubah. Ketika `job1` mulai mengeksekusi, statusnya berubah menjadi `IN_PROGRESS`. Tidak ada pemberitahuan yang dipublikasikan karena daftar pekerjaan dan pekerjaan berikutnya dalam antrian tidak berubah.

Ketika `job` ketiga (`job3`) ditambahkan, pemberitahuan ini dipublikasikan ke `jobs/notify` topik:

```
{
 "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
 }
]
}
}
```

Pemberitahuan tidak dipublikasikan ke `jobs/notify-next` topik karena pekerjaan berikutnya dalam antrian masih `job1`.

Ketika `job1` selesai, statusnya berubah menjadi `SUCCEEDED`, dan pemberitahuan ini dipublikasikan ke `jobs/notify` topik:

```
{
 "timestamp": 1517186269,
 "jobs": {
 "QUEUED": [
 {
 "jobId": "job2",
 "queuedAt": 1517017191,
 "lastUpdatedAt": 1517017191,
 "executionNumber": 1,
 "versionNumber": 1
 },
 {
 "jobId": "job3",
 "queuedAt": 1517017905,
 "lastUpdatedAt": 1517017905,
 "executionNumber": 1,
 "versionNumber": 1
 }
]
 }
}
```

Pada titik ini, `job1` telah dihapus dari antrian, dan pekerjaan berikutnya yang akan dieksekusi adalah `job2`. Pemberitahuan ini dipublikasikan ke `jobs/notify-next` topik:

```
{
 "timestamp": 1517186269,
 "execution": {
 "jobId": "job2",
```

```

 "status": "QUEUED",
 "queuedAt": 1517017191,
 "lastUpdatedAt": 1517017191,
 "versionNumber": 1,
 "executionNumber": 1,
 "jobDocument": {
 "operation": "test"
 }
 }
}

```

Jika job3 harus mulai mengeksekusi sebelumnya job2 (yang tidak disarankan), status job3 dapat diubah menjadi IN\_PROGRESS. Jika ini terjadi, job2 tidak lagi berikutnya dalam antrian, dan pemberitahuan ini diterbitkan ke jobs/notify-next topik:

```

{
 "timestamp": 1517186779,
 "execution": {
 "jobId": "job3",
 "status": "IN_PROGRESS",
 "queuedAt": 1517017905,
 "startedAt": 1517186779,
 "lastUpdatedAt": 1517186779,
 "versionNumber": 2,
 "executionNumber": 1,
 "jobDocument": {
 "operation": "test"
 }
 }
}

```

Tidak ada pemberitahuan yang dipublikasikan ke jobs/notify topik karena tidak ada pekerjaan yang ditambahkan atau dihapus.

Jika perangkat menolak job2 dan memperbarui statusnya REJECTED, pemberitahuan ini dipublikasikan ke jobs/notify topik:

```

{
 "timestamp": 1517189392,
 "jobs": {
 "IN_PROGRESS": [
 {

```

```
 "jobId": "job3",
 "queuedAt": 1517017905,
 "lastUpdatedAt": 1517186779,
 "startedAt": 1517186779,
 "executionNumber": 1,
 "versionNumber": 2
 }
]
}
```

Jika job3 (yang masih dalam proses) dihapus paksa, pemberitahuan ini dipublikasikan ke jobs/notify topik:

```
{
 "timestamp": 1517189551,
 "jobs": {}
}
```

Pada titik ini, antrian kosong. Pemberitahuan ini dipublikasikan ke jobs/notify-next topik:

```
{
 "timestamp": 1517189551
}
```

## AWS IoT pekerjaan API operations

AWS IoTJobs API dapat digunakan untuk salah satu dari kategori berikut:

- Tugas administratif seperti manajemen dan pengendalian pekerjaan. Ini adalah pesawat kontrol.
- Perangkat yang melakukan pekerjaan itu. Ini adalah pesawat data, yang memungkinkan Anda untuk mengirim dan menerima data.

Manajemen dan kontrol pekerjaan menggunakan API protokol HTTPS. Perangkat dapat menggunakan MQTT atau API protokol HTTPS. API control plane dirancang untuk volume panggilan rendah yang khas saat membuat dan melacak pekerjaan. Biasanya membuka koneksi untuk satu permintaan, dan kemudian menutup koneksi setelah respons diterima. Pesawat data HTTPS dan MQTT API mengizinkan polling panjang. Operasi API ini dirancang untuk lalu lintas dalam jumlah besar yang dapat menskalakan jutaan perangkat.

Masing-masing AWS IoT Jobs HTTPS API memiliki perintah yang sesuai yang memungkinkan Anda untuk memanggil API dari AWS Command Line Interface (AWS CLI). Perintahnya adalah huruf kecil, dengan tanda hubung antara kata-kata yang membentuk nama API. Misalnya, Anda dapat memanggil `CreateJobAPI` pada CLI dengan mengetik:

```
aws iot create-job ...
```

Jika terjadi kesalahan selama operasi, Anda mendapatkan respons kesalahan yang berisi informasi tentang kesalahan.

## ErrorResponse

Berisi informasi tentang kesalahan yang terjadi selama AWS IoT Jobs operasi layanan.

Contoh berikut menunjukkan sintaks operasi ini:

```
{
 "code": "ErrorCode",
 "message": "string",
 "clientToken": "string",
 "timestamp": timestamp,
 "executionState": JobExecutionState
}
```

Berikut ini adalah uraian tentang `ErrorResponse`:

### code

`ErrorCode` dapat diatur ke:

#### InvalidTopic

Permintaan dikirim ke topik di AWS IoT Namespace pekerjaan yang tidak memetakan ke operasi API apa pun.

#### InvalidJson

Isi permintaan tidak dapat ditafsirkan sebagai JSON yang dikodekan UTF-8 yang valid.

#### InvalidRequest

Isi permintaan tidak valid. Misalnya, kode ini dikembalikan ketika `UpdateJobExecution` permintaan berisi rincian status tidak valid. Pesan berisi rincian tentang kesalahan.

## InvalidStateTransition

Pembaruan mencoba mengubah eksekusi pekerjaan ke status yang tidak valid karena status eksekusi pekerjaan saat ini. Misalnya, upaya untuk mengubah permintaan dalam keadaan SUCCEEDED untuk menyatakan IN\_PROGRESS. Dalam hal ini, isi pesan kesalahan juga berisi `executionState` bidang.

## ResourceNotFound

Yang `JobExecution` ditentukan oleh topik permintaan tidak ada.

## VersionMismatch

Versi yang diharapkan ditentukan dalam permintaan tidak cocok dengan versi eksekusi pekerjaan di AWS IoT Jobs layanan. Dalam hal ini, isi pesan kesalahan juga berisi `executionState` bidang.

## InternalError

Ada kesalahan internal selama pemrosesan permintaan.

## RequestThrottled

Permintaan itu dibatasi.

## TerminalStateReached

Terjadi ketika perintah untuk menggambarkan pekerjaan dilakukan pada pekerjaan yang dalam keadaan terminal.

## message

String pesan kesalahan.

## clientToken

String sewenang-wenang yang digunakan untuk mengkorelasikan permintaan dengan jawabannya.

## timestamp

Waktunya, dalam hitungan detik sejak zaman.

## executionState

Sebuah objek [JobExecutionState](#). Bidang ini disertakan hanya ketika `code` bidang memiliki nilai `InvalidStateTransition` atau `VersionMismatch`. Hal ini membuatnya tidak perlu dalam



kasus ini untuk melakukan `DescribeJobExecution` permintaan untuk mendapatkan data status eksekusi pekerjaan saat ini.

Berikut ini mencantumkan operasi Jobs API dan tipe data.

- [Manajemen pekerjaan dan kontrol API dan tipe data](#)
- [Pekerjaan perangkat MQTT dan HTTPS operasi API dan tipe data](#)

## Manajemen pekerjaan dan kontrol API dan tipe data

Perintah berikut tersedia untuk manajemen dan kontrol Job di CLI dan melalui protokol HTTPS.

- [Manajemen pekerjaan dan tipe data kontrol](#)
- [Manajemen pekerjaan dan pengendalian operasi API](#)

Untuk menentukan *endpoint-url* parameter untuk perintah CLI Anda, jalankan perintah ini.

```
aws iot describe-endpoint --endpoint-type=iot:Jobs
```

Perintah ini mengembalikan output berikut.

```
{
 "endpointAddress": "account-specific-prefix.jobs.iot.aws-region.amazonaws.com"
}
```

### Note

Endpoint Lowongan Kerja tidak mendukung ALPNz-amzn-http-ca.

## Manajemen pekerjaan dan tipe data kontrol

Jenis data berikut digunakan oleh manajemen dan kontrol aplikasi untuk berkomunikasi dengan AWS IoT Pekerjaan.

## Tugas

YangJobobjek berisi rincian tentang pekerjaan. Contoh berikut menunjukkan sintaks:

```
{
 "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
}
}
```

Untuk informasi lebih lanjut, lihat [Job](#) atau [job](#).

## JobSummary

YangJobSummaryobjek berisi ringkasan pekerjaan. Contoh berikut menunjukkan sintaks:

```
{
 "jobArn": "string",
 "jobId": "string",
 "status": "IN_PROGRESS|CANCELED|SUCCEEDED|SCHEDULED",
 "targetSelection": "CONTINUOUS|SNAPSHOT",
 "thingGroupId": "string",
 "createdAt": timestamp,
 "lastUpdatedAt": timestamp,
 "completedAt": timestamp
}
```

Untuk informasi lebih lanjut, lihat [JobSummary](#) atau [job-summary](#).

## JobExecution

Yang `JobExecution` objek merupakan pelaksanaan pekerjaan pada perangkat. Contoh berikut menunjukkan sintaks:

### Note

Saat Anda menggunakan operasi API control plane, `JobExecution` tipe data tidak mengandung `JobDocument` bidang. Untuk mendapatkan informasi ini, Anda dapat menggunakan [GetJobDocument](#) Operasi API atau [get-job-document](#) Perintah 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
}
```

Untuk informasi lebih lanjut, lihat [JobExecution](#) atau [job-execution](#).

## JobExecutionSummary

Yang `JobExecutionSummary` objek berisi informasi ringkasan eksekusi pekerjaan. Contoh berikut menunjukkan sintaks:

```
{
```

```
"executionNumber": 1234567890,
"queuedAt": timestamp,
"lastUpdatedAt": timestamp,
"startedAt": timestamp,
"status": "QUEUED|IN_PROGRESS|FAILED|SUCCEEDED|CANCELED|TIMED_OUT|REJECTED|REMOVED"
}
```

Untuk informasi lebih lanjut, lihat [JobExecutionSummary](#) atau [job-execution-summary](#).

### JobExecutionSummaryForJob

Yang `JobExecutionSummaryForJob` objek berisi ringkasan informasi tentang eksekusi pekerjaan untuk pekerjaan tertentu. Contoh berikut menunjukkan sintaks:

```
{
 "executionSummaries": [
 {
 "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/MyThing",
 "jobExecutionSummary": {
 "status": "IN_PROGRESS",
 "lastUpdatedAt": 1549395301.389,
 "queuedAt": 1541526002.609,
 "executionNumber": 1
 }
 },
 ...
]
}
```

Untuk informasi lebih lanjut, lihat [JobExecutionSummaryForJob](#) atau [job-execution-summary-for-job](#).

### JobExecutionSummaryForThing

Yang `JobExecutionSummaryForThing` objek berisi ringkasan informasi tentang eksekusi pekerjaan pada hal tertentu. fContoh berikut menunjukkan sintaks:

```
{
 "executionSummaries": [
 {
 "jobExecutionSummary": {
 "status": "IN_PROGRESS",
 ...
 }
 }
]
}
```

```
 "lastUpdatedAt": 1549395301.389,
 "queuedAt": 1541526002.609,
 "executionNumber": 1
 },
 "jobId": "MyThingJob"
},
 ...
]
}
```

Untuk informasi lebih lanjut, lihat [JobExecutionSummaryForThing](#) atau [job-execution-summary-for-thing](#).

## Manajemen pekerjaan dan pengendalian operasi API

Gunakan operasi API atau perintah CLI berikut:

### AssociateTargetsWithJob

Mengaitkan kelompok dengan pekerjaan berkelanjutan. Kriteria berikut harus dipenuhi:

- Pekerjaan harus telah dibuat dengan `targetSelectionField` set ke `CONTINUOUS`.
- Status pekerjaan saat ini harus `IN_PROGRESS`.
- Jumlah total target yang terkait dengan pekerjaan tidak boleh melebihi 100.

### HTTPS request

```
POST /jobs/jobId/targets

{
 "targets": ["string"],
 "comment": "string"
}
```

Untuk informasi selengkapnya, lihat [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-jsonFormat:**

```
{
 "targets": [
 "string"
],
 "jobId": "string",
 "comment": "string"
}
```

Untuk informasi selengkapnya, lihat [associate-targets-with-job](#).

## CancelJob

Membatalkan pekerjaan.

### HTTPS request

```
PUT /jobs/jobId/cancel

{
 "force": boolean,
 "comment": "string",
 "reasonCode": "string"
}
```

Untuk informasi selengkapnya, lihat [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-jsonFormat:**

```
{
 "jobId": "string",
 "force": boolean,
 "comment": "string"
}
```

Untuk informasi selengkapnya, lihat [cancel-job](#).

## CancelJobExecution

Membatalkan eksekusi pekerjaan di perangkat.

### HTTPS request

```
PUT /things/thingName/jobs/jobId/cancel
```

```
{
 "force": boolean,
 "expectedVersion": "string",
 "statusDetails": {
 "string": "string"
 ...
 }
}
```

Untuk informasi selengkapnya, lihat [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-jsonFormat:

```
{
```



```
"jobId": "string",
"thingName": "string",
"force": boolean,
"expectedVersion": long,
"statusDetails": {
 "string": "string"
}
}
```

Untuk informasi selengkapnya, lihat [cancel-job-execution](#).

## CreateJob

Menciptakan pekerjaan. Anda dapat memberikan dokumen pekerjaan sebagai tautan ke file dalam bucket Amazon S3 (`documentSourceparameter`), atau dalam tubuh permintaan (`documentparameter`).

Pekerjaan dapat dibuat berkesinambungan dengan mengatur opsional `targetSelectionparameter` ke `CONTINUOUS` (defaultnya adalah `SNAPSHOT`). Pekerjaan berkelanjutan dapat digunakan untuk onboard atau meng-upgrade perangkat saat ditambahkan ke grup karena terus berjalan dan diluncurkan pada hal-hal yang baru ditambahkan. Hal ini dapat terjadi bahkan setelah hal-hal dalam kelompok pada saat pekerjaan itu dibuat telah menyelesaikan pekerjaan.

Pekerjaan dapat memiliki opsional [TimeoutConfig](#), yang menetapkan nilai timer dalam proses. Timer dalam proses tidak dapat diperbarui dan berlaku untuk semua eksekusi pekerjaan.

Validasi berikut dilakukan pada argumen ke `CreateJobAPI`:

- Yang `targets` argumen harus berupa daftar hal yang valid atau kelompok hal ARN. Semua hal dan kelompok hal harus ada di Akun AWS.
- Yang `documentSource` argumen harus berupa URL Amazon S3 yang valid untuk dokumen pekerjaan. URL Amazon S3 ada dalam bentuk: `https://s3.amazonaws.com/bucketName/objectName`.
- Dokumen yang disimpan dalam URL yang ditentukan oleh `documentSource` argumen harus berupa dokumen JSON yang dikodekan UTF-8.
- Ukuran dokumen pekerjaan dibatasi hingga 32 KB karena batas ukuran pesan MQTT (128 KB) dan enkripsi.
- Yang `jobId` harus unik di Akun 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
 }
 }
}

```

Untuk informasi selengkapnya, lihat [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-jsonFormat:

```

{
 "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"
}
}
```

Untuk informasi selengkapnya, lihat [create-job](#).

## DeleteJob

Menghapus pekerjaan dan eksekusi pekerjaan terkait.

Menghapus pekerjaan dapat memakan waktu, tergantung pada jumlah eksekusi pekerjaan yang dibuat untuk pekerjaan dan berbagai faktor lainnya. Sementara pekerjaan sedang dihapus, status pekerjaan ditampilkan sebagai "DELETION\_IN\_PROGRESS". Mencoba menghapus atau membatalkan pekerjaan yang statusnya sudah "DELETION\_IN\_PROGRESS" menghasilkan kesalahan.

## HTTPS request

```
DELETE /jobs/jobId?force=force
```

Untuk informasi selengkapnya, lihat [DeleteJob](#).

### CLI syntax

```
aws iot delete-job \
--job-id <value> \
[--force | --no-force] \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

### cli-input-jsonFormat:

```
{
 "jobId": "string",
 "force": boolean
}
```

Untuk informasi selengkapnya, lihat [delete-job](#).

## DeleteJobExecution

Menghapus eksekusi pekerjaan.

### HTTPS request

```
DELETE /things/thingName/jobs/jobId/executionNumber/executionNumber?force=force
```

Untuk informasi selengkapnya, lihat [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-jsonFormat:

```
{
```

```
"jobId": "string",
"thingName": "string",
"executionNumber": long,
"force": boolean
}
```

Untuk informasi selengkapnya, lihat [delete-job-execution](#).

## DescribeJob

Mendapat rincian eksekusi pekerjaan.

### HTTPS request

```
GET /jobs/jobId
```

Untuk informasi selengkapnya, lihat [DescribeJob](#).

### CLI syntax

```
aws iot describe-job \
--job-id <value> \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

cli-input-jsonFormat:

```
{
 "jobId": "string"
}
```

Untuk informasi selengkapnya, lihat [describe-job](#).

## DescribeJobExecution

Mendapat rincian eksekusi pekerjaan. Status eksekusi pekerjaan harus SUCCEEDED atau FAILED.

### HTTPS request

```
GET /things/thingName/jobs/jobId?executionNumber=executionNumber
```

Untuk informasi selengkapnya, lihat [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-jsonFormat:

```
{
 "jobId": "string",
 "thingName": "string",
 "executionNumber": long
}
```

Untuk informasi selengkapnya, lihat [describe-job-execution](#).

### GetJobDocument

Mendapat dokumen pekerjaan untuk pekerjaan.

#### Note

URL placeholder tidak diganti dengan URL Amazon S3 yang telah ditandatangani sebelumnya dalam dokumen yang dikembalikan. URL presigned dihasilkan hanya bila AWS IoT Layanan pekerjaan menerima permintaan melalui MQTT.

### HTTPS request

```
GET /jobs/jobId/job-document
```

Untuk informasi selengkapnya, lihat [GetJobDocument](#).

### CLI syntax

```
aws iot get-job-document \

```

```
--job-id <value> \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

cli-input-jsonFormat:

```
{
 "jobId": "string"
}
```

Untuk informasi selengkapnya, lihat [get-job-document](#).

## ListJobExecutionsForJob

Mendapat daftar eksekusi pekerjaan untuk pekerjaan.

HTTPS request

```
GET /jobs/jobId/things?status=status&maxResults=maxResults&nextToken=nextToken
```

Untuk informasi selengkapnya, lihat [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-jsonFormat:

```
{
 "jobId": "string",
 "status": "string",
 "maxResults": "integer",
 "nextToken": "string"
}
```



Untuk informasi selengkapnya, lihat [list-job-executions-for-job](#).

## ListJobExecutionsForThing

Mendapat daftar eksekusi pekerjaan untuk sesuatu.

### HTTPS request

```
GET /things/thingName/jobs?status=status&maxResults=maxResults&nextToken=nextToken
```

Untuk informasi selengkapnya, lihat [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-jsonFormat:

```
{
 "thingName": "string",
 "status": "string",
 "maxResults": "integer",
 "nextToken": "string"
}
```

Untuk informasi selengkapnya, lihat [list-job-executions-for-thing](#).

## ListJobs

Mendapat daftar pekerjaan diAkun AWS.

### HTTPS request

```
GET /jobs?
status=status&targetSelection=targetSelection&thingGroupName=thingGroupName&thingGroupId=thingGroupId
```

Untuk informasi selengkapnya, lihat [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-jsonFormat:

```
{
 "status": "string",
 "targetSelection": "string",
 "maxResults": "integer",
 "nextToken": "string",
 "thingGroupName": "string",
 "thingGroupId": "string"
}
```

Untuk informasi selengkapnya, lihat [list-jobs](#).

## UpdateJob

Memperbarui bidang yang didukung dari pekerjaan yang ditentukan. Nilai yang diperbarui untuk `timeoutConfig` berlaku hanya untuk peluncuran yang baru dalam proses. Saat ini, peluncuran dalam proses terus diluncurkan dengan konfigurasi batas waktu sebelumnya.

## 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
}
}

```

Untuk informasi selengkapnya, lihat [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-jsonFormat:

```
{
```

```
"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
}
}
```

Untuk informasi selengkapnya, lihat [update-job](#).

## Pekerjaan perangkat MQTT dan HTTPS operasi API dan tipe data

Perintah berikut tersedia melalui protokol MQTT dan HTTPS. Gunakan operasi API ini pada bidang data untuk perangkat yang menjalankan pekerjaan.

### Jobs perangkat MQTT dan HTTPS tipe data

Jenis data berikut digunakan untuk berkomunikasi dengan AWS IoT Layanan pekerjaan melalui protokol MQTT dan HTTPS.

## JobExecution

Yang `JobExecution` objek merupakan pelaksanaan pekerjaan pada perangkat. Contoh berikut menunjukkan sintaks:

### Note

Saat Anda menggunakan operasi API bidang data MQTT dan HTTP, `JobExecution` tipe data berisi `JobDocument` bidang. Perangkat Anda dapat menggunakan informasi ini untuk mengambil dokumen pekerjaan dari eksekusi pekerjaan.

```
{
 "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
}
```

Untuk informasi lebih lanjut, lihat [JobExecution](#) atau [job-execution](#).

## JobExecutionState

Yang `JobExecutionState` berisi informasi tentang keadaan eksekusi pekerjaan. Contoh berikut menunjukkan sintaks:

```
{
 "status": "QUEUED|IN_PROGRESS|FAILED|SUCCEEDED|CANCELED|TIMED_OUT|REJECTED|
 REMOVED",
 "statusDetails": {
 "string": "string"
 ...
 }
}
```

```
}
 "versionNumber": "number"
}
```

Untuk informasi lebih lanjut, lihat [JobExecutionState](#) atau [job-execution-state](#).

### JobExecutionSummary

Berisi subset informasi tentang eksekusi pekerjaan. Contoh berikut menunjukkan sintaks:

```
{
 "jobId": "string",
 "queuedAt": timestamp,
 "startedAt": timestamp,
 "lastUpdatedAt": timestamp,
 "versionNumber": "number",
 "executionNumber": long
}
```

Untuk informasi lebih lanjut, lihat [JobExecutionSummary](#) atau [job-execution-summary](#).

Pelajari lebih lanjut tentang operasi MQTT dan HTTPS API di bagian berikut:

- [Pekerjaan perangkat operasi MQTT API](#)
- [Lowongan kerja perangkat HTTP API](#)


## Pekerjaan perangkat operasi MQTT API

Anda dapat mengeluarkan perintah perangkat pekerjaan dengan menerbitkan pesan MQTT ke [Topik cadangan yang digunakan untuk perintah Pekerjaan](#).

Klien sisi perangkat Anda harus berlangganan topik pesan respons dari perintah ini. Jika Anda menggunakan AWS IoT Klien Perangkat, perangkat Anda akan secara otomatis berlangganan topik respons. Ini berarti bahwa broker pesan akan mempublikasikan topik pesan respons kepada klien yang menerbitkan pesan perintah, apakah klien Anda telah berlangganan topik pesan respons atau tidak. Pesan respons ini tidak melewati broker pesan dan tidak dapat berlangganan oleh klien atau aturan lain.

Saat berlangganan pekerjaan dan `jobExecution` topik acara untuk solusi pemantauan armada Anda, pertama-tama aktifkan [pekerjaan dan pelaksanaan pekerjaan](#) untuk menerima acara apa pun di sisi cloud. Pesan kemajuan pekerjaan yang diproses melalui broker pesan dan dapat digunakan

oleh AWS IoT diterbitkan sebagai [Acara Lowongan](#). Karena broker pesan menerbitkan pesan respons, bahkan tanpa berlangganan eksplisit kepada mereka, klien Anda harus dikonfigurasi untuk menerima dan mengidentifikasi pesan yang diterimanya. Klien Anda juga harus mengkonfirmasi bahwa *ThingName* dalam topik pesan masuk berlaku untuk nama hal klien sebelum klien bertindak pada pesan.

 Note

Pesan itu AWS IoT mengirim dalam menanggapi pesan perintah MQTT Jobs API dibebankan ke akun Anda, apakah Anda berlangganan mereka secara eksplisit atau tidak.

Berikut ini menunjukkan operasi API MQTT dan sintaks permintaan dan responsnya. Semua operasi MQTT API memiliki parameter berikut:

#### ClientToken

Token klien opsional yang digunakan untuk mengkorelasikan permintaan dan tanggapan. Masukkan nilai arbitrer di sini dan itu tercermin dalam respons.

#### timestamp

Waktu dalam hitungan detik sejak zaman, ketika pesan dikirim.

#### GetPendingJobExecutions

Mendapat daftar semua pekerjaan yang tidak dalam keadaan terminal, untuk hal tertentu.

Untuk memanggil API ini, publikasikan pesan di `$aws/things/thingName/jobs/get`.

Meminta muatan:

```
{ "clientToken": "string" }
```

Broker pesan akan menerbitkan `$aws/things/thingName/jobs/get/accepted` dan `$aws/things/thingName/jobs/get/rejected` bahkan tanpa berlangganan khusus untuk mereka. Namun, bagi klien Anda untuk menerima pesan, itu harus mendengarkan mereka. Untuk informasi lebih lanjut, lihat [catatan tentang pesan Jobs API](#).

Muatan respons:

```
{
 "InProgressJobs" : [JobExecutionSummary ...],
 "queuedJobs" : [JobExecutionSummary ...],
 "timestamp" : 1489096425069,
 "clientToken" : "client-001"
}
```

Dimana `InProgressJobs` dan `queuedJobs` mengembalikan daftar [JobExecutionSummary](#) objek yang memiliki status `IN_PROGRESS` atau `QUEUED`.

### StartNextPendingJobExecution

Mendapat dan memulai eksekusi pekerjaan tertunda berikutnya untuk suatu hal (status `IN_PROGRESS` atau `QUEUED`).

- Setiap eksekusi pekerjaan dengan status `IN_PROGRESS` dikembalikan terlebih dahulu.
- Eksekusi pekerjaan dikembalikan sesuai urutan antrian. Ketika sesuatu ditambahkan atau dihapus dari grup target untuk pekerjaan Anda, konfirmasi urutan peluncuran eksekusi pekerjaan baru apa pun dibandingkan dengan eksekusi pekerjaan yang ada.
- Jika eksekusi pekerjaan tertunda berikutnya adalah `QUEUED`, kondisinya berubah menjadi `IN_PROGRESS` dan rincian status eksekusi pekerjaan ditetapkan seperti yang ditentukan.
- Jika eksekusi pekerjaan tertunda berikutnya sudah `IN_PROGRESS`, Rincian statusnya tidak berubah.
- Jika tidak ada eksekusi pekerjaan yang tertunda, respons tidak menyertakan `execution` bidang.
- Opsional, Anda dapat membuat timer langkah dengan menetapkan nilai untuk `stepTimeoutInMinutes` properti. Jika Anda tidak memperbarui nilai properti ini dengan menjalankan `UpdateJobExecution`, eksekusi pekerjaan kali keluar ketika timer langkah berakhir.

Untuk memanggil API ini, publikasikan pesan di `$aws/things/thingName/jobs/start-next`.

Meminta muatan:

```
{
 "statusDetails": {
 "string": "job-execution-state"
 ...
 },
 "stepTimeoutInMinutes": long,
```



```
"clientToken": "string"
}
```

## statusDetails

Kumpulan pasangan nama-nilai yang menggambarkan status eksekusi pekerjaan. Jika tidak ditentukan, `statusDetail` tidak berubah.

## stepTimeoutInMinutes

Menentukan jumlah waktu perangkat ini harus menyelesaikan pelaksanaan pekerjaan ini. Jika status eksekusi pekerjaan tidak disetel ke status terminal sebelum timer ini berakhir, atau sebelum pengatur waktu disetel ulang, (dengan memanggil `UpdateJobExecution`, mengatur status ke `IN_PROGRESS` dan menentukan nilai batas waktu baru di lapangan `stepTimeoutInMinutes`) status eksekusi pekerjaan diatur ke `TIMED_OUT`. Menetapkan batas waktu ini tidak berpengaruh pada batas waktu eksekusi pekerjaan yang mungkin telah ditentukan saat pekerjaan dibuat (`CreateJob` menggunakan `timeoutConfig` bidang).

Broker pesan akan menerbitkan `$aws/things/thingName/jobs/start-next/accepted` dan `$aws/things/thingName/jobs/start-next/rejected` bahkan tanpa berlangganan khusus untuk mereka. Namun, bagi klien Anda untuk menerima pesan, itu harus mendengarkan mereka. Untuk informasi lebih lanjut, lihat [catatan tentang pesan Jobs API](#).

Muatan respons:

```
{
 "execution" : JobExecutionData,
 "timestamp" : timestamp,
 "clientToken" : "string"
}
```

Dimana `execution` adalah [JobExecution](#) objek. Misalnya:

```
{
 "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

Mendapat informasi rinci tentang eksekusi pekerjaan.

Anda dapat mengatur `jobId` kepada `$next` untuk mengembalikan eksekusi pekerjaan tertunda berikutnya untuk suatu hal (dengan status `IN_PROGRESS` atau `QUEUED`).

Untuk memanggil API ini, publikasikan pesan di `aws/things/thingName/jobs/jobId/get`.

Meminta muatan:

```
{
"jobId" : "022",
"thingName" : "MyThing",
"executionNumber": long,
"includeJobDocument": boolean,
"clientToken": "string"
}
```

### thingName

Nama hal yang terkait dengan perangkat.

### jobId

Pengenal unik ditugaskan untuk pekerjaan ini ketika itu dibuat.

Atau gunakan `$next` untuk mengembalikan eksekusi pekerjaan tertunda berikutnya untuk suatu hal (dengan status `IN_PROGRESS` atau `QUEUED`). Dalam hal ini, setiap eksekusi pekerjaan dengan status `IN_PROGRESS` dikembalikan terlebih dahulu. Eksekusi pekerjaan dikembalikan dalam urutan di mana mereka diciptakan.

### executionNumber

(Opsional) Angka yang mengidentifikasi eksekusi pekerjaan di perangkat. Jika tidak ditentukan, eksekusi pekerjaan terbaru dikembalikan.

## includeJobDocument

(Opsional) Kecuali diatur ke `false`, respon berisi dokumen pekerjaan. Defaultnya adalah `true`.

Broker pesan akan menerbitkan `$aws/things/thingName/jobs/jobId/get/accepted` dan `$aws/things/thingName/jobs/jobId/get/rejected` bahkan tanpa berlangganan khusus untuk mereka. Namun, bagi klien Anda untuk menerima pesan, itu harus mendengarkan mereka. Untuk informasi lebih lanjut, lihat [catatan tentang pesan Jobs API](#).

Muatan respons:

```
{
 "execution" : JobExecutionData,
 "timestamp": "timestamp",
 "clientToken": "string"
}
```

Dimana `execution` adalah [JobExecution](#) objek.

## UpdateJobExecution

Memperbarui status eksekusi pekerjaan. Anda opsional dapat membuat timer langkah dengan menetapkan nilai untuk `stepTimeoutInMinutes` properti. Jika Anda tidak memperbarui nilai properti ini dengan menjalankan `UpdateJobExecution` sekali lagi, eksekusi pekerjaan kali keluar ketika timer langkah berakhir.

Untuk memanggil API ini, publikasikan pesan di `$aws/things/thingName/jobs/jobId/update`.

Meminta muatan:

```
{
 "status": "job-execution-state",
 "statusDetails": {
 "string": "string"
 ...
 },
 "expectedVersion": "number",
 "executionNumber": long,
 "includeJobExecutionState": boolean,
 "includeJobDocument": boolean,
 "stepTimeoutInMinutes": long,
```

```
"clientToken": "string"
}
```

## status

Status baru untuk eksekusi pekerjaan (IN\_PROGRESS, FAILED, SUCCEEDED, atau REJECTED). Ini harus ditentukan pada setiap pembaruan.

## statusDetails

Kumpulan pasangan nama-nilai yang menggambarkan status eksekusi pekerjaan. Jika tidak ditentukan, `statusDetails` tidak berubah.

## expectedVersion

Versi saat ini diharapkan dari eksekusi pekerjaan. Setiap kali Anda memperbarui eksekusi pekerjaan, versinya bertambah. Jika versi eksekusi pekerjaan disimpan dalam AWS IoT Layanan pekerjaan tidak cocok, pembaruan ditolak dengan `VersionMismatch` kesalahan. Sebuah [ErrorResponse](#) yang berisi data status eksekusi pekerjaan saat ini juga dikembalikan. (Ini membuatnya tidak perlu untuk melakukan yang terpisah `DescribeJobExecution` permintaan untuk mendapatkan data status eksekusi pekerjaan.)

## executionNumber

(Opsional) Angka yang mengidentifikasi eksekusi pekerjaan di perangkat. Jika tidak ditentukan, eksekusi pekerjaan terbaru digunakan.

## includeJobExecutionState

(Opsional) Bila disertakan dan diatur ke `true`, respon berisi `JobExecutionState` bidang. Defaultnya adalah `false`.

## includeJobDocument

(Opsional) Bila disertakan dan diatur ke `true`, respon berisi `JobDocument`. Defaultnya adalah `false`.

## stepTimeoutInMinutes

Menentukan jumlah waktu perangkat ini harus menyelesaikan pelaksanaan pekerjaan ini. Jika status eksekusi pekerjaan tidak disetel ke status terminal sebelum timer ini berakhir, atau sebelum timer disetel ulang, status eksekusi pekerjaan diatur ke `TIMED_OUT`. Mengatur atau mengatur ulang batas waktu ini tidak berpengaruh pada batas waktu eksekusi pekerjaan yang mungkin telah ditentukan saat pekerjaan dibuat.

Broker pesan akan menerbitkan `$aws/things/thingName/jobs/jobId/update/accepted` dan `$aws/things/thingName/jobs/jobId/update/rejected` bahkan tanpa berlangganan khusus untuk mereka. Namun, bagi klien Anda untuk menerima pesan, itu harus mendengarkan mereka. Untuk informasi lebih lanjut, lihat [catatan tentang pesan Jobs API](#).

Muatan respons:

```
{
 "executionState": JobExecutionState,
 "jobDocument": "string",
 "timestamp": timestamp,
 "clientToken": "string"
}
```

`executionState`

Sebuah objek [JobExecutionState](#).

`jobDocument`

SEBUAH [dokumen pekerjaan](#) objek.

`timestamp`

Waktu dalam hitungan detik sejak zaman, ketika pesan dikirim.

`clientToken`

Token klien yang digunakan untuk mengkorelasikan permintaan dan tanggapan.

Ketika Anda menggunakan protokol MQTT, Anda juga dapat melakukan pemutakhiran berikut:

`JobExecutionsChanged`

Dikirim setiap kali eksekusi pekerjaan ditambahkan ke atau dihapus dari daftar eksekusi pekerjaan yang tertunda untuk suatu hal.

Gunakan topik:

`$aws/things/thingName/jobs/notify`

Muatan pesan:

```
{
 "jobs" : {
 "JobExecutionState": [JobExecutionSummary ...]
 },
 "timestamp": timestamp
}
```

## NextJobExecutionChanged

Dikirim setiap kali ada perubahan yang eksekusi pekerjaan berikutnya pada daftar eksekusi pekerjaan yang tertunda untuk suatu hal, sebagaimana didefinisikan untuk [DescribeJobExecution](#) bersama `jobId $next`. Pesan ini tidak dikirim ketika rincian eksekusi pekerjaan berikutnya berubah, hanya ketika pekerjaan berikutnya yang akan dikembalikan oleh [DescribeJobExecution](#) bersama `jobId $next` telah berubah. Pertimbangkan eksekusi pekerjaan J1 dan J2 dengan status `QUEUED`. J1 adalah berikutnya pada daftar eksekusi pekerjaan yang tertunda. Jika status J2 diubah menjadi `IN_PROGRESS` sementara keadaan J1 tetap tidak berubah, maka pemberitahuan ini dikirim dan berisi rincian J2.

Gunakan topik:

`$aws/things/thingName/jobs/notify-next`

Muatan pesan:

```
{
 "execution" : JobExecution,
 "timestamp": timestamp,
}
```

## Lowongan kerja perangkat HTTP API

Perangkat dapat berkomunikasi dengan AWS IoT Pekerjaan menggunakan HTTP Signature Versi 4 pada port 443. Ini adalah metode yang digunakan oleh AWS SDK dan CLI. Untuk informasi selengkapnya tentang alat tersebut, lihat [AWS CLI Referensi Perintah:iot-jobs-data](#) atau [AWS SDK dan Alat](#).

Perintah berikut tersedia untuk perangkat yang menjalankan pekerjaan. Untuk informasi tentang penggunaan operasi API dengan protokol MQTT, lihat [Pekerjaan perangkat operasi MQTT API](#).

## GetPendingJobExecutions

Mendapat daftar semua pekerjaan yang tidak dalam keadaan terminal, untuk hal tertentu.

### HTTPS request

```
GET /things/thingName/jobs
```

### Jawaban:

```
{
 "inProgressJobs" : [JobExecutionSummary ...],
 "queuedJobs" : [JobExecutionSummary ...]
}
```

Untuk informasi selengkapnya, lihat [GetPendingJobExecutions](#).

### CLI syntax

```
aws iot-jobs-data get-pending-job-executions \
--thing-name <value> \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

### cli-input-jsonFormat:

```
{
 "thingName": "string"
}
```

Untuk informasi selengkapnya, lihat [get-pending-job-executions](#).

## StartNextPendingJobExecution

Mendapat dan memulai eksekusi pekerjaan tertunda berikutnya untuk suatu hal (dengan statusIN\_PROGRESSatauQUEUED).

- Setiap eksekusi pekerjaan dengan statusIN\_PROGRESSdikembalikan terlebih dahulu.
- Eksekusi pekerjaan dikembalikan dalam urutan di mana mereka diciptakan.

- Jika eksekusi pekerjaan tertunda berikutnya adalah `QUEUED`, statusnya berubah menjadi `IN_PROGRESS` dan rincian status eksekusi pekerjaan ditetapkan seperti yang ditentukan.
- Jika eksekusi pekerjaan tertunda berikutnya sudah `IN_PROGRESS`, detail statusnya tidak berubah.
- Jika tidak ada eksekusi pekerjaan yang tertunda, respons tidak menyertakan `execution` bidang.
- Opsional, Anda dapat membuat timer langkah dengan menetapkan nilai untuk `stepTimeoutInMinutes` properti. Jika Anda tidak memperbarui nilai properti ini dengan menjalankan `UpdateJobExecution`, eksekusi pekerjaan kali keluar ketika timer langkah berakhir.

## HTTPS request

Contoh berikut menunjukkan sintaks permintaan:

```
PUT /things/thingName/jobs/$next
{
 "statusDetails": {
 "string": "string"
 ...
 },
 "stepTimeoutInMinutes": long
}
```

Untuk informasi selengkapnya, lihat [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` Format:

```
{
 "thingName": "string",
 "statusDetails": {
 "string": "string"
```



```
},
"stepTimeoutInMinutes": long
}
```

Untuk informasi selengkapnya, lihat [start-next-pending-job-execution](#).

## DescribeJobExecution

Mendapat informasi rinci tentang eksekusi pekerjaan.

Anda dapat mengatur `jobId` kepada `$next` untuk mengembalikan eksekusi pekerjaan tertunda berikutnya untuk sesuatu. Status eksekusi pekerjaan harus `QUEUED` atau `IN_PROGRESS`.

## HTTPS request

Permintaan:

```
GET /things/thingName/jobs/jobId?
executionNumber=executionNumber&includeJobDocument=includeJobDocument
```

Jawaban:

```
{
 "execution" : JobExecution,
}
```

Untuk informasi selengkapnya, lihat [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` Format:

```
{
 "jobId": "string",
 "thingName": "string",
 "includeJobDocument": boolean,
 "executionNumber": long
}
```

Untuk informasi selengkapnya, lihat [describe-job-execution](#).

## UpdateJobExecution

Memperbarui status eksekusi pekerjaan. Opsional, Anda dapat membuat timer langkah dengan menetapkan nilai untuk `stepTimeoutInMinutes` properti. Jika Anda tidak memperbarui nilai properti ini dengan menjalankan `UpdateJobExecution` sekali lagi, eksekusi pekerjaan kali keluar ketika timer langkah berakhir.

## HTTPS request

Permintaan:

```
POST /things/thingName/jobs/jobId
{
 "status": "job-execution-state",
 "statusDetails": {
 "string": "string"
 ...
 },
 "expectedVersion": "number",
 "includeJobExecutionState": boolean,
 "includeJobDocument": boolean,
 "stepTimeoutInMinutes": long,
 "executionNumber": long
}
```

Untuk informasi selengkapnya, lihat [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`Format:

```
{
 "jobId": "string",
 "thingName": "string",
 "status": "string",
 "statusDetails": {
 "string": "string"
 },
 "stepTimeoutInMinutes": number,
 "expectedVersion": long,
 "includeJobExecutionState": boolean,
 "includeJobDocument": boolean,
 "executionNumber": long
}
```

Untuk informasi selengkapnya, lihat [update-job-execution](#).

## Mengamankan pengguna dan perangkat dengan AWS IoT Jobs

Untuk mengizinkan pengguna menggunakan AWS IoT Lowongan dengan perangkat mereka, Anda harus memberi mereka izin dengan menggunakan kebijakan IAM. Perangkat kemudian harus diotorisasi dengan menggunakan AWS IoT Core kebijakan untuk terhubung dengan aman AWS IoT, menerima eksekusi pekerjaan, dan memperbarui status eksekusi.

### Jenis kebijakan yang diperlukan untuk AWS IoT Pekerjaan

Tabel berikut menunjukkan berbagai jenis kebijakan yang harus Anda gunakan untuk otorisasi. Untuk informasi selengkapnya tentang kebijakan yang diperlukan untuk digunakan, lihat [Otorisasi](#).

## Jenis kebijakan yang diperlukan

| Kasus penggunaan                                                                            | Protokol    | Autentikasi                                               | Bidang kontrol/bidang data          | Jenis identitas                                       | Jenis kebijakan yang diperlukan |
|---------------------------------------------------------------------------------------------|-------------|-----------------------------------------------------------|-------------------------------------|-------------------------------------------------------|---------------------------------|
| Otorisasi administrator, operator, atau Layanan Cloud untuk bekerja dengan aman dengan Jobs | HTTPS       | AWS Otentikasi Tanda Tangan Versi 4 (port 443)            | Baik bidang kontrol dan bidang data | Identitas Amazon Cognito, IAM, atau pengguna federasi | Kebijakan IAM                   |
| Otorisasi perangkat IoT Anda agar bekerja dengan aman dengan Jobs                           | MQTT/HTTP S | Otentikasi timbal balik TCP atau TLS (port 8883 atau 443) | Bidang data                         | Sertifikat X.509                                      | AWS IoT Core kebijakan          |

Untuk mengotorisasi operasi AWS IoT Pekerjaan yang dapat dilakukan baik pada bidang kontrol dan bidang data, Anda harus menggunakan kebijakan IAM. Identitas harus telah diautentikasi dengan AWS IoT untuk melakukan operasi ini, yang harus [Identitas Amazon Cognito](#) atau [Pengguna IAM, grup IAM, dan IAM role](#) Untuk informasi selengkapnya tentang otentikasi, lihat [Autentikasi](#).

Perangkat sekarang harus diotorisasi pada bidang data dengan menggunakan AWS IoT Core kebijakan untuk terhubung dengan aman ke gateway perangkat. Gateway perangkat memungkinkan perangkat untuk berkomunikasi dengan aman AWS IoT, menerima eksekusi pekerjaan, dan memperbarui status eksekusi pekerjaan. Komunikasi perangkat diamankan dengan menggunakan protokol aman [MQTT](#) atau [HTTPS](#) komunikasi. Protokol ini digunakan [Sertifikat klien X.509](#) yang disediakan oleh AWS IoT untuk mengautentikasi koneksi perangkat.

Berikut ini menunjukkan cara Anda memberi wewenang kepada pengguna, layanan keras, dan perangkat untuk menggunakan AWS IoT Jobs. Untuk informasi tentang bidang kontrol dan operasi API bidang data, lihat [AWS IoT pekerjaan API operations](#).

## Topik

- [Mengotorisasi pengguna dan layanan cloud untuk menggunakan AWS IoT Jobs](#)
- [Mengotorisasi perangkat Anda untuk menggunakan AWS IoT Pekerjaan dengan aman di bidang data](#)

## Mengotorisasi pengguna dan layanan cloud untuk menggunakan AWS IoT Jobs

Untuk mengotorisasi pengguna dan layanan cloud Anda, Anda harus menggunakan kebijakan IAM pada bidang kontrol dan bidang data. Kebijakan harus digunakan dengan protokol HTTPS dan harus menggunakan otentikasi AWS Signature Version 4 (port 443) untuk mengautentikasi pengguna.

### Note

AWS IoT Core Kebijakan tidak boleh digunakan pada pesawat kontrol. Hanya kebijakan IAM yang digunakan untuk mengotorisasi pengguna atau Layanan Cloud. Untuk informasi selengkapnya tentang menggunakan jenis kebijakan yang diperlukan, lihat [Jenis kebijakan yang diperlukan untuk AWS IoT Pekerjaan](#).

Kebijakan IAM adalah dokumen JSON yang berisi pernyataan kebijakan. Pernyataan kebijakan menggunakan elemen Efek, Tindakan, dan Sumber Daya untuk menentukan sumber daya, tindakan yang diizinkan atau ditolak, dan kondisi di mana tindakan diizinkan atau ditolak. Untuk informasi selengkapnya, lihat [Referensi Elemen Kebijakan IAM JSON](#) di Panduan pengguna IAM.

### Warning

Kami menyarankan agar Anda tidak menggunakan izin wildcard, seperti "Action": ["iot:\*"] dalam kebijakan atau kebijakan IAM Anda. AWS IoT Core Menggunakan izin wildcard bukanlah praktik terbaik keamanan yang disarankan. Untuk informasi selengkapnya, lihat [AWS IoT kebijakan yang terlalu permisif](#).

## Kebijakan IAM di bidang kontrol

Pada bidang kontrol, kebijakan IAM menggunakan `iot:` awalan dengan tindakan untuk mengotorisasi operasi API pekerjaan terkait. Misalnya, tindakan `iot:CreateJob` kebijakan memberikan izin kepada pengguna untuk menggunakan [CreateJob](#) API.

### Tindakan kebijakan

Tabel berikut menunjukkan daftar tindakan kebijakan IAM dan izin untuk menggunakan tindakan API. Untuk informasi tentang jenis sumber daya, lihat [Jenis sumber daya yang ditentukan oleh AWS IoT](#). Untuk informasi selengkapnya tentang AWS IoT tindakan, lihat [Tindakan yang ditentukan oleh AWS IoT](#).

### Tindakan kebijakan IAM pada control plane

| Tindakan kebijakan                       | Operasi API                             | Jenis sumber daya                                                                            | Deskripsi                                                                                                                                                                                |
|------------------------------------------|-----------------------------------------|----------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>iot:AssociateTargetsWithJob</code> | <a href="#">AssociateTargetsWithJob</a> | <ul style="list-style-type: none"> <li>pekerjaan</li> <li>hal</li> <li>thinggroup</li> </ul> | Merupakan izin untuk mengasosiasikan grup dengan pekerjaan berkelanjutan. <code>iot:AssociateTargetsWithJob</code> Izin diperiksa setiap kali permintaan dibuat untuk mengaitkan target. |
| <code>iot:CancelJob</code>               | <a href="#">CancelJob</a>               | pekerjaan                                                                                    | Merupakan izin untuk membatalkan pekerjaan. <code>iot:CancelJob</code> Izin diperiksa setiap kali permintaan dibuat untuk membatalkan pekerjaan.                                         |
| <code>iot:CancelJobExecution</code>      | <a href="#">CancelJobExecution</a>      | <ul style="list-style-type: none"> <li>pekerjaan</li> <li>hal</li> </ul>                     | Merupakan izin untuk membatalkan eksekusi pekerjaan. <code>iot:CancelJobExecution</code> Izin diperiksa setiap kali permintaan dibuat untuk membatalkan eksekusi pekerjaan.              |
| <code>iot:CreateJob</code>               | <a href="#">CreateJob</a>               | <ul style="list-style-type: none"> <li>pekerjaan</li> <li>hal</li> </ul>                     | Merupakan izin untuk membuat pekerjaan. <code>iot:CreateJob</code> Izin diperiksa setiap kali permintaan dibuat untuk membuat pekerjaan.                                                 |

| Tindakan kebijakan     | Operasi API                       | Jenis sumber daya                                                                                        | Deskripsi                                                                                                                                                   |
|------------------------|-----------------------------------|----------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                        |                                   | <ul style="list-style-type: none"> <li>• thinggroup</li> <li>• Jobtemplate</li> <li>• package</li> </ul> |                                                                                                                                                             |
| iot:CreateJobTemplate  | <a href="#">CreateJobTemplate</a> | <ul style="list-style-type: none"> <li>• pekerjaan</li> <li>• Jobtemplate</li> <li>• package</li> </ul>  | Merupakan izin untuk membuat template pekerjaan. iot: CreateJobTemplate Izin diperiksa setiap kali permintaan dibuat untuk membuat template pekerjaan.      |
| iot>DeleteJob          | <a href="#">DeleteJob</a>         | pekerjaan                                                                                                | Merupakan izin untuk menghapus pekerjaan. iot: DeleteJob Izin diperiksa setiap kali permintaan dibuat untuk menghapus pekerjaan.                            |
| iot>DeleteJobTemplate  | <a href="#">DeleteJobTemplate</a> | Jobtemplate                                                                                              | Merupakan izin untuk menghapus template pekerjaan. iot: CreateJobTemplate Izin diperiksa setiap kali permintaan dibuat untuk menghapus template pekerjaan.  |
| iot>DeleteJobExecution | <a href="#">DeleteJobTemplate</a> | <ul style="list-style-type: none"> <li>• pekerjaan</li> <li>• hal</li> </ul>                             | Merupakan izin untuk menghapus eksekusi pekerjaan. iot: DeleteJobExecution Izin diperiksa setiap kali permintaan dibuat untuk menghapus eksekusi pekerjaan. |
| iot:DescribeJob        | <a href="#">DescribeJob</a>       | pekerjaan                                                                                                | Merupakan izin untuk menggambarkan pekerjaan. iot: DescribeJob Izin diperiksa setiap kali permintaan dibuat untuk menggambarkan suatu pekerjaan.            |

| Tindakan kebijakan                          | Operasi API                                | Jenis sumber daya                                                        | Deskripsi                                                                                                                                                                                                                         |
|---------------------------------------------|--------------------------------------------|--------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>iot:DescribeJobExecution</code>       | <a href="#">DescribeJobExecution</a>       | <ul style="list-style-type: none"> <li>pekerjaan</li> <li>hal</li> </ul> | Merupakan izin untuk menggambarkan eksekusi pekerjaan. <code>iot: DescribeJobExecution</code> Izin diperiksa setiap kali permintaan dibuat untuk menggambarkan eksekusi pekerjaan.                                                |
| <code>iot:DescribeJobTemplate</code>        | <a href="#">DescribeJobTemplate</a>        | Jobtemplate                                                              | Merupakan izin untuk menggambarkan template pekerjaan. <code>iot: DescribeJobTemplate</code> Izin diperiksa setiap kali permintaan dibuat untuk menggambarkan template pekerjaan.                                                 |
| <code>iot:DescribeManagedJobTemplate</code> | <a href="#">DescribeManagedJobTemplate</a> | Jobtemplate                                                              | Merupakan izin untuk mendeskripsikan template pekerjaan terkelola. <code>iot: DescribeManagedJobTemplate</code> Izin diperiksa setiap kali permintaan dibuat untuk menggambarkan template pekerjaan yang dikelola.                |
| <code>iot:GetJobDocument</code>             | <a href="#">GetJobDocument</a>             | pekerjaan                                                                | Merupakan izin untuk mendapatkan dokumen pekerjaan untuk suatu pekerjaan. <code>iot: GetJobDocument</code> Izin diperiksa setiap kali permintaan dibuat untuk mendapatkan dokumen pekerjaan.                                      |
| <code>iot:ListJobExecutionsForJob</code>    | <a href="#">ListJobExecutionsForJob</a>    | pekerjaan                                                                | Merupakan izin untuk membuat daftar eksekusi pekerjaan untuk suatu pekerjaan. <code>iot: ListJobExecutionsForJob</code> Izin diperiksa setiap kali permintaan dibuat untuk mencantumkan eksekusi pekerjaan untuk suatu pekerjaan. |



| Tindakan kebijakan                         | Operasi API                               | Jenis sumber daya | Deskripsi                                                                                                                                                                                                                    |
|--------------------------------------------|-------------------------------------------|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>iot:ListJobExecutionsForThing</code> | <a href="#">ListJobExecutionsForThing</a> | hal               | Merupakan izin untuk membuat daftar eksekusi pekerjaan untuk suatu pekerjaan. <code>iot:ListJobExecutionsForThing</code> Izin diperiksa setiap kali permintaan dibuat untuk mencantumkan eksekusi pekerjaan untuk suatu hal. |
| <code>iot:ListJobs</code>                  | <a href="#">ListJobs</a>                  | none              | Merupakan izin untuk membuat daftar pekerjaan. <code>iot:ListJobs</code> Izin diperiksa setiap kali permintaan dibuat untuk membuat daftar pekerjaan.                                                                        |
| <code>iot:ListJobTemplates</code>          | <a href="#">ListJobTemplates</a>          | none              | Merupakan izin untuk membuat daftar template pekerjaan. <code>iot:ListJobTemplates</code> Izin diperiksa setiap kali permintaan dibuat untuk membuat daftar templat pekerjaan.                                               |
| <code>iot:ListManagedJobTemplates</code>   | <a href="#">ListManagedJobTemplates</a>   | none              | Merupakan izin untuk membuat daftar templat pekerjaan yang dikelola. <code>iot:ListManagedJobTemplates</code> Izin diperiksa setiap kali permintaan dibuat untuk mencantumkan templat pekerjaan yang dikelola.               |
| <code>iot:UpdateJob</code>                 | <a href="#">UpdateJob</a>                 | pekerjaan         | Merupakan izin untuk memperbarui pekerjaan. <code>iot:UpdateJob</code> Izin diperiksa setiap kali permintaan dibuat untuk memperbarui pekerjaan.                                                                             |

| Tindakan kebijakan | Operasi API                   | Jenis sumber daya                                                                                   | Deskripsi                                                 |
|--------------------|-------------------------------|-----------------------------------------------------------------------------------------------------|-----------------------------------------------------------|
| iot:TagResource    | <a href="#">TagResource</a>   | <ul style="list-style-type: none"> <li>• pekerjaan</li> <li>• Jobtemplate</li> <li>• hal</li> </ul> | Memberikan izin untuk menandai sumber daya tertentu.      |
| iot:UntagResource  | <a href="#">UntagResource</a> | <ul style="list-style-type: none"> <li>• pekerjaan</li> <li>• Jobtemplate</li> <li>• hal</li> </ul> | Memberikan izin untuk menghapus tag sumber daya tertentu. |

### Contoh kebijakan IAM dasar

Contoh berikut menunjukkan kebijakan IAM yang memungkinkan izin pengguna untuk melakukan tindakan berikut untuk hal IoT Anda dan grup benda.

Dalam contoh, ganti:

- *daerah* dengan Anda Wilayah AWS, seperti `us-east-1`.
- *account-id* dengan Akun AWS nomor Anda, seperti `57EXAMPLE833`
- *thing-group-name* dengan nama grup IoT Anda yang Anda targetkan pekerjaan, seperti `FirmwareUpdateGroup`
- *thing-name* dengan nama hal IoT Anda yang Anda targetkan pekerjaan, seperti `MyIoTThing`

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Action": [
 "iot:CreateJobTemplate",
 "iot:CreateJob",
],
 "Effect": "Allow",
 "Resource": "arn:aws:iot:region:account-id:thinggroup/thing-group-name"
 }
]
}
```

```

 },
 {
 "Action": [
 "iot:DescribeJob",
 "iot:CancelJob",
 "iot>DeleteJob",
],
 "Effect": "Allow",
 "Resource": "arn:aws:iot:region:account-id:job/*"
 },
 {
 "Action": [
 "iot:DescribeJobExecution",
 "iot:CancelJobExecution",
 "iot>DeleteJobExecution",
],
 "Effect": "Allow",
 "Resource": [
 "arn:aws:iot:region:account-id:thing/thing-name",
 "arn:aws:iot:region:account-id:job/*"
]
 }
]
}

```

## Contoh kebijakan IAM untuk otorisasi berbasis IP

Anda dapat membatasi prinsipal agar tidak melakukan panggilan API ke titik akhir bidang kontrol Anda dari alamat IP tertentu. Untuk menentukan alamat IP yang dapat diizinkan, dalam elemen Kondisi kebijakan IAM Anda, gunakan kunci kondisi [aws:SourceIp](#)global.

Menggunakan kunci kondisi ini juga dapat menolak akses ke Layanan AWS s lain dari melakukan panggilan API ini atas nama Anda, seperti AWS CloudFormation. Untuk mengizinkan akses ke layanan ini, gunakan kunci kondisi [aws:ViaAWSService](#)global dengan SourceIp kunci aws:. Ini memastikan bahwa pembatasan akses alamat IP sumber hanya berlaku untuk permintaan yang dibuat langsung oleh prinsipal. Untuk informasi selengkapnya, lihat [AWS: Menolak akses AWS berdasarkan IP sumber](#).

Contoh berikut menunjukkan cara mengizinkan hanya alamat IP tertentu yang dapat membuat panggilan API ke titik akhir bidang kontrol. `aws:ViaAWSService`Kuncinya disetel ke `true`, yang memungkinkan layanan lain melakukan panggilan API atas nama Anda.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "iot:CreateJobTemplate",
 "iot:CreateJob"
],
 "Resource": ["*"],
 "Condition": {
 "IpAddress": {
 "aws:SourceIp": "123.45.167.89"
 }
 },
 "Bool": {"aws:ViaAWSService": "true"}
 }
],
}
```

## Kebijakan IAM tentang pesawat data

Kebijakan IAM pada bidang data menggunakan `iotjobsdata:` awalan untuk mengotorisasi operasi API lowongan yang dapat dilakukan pengguna. Pada bidang data, Anda dapat memberikan izin kepada pengguna untuk menggunakan [DescribeJobExecution](#) API menggunakan tindakan `iotjobsdata:DescribeJobExecution` kebijakan.

### Warning

Menggunakan kebijakan IAM di bidang data tidak disarankan saat menargetkan AWS IoT Lowongan untuk perangkat Anda. Kami menyarankan Anda menggunakan kebijakan IAM pada bidang kontrol bagi pengguna untuk membuat dan mengelola pekerjaan. Pada bidang data, untuk mengotorisasi perangkat untuk mengambil eksekusi pekerjaan dan memperbarui status eksekusi, gunakan. [AWS IoT Core kebijakan untuk protokol HTTPS](#)

## Contoh kebijakan IAM dasar

Operasi API yang harus diotorisasi biasanya dilakukan oleh Anda mengetik perintah CLI. Berikut ini menunjukkan contoh pengguna yang melakukan `DescribeJobExecution` operasi.

Dalam contoh, ganti:

- *daerah* dengan Anda Wilayah AWS, seperti `us-east-1`.
- *account-id* dengan Akun AWS nomor Anda, seperti `57EXAMPLE833`
- *thing-name* dengan nama hal IoT Anda yang Anda targetkan pekerjaan, seperti `myRegisteredThing`
- *job-id* adalah pengidentifikasi unik untuk pekerjaan yang ditargetkan menggunakan 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
```

Berikut ini menunjukkan contoh kebijakan IAM yang mengizinkan tindakan ini:

```
{
 "Version": "2012-10-17",
 "Statement":
 {
 "Action": ["iotjobsdata:DescribeJobExecution"],
 "Effect": "Allow",
 "Resource": "arn:aws:iot:region:account-id:thing/thing-name",
 }
}
```

Contoh kebijakan IAM untuk otorisasi berbasis IP

Anda dapat membatasi prinsipal agar tidak melakukan panggilan API ke titik akhir bidang data Anda dari alamat IP tertentu. Untuk menentukan alamat IP yang dapat diizinkan, dalam elemen Kondisi kebijakan IAM Anda, gunakan kunci kondisi [aws:SourceIp](#) global.

Menggunakan kunci kondisi ini juga dapat menolak akses ke Layanan AWS s lain dari melakukan panggilan API ini atas nama Anda, seperti AWS CloudFormation. Untuk mengizinkan akses ke layanan ini, gunakan kunci kondisi [aws:ViaAWSService](#) global dengan kunci `aws:SourceIp` kondisi. Hal ini memastikan bahwa pembatasan akses alamat IP hanya berlaku untuk permintaan yang langsung dibuat oleh prinsipal. Untuk informasi selengkapnya, lihat [AWS: Menolak akses AWS berdasarkan IP sumber](#).

Contoh berikut menunjukkan cara mengizinkan hanya alamat IP tertentu yang dapat membuat panggilan API ke titik akhir bidang data.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": ["iotjobsdata:*"],
 "Resource": ["*"],
 "Condition": {
 "IpAddress": {
 "aws:SourceIp": "123.45.167.89"
 }
 },
 "Bool": {"aws:ViaAWSService": "false"}
 }
],
}
```

Contoh berikut menunjukkan cara membatasi alamat IP atau rentang alamat tertentu dari membuat panggilan API ke titik akhir bidang data.

```
{
 "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": ["*"],
 }
],
}
```

## Contoh kebijakan IAM untuk bidang kontrol dan bidang data

Jika Anda melakukan operasi API pada bidang kontrol dan bidang data, tindakan kebijakan bidang kontrol Anda harus menggunakan `iot: awalan`, dan tindakan kebijakan bidang data Anda harus menggunakan `iotjobsdata: awalan`.

Misalnya, `DescribeJobExecution` API dapat digunakan di bidang kontrol dan bidang data. Pada bidang kontrol, [DescribeJobExecution](#) API digunakan untuk menggambarkan eksekusi pekerjaan. Pada bidang data, [DescribeJobExecution](#) API digunakan untuk mendapatkan detail eksekusi pekerjaan.

Kebijakan IAM berikut mengotorisasi izin pengguna untuk menggunakan `DescribeJobExecution` API pada bidang kontrol dan bidang data.

Dalam contoh, ganti:

- *daerah* dengan Anda Wilayah AWS, seperti `us-east-1`.
- *account-id* dengan Akun AWS nomor Anda, seperti `57EXAMPLE833`
- *thing-name* dengan nama hal IoT Anda yang Anda targetkan pekerjaan, seperti `MyIoTThing`

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Action": ["iotjobsdata:DescribeJobExecution"],
 "Effect": "Allow",
 "Resource": "arn:aws:iot:region:account-id:thing/thing-name"
 },
 {
 "Action": [
 "iot:DescribeJobExecution",
 "iot:CancelJobExecution",
 "iot>DeleteJobExecution",
],
 "Effect": "Allow",
 "Resource": [
 "arn:aws:iot:region:account-id:thing/thing-name"
 "arn:aws:iot:region:account-id:job/*"
]
 }
]
}
```

```
}
```

## Otorisasi penandaan sumber daya IoT

Untuk kontrol yang lebih baik atas pekerjaan dan templat pekerjaan yang dapat Anda buat, modifikasi, atau gunakan, Anda dapat melampirkan tag ke pekerjaan atau templat pekerjaan. Tag juga membantu Anda membedakan kepemilikan dan menetapkan serta mengalokasikan biaya dengan menempatkannya dalam grup penagihan dan melampirkan tag pada mereka.

Ketika pengguna ingin menandai pekerjaan atau templat pekerjaan mereka yang mereka buat dengan menggunakan AWS Management Console atau AWS CLI, kebijakan IAM Anda harus memberikan izin kepada pengguna untuk menandai mereka. Untuk memberikan izin, kebijakan IAM Anda harus menggunakan tindakan tersebut. `iot:TagResource`

### Note

Jika kebijakan IAM Anda tidak menyertakan `iot:TagResource` tindakan, maka salah satu [CreateJob](#) atau [CreateJobTemplate](#) dengan tag akan menampilkan `AccessDeniedException` kesalahan.

Ketika Anda ingin menandai pekerjaan atau templat pekerjaan yang Anda buat dengan menggunakan AWS Management Console atau AWS CLI, kebijakan IAM Anda harus memberikan izin untuk menandai mereka. Untuk memberikan izin, kebijakan IAM Anda harus menggunakan tindakan tersebut. `iot:TagResource`

Untuk informasi umum tentang menandai sumber daya Anda, lihat [Menandai sumber daya Anda AWS IoT](#).

### Contoh kebijakan IAM

Lihat contoh kebijakan IAM berikut yang memberikan izin penandaan:

#### Contoh 1

Pengguna yang menjalankan perintah berikut untuk membuat pekerjaan dan menandainya ke lingkungan tertentu.

Dalam contoh ini, ganti:



- *daerah* dengan Anda Wilayah AWS, seperti `us-east-1`.
- *account-id* dengan Akun AWS nomor Anda, seperti `57EXAMPLE833`
- *thing-name* dengan nama hal IoT Anda yang Anda targetkan pekerjaan, seperti `MyIoTThing`

```
aws iot create-job
 --job-id test_job
 --targets "arn:aws:iot:region:account-id:thing/thingOne"
 --document-source "https://s3.amazonaws.com/my-s3-bucket/job-document.json"
 --description "test job description"
 --tags Key=environment,Value=beta
```

Untuk contoh ini, Anda harus menggunakan kebijakan IAM berikut:

```
{
 "Version": "2012-10-17",
 "Statement":
 {
 "Action": ["iot:CreateJob", "iot:CreateJobTemplate", "iot:TagResource"],
 "Effect": "Allow",
 "Resource": [
 "arn:aws:iot:aws-region:account-id:job/*",
 "arn:aws:iot:aws-region:account-id:jobtemplate/*"
]
 }
}
```

## Mengotorisasi perangkat Anda untuk menggunakan AWS IoT Pekerjaan dengan aman di bidang data

Untuk mengizinkan perangkat Anda berinteraksi secara aman dengan AWS IoT Jobs di bidang data, Anda harus menggunakan AWS IoT Core kebijakan. AWS IoT Core kebijakan untuk lowongan kerja adalah dokumen JSON yang berisi pernyataan kebijakan. Kebijakan ini juga menggunakan elemen Efek, Tindakan, dan Sumber Daya, dan mengikuti konvensi serupa dengan kebijakan IAM. Untuk informasi selengkapnya tentang elemen, lihat [Referensi Elemen Kebijakan IAM JSON](#) di Panduan pengguna IAM.

Kebijakan dapat digunakan dengan protokol MQTT dan HTTPS dan harus menggunakan otentikasi timbal balik TCP atau TLS untuk mengautentikasi perangkat. Berikut ini menunjukkan cara menggunakan kebijakan ini di berbagai protokol komunikasi.

**⚠ Warning**

Kami menyarankan agar Anda tidak menggunakan izin wildcard, seperti "Action": ["iot:\*"] dalam kebijakan atau kebijakan IAM Anda. AWS IoT Core Menggunakan izin wildcard bukanlah praktik terbaik keamanan yang disarankan. Untuk informasi selengkapnya, lihat [AWS IoT kebijakan yang terlalu permisif](#).

## AWS IoT Core kebijakan untuk protokol MQTT

AWS IoT Core kebijakan untuk protokol MQTT memberi Anda izin untuk menggunakan tindakan API MQTT perangkat pekerjaan. Operasi MQTT API digunakan untuk bekerja dengan topik MQTT yang dicadangkan untuk perintah pekerjaan. Untuk informasi selengkapnya tentang operasi API ini, lihat [Pekerjaan perangkat operasi MQTT API](#).

Kebijakan MQTT menggunakan tindakan kebijakan seperti `iot:Connect`, `iot:Publish`, `iot:Subscribe`, dan `iot:Receive` untuk bekerja dengan topik pekerjaan. Kebijakan ini memungkinkan Anda untuk terhubung ke broker pesan, berlangganan topik MQTT pekerjaan, dan mengirim dan menerima pesan MQTT antara perangkat Anda dan cloud. Untuk informasi lebih lanjut tentang tindakan ini, lihat [AWS IoT Core tindakan kebijakan](#).

Untuk informasi tentang topik untuk AWS IoT Pekerjaan, lihat [Topik Job](#).

### Contoh kebijakan MQTT dasar

Contoh berikut menunjukkan bagaimana Anda dapat menggunakan `iot:Publish` dan `iot:Subscribe` mempublikasikan dan berlangganan pekerjaan dan eksekusi pekerjaan.

Dalam contoh, ganti:

- *daerah* dengan Anda Wilayah AWS, seperti `us-east-1`.
- *account-id* dengan Akun AWS nomor Anda, seperti `57EXAMPLE833`
- *thing-name* dengan nama hal IoT Anda yang Anda targetkan pekerjaan, seperti `MyIoTThing`

```
{
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
```

```
 "iot:Publish",
 "iot:Subscribe"
],
 "Resource": [
 "arn:aws:iot:region:account-id:topic/$aws/events/job/*",
 "arn:aws:iot:region:account-id:topic/$aws/events/jobExecution/*",
 "arn:aws:iot:region:account-id:topic/$aws/things/thing-name/jobs/*"
]
}
],
"Version": "2012-10-17"
}
```

## AWS IoT Core kebijakan untuk protokol HTTPS

AWS IoT Core kebijakan pada bidang data juga dapat menggunakan protokol HTTPS dengan mekanisme otentikasi TLS untuk mengotorisasi perangkat Anda. Pada bidang data, kebijakan menggunakan `iotjobsdata:` awalan untuk mengotorisasi operasi API lowongan yang dapat dilakukan perangkat Anda. Misalnya, tindakan `iotjobsdata:DescribeJobExecution` kebijakan memberikan izin kepada pengguna untuk menggunakan [DescribeJobExecution](#) API.

### Note

Tindakan kebijakan bidang data harus menggunakan `iotjobsdata:` awalan. Pada bidang kontrol, tindakan harus menggunakan `iot:` awalan. Untuk contoh kebijakan IAM saat tindakan kebijakan bidang kontrol dan bidang data digunakan, lihat [Contoh kebijakan IAM untuk bidang kontrol dan bidang data](#).

## Tindakan kebijakan

Tabel berikut menunjukkan daftar tindakan AWS IoT Core kebijakan dan izin untuk mengotorisasi perangkat agar menggunakan tindakan API. Untuk daftar operasi API yang dapat Anda lakukan di bidang data, lihat [Lowongan kerja perangkat HTTP API](#).

### Note

Tindakan kebijakan eksekusi pekerjaan ini hanya berlaku untuk titik akhir HTTP TLS. Jika Anda menggunakan titik akhir MQTT, Anda harus menggunakan tindakan kebijakan MQTT yang ditentukan sebelumnya.

## AWS IoT Core tindakan kebijakan pada pesawat data

| Tindakan kebijakan                       | Operasi API                                  | Jenis sumber daya                                                        | Deskripsi                                                                                                                                                                                                                                                                                                                              |
|------------------------------------------|----------------------------------------------|--------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| iotjobsdata:DescribeJobExecution         | <a href="#">DescribeJobExecution</a>         | <ul style="list-style-type: none"> <li>pekerjaan</li> <li>hal</li> </ul> | Merupakan izin untuk mengambil eksekusi pekerjaan. <code>iotjobsdata:DescribeJobExecution</code> Izin diperiksa setiap kali permintaan dibuat untuk mengambil eksekusi pekerjaan.                                                                                                                                                      |
| iotjobsdata:GetPendingJobExecutions      | <a href="#">GetPendingJobExecutions</a>      | hal                                                                      | Merupakan izin untuk mengambil daftar pekerjaan yang tidak dalam status terminal untuk suatu hal. <code>iotjobsdata:GetPendingJobExecutions</code> Izin diperiksa setiap kali permintaan dibuat untuk mengambil daftar.                                                                                                                |
| iotjobsdata:StartNextPendingJobExecution | <a href="#">StartNextPendingJobExecution</a> | hal                                                                      | Merupakan izin untuk mendapatkan dan memulai eksekusi pekerjaan tertunda berikutnya untuk suatu hal. Yaitu, untuk memperbarui eksekusi pekerjaan dengan status QUEUED ke IN_PROGRESS. <code>iot:StartNextPendingJobExecution</code> Izin diperiksa setiap kali permintaan dibuat untuk memulai eksekusi pekerjaan tertunda berikutnya. |
| iotjobsdata:UpdateJobExecution           | <a href="#">UpdateJobExecution</a>           | hal                                                                      | Merupakan izin untuk memperbarui eksekusi pekerjaan. <code>iot:UpdateJobExecution</code> Izin diperiksa setiap kali permintaan dibuat untuk memperbarui status eksekusi pekerjaan.                                                                                                                                                     |

## Contoh kebijakan dasar

Berikut ini menunjukkan contoh AWS IoT Core kebijakan yang memberikan izin untuk melakukan tindakan pada operasi API bidang data untuk sumber daya apa pun. Anda dapat membuat cakupan kebijakan Anda ke sumber daya tertentu, seperti hal IoT. Dalam contoh Anda, ganti:

- *daerah* dengan Wilayah AWS seperti `Andaus-east-1`.
- *account-id* dengan Akun AWS nomor Anda, seperti. `57EXAMPLE833`
- *thing-name* dengan nama hal IoT, seperti. `MyIoTthing`

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Action": [
 "iotjobsdata:GetPendingJobExecutions",
 "iotjobsdata:StartNextPendingJobExecution",
 "iotjobsdata:DescribeJobExecution",
 "iotjobsdata:UpdateJobExecution"
],
 "Effect": "Allow",
 "Resource": "arn:aws:iot:region:account-id:thing/thing-name"
 }
]
}
```

Contoh kapan Anda harus menggunakan kebijakan ini adalah ketika perangkat IoT Anda menggunakan AWS IoT Core kebijakan untuk mengakses salah satu operasi API ini, seperti contoh API berikut: `DescribeJobExecution`

```
GET /things/thingName/jobs/jobId?
executionNumber=executionNumber&includeJobDocument=includeJobDocument&namespaceId=namespaceId
HTTP/1.1
```

## Batas Job

AWS IoT Jobs memiliki kuota Layanan, atau batasan, yang sesuai dengan jumlah maksimum sumber daya layanan atau operasi untuk Anda Akun AWS.

## Batas pekerjaan aktif dan bersamaan

Bagian ini akan membantu Anda mempelajari lebih lanjut tentang pekerjaan aktif dan bersamaan serta batasan yang berlaku untuk mereka.

### Pekerjaan aktif dan batas pekerjaan aktif

Saat Anda membuat pekerjaan menggunakan AWS IoT konsol atau `CreateJob` API, status pekerjaan akan berubah menjadi `IN_PROGRESS`. Semua pekerjaan yang sedang berlangsung adalah pekerjaan aktif dan diperhitungkan dalam batas pekerjaan aktif. Ini termasuk pekerjaan yang meluncurkan eksekusi pekerjaan baru, atau pekerjaan yang menunggu perangkat untuk menyelesaikan eksekusi pekerjaan mereka. Batas ini berlaku untuk pekerjaan berkelanjutan dan snapshot.

### Pekerjaan bersamaan dan batas konkurensi pekerjaan

Pekerjaan dalam proses yang meluncurkan eksekusi pekerjaan baru, atau pekerjaan yang membatalkan eksekusi pekerjaan yang dibuat sebelumnya adalah pekerjaan bersamaan dan diperhitungkan dalam batas konkurensi pekerjaan. AWS IoT Jobs dapat meluncurkan dan membatalkan eksekusi pekerjaan dengan cepat dengan kecepatan 1000 perangkat per menit. Setiap pekerjaan adalah `concurrent` dan diperhitungkan dalam batas konkurensi pekerjaan hanya untuk waktu yang singkat. Setelah eksekusi pekerjaan diluncurkan atau dibatalkan, pekerjaan tidak lagi bersamaan dan tidak diperhitungkan dalam batas konkurensi pekerjaan. Anda dapat menggunakan konkurensi pekerjaan untuk membuat sejumlah besar pekerjaan sambil menunggu perangkat menyelesaikan eksekusi pekerjaan.

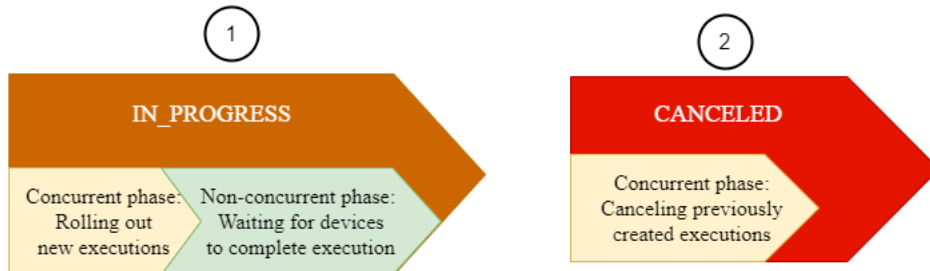
#### Note

Jika pekerjaan dengan konfigurasi penjadwalan opsional dan peluncuran dokumen pekerjaan yang dijadwalkan berlangsung selama jendela pemeliharaan mencapai yang dipilih `startTime` dan Anda berada pada batas konkurensi pekerjaan maksimum Anda, maka pekerjaan terjadwal tersebut akan pindah ke status `CANCELED`.

Untuk menentukan apakah suatu pekerjaan bersamaan, Anda dapat menggunakan `IsConcurrent` properti pekerjaan dari AWS IoT konsol, atau dengan menggunakan `ListJob API DescribeJob` atau. Batas ini berlaku untuk pekerjaan berkelanjutan dan snapshot.

Untuk melihat batasan konkurensi pekerjaan dan lowongan kerja aktif serta kuota AWS IoT Pekerjaan lainnya untuk Anda Akun AWS dan untuk meminta peningkatan batas, lihat [titik akhir Manajemen AWS IoT Perangkat dan kuota](#) di. Referensi Umum AWS

Diagram berikut menunjukkan bagaimana konkurensi pekerjaan berlaku untuk pekerjaan yang sedang berlangsung dan pekerjaan yang dibatalkan.



### Note

Pekerjaan baru dengan opsional `SchedulingConfig` akan mempertahankan status awal `SCHEDULED` dan diperbarui `IN_PROGRESS` setelah mencapai yang dipilih `startTime`. Setelah pekerjaan baru dengan opsional `SchedulingConfig` mencapai yang dipilih `startTime` dan diperbarui `IN_PROGRESS`, itu akan dihitung terhadap batas pekerjaan aktif dan batas konkurensi pekerjaan. Pekerjaan dengan status `SCHEDULED` akan dihitung terhadap batas pekerjaan aktif, tetapi tidak akan dihitung terhadap batas konkurensi pekerjaan.

Tabel berikut menunjukkan batasan yang berlaku untuk pekerjaan aktif dan bersamaan dan fase bersamaan dan tidak bersamaan dari status pekerjaan.

### Batas pekerjaan aktif dan bersamaan

| Status pekerjaan | Fase                                                                                                                                       | Batas pekerjaan aktif | Batas konkurensi Job |
|------------------|--------------------------------------------------------------------------------------------------------------------------------------------|-----------------------|----------------------|
| SCHEDULED        | Fase tidak bersamaan: AWS IoT Pekerjaan menunggu jadwal <code>startTime</code> pekerjaan untuk memulai pemberitahuan pelaksanaan pekerjaan | Berlaku               | Tidak berlaku        |

| Status pekerjaan | Fase                                                                                                                                                                                                                                                                                                                                                                           | Batas pekerjaan aktif | Batas konkurensi Job |
|------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------|----------------------|
|                  | ke perangkat Anda. Pekerjaan dalam fase ini hanya dihitung terhadap batas pekerjaan aktif dan <code>IsConcurrent</code> properti akan disetel ke <code>false</code> .                                                                                                                                                                                                          |                       |                      |
| IN_PROGRES<br>S  | Fase bersamaan: AWS IoT Pekerjaan menerima permintaan untuk membuat pekerjaan dan mulai meluncurkan pemberitahuan eksekusi pekerjaan ke perangkat Anda. Pekerjaan dalam fase ini bersamaan, sebagaimana dilambangkan dengan <code>IsConcurrent</code> properti yang disetel ke <code>true</code> , dan diperhitungkan terhadap pekerjaan aktif dan batas konkurensi pekerjaan. | Berlaku               | Berlaku              |
|                  | Fase tidak bersamaan: AWS IoT Pekerjaan menunggu perangkat melaporkan hasil eksekusi pekerjaan mereka. Pekerjaan dalam fase ini hanya dihitung terhadap batas pekerjaan aktif dan <code>IsConcurrent</code> properti akan disetel ke <code>false</code> .                                                                                                                      | Berlaku               | Tidak berlaku        |



| Status pekerjaan | Fase                                                                                                                                                                                                                                                                                                                                                                                                                                 | Batas pekerjaan aktif | Batas konkurensi Job |
|------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------|----------------------|
| Cancelled        | Fase bersamaan: AWS IoT Pekerjaan menerima permintaan untuk membatalkan pekerjaan dan mulai membatalkan eksekusi pekerjaan yang sebelumnya dibuat untuk perangkat Anda. Pekerjaan dalam fase ini bersamaan dan <code>IsConcurrent</code> properti akan disetel ke <code>true</code> . Setelah eksekusi pekerjaan dan pekerjaan dibatalkan, pekerjaan tidak lagi bersamaan dan tidak diperhitungkan dalam batas konkurensi pekerjaan. | Tidak berlaku         | Berlaku              |

**Note**

Durasi maksimal jendela pemeliharaan berulang adalah 23 jam, 50 menit.

# AWS IoT terowongan aman

Saat perangkat digunakan di balik firewall terbatas di situs jarak jauh, Anda memerlukan cara untuk mendapatkan akses ke perangkat tersebut untuk pemecahan masalah, pembaruan konfigurasi, dan tugas operasional lainnya. Gunakan tunneling aman untuk membangun komunikasi dua arah ke perangkat jarak jauh melalui koneksi aman yang dikelola oleh AWS IoT Tunneling aman tidak memerlukan pembaruan pada aturan firewall masuk yang ada, sehingga Anda dapat menjaga tingkat keamanan yang sama yang disediakan oleh aturan firewall di situs jarak jauh.

Misalnya, perangkat sensor yang terletak di pabrik yang berjarak beberapa ratus mil jauhnya mengalami kesulitan mengukur suhu pabrik. Anda dapat menggunakan tunneling aman untuk membuka dan memulai sesi dengan cepat ke perangkat sensor itu. Setelah Anda mengidentifikasi masalah (misalnya, file konfigurasi yang buruk), Anda dapat mengatur ulang file dan memulai ulang perangkat sensor melalui sesi yang sama. Dibandingkan dengan pemecahan masalah yang lebih tradisional (misalnya, mengirim teknisi ke pabrik untuk menyelidiki perangkat sensor), tunneling aman mengurangi respons insiden dan waktu pemulihan serta biaya operasional.

## Apa itu terowongan aman?

Gunakan tunneling aman untuk mengakses perangkat yang digunakan di belakang firewall yang dibatasi port di lokasi terpencil. Anda dapat terhubung ke perangkat tujuan dari laptop atau komputer desktop Anda sebagai perangkat sumber dengan menggunakan AWS Cloud. Sumber dan tujuan berkomunikasi dengan menggunakan proxy lokal open source yang berjalan di setiap perangkat. Proxy lokal berkomunikasi dengan menggunakan port terbuka yang diizinkan oleh firewall, biasanya 443. AWS Cloud Data yang ditransmisikan melalui terowongan dienkripsi menggunakan Transported Layer Security (TLS).

### Topik

- [Konsep terowongan yang aman](#)
- [Cara kerja tunneling yang aman](#)
- [Siklus hidup terowongan yang aman](#)

## Konsep terowongan yang aman

Istilah-istilah berikut digunakan oleh tunneling aman saat membangun komunikasi dengan perangkat jarak jauh. Untuk informasi tentang cara kerja tunneling aman, lihat [Cara kerja tunneling yang aman](#)

## Token akses klien (CAT)

Sepasang token yang dihasilkan oleh terowongan aman saat terowongan baru dibuat. CAT digunakan oleh perangkat sumber dan tujuan untuk terhubung ke layanan tunneling yang aman. CAT hanya dapat digunakan sekali untuk terhubung ke terowongan. Untuk menyambung kembali ke terowongan, putar token akses klien menggunakan operasi [RotateTunnelAccessTokenAPI](#) atau perintah [rotate-tunnel-access-tokenCLI](#).

## Token klien

Nilai unik yang dihasilkan oleh klien yang dapat digunakan tunneling AWS IoT aman untuk semua koneksi coba lagi berikutnya ke terowongan yang sama. Bidang ini bersifat opsional. Jika token klien tidak disediakan, maka token akses klien (CAT) hanya dapat digunakan sekali untuk terowongan yang sama. Upaya koneksi selanjutnya menggunakan CAT yang sama akan ditolak. Untuk informasi selengkapnya tentang penggunaan token klien, lihat [implementasi referensi proxy lokal di GitHub](#).

## Aplikasi tujuan

Aplikasi yang berjalan pada perangkat tujuan. Misalnya, aplikasi tujuan dapat menjadi daemon SSH untuk membuat sesi SSH menggunakan tunneling aman.

## Perangkat tujuan

Perangkat jarak jauh yang ingin Anda akses.

## Agen perangkat

Aplikasi IoT yang terhubung ke gateway AWS IoT perangkat dan mendengarkan notifikasi terowongan baru melalui MQTT. Untuk informasi selengkapnya, lihat [Cuplikan agen IoT](#).

## Proksi lokal

Proxy perangkat lunak yang berjalan pada perangkat sumber dan tujuan dan menyampaikan aliran data antara tunneling aman dan aplikasi perangkat. Proxy lokal dapat dijalankan dalam mode sumber atau mode tujuan. Untuk informasi selengkapnya, lihat [Proksi lokal](#).

## Perangkat sumber

Perangkat yang digunakan operator untuk memulai sesi ke perangkat tujuan, biasanya laptop atau komputer desktop.

## Terowongan

Jalur logis melalui AWS IoT itu memungkinkan komunikasi dua arah antara perangkat sumber dan perangkat tujuan.

## Cara kerja tunneling yang aman

Berikut ini menunjukkan bagaimana tunneling aman membuat koneksi antara perangkat sumber dan tujuan Anda. Untuk informasi tentang istilah yang berbeda seperti token akses klien (CAT), lihat [Konsep terowongan yang aman](#).

### 1. Buka terowongan

[Untuk membuka terowongan untuk memulai sesi dengan perangkat tujuan jarak jauh, Anda dapat menggunakan perintah AWS Management Console, AWS CLI terowongan terbuka, atau API. OpenTunnel](#)

### 2. Unduh pasangan token akses klien

Setelah membuka terowongan, Anda dapat mengunduh token akses klien (CAT) untuk sumber dan tujuan Anda dan menyimpannya di perangkat sumber Anda. Anda harus mengambil CAT dan menyimpannya sekarang sebelum memulai proxy lokal.

### 3. Mulai proxy lokal dalam mode tujuan

Agan IoT yang telah diinstal dan berjalan di perangkat tujuan Anda akan berlangganan topik `$aws/things/thing-name/tunnels/notify` MQTT yang dipesan dan akan menerima CAT. Di sini, *thing-name* adalah nama AWS IoT benda yang Anda buat untuk tujuan Anda. Untuk informasi selengkapnya, lihat [Topik tunneling yang aman](#).

Agan IoT kemudian menggunakan CAT untuk memulai proxy lokal dalam mode tujuan dan mengatur koneksi di sisi tujuan terowongan. Untuk informasi selengkapnya, lihat [Cuplikan agen IoT](#).

### 4. Mulai proxy lokal dalam mode sumber

Setelah terowongan dibuka, AWS IoT Device Management sediakan CAT untuk sumber yang dapat Anda unduh di perangkat sumber. Anda dapat menggunakan CAT untuk memulai proxy lokal dalam mode sumber, yang kemudian menghubungkan sisi sumber terowongan. Untuk informasi selengkapnya tentang proxy lokal, lihat [Proksi lokal](#).

### 5. Buka sesi SSH

Karena kedua sisi terowongan terhubung, Anda dapat memulai sesi SSH dengan menggunakan proxy lokal di sisi sumber.

Untuk informasi lebih lanjut tentang cara menggunakan AWS Management Console untuk membuka terowongan dan memulai sesi SSH, lihat [Buka terowongan dan mulai sesi SSH ke perangkat jarak jauh](#).

Video berikut menjelaskan cara kerja tunneling aman dan memandu Anda melalui proses pengaturan sesi SSH ke perangkat Raspberry Pi.

## Siklus hidup terowongan yang aman

Terowongan dapat memiliki status OPEN atau CLOSED. Koneksi ke terowongan dapat memiliki status CONNECTED atau DISCONNECTED. Berikut ini menunjukkan bagaimana status terowongan dan koneksi yang berbeda bekerja.

1. Ketika Anda membuka terowongan, itu memiliki status OPEN. Status koneksi sumber dan tujuan terowongan diatur ke DISCONNECTED.
2. Ketika perangkat (sumber atau tujuan) terhubung ke terowongan, status koneksi yang sesuai berubah menjadi CONNECTED.
3. Ketika perangkat terputus dari terowongan sementara status terowongan tetap ada OPEN, status koneksi yang sesuai akan berubah kembali ke DISCONNECTED. Sebuah perangkat dapat terhubung ke dan memutuskan sambungan dari terowongan berulang kali selama terowongan tetap ada OPEN.

### Note

Token akses klien (CAT) hanya dapat digunakan sekali untuk terhubung ke terowongan. Untuk menyambung kembali ke terowongan, putar token akses klien menggunakan operasi [Rotate Tunnel Access Token API](#) atau perintah [rotate-tunnel-access-token CLI](#). Sebagai contoh, lihat [Menyelesaikan masalah konektivitas tunneling yang AWS IoT aman dengan memutar token akses klien](#).

4. Saat Anda menelepon `CloseTunnel` atau terowongan tetap lebih OPEN lama dari `MaxLifetimeTimeout` nilainya, status terowongan menjadi CLOSED. Anda dapat mengonfigurasi `MaxLifetimeTimeout` saat menelepon `OpenTunnel`. `MaxLifetimeTimeoutdefault` ke 12 jam jika Anda tidak menentukan nilai.

### Note

Sebuah terowongan tidak dapat dibuka kembali ketika itu CLOSED.

5. Anda dapat memanggil `DescribeTunnel` dan `ListTunnels` melihat metadata terowongan saat terowongan terlihat. Terowongan dapat terlihat di AWS IoT konsol setidaknya selama tiga jam sebelum dihapus.

## AWS IoT tutorial tunneling aman

AWS IoT Tunneling aman membantu pelanggan membangun komunikasi dua arah ke perangkat jarak jauh yang berada di belakang firewall melalui koneksi aman yang dikelola oleh AWS IoT

Untuk demo tunneling AWS IoT aman, gunakan demo [tunneling AWS IoT aman](#) kami di GitHub

Tutorial berikut akan membantu Anda mempelajari cara memulai dan menggunakan tunneling yang aman. Anda akan belajar cara:

1. Buat terowongan aman menggunakan pengaturan cepat dan metode pengaturan manual untuk mengakses perangkat jarak jauh.
2. Konfigurasi proxy lokal saat menggunakan metode pengaturan manual dan sambungkan ke terowongan untuk mengakses perangkat tujuan.
3. SSH ke perangkat jarak jauh dari browser tanpa harus mengkonfigurasi proxy lokal.
4. Mengkonversi terowongan yang dibuat menggunakan AWS CLI atau menggunakan metode pengaturan manual untuk menggunakan metode penyiapan cepat.

### Tutorial di bagian ini

Tutorial di bagian ini berfokus pada pembuatan terowongan menggunakan AWS Management Console dan Referensi AWS IoT API. Di AWS IoT konsol, Anda dapat membuat terowongan dari halaman [hub Tunnels](#) atau dari halaman detail sesuatu yang Anda buat. Untuk informasi selengkapnya, lihat [Metode pembuatan terowongan di AWS IoT konsol](#).

Berikut ini menunjukkan tutorial di bagian ini:

- [Buka terowongan dan gunakan SSH berbasis browser untuk mengakses perangkat jarak jauh](#)

Tutorial ini menunjukkan cara membuka terowongan dari halaman [hub Tunnels](#) menggunakan metode penyiapan cepat. Anda juga akan belajar cara menggunakan SSH berbasis browser untuk mengakses perangkat jarak jauh menggunakan antarmuka baris perintah dalam konteks di dalam konsol AWS IoT

- [Buka terowongan menggunakan pengaturan manual dan sambungkan ke perangkat jarak jauh](#)

Tutorial ini menunjukkan cara membuka terowongan dari halaman [hub Tunnels](#) menggunakan metode pengaturan manual. Anda juga akan belajar cara mengkonfigurasi dan memulai proxy lokal dari terminal di perangkat sumber Anda dan terhubung ke terowongan.

- [Buka terowongan untuk perangkat jarak jauh dan gunakan SSH berbasis browser](#)

Tutorial ini menunjukkan cara membuka terowongan dari halaman detail sesuatu yang Anda buat. Anda akan belajar cara membuat terowongan baru dan menggunakan terowongan yang ada. Terowongan yang ada sesuai dengan terowongan terbuka terbaru yang dibuat untuk perangkat. Anda juga dapat menggunakan SSH berbasis browser untuk mengakses perangkat jarak jauh.

AWS IoT tutorial tunneling aman

- [Buka terowongan dan mulai sesi SSH ke perangkat jarak jauh](#)
- [Buka terowongan untuk perangkat jarak jauh dan gunakan SSH berbasis browser](#)

## Buka terowongan dan mulai sesi SSH ke perangkat jarak jauh

Dalam tutorial ini, Anda akan belajar cara mengakses perangkat yang berada di belakang firewall dari jarak jauh. Anda tidak dapat memulai sesi SSH langsung ke perangkat karena firewall memblokir semua lalu lintas masuk. Tutorial menunjukkan kepada Anda bagaimana Anda dapat membuka terowongan dan kemudian menggunakan terowongan itu untuk memulai sesi SSH ke perangkat jarak jauh.

### Prasyarat untuk tutorial

Prasyarat untuk menjalankan tutorial dapat bervariasi tergantung pada apakah Anda menggunakan metode pengaturan manual atau cepat untuk membuka terowongan dan mengakses perangkat jarak jauh.

#### Note

Untuk kedua metode pengaturan, Anda harus mengizinkan lalu lintas keluar pada port 443.

- Untuk informasi tentang prasyarat untuk tutorial metode penyiapan cepat, lihat. [Prasyarat untuk metode penyiapan cepat](#)

- Untuk informasi tentang prasyarat untuk tutorial metode penyiapan manual, lihat [Prasyarat untuk metode pengaturan manual](#). Jika Anda menggunakan metode penyiapan ini, Anda harus mengonfigurasi proxy lokal di perangkat sumber Anda. Untuk mengunduh kode sumber proxy lokal, lihat [Implementasi referensi proxy lokal di GitHub](#).

## Metode pengaturan terowongan

Dalam tutorial ini, Anda akan belajar tentang metode pengaturan manual dan cepat untuk membuka terowongan dan menghubungkan ke perangkat jarak jauh. Tabel berikut menunjukkan perbedaan antara metode setup. Setelah Anda membuat terowongan, Anda dapat menggunakan antarmuka baris perintah di browser ke SSH ke perangkat jarak jauh. Jika Anda salah menempatkan token atau terowongan terputus, Anda dapat mengirim token akses baru untuk terhubung kembali ke terowongan.

### Metode penyiapan cepat dan manual

| Kriteria             | Pengaturan cepat                                                                                                                                                                                                                                    | Pengaturan manual                                                                                                                                                                                                                                               |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Pembuatan terowongan | Buat terowongan baru dengan konfigurasi default yang dapat diedit. Untuk mengakses perangkat jarak jauh Anda, Anda hanya dapat menggunakan SSH sebagai layanan tujuan.                                                                              | Buat terowongan dengan menentukan konfigurasi terowongan secara manual. Anda dapat menggunakan metode ini untuk terhubung ke perangkat jarak jauh menggunakan layanan selain SSH.                                                                               |
| Token akses          | Token akses tujuan akan secara otomatis dikirimkan ke perangkat Anda pada <a href="#">topik MQTT yang dipesan</a> , jika nama sesuatu ditentukan saat membuat terowongan. Anda tidak perlu mengunduh atau mengelola token di perangkat sumber Anda. | Anda harus mengunduh dan mengelola token secara manual di perangkat sumber Anda. Token akses tujuan secara otomatis dikirimkan ke perangkat jarak jauh pada <a href="#">topik MQTT yang dicadangkan</a> , jika nama sesuatu ditentukan saat membuat terowongan. |
| Proksi lokal         | Proxy lokal berbasis web secara otomatis dikonfigurasi untuk Anda untuk berinteraksi dengan perangkat . Anda tidak perlu mengonfigurasi proxy lokal secara manual.                                                                                  | Anda harus mengkonfigurasi dan meluncurkan proxy lokal secara manual. Untuk mengonfigurasi proxy lokal, Anda dapat menggunakan AWS IoT Device Client atau                                                                                                       |



| Kriteria | Pengaturan cepat | Pengaturan manual                                                     |
|----------|------------------|-----------------------------------------------------------------------|
|          |                  | mengunduh <a href="#">implementasi referensi proxy lokal GitHub</a> . |

## Metode pembuatan terowongan di AWS IoT konsol

Tutorial di bagian ini menunjukkan cara membuat terowongan menggunakan AWS Management Console dan [OpenTunnelAPI](#). Jika Anda mengonfigurasi tujuan saat membuat terowongan, tunneling AWS IoT aman mengirimkan token akses klien tujuan ke perangkat jarak jauh melalui MQTT dan topik MQTT yang dicadangkan,). \$aws/things/RemoteDeviceA/tunnels/notify Saat menerima pesan MQTT, agen IoT pada perangkat jarak jauh memulai proxy lokal dalam mode tujuan. Untuk informasi selengkapnya, lihat [Topik yang dipesan](#).

### Note

Anda dapat menghilangkan konfigurasi tujuan jika Anda ingin mengirimkan token akses klien tujuan ke perangkat jarak jauh melalui metode lain. Untuk informasi selengkapnya, lihat [Mengkonfigurasi perangkat jarak jauh dan menggunakan agen IoT](#).

Di AWS IoT konsol, Anda dapat membuat terowongan menggunakan salah satu metode berikut. Untuk informasi tentang tutorial yang akan membantu Anda belajar membuat terowongan menggunakan metode ini, lihat [Tutorial di bagian ini](#).

- [Terowongan hub](#)

Saat Anda membuat terowongan, Anda akan dapat menentukan apakah akan menggunakan pengaturan cepat atau metode pengaturan manual untuk membuat terowongan dan memberikan detail konfigurasi terowongan opsional. Detail konfigurasi juga menyertakan nama perangkat tujuan dan layanan yang ingin Anda gunakan untuk menghubungkan ke perangkat. Setelah Anda membuat terowongan, Anda dapat SSH di dalam browser atau membuka terminal di luar AWS IoT konsol untuk mengakses perangkat jarak jauh Anda.

- Halaman detail hal

Saat membuat terowongan, Anda juga dapat menentukan apakah akan menggunakan terowongan terbuka terbaru atau membuat terowongan baru untuk perangkat, selain memilih metode penyiapan dan memberikan detail konfigurasi terowongan opsional. Anda tidak dapat mengedit

detail konfigurasi terowongan yang ada. Anda dapat menggunakan metode pengaturan cepat untuk memutar token akses dan SSH ke perangkat jarak jauh di dalam browser. Untuk membuka terowongan menggunakan metode ini, Anda harus membuat hal IoT (misalnya, RemoteDeviceA) di registri. AWS IoT Untuk informasi selengkapnya, lihat [Mendaftarkan perangkat di AWS IoT registri](#).

Tutorial di bagian ini

- [Buka terowongan dan gunakan SSH berbasis browser untuk mengakses perangkat jarak jauh](#)
- [Buka terowongan menggunakan pengaturan manual dan sambungkan ke perangkat jarak jauh](#)

## Buka terowongan dan gunakan SSH berbasis browser untuk mengakses perangkat jarak jauh

Anda dapat menggunakan pengaturan cepat atau metode pengaturan manual untuk membuat terowongan. Tutorial ini menunjukkan cara membuka terowongan menggunakan metode pengaturan cepat dan menggunakan SSH berbasis browser untuk terhubung ke perangkat jarak jauh. Untuk contoh yang menunjukkan cara membuka terowongan menggunakan metode penyiapan manual, lihat [Buka terowongan menggunakan pengaturan manual dan sambungkan ke perangkat jarak jauh](#).

Dengan menggunakan metode penyiapan cepat, Anda dapat membuat terowongan baru dengan konfigurasi default yang dapat diedit. Proxy lokal berbasis web dikonfigurasi untuk Anda dan token akses secara otomatis dikirimkan ke perangkat tujuan jarak jauh Anda menggunakan MQTT. Setelah membuat terowongan, Anda dapat mulai berinteraksi dengan perangkat jarak jauh menggunakan antarmuka baris perintah di dalam konsol.

Dengan metode pengaturan cepat, Anda harus menggunakan SSH sebagai layanan tujuan untuk mengakses perangkat jarak jauh. Untuk informasi selengkapnya tentang metode penyiapan yang berbeda, lihat [Metode pengaturan terowongan](#).

Prasyarat untuk metode penyiapan cepat

- Firewall yang berada di belakang perangkat jarak jauh harus memungkinkan lalu lintas keluar pada port 443. Terowongan yang Anda buat akan menggunakan port ini untuk terhubung ke perangkat jarak jauh.
- Anda memiliki agen perangkat IoT (lihat [Cuplikan agen IoT](#)) yang berjalan di perangkat jarak jauh yang terhubung ke gateway AWS IoT perangkat dan dikonfigurasi dengan langganan topik MQTT. Untuk informasi selengkapnya, lihat [menghubungkan perangkat ke gateway AWS IoT perangkat](#).

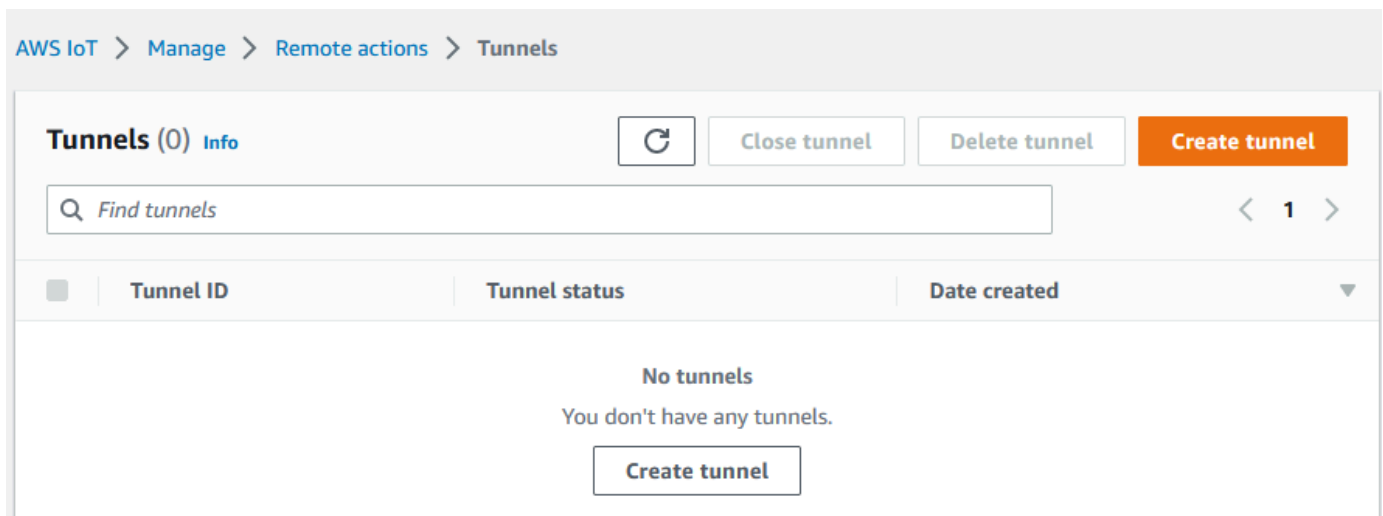
- Anda harus memiliki daemon SSH yang berjalan di perangkat jarak jauh.

## Buka terowongan

Anda dapat membuka terowongan aman menggunakan AWS Management Console, Referensi AWS IoT API, atau AWS CLI. Anda dapat secara opsional mengonfigurasi nama tujuan tetapi tidak diperlukan untuk tutorial ini. Jika Anda mengonfigurasi tujuan, tunneling aman akan secara otomatis mengirimkan token akses ke perangkat jarak jauh menggunakan MQTT. Untuk informasi selengkapnya, lihat [Metode pembuatan terowongan di AWS IoT konsol](#).

Untuk membuka terowongan menggunakan konsol

1. Pergi ke [hub Tunnels AWS IoT konsol](#) dan pilih Buat terowongan.



2. Untuk tutorial ini, pilih Quick setup sebagai metode pembuatan terowongan dan kemudian pilih Berikutnya.

### Note

Jika Anda membuat terowongan aman dari halaman detail sesuatu yang Anda buat, Anda dapat memilih apakah akan membuat terowongan baru atau menggunakan terowongan yang sudah ada. Untuk informasi selengkapnya, lihat [Buka terowongan untuk perangkat jarak jauh dan gunakan SSH berbasis browser](#).

### Setup method

Quick setup (SSH)

Manual setup

#### Quick setup (SSH)

Use quick setup to create a new tunnel with default, editable tunnel configurations. When you use quick setup:

- A web-based local proxy will be automatically configured for you to SSH into the remote device.
- The destination access token will be automatically delivered to your device on the [reserved MQTT topic](#), if a thing name is specified.

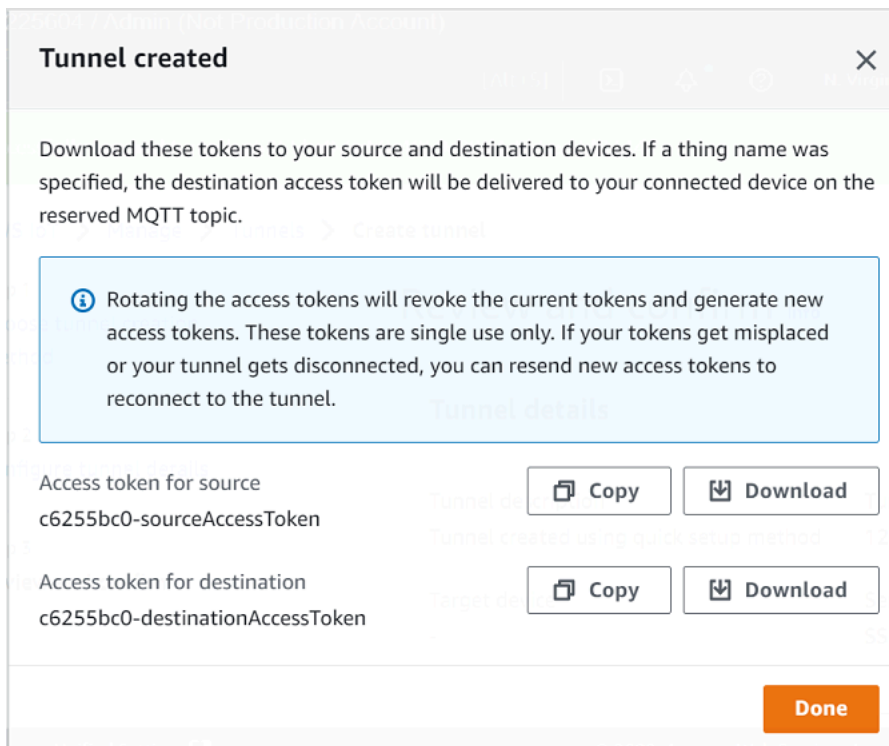
3. Tinjau dan konfirmasi detail konfigurasi terowongan. Untuk membuat terowongan, pilih Konfirmasi dan buat. Jika Anda ingin mengedit detail ini, pilih Sebelumnya untuk kembali ke halaman sebelumnya dan kemudian konfirmasi dan buat terowongan.

#### Note

Saat menggunakan pengaturan cepat, nama layanan tidak dapat diedit. Anda harus menggunakan SSH sebagai Layanan.

4. Untuk membuat terowongan, pilih Selesai.

Untuk tutorial ini, Anda tidak perlu mengunduh token akses sumber atau tujuan. Token ini hanya dapat digunakan sekali untuk terhubung ke terowongan. Jika terowongan Anda terputus, Anda dapat membuat dan mengirim token baru ke perangkat jarak jauh Anda untuk menyambung kembali ke terowongan. Untuk informasi selengkapnya, lihat [Kirim ulang token akses terowongan](#).



Untuk membuka terowongan menggunakan API

Untuk membuka terowongan baru, Anda dapat menggunakan operasi [OpenTunnel](#) API.

#### Note

Anda dapat membuat terowongan menggunakan metode penyiapan cepat hanya dari AWS IoT konsol. Saat Anda menggunakan AWS IoT API Referensi API atau API AWS CLI, itu akan menggunakan metode penyiapan manual. Anda dapat membuka terowongan yang ada yang Anda buat dan kemudian mengubah metode penyiapan terowongan untuk menggunakan pengaturan cepat. Untuk informasi selengkapnya, lihat [Buka terowongan yang ada dan gunakan SSH berbasis browser](#).

Berikut ini menunjukkan contoh bagaimana menjalankan operasi API ini. Secara opsional, jika Anda ingin menentukan nama benda dan layanan tujuan, gunakan `DestinationConfig` parameter. Untuk contoh yang menunjukkan cara menggunakan parameter ini, lihat [Buka terowongan baru untuk perangkat jarak jauh](#).

```
aws iotsecuretunneling open-tunnel
```

Menjalankan perintah ini membuat terowongan baru dan memberi Anda token akses sumber dan tujuan.

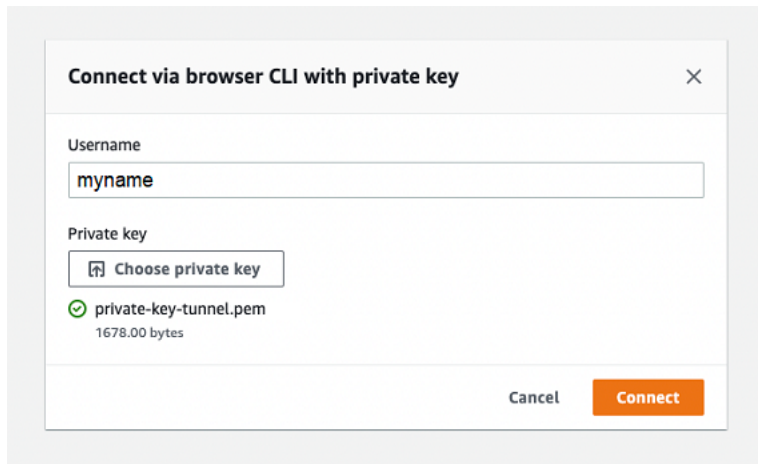
```
{
 "tunnelId": "01234567-89ab-0123-4c56-789a01234bcd",
 "tunnelArn": "arn:aws:iot:us-
east-1:123456789012:tunnel/01234567-89ab-0123-4c56-789a01234bcd",
 "sourceAccessToken": "<SOURCE_ACCESS_TOKEN>",
 "destinationAccessToken": "<DESTINATION_ACCESS_TOKEN>"
}
```

## Menggunakan SSH berbasis browser

Setelah Anda membuat terowongan menggunakan metode penyiapan cepat, dan perangkat tujuan Anda telah terhubung ke terowongan, Anda dapat mengakses perangkat jarak jauh menggunakan SSH berbasis browser. Menggunakan SSH berbasis browser, Anda dapat langsung berkomunikasi dengan perangkat jarak jauh dengan memasukkan perintah ke dalam antarmuka baris perintah dalam konteks dalam konsol. Fitur ini memudahkan Anda untuk berinteraksi dengan perangkat jarak jauh karena Anda tidak perlu membuka terminal di luar konsol atau mengonfigurasi proxy lokal.

### Untuk menggunakan SSH berbasis browser

1. Buka [hub Terowongan AWS IoT konsol](#) dan pilih terowongan yang Anda buat untuk melihat detailnya.
2. Buka bagian Secure Shell (SSH) dan kemudian pilih Connect.
3. Pilih apakah Anda ingin mengautentikasi ke koneksi SSH dengan memberikan nama pengguna dan kata sandi Anda, atau, untuk otentikasi yang lebih aman, Anda dapat menggunakan kunci pribadi perangkat Anda. Jika Anda mengautentikasi menggunakan kunci pribadi, Anda dapat menggunakan tipe kunci RSA, DSA, ECDSA (nistp-\*) dan ED25519, dalam format PEM (PKCS #1, PKCS #8) dan OpenSSH.
  - Untuk terhubung menggunakan nama pengguna dan kata sandi Anda, pilih Gunakan kata sandi. Anda kemudian dapat memasukkan nama pengguna dan kata sandi Anda dan mulai menggunakan CLI di browser.
  - Untuk terhubung menggunakan kunci pribadi perangkat tujuan Anda, pilih Gunakan kunci pribadi. Tentukan nama pengguna Anda dan unggah file kunci pribadi perangkat, lalu pilih Connect untuk mulai menggunakan CLI dalam browser.



Setelah Anda mengautentikasi ke koneksi SSH, Anda dapat dengan cepat memulai memasukkan perintah dan berinteraksi dengan perangkat menggunakan CLI browser, karena proxy lokal telah dikonfigurasi untuk Anda.

▼ Comand line interface [info](#)

```
const [preferences, setPreferences] = React.useState(
 undefined
);
const [loading, setLoading] = React.useState(false);
return (
 <CodeEditor
 ace={ace}
 language="javascript"
 value="const pi = 3.14;"
 preferences={preferences}
 onPreferencesChange={e => setPreferences(e.detail)}
 loading={loading}
 />
);
```

Jika CLI browser tetap terbuka setelah durasi terowongan, mungkin waktu habis, menyebabkan antarmuka baris perintah terputus. Anda dapat menduplikasi terowongan dan memulai sesi lain untuk berinteraksi dengan perangkat jarak jauh di dalam konsol itu sendiri.

## Memecahkan masalah saat menggunakan SSH berbasis browser

Berikut ini menunjukkan cara memecahkan masalah beberapa masalah yang mungkin Anda hadapi saat menggunakan SSH berbasis browser.

- Anda melihat kesalahan alih-alih antarmuka baris perintah

Anda mungkin melihat kesalahan karena perangkat tujuan Anda terputus. Anda dapat memilih Hasilkan token akses baru untuk menghasilkan token akses baru dan mengirim token ke perangkat jarak jauh Anda menggunakan MQTT. Token baru dapat digunakan untuk menyambung kembali ke terowongan. Menghubungkan kembali ke terowongan menghapus riwayat dan menyegarkan sesi baris perintah.

- Anda melihat kesalahan terputus terowongan saat mengautentikasi menggunakan kunci pribadi

Anda mungkin melihat kesalahan karena kunci pribadi Anda mungkin tidak diterima oleh perangkat tujuan. Untuk memecahkan masalah kesalahan ini, periksa file kunci pribadi yang Anda unggah untuk otentikasi. Jika Anda masih melihat kesalahan, periksa log perangkat Anda. Anda juga dapat mencoba menghubungkan kembali ke terowongan dengan mengirimkan token akses baru ke perangkat jarak jauh Anda.

- Terowongan Anda ditutup saat menggunakan sesi

Jika terowongan Anda ditutup karena tetap terbuka selama lebih dari durasi yang ditentukan, sesi baris perintah Anda mungkin terputus. Terowongan tidak dapat dibuka kembali setelah ditutup. Untuk menyambung kembali, Anda harus membuka terowongan lain ke perangkat.

Anda dapat menduplikasi terowongan untuk membuat terowongan baru dengan konfigurasi yang sama dengan terowongan tertutup. Anda dapat menduplikasi terowongan tertutup dari AWS IoT konsol. Untuk menduplikasi terowongan, pilih terowongan yang ditutup untuk melihat detailnya, lalu pilih Terowongan duplikat. Tentukan durasi terowongan yang ingin Anda gunakan dan kemudian buat terowongan baru.

## Membersihkan

- Tutup terowongan

Kami menyarankan Anda menutup terowongan setelah Anda selesai menggunakannya. Terowongan juga dapat ditutup jika tetap terbuka lebih lama dari durasi terowongan yang ditentukan. Terowongan tidak dapat dibuka kembali setelah ditutup. Anda masih dapat menduplikasi terowongan dengan memilih terowongan tertutup dan kemudian memilih terowongan



Duplikat. Tentukan durasi terowongan yang ingin Anda gunakan dan kemudian buat terowongan baru.

- Untuk menutup terowongan individu atau beberapa terowongan dari AWS IoT konsol, buka [hub Tunnels](#), pilih terowongan yang ingin Anda tutup, lalu pilih Tutup terowongan.
- Untuk menutup terowongan individu atau beberapa terowongan menggunakan AWS IoT API Referensi API, gunakan [CloseTunnel](#)API.

```
aws iotsecuretunneling close-tunnel \
 --tunnel-id "01234567-89ab-0123-4c56-789a01234bcd"
```

- Hapus terowongan

Anda dapat menghapus terowongan secara permanen dari Anda Akun AWS.

#### Warning

Tindakan penghapusan bersifat permanen dan tidak dapat dibatalkan.

- Untuk menghapus terowongan individual atau beberapa terowongan dari AWS IoT konsol, buka [hub Tunnels](#), pilih terowongan yang ingin Anda hapus, lalu pilih Delete tunnel.
- Untuk menghapus terowongan individual atau beberapa terowongan menggunakan AWS IoT API Referensi API, gunakan [CloseTunnel](#)API. Saat menggunakan API, setel `delete` flag ke `true`.

```
aws iotsecuretunneling close-tunnel \
 --tunnel-id "01234567-89ab-0123-4c56-789a01234bcd" \
 --delete true
```

## Buka terowongan menggunakan pengaturan manual dan sambungkan ke perangkat jarak jauh

Saat Anda membuka terowongan, Anda dapat memilih pengaturan cepat atau metode pengaturan manual untuk membuka terowongan ke perangkat jarak jauh. Tutorial ini menunjukkan cara membuka terowongan menggunakan metode pengaturan manual dan mengkonfigurasi dan memulai proxy lokal untuk terhubung ke perangkat jarak jauh.

Saat Anda menggunakan metode pengaturan manual, Anda harus menentukan konfigurasi terowongan secara manual saat membuat terowongan. Setelah membuat terowongan, Anda dapat SSH di dalam browser atau membuka terminal di luar AWS IoT konsol. Tutorial ini menunjukkan cara menggunakan terminal di luar konsol untuk mengakses perangkat jarak jauh. Anda juga akan mempelajari cara mengonfigurasi proxy lokal dan kemudian terhubung ke proxy lokal untuk berinteraksi dengan perangkat jarak jauh. Untuk terhubung ke proxy lokal, Anda harus mengunduh token akses sumber saat membuat terowongan.

Dengan metode pengaturan ini, Anda dapat menggunakan layanan selain SSH, seperti FTP untuk terhubung ke perangkat jarak jauh. Untuk informasi selengkapnya tentang metode penyiapan yang berbeda, lihat [Metode pengaturan terowongan](#).

Prasyarat untuk metode pengaturan manual

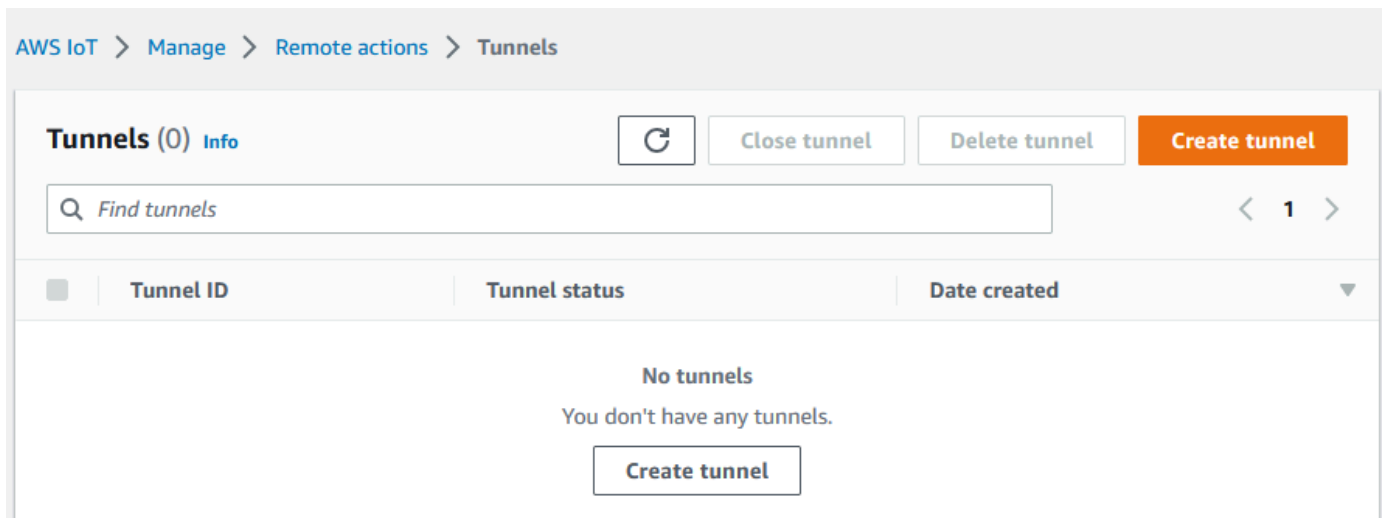
- Firewall yang berada di belakang perangkat jarak jauh harus memungkinkan lalu lintas keluar pada port 443. Terowongan yang Anda buat akan menggunakan port ini untuk terhubung ke perangkat jarak jauh.
- Anda memiliki agen perangkat IoT (lihat [Cuplikan agen IoT](#)) yang berjalan di perangkat jarak jauh yang terhubung ke gateway AWS IoT perangkat dan dikonfigurasi dengan langganan topik MQTT. Untuk informasi selengkapnya, lihat [menghubungkan perangkat ke gateway AWS IoT perangkat](#).
- Anda harus memiliki daemon SSH yang berjalan di perangkat jarak jauh.
- Anda telah mengunduh kode sumber proxy lokal dari [GitHub](#) dan membangunnya untuk platform pilihan Anda. Kita akan merujuk ke file executable proxy lokal yang dibangun seperti `localproxy` dalam tutorial ini.

Buka terowongan

Anda dapat membuka terowongan aman menggunakan AWS Management Console, Referensi AWS IoT API, atau AWS CLI. Anda dapat secara opsional mengonfigurasi nama tujuan tetapi tidak diperlukan untuk tutorial ini. Jika Anda mengonfigurasi tujuan, tunneling aman akan secara otomatis mengirimkan token akses ke perangkat jarak jauh menggunakan MQTT. Untuk informasi selengkapnya, lihat [Metode pembuatan terowongan di AWS IoT konsol](#).

Untuk membuka terowongan di konsol

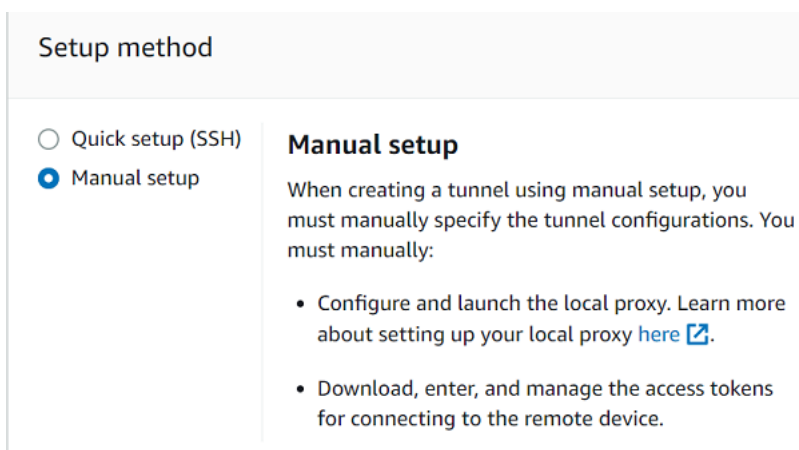
1. Pergi ke [hub Tunnels AWS IoT konsol](#) dan pilih Buat terowongan.



- Untuk tutorial ini, pilih Pengaturan manual sebagai metode pembuatan terowongan dan kemudian pilih Berikutnya. Untuk informasi tentang menggunakan metode penyiapan cepat untuk membuat terowongan, lihat [Buka terowongan dan gunakan SSH berbasis browser untuk mengakses perangkat jarak jauh](#).

#### Note

Jika Anda membuat terowongan aman dari halaman detail sesuatu, Anda dapat memilih apakah akan membuat terowongan baru atau menggunakan terowongan yang sudah ada. Untuk informasi selengkapnya, lihat [Buka terowongan untuk perangkat jarak jauh dan gunakan SSH berbasis browser](#).



- (Opsional) Masukkan pengaturan konfigurasi untuk terowongan Anda. Anda juga dapat melewati langkah ini dan melanjutkan ke langkah berikutnya untuk membuat terowongan.

Masukkan deskripsi terowongan, durasi batas waktu terowongan, dan tag sumber daya sebagai pasangan nilai kunci untuk membantu Anda mengidentifikasi sumber daya Anda. Untuk tutorial ini, Anda dapat melewati konfigurasi tujuan.

#### Note

Anda tidak akan dikenakan biaya berdasarkan durasi terowongan tetap terbuka. Anda hanya dikenakan biaya saat membuat terowongan baru. [Untuk informasi harga, lihat Terowongan Aman dalam AWS IoT Device Management harga.](#)

4. Unduh token akses klien dan kemudian pilih Selesai. Token tidak akan tersedia untuk diunduh setelah Anda memilih Selesai.

Token ini hanya dapat digunakan sekali untuk terhubung ke terowongan. Jika Anda salah menempatkan token atau terowongan terputus, Anda dapat membuat dan mengirim token baru ke perangkat jarak jauh Anda untuk menyambung kembali ke terowongan.

**Tunnel created**

Download these tokens to your source and destination devices. If a thing name was specified, the destination access token will be delivered to your connected device on the reserved MQTT topic.

Rotating the access tokens will revoke the current tokens and generate new access tokens. These tokens are single use only. If your tokens get misplaced or your tunnel gets disconnected, you can resend new access tokens to reconnect to the tunnel.

Access token for source  
c6255bc0-sourceAccessToken

Access token for destination  
c6255bc0-destinationAccessToken

Done

Untuk membuka terowongan menggunakan API

Untuk membuka terowongan baru, Anda dapat menggunakan operasi [OpenTunnel](#) API. Anda juga dapat menentukan konfigurasi tambahan menggunakan API, seperti durasi terowongan dan konfigurasi tujuan.

```
aws iotsecuretunneling open-tunnel \
 --region us-east-1 \
 --endpoint https://api.us-east-1.tunneling.iot.amazonaws.com
```

Menjalankan perintah ini membuat terowongan baru dan memberi Anda token akses sumber dan tujuan.

```
{
 "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>"
}
```

Kirim ulang token akses terowongan

Token yang Anda peroleh saat membuat terowongan hanya dapat digunakan sekali untuk terhubung ke terowongan. Jika Anda salah menempatkan token akses atau terowongan terputus, Anda dapat mengirim ulang token akses baru ke perangkat jarak jauh menggunakan MQTT tanpa biaya tambahan. AWS IoT Tunneling aman akan mencabut token saat ini dan mengembalikan token akses baru untuk menghubungkan kembali ke terowongan.

Untuk memutar token dari konsol

1. Pergi ke [hub Terowongan AWS IoT konsol](#) dan pilih terowongan yang Anda buat.
2. Di halaman detail terowongan, pilih Hasilkan token akses baru, lalu pilih Berikutnya.
3. Unduh token akses baru untuk terowongan Anda dan pilih Selesai. Token ini hanya dapat digunakan sekali. Jika Anda salah menempatkan token ini atau terowongan terputus, Anda dapat mengirim ulang token akses baru.

**Tokens rotated**

Download these tokens to your source and destination devices. If a thing name was specified, the destination access token will be delivered to your connected device on the reserved MQTT topic.

**i** Rotating the access tokens will revoke the current tokens and generate new access tokens. These tokens are single use only. If your tokens get misplaced or your tunnel gets disconnected, you can resend new access tokens to reconnect to the tunnel.

Access token for source  
c6255bc0-sourceAccessToken

Access token for destination  
c6255bc0-destinationAccessToken

Done

Untuk memutar token akses menggunakan API

Untuk memutar token akses terowongan, Anda dapat menggunakan operasi [RotateTunnelAccessToken](#) API untuk mencabut token saat ini dan mengembalikan token akses baru untuk menyambung kembali ke terowongan. Misalnya, perintah berikut memutar token akses untuk perangkat tujuan, *RemoteThing1*.

```
aws iotsecuretunneling rotate-tunnel-access-token \
 --tunnel-id <tunnel-id> \
 --client-mode DESTINATION \
 --destination-config thingName=<RemoteThing1>,services=SSH \
 --region <region>
```

Menjalankan perintah ini menghasilkan token akses baru seperti yang ditunjukkan pada contoh berikut. Token kemudian dikirim ke perangkat menggunakan MQTT untuk terhubung ke terowongan, jika agen perangkat diatur dengan benar.

```
{
 "destinationAccessToken": "destination-access-token",
 "tunnelArn": "arn:aws:iot:region:account-id:tunnel/tunnel-id"
}
```

Untuk contoh yang menunjukkan bagaimana dan kapan harus memutar token akses, lihat [Menyelesaikan masalah konektivitas tunneling yang AWS IoT aman dengan memutar token akses klien](#).

Konfigurasi dan mulai proxy lokal

Untuk terhubung ke perangkat jarak jauh, buka terminal di laptop Anda dan konfigurasi dan mulai proxy lokal. Proxy lokal mentransmisikan data yang dikirim oleh aplikasi yang berjalan pada perangkat sumber dengan menggunakan tunneling aman melalui koneksi aman. WebSocket Anda dapat mengunduh sumber proxy lokal dari [GitHub](#).

Setelah Anda mengonfigurasi proxy lokal, salin token akses klien sumber, dan gunakan untuk memulai proxy lokal dalam mode sumber. Berikut ini menunjukkan contoh perintah untuk memulai proxy lokal. Dalam perintah berikut, proxy lokal dikonfigurasi untuk mendengarkan koneksi baru pada port 5555. Dalam perintah ini:

- -r menentukan Wilayah AWS, yang harus menjadi Wilayah yang sama di mana terowongan Anda dibuat.
- -s menentukan port yang harus dihubungkan oleh proxy.
- -t menentukan teks token klien.

```
./localproxy -r us-east-1 -s 5555 -t source-client-access-token
```

Menjalankan perintah ini akan memulai proxy lokal dalam mode sumber. Jika Anda menerima kesalahan berikut setelah menjalankan perintah, atur jalur CA. Untuk selengkapnya, lihat [Proksi lokal tunneling aman](#) aktif. GitHub

```
Could not perform SSL handshake with proxy server: certificate verify failed
```

Berikut ini menunjukkan contoh output menjalankan proxy lokal dalam source mode.

```
...
...
```

**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
```

## Memulai sesi SSH

Buka terminal lain dan gunakan perintah berikut untuk memulai sesi SSH baru dengan menghubungkan ke proxy lokal pada port 5555.

```
ssh username@localhost -p 5555
```

Anda mungkin akan diminta untuk password untuk sesi SSH. Setelah selesai dengan sesi SSH, ketik **exit** untuk menutup sesi.

## Membersihkan

- Tutup terowongan

Kami menyarankan Anda menutup terowongan setelah Anda selesai menggunakannya. Terowongan juga dapat ditutup jika tetap terbuka lebih lama dari durasi terowongan yang ditentukan. Terowongan tidak dapat dibuka kembali setelah ditutup. Anda masih dapat



menduplikasi terowongan dengan membuka terowongan tertutup dan kemudian memilih terowongan Duplikat. Tentukan durasi terowongan yang ingin Anda gunakan dan kemudian buat terowongan baru.

- Untuk menutup terowongan individu atau beberapa terowongan dari AWS IoT konsol, buka [hub Tunnels](#), pilih terowongan yang ingin Anda tutup, lalu pilih Tutup terowongan.
- Untuk menutup terowongan individual atau beberapa terowongan menggunakan AWS IoT API Referensi API, gunakan operasi [CloseTunnelAPI](#).

```
aws iotsecuretunneling close-tunnel \
 --tunnel-id "01234567-89ab-0123-4c56-789a01234bcd"
```

- Hapus terowongan

Anda dapat menghapus terowongan secara permanen dari Anda Akun AWS.

#### Warning

Tindakan penghapusan bersifat permanen dan tidak dapat dibatalkan.

- Untuk menghapus terowongan individual atau beberapa terowongan dari AWS IoT konsol, buka [hub Tunnels](#), pilih terowongan yang ingin Anda hapus, lalu pilih Delete tunnel.
- Untuk menghapus terowongan individual atau beberapa terowongan menggunakan AWS IoT API Referensi API, gunakan operasi [CloseTunnelAPI](#). Saat menggunakan API, setel `delete` flag ke `true`.

```
aws iotsecuretunneling close-tunnel \
 --tunnel-id "01234567-89ab-0123-4c56-789a01234bcd" \
 --delete true
```

## Buka terowongan untuk perangkat jarak jauh dan gunakan SSH berbasis browser

Dari AWS IoT konsol, Anda dapat membuat terowongan baik dari hub Tunnels atau dari halaman detail hal IoT yang Anda buat. Saat Anda membuat terowongan dari hub Tunnels, Anda dapat menentukan apakah akan membuat terowongan menggunakan pengaturan cepat atau pengaturan manual. Untuk tutorial contoh, lihat [Buka terowongan dan mulai sesi SSH ke perangkat jarak jauh](#).

Saat Anda membuat terowongan dari halaman detail hal AWS IoT konsol, Anda juga dapat menentukan apakah akan membuat terowongan baru atau membuka terowongan yang ada untuk hal itu seperti yang diilustrasikan dalam tutorial ini. Jika Anda memilih terowongan yang ada, Anda dapat mengakses terowongan terbuka terbaru yang Anda buat untuk perangkat ini. Anda kemudian dapat menggunakan antarmuka baris perintah di dalam terminal untuk SSH ke dalam perangkat.

## Prasyarat

- Firewall yang berada di belakang perangkat jarak jauh harus memungkinkan lalu lintas keluar pada port 443. Terowongan yang Anda buat akan menggunakan port ini untuk terhubung ke perangkat jarak jauh.
- Anda telah membuat hal IoT (misalnya, `RemoteDevice1`) di registri. AWS IoT Hal ini sesuai dengan representasi perangkat jarak jauh Anda di cloud. Untuk informasi selengkapnya, lihat [Mendaftarkan perangkat di AWS IoT registri](#).
- Anda memiliki agen perangkat IoT (lihat [Cuplikan agen IoT](#)) yang berjalan di perangkat jarak jauh yang terhubung ke gateway AWS IoT perangkat dan dikonfigurasi dengan langganan topik MQTT. Untuk informasi selengkapnya, lihat [menghubungkan perangkat ke gateway AWS IoT perangkat](#).
- Anda harus memiliki daemon SSH yang berjalan di perangkat jarak jauh.

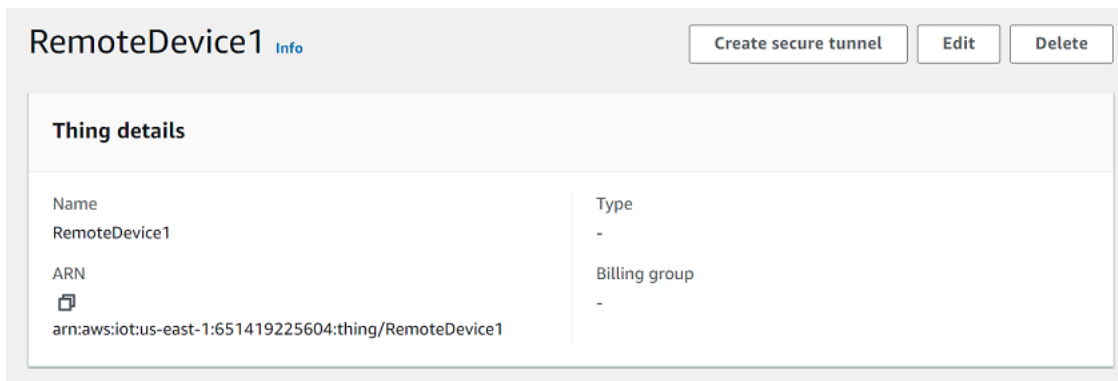
## Buka terowongan baru untuk perangkat jarak jauh

Katakanlah Anda ingin membuka terowongan ke perangkat jarak jauh Anda, `RemoteDevice1`. Pertama, buat hal IoT dengan nama `RemoteDevice1` di registri. AWS IoT Anda kemudian dapat membuat terowongan menggunakan AWS IoT API Referensi API, atau AWS CLI. AWS Management Console

Dengan mengonfigurasi tujuan saat membuat terowongan, layanan tunneling aman mengirimkan token akses klien tujuan ke perangkat jarak jauh melalui MQTT dan topik MQTT yang dicadangkan (). `$aws/things/RemoteDeviceA/tunnels/notify` Untuk informasi selengkapnya, lihat [Metode pembuatan terowongan di AWS IoT konsol](#).

Untuk membuat terowongan untuk perangkat jarak jauh dari konsol

1. Pilih hal, `RemoteDevice1`, untuk melihat detailnya, dan kemudian pilih Buat terowongan aman.



- Pilih apakah akan membuat terowongan baru atau membuka terowongan yang ada. Untuk membuat terowongan baru, pilih Buat terowongan baru. Anda kemudian dapat memilih apakah akan menggunakan pengaturan manual atau metode pengaturan cepat untuk membuat terowongan. Lihat informasi yang lebih lengkap di [Buka terowongan menggunakan pengaturan manual dan sambungkan ke perangkat jarak jauh](#) dan [Buka terowongan dan gunakan SSH berbasis browser untuk mengakses perangkat jarak jauh](#).

Untuk membuat terowongan untuk perangkat jarak jauh menggunakan API

Untuk membuka terowongan baru, Anda dapat menggunakan operasi [OpenTunnel](#) API. Kode berikut menunjukkan contoh menjalankan perintah ini.

```
aws iotsecuretunneling open-tunnel \
 --region us-east-1 \
 --endpoint https://api.us-east-1.tunneling.iot.amazonaws.com
 --cli-input-json file://input.json
```

Berikut ini menunjukkan isi untuk `input.json` file. Anda dapat menggunakan `destinationConfig` parameter untuk menentukan nama perangkat tujuan (misalnya, *RemoteDevice1*) dan layanan yang ingin Anda gunakan untuk mengakses perangkat tujuan, seperti *SSH*. Secara opsional, Anda juga dapat menentukan parameter tambahan seperti deskripsi terowongan dan tag.

Isi `input.json`

```
{
 "description": "Tunnel to remote device1",
 "destinationConfig": {
 "services": ["SSH"],
 "thingName": "RemoteDevice1"
 }
}
```

```
}
}
```

Menjalankan perintah ini membuat terowongan baru dan memberi Anda token akses sumber dan tujuan.

```
{
 "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>"
}
```

## Buka terowongan yang ada dan gunakan SSH berbasis browser

Katakanlah Anda membuat terowongan untuk perangkat jarak jauh `RemoteDevice1`, menggunakan metode penyiapan manual atau menggunakan AWS IoT API Referensi API. Anda kemudian dapat membuka terowongan yang ada untuk perangkat dan memilih Pengaturan cepat untuk menggunakan fitur SSH berbasis browser. Konfigurasi terowongan yang ada tidak dapat diedit sehingga Anda tidak dapat menggunakan metode penyiapan manual.

Untuk menggunakan fitur SSH berbasis browser, Anda tidak perlu mengunduh token akses sumber atau mengonfigurasi proxy lokal. Proxy lokal berbasis web akan dikonfigurasi secara otomatis untuk Anda sehingga Anda dapat mulai berinteraksi dengan perangkat jarak jauh Anda.

Untuk menggunakan metode penyiapan cepat dan SSH berbasis browser

1. Buka halaman detail dari hal yang Anda buat, `RemoteDevice1`, dan Buat terowongan aman.
2. Pilih Gunakan terowongan yang ada untuk membuka terowongan terbuka terbaru yang Anda buat untuk perangkat jarak jauh. Konfigurasi terowongan tidak dapat diedit sehingga Anda tidak dapat menggunakan metode penyiapan manual untuk terowongan. Untuk menggunakan metode penyiapan cepat, pilih Pengaturan cepat.
3. Lanjutkan untuk meninjau dan mengonfirmasi detail konfigurasi terowongan dan membuat terowongan. Konfigurasi terowongan tidak dapat diedit.

Saat Anda membuat terowongan, tunneling aman akan menggunakan operasi [RotateTunnelAccessToken](#) API untuk mencabut token akses asli dan menghasilkan token akses baru. Jika perangkat jarak jauh Anda menggunakan MQTT, token ini akan secara otomatis

dikirimkan ke perangkat jarak jauh pada topik MQTT yang dilangganannya. Anda juga dapat memilih untuk mengunduh token ini secara manual ke perangkat sumber Anda.

Setelah membuat terowongan, Anda dapat menggunakan SSH berbasis browser untuk berinteraksi dengan perangkat jarak jauh langsung dari konsol menggunakan antarmuka baris perintah dalam konteks. Untuk menggunakan antarmuka baris perintah ini, pilih terowongan untuk hal yang Anda buat, dan di halaman detail, perluas bagian antarmuka baris perintah. Karena proxy lokal telah dikonfigurasi untuk Anda, Anda dapat mulai memasukkan perintah untuk memulai dengan cepat mengakses dan berinteraksi dengan perangkat jarak jauh Anda, `RemoteDevice1`

Untuk informasi selengkapnya tentang metode penyiapan cepat dan menggunakan SSH berbasis browser, lihat [Buka terowongan dan gunakan SSH berbasis browser untuk mengakses perangkat jarak jauh](#)

## Membersihkan

- Tutup terowongan

Kami menyarankan Anda menutup terowongan setelah Anda selesai menggunakannya. Terowongan juga dapat ditutup jika tetap terbuka lebih lama dari durasi terowongan yang ditentukan. Terowongan tidak dapat dibuka kembali setelah ditutup. Anda masih dapat menduplikasi terowongan dengan membuka terowongan tertutup dan kemudian memilih terowongan Duplikat. Tentukan durasi terowongan yang ingin Anda gunakan dan kemudian buat terowongan baru.

- Untuk menutup terowongan individu atau beberapa terowongan dari AWS IoT konsol, buka [hub Tunnels](#), pilih terowongan yang ingin Anda tutup, lalu pilih Tutup terowongan.
- Untuk menutup terowongan individual atau beberapa terowongan menggunakan AWS IoT API Referensi API, gunakan operasi [CloseTunnelAPI](#).

```
aws iotsecuretunneling close-tunnel \
 --tunnel-id "01234567-89ab-0123-4c56-789a01234bcd"
```

- Hapus terowongan

Anda dapat menghapus terowongan secara permanen dari Anda Akun AWS.

**⚠ Warning**

Tindakan penghapusan bersifat permanen dan tidak dapat dibatalkan.

- Untuk menghapus terowongan individual atau beberapa terowongan dari AWS IoT konsol, buka [hub Tunnels](#), pilih terowongan yang ingin Anda hapus, lalu pilih Delete tunnel.
- Untuk menghapus terowongan individual atau beberapa terowongan menggunakan AWS IoT API Referensi API, gunakan operasi [CloseTunnelAPI](#). Saat menggunakan API, setel `delete` flag ke `true`.

```
aws iotsecuretunneling close-tunnel \
 --tunnel-id "01234567-89ab-0123-4c56-789a01234bcd"
 --delete true
```

## Proksi lokal

Proxy lokal mentransmisikan data yang dikirim oleh aplikasi yang berjalan pada perangkat sumber dengan menggunakan tunneling aman melalui koneksi aman. WebSocket Anda dapat mengunduh sumber proxy lokal dari [GitHub](#).

Proxy lokal dapat berjalan dalam dua mode: `source` atau `destination`. Dalam mode sumber, proxy lokal berjalan pada perangkat atau jaringan yang sama dengan aplikasi klien yang memulai koneksi TCP. Dalam mode tujuan, proxy lokal berjalan pada perangkat jarak jauh, bersama dengan aplikasi tujuan. Sebuah terowongan tunggal dapat mendukung hingga tiga aliran data pada satu waktu dengan menggunakan multiplexing terowongan. Untuk setiap aliran data, tunneling aman menggunakan beberapa koneksi TCP, yang mengurangi potensi waktu istirahat. Untuk informasi selengkapnya, lihat [Aliran data multipleks dan menggunakan koneksi TCP simultan di terowongan aman](#).

## Cara menggunakan proxy lokal

Anda dapat menjalankan proxy lokal pada perangkat sumber dan tujuan untuk mengirimkan data ke titik akhir tunneling yang aman. Jika perangkat Anda berada dalam jaringan yang menggunakan proxy web, proxy web dapat mencegah koneksi sebelum meneruskannya ke internet. Dalam hal

ini, Anda harus mengonfigurasi proxy lokal Anda untuk menggunakan proxy web. Untuk informasi selengkapnya, lihat [Konfigurasi proxy lokal untuk perangkat yang menggunakan proxy web](#).

## Alur kerja proxy lokal

Langkah-langkah berikut menunjukkan bagaimana proxy lokal dijalankan pada perangkat sumber dan tujuan.

### 1. Connect proxy lokal untuk mengamankan tunneling

Pertama, proxy lokal harus membuat koneksi untuk mengamankan tunneling. Saat Anda memulai proxy lokal, gunakan argumen berikut:

- `-r` Argumen untuk menentukan Wilayah AWS di mana terowongan dibuka.
- `-t` Argumen untuk meneruskan token akses klien sumber atau tujuan dikembalikan dari `OpenTunnel`.

#### Note

Dua proxy lokal yang menggunakan nilai token akses klien yang sama tidak dapat dihubungkan pada saat yang bersamaan.

### 2. Lakukan tindakan sumber atau tujuan

Setelah WebSocket koneksi dibuat, proxy lokal melakukan mode sumber atau tindakan mode tujuan, tergantung pada konfigurasinya.

Secara default, proxy lokal mencoba untuk menyambung kembali untuk mengamankan tunneling jika terjadi kesalahan input/output (I/O) atau jika koneksi ditutup secara tidak terduga. WebSocket Hal ini menyebabkan koneksi TCP ditutup. Jika terjadi kesalahan soket TCP, proxy lokal mengirim pesan melalui terowongan untuk memberi tahu pihak lain untuk menutup koneksi TCP-nya. Secara default, proxy lokal selalu menggunakan komunikasi SSL.

### 3. Hentikan proxy lokal

Setelah Anda menggunakan terowongan, aman untuk menghentikan proses proxy lokal. Kami menyarankan Anda menutup terowongan secara eksplisit dengan menelepon `CloseTunnel`. Klien terowongan aktif mungkin tidak ditutup tepat setelah menelepon `CloseTunnel`.

Untuk informasi lebih lanjut tentang cara menggunakan AWS Management Console untuk membuka terowongan dan memulai sesi SSH, lihat [Buka terowongan dan mulai sesi SSH ke perangkat jarak jauh](#).

## Praktik terbaik proxy lokal

Saat menjalankan proxy lokal, ikuti praktik terbaik berikut:

- Hindari penggunaan argumen proxy -t lokal untuk meneruskan token akses. Kami menyarankan Anda menggunakan variabel `AWSIoT_TUNNEL_ACCESS_TOKEN` lingkungan untuk mengatur token akses untuk proxy lokal.
- Jalankan proxy lokal yang dapat dieksekusi dengan hak istimewa paling sedikit di sistem operasi atau lingkungan.
  - Hindari menjalankan proxy lokal sebagai administrator di Windows.
  - Hindari menjalankan proxy lokal sebagai root di Linux dan macOS.
- Pertimbangkan untuk menjalankan proxy lokal pada host, kontainer, kotak pasir, penjara chroot, atau lingkungan virtual yang terpisah.
- Buat proxy lokal dengan flag keamanan yang relevan, tergantung pada toolchain Anda.
- Pada perangkat dengan beberapa antarmuka jaringan, gunakan -b argumen untuk mengikat soket TCP ke antarmuka jaringan yang digunakan untuk berkomunikasi dengan aplikasi tujuan.

## Contoh perintah dan output

Berikut ini menunjukkan contoh perintah yang Anda jalankan dan output yang sesuai. Contoh menunjukkan bagaimana proxy lokal dapat dikonfigurasi dalam kedua `source` dan `destination` mode. Proxy lokal meningkatkan protokol HTTPS WebSockets untuk membuat koneksi berumur panjang dan kemudian mulai mentransmisikan data melalui koneksi ke titik akhir perangkat tunneling aman.

Sebelum Anda menjalankan perintah ini:

Anda harus telah membuka terowongan dan memperoleh token akses klien untuk sumber dan tujuan. Anda juga harus membangun proxy lokal seperti yang dijelaskan sebelumnya. Untuk membangun proxy lokal, buka [kode sumber proxy lokal](#) di GitHub repositori dan ikuti instruksi untuk membangun dan menginstal proxy lokal.



**Note**

Perintah berikut yang digunakan dalam contoh menggunakan `verbosity` bendera untuk mengilustrasikan ikhtisar langkah-langkah berbeda yang dijelaskan sebelumnya setelah Anda menjalankan proxy lokal. Kami menyarankan Anda menggunakan bendera ini hanya untuk tujuan pengujian.

## Menjalankan proxy lokal dalam mode sumber

Perintah berikut menunjukkan cara menjalankan proxy lokal dalam mode sumber.

### Linux/macOS

Di Linux atau macOS, jalankan perintah berikut di terminal untuk mengonfigurasi dan memulai proxy lokal di sumber Anda.

```
export AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
./localproxy -s 5555 -v 5 -r us-west-2
```

Di mana:

- `-s` adalah port mendengarkan sumber, yang memulai proxy lokal dalam mode sumber.
- `-v` adalah verbositas output, yang bisa menjadi nilai antara nol dan enam.
- `-r` adalah wilayah titik akhir tempat terowongan dibuka.

Untuk informasi selengkapnya tentang parameter, lihat [Opsinya yang disetel menggunakan argumen baris perintah](#).

### Windows

Di Windows, Anda mengonfigurasi proxy lokal yang mirip dengan yang Anda lakukan untuk Linux atau macOS, tetapi bagaimana Anda mendefinisikan variabel lingkungan berbeda dari platform lain. Jalankan perintah berikut di cmd jendela untuk mengkonfigurasi dan memulai proxy lokal di sumber Anda.

```
set AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
.\localproxy -s 5555 -v 5 -r us-west-2
```

Di mana:

- -s adalah port mendengarkan sumber, yang memulai proxy lokal dalam mode sumber.
- -v adalah verbositas output, yang bisa menjadi nilai antara nol dan enam.
- -r adalah wilayah titik akhir tempat terowongan dibuka.

Untuk informasi selengkapnya tentang parameter, lihat [Opsi yang disetel menggunakan argumen baris perintah](#).

Berikut ini menunjukkan contoh output menjalankan proxy lokal dalam source mode.

```
...
...

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.secure tunneling-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
```

Menjalankan proxy lokal dalam mode tujuan

Perintah berikut menunjukkan cara menjalankan proxy lokal dalam mode tujuan.

Linux/macOS

Di Linux atau macOS, jalankan perintah berikut di terminal untuk mengonfigurasi dan memulai proxy lokal di tujuan Anda.

```
export AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
./localproxy -d 22 -v 5 -r us-west-2
```

Di mana:

- -d adalah aplikasi tujuan yang memulai proxy lokal dalam mode tujuan.
- -v adalah verbositas output, yang bisa menjadi nilai antara nol dan enam.
- -r adalah wilayah titik akhir tempat terowongan dibuka.

Untuk informasi selengkapnya tentang parameter, lihat [Opsi yang disetel menggunakan argumen baris perintah](#).

Windows

Di Windows, Anda mengonfigurasi proxy lokal yang mirip dengan yang Anda lakukan untuk Linux atau macOS, tetapi bagaimana Anda mendefinisikan variabel lingkungan berbeda dari platform lain. Jalankan perintah berikut di cmd jendela untuk mengkonfigurasi dan memulai proxy lokal di tujuan Anda.

```
set AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
.\localproxy -d 22 -v 5 -r us-west-2
```

Di mana:

- -d adalah aplikasi tujuan yang memulai proxy lokal dalam mode tujuan.
- -v adalah verbositas output, yang bisa menjadi nilai antara nol dan enam.
- -r adalah wilayah titik akhir tempat terowongan dibuka.

Untuk informasi selengkapnya tentang parameter, lihat [Opsi yang disetel menggunakan argumen baris perintah](#).

Berikut ini menunjukkan contoh output menjalankan proxy lokal dalam destination mode.

```
...
...

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.secure tunneling-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...
```

## Konfigurasi proxy lokal untuk perangkat yang menggunakan proxy web

Anda dapat menggunakan proxy lokal pada AWS IoT perangkat untuk berkomunikasi dengan API tunneling yang AWS IoT aman. Proxy lokal mentransmisikan data yang dikirim oleh aplikasi perangkat menggunakan tunneling aman melalui koneksi aman. WebSocket Proxy lokal dapat

bekerja dalam `destination mode source` atau. Dalam `source mode`, ini berjalan pada perangkat atau jaringan yang sama yang memulai koneksi TCP. Dalam `destination mode`, proxy lokal berjalan pada perangkat jarak jauh, bersama dengan aplikasi tujuan. Untuk informasi selengkapnya, lihat [Proksi lokal](#).

Proxy lokal perlu terhubung langsung ke internet untuk menggunakan tunneling yang AWS IoT aman. Untuk koneksi TCP berumur panjang dengan tunneling aman, proxy lokal memutarhkirkan permintaan HTTPS untuk membuat WebSockets koneksi ke salah satu titik akhir koneksi perangkat tunneling yang [aman](#).

Jika perangkat Anda berada dalam jaringan yang menggunakan proxy web, proxy web dapat mencegat koneksi sebelum meneruskannya ke internet. [Untuk membuat koneksi berumur panjang ke titik akhir koneksi perangkat tunneling aman, konfigurasi proxy lokal Anda untuk menggunakan proxy web seperti yang dijelaskan dalam spesifikasi websocket.](#)

#### Note

[AWS IoT Klien Perangkat](#) Tidak mendukung perangkat yang menggunakan proxy web. Untuk bekerja dengan proxy web, Anda harus menggunakan proxy lokal dan mengonfigurasinya agar berfungsi dengan proxy web seperti yang dijelaskan di bawah ini.

Langkah-langkah berikut menunjukkan bagaimana proxy lokal bekerja dengan proxy web.

1. Proxy lokal mengirimkan CONNECT permintaan HTTP ke proxy web yang berisi alamat jarak jauh dari layanan tunneling aman, bersama dengan informasi otentikasi proxy web.
2. Proxy web kemudian akan membuat koneksi berumur panjang ke titik akhir terowongan aman jarak jauh.
3. Koneksi TCP dibuat dan proxy lokal sekarang akan bekerja dalam mode sumber dan tujuan untuk transmisi data.

Untuk menyelesaikan prosedur ini, lakukan langkah-langkah berikut.

- [Membangun proxy lokal](#)
- [Konfigurasi proxy web Anda](#)
- [Konfigurasi dan mulai proxy lokal](#)

## Membangun proxy lokal

Buka [kode sumber proxy lokal](#) di GitHub repositori dan ikuti instruksi untuk membangun dan menginstal proxy lokal.

## Konfigurasi proxy web Anda

[Proxy lokal bergantung pada mekanisme tunneling HTTP yang dijelaskan oleh spesifikasi HTTP/1.1.](#)

Untuk mematuhi spesifikasi, proxy web Anda harus mengizinkan perangkat untuk menggunakan CONNECT metode ini.

Cara Anda mengonfigurasi proxy web Anda tergantung pada proxy web yang Anda gunakan dan versi proxy web. Untuk memastikan Anda mengonfigurasi proxy web dengan benar, periksa dokumentasi proxy web Anda.

Untuk mengonfigurasi proxy web Anda, pertama-tama identifikasi URL proxy web Anda dan konfirmasi apakah proxy web Anda mendukung tunneling HTTP. URL proxy web akan digunakan nanti saat Anda mengonfigurasi dan memulai proxy lokal.

### 1. Identifikasi URL proxy web Anda

URL proxy web Anda akan dalam format berikut.

```
protocol://web_proxy_host_domain:web_proxy_port
```

AWS IoT tunneling aman hanya mendukung otentikasi dasar untuk proxy web. Untuk menggunakan otentikasi dasar, Anda harus menentukan **username** dan **password** sebagai bagian dari URL proxy web. URL proxy web akan dalam format berikut.

```
protocol://username:password@web_proxy_host_domain:web_proxy_port
```

- *Protokol* bisa http atau https. Kami menyarankan Anda menggunakan https.
- *web\_proxy\_host\_domain* adalah alamat IP proxy web Anda atau nama DNS yang menyelesaikan ke alamat IP proxy web Anda.
- *web\_proxy\_port adalah port* tempat proxy web mendengarkan.
- Proxy web menggunakan ini **username** dan **password** untuk mengotentikasi permintaan.

### 2. Uji URL proxy web Anda

Untuk mengonfirmasi apakah proxy web Anda mendukung tunneling TCP, gunakan `curl` perintah dan pastikan Anda mendapatkan respons 2xx atau respons. 3xx

Misalnya, jika URL proxy web Anda `https://server.com:1235`, gunakan `proxy-insecure` tanda dengan `curl` perintah karena proxy web mungkin bergantung pada sertifikat yang ditandatangani sendiri.

```
export HTTPS_PROXY=https://server.com:1235
curl -I https://aws.amazon.com --proxy-insecure
```

Jika URL proxy web Anda memiliki http port (misalnya, `http://server.com:1234`), Anda tidak perlu menggunakan `proxy-insecure` bendera.

```
export HTTPS_PROXY=http://server.com:1234
curl -I https://aws.amazon.com
```

## Konfigurasi dan mulai proxy lokal

Untuk mengonfigurasi proxy lokal untuk menggunakan proxy web, Anda harus mengonfigurasi variabel `HTTPS_PROXY` lingkungan dengan nama domain DNS atau alamat IP dan nomor port yang digunakan proxy web Anda.

Setelah mengonfigurasi proxy lokal, Anda dapat menggunakan proxy lokal seperti yang dijelaskan dalam dokumen [README](#) ini.

### Note

Deklarasi variabel lingkungan Anda peka huruf besar/kecil. Kami menyarankan Anda mendefinisikan setiap variabel sekali menggunakan semua huruf besar atau semua huruf kecil. Contoh berikut menunjukkan variabel lingkungan dideklarasikan dalam huruf besar. Jika variabel yang sama ditentukan menggunakan huruf besar dan kecil, variabel yang ditentukan menggunakan huruf kecil diutamakan.

Perintah berikut menunjukkan cara mengonfigurasi proxy lokal yang berjalan di tujuan Anda untuk menggunakan proxy web dan memulai proxy lokal.

- `AWSIOT_TUNNEL_ACCESS_TOKEN`: Variabel ini menyimpan token akses klien (CAT) untuk tujuan.
- `HTTPS_PROXY`: Variabel ini menyimpan URL proxy web atau alamat IP untuk mengkonfigurasi proxy lokal.

Perintah yang ditampilkan dalam contoh berikut bergantung pada sistem operasi yang Anda gunakan dan apakah proxy web mendengarkan pada HTTP atau port HTTPS.

Proxy web mendengarkan pada port HTTP

Jika proxy web Anda mendengarkan pada port HTTP, Anda dapat memberikan URL proxy web atau alamat IP untuk `HTTPS_PROXY` variabel tersebut.

Linux/macOS

Di Linux atau macOS, jalankan perintah berikut di terminal untuk mengonfigurasi dan memulai proxy lokal di tujuan Anda untuk menggunakan proxy web yang mendengarkan port HTTP.

```
export AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
export HTTPS_PROXY=http:proxy.example.com:1234
./localproxy -r us-east-1 -d 22
```

Jika Anda harus mengautentikasi dengan proxy, Anda harus menentukan **username** dan **password** sebagai bagian dari `HTTPS_PROXY` variabel.

```
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

Di Windows, Anda mengonfigurasi proxy lokal yang mirip dengan yang Anda lakukan untuk Linux atau macOS, tetapi bagaimana Anda mendefinisikan variabel lingkungan berbeda dari platform lain. Jalankan perintah berikut di cmd jendela untuk mengkonfigurasi dan memulai proxy lokal di tujuan Anda untuk menggunakan proxy web mendengarkan port HTTP.

```
set AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
set HTTPS_PROXY=http://proxy.example.com:1234
.\localproxy -r us-east-1 -d 22
```



Jika Anda harus mengautentikasi dengan proxy, Anda harus menentukan **username** dan **password** sebagai bagian dari HTTPS\_PROXY variabel.

```
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
```

Proxy web mendengarkan pada port HTTPS

Jalankan perintah berikut jika proxy web Anda mendengarkan pada port HTTPS.

#### Note

Jika Anda menggunakan sertifikat yang ditandatangani sendiri untuk proxy web atau jika Anda menjalankan proxy lokal pada OS yang tidak memiliki dukungan OpenSSL asli dan konfigurasi default, Anda harus menyiapkan sertifikat proxy web Anda seperti yang dijelaskan di [bagian Penyiapan sertifikat di repositori](#). GitHub

Perintah berikut akan terlihat mirip dengan cara Anda mengonfigurasi proxy web Anda untuk proxy HTTP, dengan pengecualian bahwa Anda juga akan menentukan jalur ke file sertifikat yang Anda instal seperti yang dijelaskan sebelumnya.

Linux/macOS

Di Linux atau macOS, jalankan perintah berikut di terminal untuk mengonfigurasi proxy lokal yang berjalan di tujuan Anda untuk menggunakan proxy web yang mendengarkan port 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
```

Jika Anda harus mengautentikasi dengan proxy, Anda harus menentukan **username** dan **password** sebagai bagian dari HTTPS\_PROXY variabel.

```
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

Di Windows, jalankan perintah berikut di cmd jendela untuk mengkonfigurasi dan memulai proxy lokal yang berjalan di tujuan Anda untuk menggunakan proxy web mendengarkan port 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
```

Jika Anda harus mengautentikasi dengan proxy, Anda harus menentukan **username** dan **password** sebagai bagian dari HTTPS\_PROXY variabel.

```
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
```

### Contoh perintah dan output

Berikut ini menunjukkan contoh perintah yang Anda jalankan pada OS Linux dan output yang sesuai. Contoh menunjukkan proxy web yang mendengarkan pada port HTTP dan bagaimana proxy lokal dapat dikonfigurasi untuk menggunakan proxy web dalam kedua destination mode source dan mode. Sebelum Anda dapat menjalankan perintah ini, Anda harus sudah membuka terowongan dan memperoleh token akses klien untuk sumber dan tujuan. Anda juga harus membangun proxy lokal dan mengonfigurasi proxy web Anda seperti yang dijelaskan sebelumnya.

Berikut ikhtisar langkah-langkah setelah Anda memulai proxy lokal. Proksi lokal:

- Mengidentifikasi URL proxy web sehingga dapat menggunakan URL untuk terhubung ke server proxy.
- Menetapkan koneksi TCP dengan proxy web.
- Mengirim CONNECT permintaan HTTP ke proxy web dan menunggu HTTP/1.1 200 respons, yang menunjukkan bahwa koneksi telah dibuat.
- Upgrade protokol HTTPS WebSockets untuk membuat koneksi berumur panjang.
- Mulai mentransmisikan data melalui koneksi ke titik akhir perangkat tunneling yang aman.

**Note**

Perintah berikut yang digunakan dalam contoh menggunakan verbosity bendera untuk mengilustrasikan ikhtisar langkah-langkah berbeda yang dijelaskan sebelumnya setelah Anda menjalankan proxy lokal. Kami menyarankan Anda menggunakan bendera ini hanya untuk tujuan pengujian.

**Menjalankan proxy lokal dalam mode sumber**

Perintah berikut menunjukkan cara menjalankan proxy lokal dalam mode sumber.

```
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
```

Berikut ini menunjukkan contoh output menjalankan proxy lokal dalam source mode.

```
...

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

```

## Menjalankan proxy lokal dalam mode tujuan

Perintah berikut menunjukkan cara menjalankan proxy lokal dalam mode tujuan.

```

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

```

Berikut ini menunjukkan contoh output menjalankan proxy lokal dalam destination mode.

```

...

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: 06717bffffed3fd05-00001355-000b8315-da3109a85da804dd-24c3d10d
Web socket subprotocol selected: aws.iot.secure tunneling-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...
```

## Aliran data multipleks dan menggunakan koneksi TCP simultan di terowongan aman

Anda dapat menggunakan beberapa aliran data per terowongan dengan menggunakan fitur multiplexing tunneling aman. Dengan multiplexing, Anda dapat memecahkan masalah perangkat menggunakan beberapa aliran data. Anda juga dapat mengurangi beban operasional dengan menghilangkan kebutuhan untuk membangun, menyebarkan, dan memulai beberapa proxy lokal atau membuka beberapa terowongan ke perangkat yang sama. Misalnya, multiplexing dapat digunakan dalam kasus browser web yang memerlukan pengiriman beberapa aliran data HTTP dan SSH.

Untuk setiap aliran data, tunneling AWS IoT aman mendukung koneksi TCP simultan. Menggunakan koneksi simultan mengurangi potensi time-out jika terjadi beberapa permintaan dari klien. Misalnya, ini dapat mengurangi waktu pemuatan saat mengakses server web dari jarak jauh yang lokal ke perangkat tujuan.

Bagian berikut menjelaskan lebih lanjut tentang multiplexing dan menggunakan koneksi TCP simultan, dan kasus penggunaannya yang berbeda.

### Topik

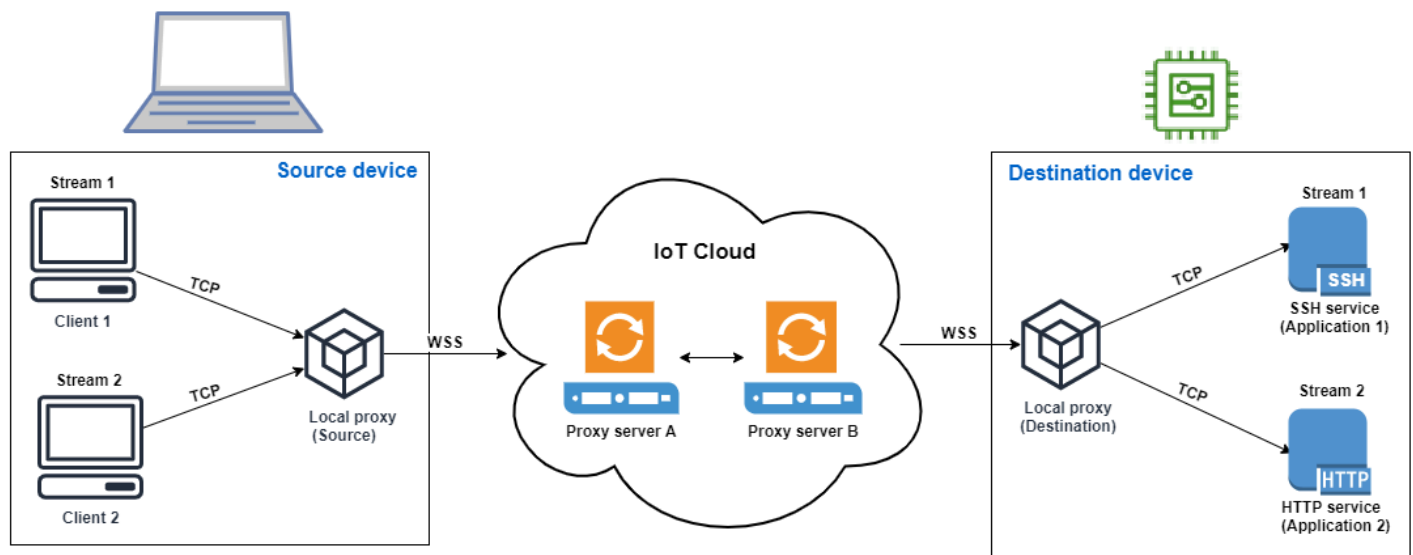
- [Multiplexing beberapa aliran data dalam terowongan aman](#)
- [Menggunakan koneksi TCP simultan di terowongan aman](#)

## Multiplexing beberapa aliran data dalam terowongan aman

Anda dapat menggunakan fitur multiplexing untuk perangkat yang menggunakan beberapa koneksi atau port. Multiplexing juga dapat digunakan ketika Anda memerlukan beberapa koneksi ke perangkat jarak jauh untuk memecahkan masalah apa pun. Misalnya, dapat digunakan dalam kasus browser web yang memerlukan pengiriman beberapa aliran data HTTP dan SSH. Data aplikasi dari kedua aliran dikirim ke perangkat secara bersamaan melalui terowongan multipleks.

## Contoh kasus penggunaan

Katakanlah Anda perlu terhubung ke aplikasi web di perangkat untuk mengubah beberapa parameter jaringan, sekaligus mengeluarkan perintah shell melalui terminal untuk memverifikasi bahwa perangkat berfungsi dengan baik dengan parameter jaringan baru. Dalam skenario ini, Anda mungkin perlu terhubung ke perangkat melalui HTTP dan SSH dan mentransfer dua aliran data paralel untuk mengakses aplikasi web dan terminal secara bersamaan. Dengan fitur multiplexing, kedua aliran independen ini dapat ditransfer melalui terowongan yang sama pada saat yang bersamaan.



## Cara mengatur terowongan multipleks

Prosedur berikut memandu Anda melalui cara mengatur terowongan multipleks untuk perangkat pemecahan masalah menggunakan aplikasi yang memerlukan koneksi ke beberapa port. Anda akan mengatur satu terowongan dengan dua aliran multipleks: satu aliran HTTP dan satu aliran SSH.

### 1. (Opsional) Buat file konfigurasi

Anda dapat secara opsional mengonfigurasi perangkat sumber dan tujuan dengan file konfigurasi. Gunakan file konfigurasi jika pemetaan port Anda cenderung sering berubah. Anda dapat melewati langkah ini jika Anda lebih suka menentukan pemetaan port secara eksplisit menggunakan CLI, atau jika Anda tidak perlu memulai proxy lokal pada port mendengarkan yang ditunjuk. Untuk informasi selengkapnya tentang cara menggunakan file konfigurasi, lihat [Opsional yang disetel melalui --config](#) di GitHub

1. Di perangkat sumber Anda, di folder tempat proxy lokal Anda akan berjalan, buat folder konfigurasi bernama `Config`. Di dalam folder ini, buat file yang disebut `SSHSource.ini` dengan konten berikut:

```
HTTP1 = 5555
SSH1 = 3333
```

2. Di perangkat tujuan Anda, di folder tempat proxy lokal Anda akan berjalan, buat folder konfigurasi bernama `Config`. Di dalam folder ini, buat file yang disebut `SSHDestination.ini` dengan konten berikut:

```
HTTP1 = 80
SSH1 = 22
```

## 2. Buka terowongan

Buka terowongan menggunakan operasi `OpenTunnel` API atau perintah `open-tunnel` CLI. Konfigurasikan tujuan dengan menentukan `SSH1` dan `HTTP1` sebagai layanan dan nama AWS IoT benda yang sesuai dengan perangkat jarak jauh Anda. Aplikasi SSH dan HTTP Anda berjalan di perangkat jarak jauh ini. Anda pasti sudah membuat IoT di registri. AWS IoT Untuk informasi selengkapnya, lihat [Cara mengelola sesuatu dengan registri](#).

```
aws iotsecuretunneling open-tunnel \
--destination-config thingName=RemoteDevice1,services=HTTP1,SSH1
```

Menjalankan perintah ini menghasilkan token akses sumber dan tujuan yang akan Anda gunakan untuk menjalankan proxy lokal.

```
{
 "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. Konfigurasi dan mulai proxy lokal

Sebelum Anda dapat menjalankan proxy lokal, siapkan AWS IoT Device Client, atau unduh kode sumber proxy lokal dari [GitHub](#) dan buat untuk platform pilihan Anda. Anda kemudian dapat memulai tujuan dan proxy lokal sumber untuk terhubung ke terowongan aman. Untuk informasi selengkapnya tentang mengonfigurasi dan menggunakan proxy lokal, lihat [Cara menggunakan proxy lokal](#).

#### Note

Di perangkat sumber Anda, jika Anda tidak menggunakan file konfigurasi apa pun atau menentukan pemetaan port menggunakan CLI, Anda masih dapat menggunakan perintah yang sama untuk menjalankan proxy lokal. Proxy lokal dalam mode sumber akan secara otomatis mengambil port yang tersedia untuk digunakan dan pemetaan untuk Anda.

#### Start local proxy using configuration files

Jalankan perintah berikut untuk menjalankan proxy lokal dalam mode sumber dan tujuan menggunakan file konfigurasi.

```
// ----- 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

Jalankan perintah berikut untuk menjalankan proxy lokal dalam mode sumber dan tujuan dengan menentukan pemetaan port secara eksplisit menggunakan CLI.

```
// ----- Start the destination local proxy

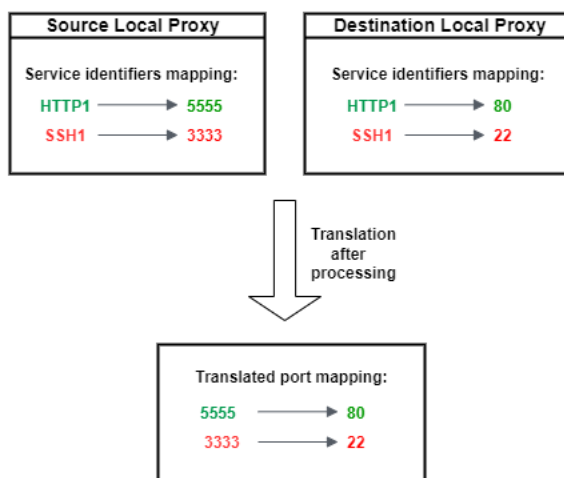
./localproxy -r us-east-1 -d HTTP1=80,SSH1=22 -t destination_client_access_token
```



```
// ----- Start the source local proxy

./localproxy -r us-east-1 -s HTTP1=5555,SSH1=33 -t source_client_access_token
```

Data aplikasi dari koneksi SSH dan HTTP sekarang dapat ditransfer secara bersamaan melalui terowongan multiplex. Seperti yang terlihat pada peta di bawah ini, pengenalan layanan bertindak sebagai format yang dapat dibaca untuk menerjemahkan pemetaan port antara perangkat sumber dan tujuan. *Dengan konfigurasi ini, tunneling aman meneruskan lalu lintas HTTP yang masuk dari port 5555 pada perangkat sumber ke port 80 pada perangkat tujuan, dan setiap lalu lintas SSH yang masuk dari port 3333 ke port 22 pada perangkat tujuan.*



## Menggunakan koneksi TCP simultan di terowongan aman

AWS IoT tunneling aman mendukung lebih dari satu koneksi TCP secara bersamaan untuk setiap aliran data. Anda dapat menggunakan kemampuan ini ketika Anda memerlukan koneksi simultan ke perangkat jarak jauh. Menggunakan koneksi TCP simultan mengurangi potensi time-out jika terjadi beberapa permintaan dari klien. Misalnya, saat mengakses server web yang memiliki beberapa komponen yang berjalan di atasnya, koneksi TCP simultan dapat mengurangi waktu yang diperlukan untuk memuat situs.

### Note

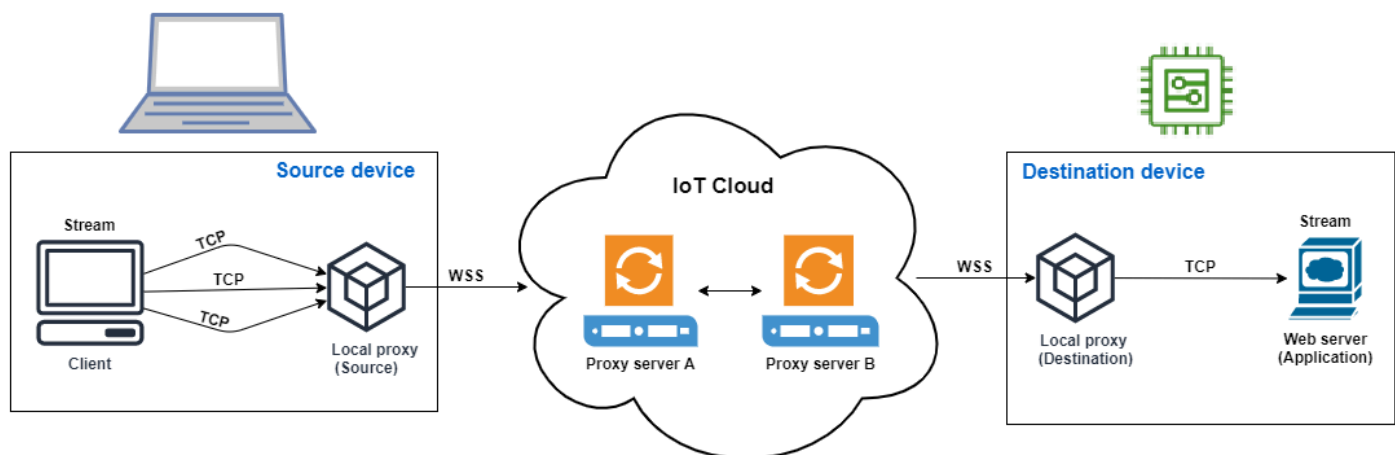
Koneksi TCP simultan memiliki batas bandwidth 800 Kilobyte per detik untuk masing-masing. Akun AWS IoT tunneling aman dapat mengonfigurasi batas ini untuk Anda tergantung pada jumlah permintaan yang masuk.

## Contoh kasus penggunaan

Katakanlah Anda perlu mengakses server web dari jarak jauh yang lokal ke perangkat tujuan dan memiliki beberapa komponen yang berjalan di atasnya. Dengan koneksi TCP tunggal, saat mencoba mengakses server web, pemuatan berurutan dapat meningkatkan jumlah waktu yang diperlukan untuk memuat sumber daya di situs. Koneksi TCP simultan dapat mengurangi waktu pemuatan dengan memenuhi persyaratan sumber daya situs, sehingga mengurangi waktu akses. Diagram berikut menunjukkan bagaimana koneksi TCP simultan didukung untuk aliran data ke aplikasi server web yang berjalan pada perangkat jarak jauh.

### Note

Jika Anda ingin mengakses beberapa aplikasi yang berjalan di perangkat jarak jauh menggunakan terowongan, Anda dapat menggunakan multiplexing terowongan. Untuk informasi selengkapnya, lihat [Multiplexing beberapa aliran data dalam terowongan aman](#).



## Cara menggunakan koneksi TCP simultan

Prosedur berikut memandu Anda melalui cara menggunakan koneksi TCP simultan untuk mengakses browser web pada perangkat jarak jauh. Ketika ada beberapa permintaan dari klien, tunneling AWS IoT aman secara otomatis mengatur koneksi TCP simultan untuk menangani permintaan, sehingga mengurangi waktu pemuatan.

## 1. Buka terowongan

Buka terowongan menggunakan operasi OpenTunnel API atau perintah `open-tunnel` CLI. Konfigurasi tujuan dengan menentukan HTTP sebagai layanan dan nama AWS IoT benda yang sesuai dengan perangkat jarak jauh Anda. Aplikasi server web Anda berjalan di perangkat jarak jauh ini. Anda pasti sudah membuat IoT di registri. AWS IoT Untuk informasi selengkapnya, lihat [Cara mengelola sesuatu dengan registri](#).

```
aws iotsecuretunneling open-tunnel \
 --destination-config thingName=RemoteDevice1,services=HTTP
```

Menjalankan perintah ini menghasilkan token akses sumber dan tujuan yang akan Anda gunakan untuk menjalankan proxy lokal.

```
{
 "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. Konfigurasi dan mulai proxy lokal

Sebelum Anda dapat menjalankan proxy lokal, unduh kode sumber proxy lokal dari [GitHub](#) dan buat untuk platform pilihan Anda. Anda kemudian dapat memulai tujuan dan proxy lokal sumber untuk terhubung ke terowongan aman dan mulai menggunakan aplikasi server web jarak jauh.

### Note

Agar tunneling AWS IoT aman menggunakan koneksi TCP simultan, Anda harus meningkatkan ke versi terbaru dari proxy lokal. Fitur ini tidak tersedia jika Anda mengonfigurasi proxy lokal menggunakan AWS IoT Device Client.

```
// 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
```

Untuk informasi selengkapnya tentang mengonfigurasi dan menggunakan proxy lokal, lihat [Cara menggunakan proxy lokal](#).

Anda sekarang dapat menggunakan terowongan untuk mengakses aplikasi server web. AWS IoT tunneling aman akan secara otomatis mengatur dan menangani koneksi TCP simultan ketika ada beberapa permintaan dari klien.

## Mengkonfigurasi perangkat jarak jauh dan menggunakan agen IoT

Agan IoT digunakan untuk menerima pesan MQTT yang menyertakan token akses klien dan memulai proxy lokal pada perangkat jarak jauh. Anda harus menginstal dan menjalankan agen IoT pada perangkat jarak jauh jika Anda ingin tunneling aman untuk mengirimkan token akses klien menggunakan MQTT. Agan IoT harus berlangganan topik IoT MQTT yang dipesan berikut:

### Note

Jika Anda ingin mengirimkan token akses klien tujuan ke perangkat jarak jauh melalui metode selain berlangganan topik MQTT yang dicadangkan, Anda mungkin memerlukan pendengar token akses klien tujuan (CAT) dan proxy lokal. Pendengar CAT harus bekerja dengan mekanisme pengiriman token akses klien yang Anda pilih dan dapat memulai proxy lokal dalam mode tujuan.

## Cuplikan agen IoT

Agan IoT harus berlangganan topik IoT MQTT yang dicadangkan berikut sehingga dapat menerima pesan MQTT dan memulai proxy lokal:

```
$aws/things/thing-name/tunnels/notify
```

Di *thing-name* mana nama AWS IoT benda yang terkait dengan perangkat jarak jauh.

Berikut ini adalah contoh payload pesan MQTT:

```
{
```

```
"clientAccessToken": "destination-client-access-token",
"clientMode": "destination",
"region": "aws-region",
"services": ["destination-service"]
}
```

Setelah menerima pesan MQTT, agen IoT harus memulai proxy lokal pada perangkat jarak jauh dengan parameter yang sesuai.

Kode Java berikut menunjukkan cara menggunakan [AWS IoT Device SDK](#) dan [ProcessBuilder](#) dari pustaka Java untuk membangun agen IoT sederhana untuk bekerja dengan tunneling aman.

```
// Find the IoT device endpoint for your Akun 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);
 }
}
}
```

## Mengontrol akses ke terowongan

Tunneling aman menyediakan tindakan, sumber daya, dan kunci konteks kondisi khusus layanan untuk digunakan dalam kebijakan izin IAM.

## Prasyarat akses terowongan

- Pelajari cara mengamankan AWS sumber daya dengan menggunakan [kebijakan IAM](#).

- Pelajari cara membuat dan mengevaluasi [kondisi IAM](#).
- Pelajari cara mengamankan AWS sumber daya menggunakan [tag sumber daya](#).

## Kebijakan akses terowongan

Anda harus menggunakan kebijakan berikut untuk mengotorisasi izin untuk menggunakan API tunneling aman. Untuk informasi lebih lanjut tentang AWS IoT keamanan, lihat [Identitas dan manajemen akses untuk AWS IoT](#).

IOT: OpenTunnel

Tindakan `iot:OpenTunnel` kebijakan memberikan izin utama untuk menelepon [OpenTunnel](#).

Dalam Resource elemen pernyataan kebijakan IAM:

- Tentukan terowongan wildcard ARN:

```
arn:aws:iot:aws-region:aws-account-id:tunnel/*
```

- Tentukan hal ARN untuk mengelola OpenTunnel izin untuk hal-hal IoT tertentu:

```
arn:aws:iot:aws-region:aws-account-id:thing/thing-name
```

Misalnya, pernyataan kebijakan berikut memungkinkan Anda untuk membuka terowongan ke benda IoT bernama `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"
]
}
```

Tindakan `iot:OpenTunnel` kebijakan mendukung kunci kondisi berikut:

- `iot:ThingGroupArn`
- `iot:TunnelDestinationService`
- `aws:RequestTag/tombol tag`

- `aws:SecureTransport`
- `aws:TagKeys`

Pernyataan kebijakan berikut memungkinkan Anda untuk membuka terowongan ke benda tersebut jika benda itu milik grup benda dengan nama yang dimulai dengan `TestGroup` dan layanan tujuan yang dikonfigurasi pada terowongan adalah 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"
]
 }
 }
}
```

Anda juga dapat menggunakan tag sumber daya untuk mengontrol izin membuka terowongan. Misalnya, pernyataan kebijakan berikut memungkinkan terowongan dibuka jika kunci tag `Owner` hadir dengan nilai `Admin` dan tidak ada tag lain yang ditentukan. Untuk informasi umum tentang penggunaan tag, lihat [Menandai sumber daya Anda 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

Tindakan `iot:RotateTunnelAccessToken` kebijakan memberikan izin utama untuk menelepon [RotateTunnelAccessToken](#).

Dalam Resource elemen pernyataan kebijakan IAM:

- Tentukan ARN terowongan yang sepenuhnya memenuhi syarat:

```
arn:aws:iot:aws-region: aws-account-id:tunnel/tunnel-id
```

Anda juga dapat menggunakan terowongan wildcard ARN:

```
arn:aws:iot:aws-region:aws-account-id:tunnel/*
```

- Tentukan hal ARN untuk mengelola RotateTunnelAccessToken izin untuk hal-hal IoT tertentu:

```
arn:aws:iot:aws-region:aws-account-id:thing/thing-name
```

Misalnya, pernyataan kebijakan berikut memungkinkan Anda untuk memutar token akses sumber terowongan atau token akses tujuan klien untuk hal IoT bernama. `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"
]
}

```

Tindakan `iot:RotateTunnelAccessToken` kebijakan mendukung kunci kondisi berikut:

- `iot:ThingGroupArn`
- `iot:TunnelDestinationService`

- `iot:ClientMode`
- `aws:SecureTransport`

Pernyataan kebijakan berikut memungkinkan Anda untuk memutar token akses tujuan ke benda tersebut jika benda itu milik grup benda dengan nama yang dimulai dengan `TestGroup`, layanan tujuan yang dikonfigurasi pada terowongan adalah SSH, dan klien dalam `DESTINATION` mode.

```
{
 "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

Tindakan `iot:DescribeTunnel` kebijakan memberikan izin utama untuk menelepon [DescribeTunnel](#).

Dalam `Resource` elemen pernyataan kebijakan IAM, tentukan ARN terowongan yang sepenuhnya memenuhi syarat:

```
arn:aws:iot:aws-region: aws-account-id:tunnel/tunnel-id
```

Anda juga dapat menggunakan ARN wildcard:

```
arn:aws:iot:aws-region:aws-account-id:tunnel/*
```

Tindakan `iot:DescribeTunnel` kebijakan mendukung kunci kondisi berikut:

- `aws:ResourceTag/tag-key`
- `aws:SecureTransport`

Pernyataan kebijakan berikut memungkinkan Anda untuk memanggil `DescribeTunnel` jika terowongan yang diminta ditandai dengan kunci `Owner` dengan nilai `Admin`

```
{
 "Effect": "Allow",
 "Action": "iot:DescribeTunnel",
 "Resource": [
 "arn:aws:iot:aws-region:aws-account-id:tunnel/*"
],
 "Condition": {
 "StringEquals": {
 "aws:ResourceTag/Owner": "Admin"
 }
 }
}
```

## IOT: ListTunnels

Tindakan `iot:ListTunnels` kebijakan memberikan izin utama untuk menelepon [ListTunnels](#).

Dalam `Resource` elemen pernyataan kebijakan IAM:

- Tentukan terowongan wildcard ARN:  
`arn:aws:iot:aws-region:aws-account-id:tunnel/*`
- Tentukan hal ARN untuk mengelola `ListTunnels` izin pada hal-hal IoT yang dipilih:  
`arn:aws:iot:aws-region:aws-account-id:thing/thing-name`

Tindakan `iot:ListTunnels` kebijakan mendukung kunci kondisi `aws:SecureTransport`.

Pernyataan kebijakan berikut memungkinkan Anda untuk daftar terowongan untuk hal yang bernama `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

Tindakan `iot:ListTagsForResource` kebijakan memberikan izin utama untuk menelepon `ListTagsForResource`.

Dalam `Resource` elemen pernyataan kebijakan IAM, tentukan ARN terowongan yang sepenuhnya memenuhi syarat:

```
arn:aws:iot:aws-region: aws-account-id:tunnel/tunnel-id
```

Anda juga dapat menggunakan terowongan wildcard ARN:

```
arn:aws:iot:aws-region:aws-account-id:tunnel/*
```

Tindakan `iot:ListTagsForResource` kebijakan mendukung kunci kondisi `aws:SecureTransport`.

## IOT: CloseTunnel

Tindakan `iot:CloseTunnel` kebijakan memberikan izin utama untuk menelepon [CloseTunnel](#).

Dalam `Resource` elemen pernyataan kebijakan IAM, tentukan ARN terowongan yang sepenuhnya memenuhi syarat:

```
arn:aws:iot:aws-region: aws-account-id:tunnel/tunnel-id
```

Anda juga dapat menggunakan terowongan wildcard ARN:

```
arn:aws:iot:aws-region:aws-account-id:tunnel/*
```

Tindakan `iot:CloseTunnel` kebijakan mendukung kunci kondisi berikut:

- `iot>Delete`
- `aws:ResourceTag/tag-key`
- `aws:SecureTransport`

Pernyataan kebijakan berikut memungkinkan Anda untuk memanggil `CloseTunnel` jika `Delete` parameter permintaan adalah `false` dan permintaan ditandai dengan kunci `Owner` dengan nilai `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

Tindakan `iot:TagResource` kebijakan memberikan izin utama untuk menelepon `TagResource`.

Dalam `Resource` elemen pernyataan kebijakan IAM, tentukan ARN terowongan yang sepenuhnya memenuhi syarat:

```
arn:aws:iot:aws-region: aws-account-id:tunnel/tunnel-id
```

Anda juga dapat menggunakan terowongan wildcard ARN:

```
arn:aws:iot:aws-region:aws-account-id:tunnel/*
```

Tindakan `iot:TagResource` kebijakan mendukung kunci kondisi `aws:SecureTransport`.

## IOT: UntagResource

Tindakan `iot:UntagResource` kebijakan memberikan izin utama untuk menelepon `UntagResource`.

Dalam `Resource` elemen pernyataan kebijakan IAM, tentukan ARN terowongan yang sepenuhnya memenuhi syarat:

```
arn:aws:iot:aws-region: aws-account-id:tunnel/tunnel-id
```

Anda juga dapat menggunakan terowongan wildcard ARN:

```
arn:aws:iot:aws-region:aws-account-id:tunnel/*
```

Tindakan `iot:UntagResource` kebijakan mendukung kunci kondisi `aws:SecureTransport`.

## Menyelesaikan masalah konektivitas tunneling yang AWS IoT aman dengan memutar token akses klien

Saat Anda menggunakan tunneling AWS IoT aman, Anda mungkin mengalami masalah konektivitas meskipun terowongan terbuka. Bagian berikut menunjukkan beberapa kemungkinan masalah dan bagaimana Anda dapat menyelesaikannya dengan memutar token akses klien. Untuk memutar token akses klien (CAT), gunakan [RotateTunnelAccessToken](#) API atau [rotate-tunnel-access-token](#) AWS CLI. Bergantung pada apakah Anda mengalami kesalahan dengan menggunakan klien dalam mode sumber atau tujuan, Anda dapat memutar CAT baik dalam mode sumber atau tujuan, atau keduanya.

### Note

- Jika Anda tidak yakin apakah CAT perlu diputar pada sumber atau tujuan, Anda dapat memutar CAT pada sumber dan tujuan dengan menyetel `ClientMode` ke `ALL` saat menggunakan `RotateTunnelAccessToken` API.
- Memutar CAT tidak memperpanjang durasi terowongan. Misalnya, katakanlah durasi terowongan adalah 12 jam dan terowongan sudah buka selama 4 jam. Saat Anda memutar token akses, token baru yang dihasilkan hanya dapat digunakan selama 8 jam tersisa.

### Topik

- [Kesalahan token akses klien tidak valid](#)
- [Kesalahan ketidakcocokan token klien](#)
- [Masalah konektivitas perangkat jarak jauh](#)

## Kesalahan token akses klien tidak valid

Saat menggunakan tunneling AWS IoT aman, Anda dapat mengalami kesalahan koneksi saat menggunakan token akses klien (CAT) yang sama untuk menyambung kembali ke terowongan yang

sama. Dalam hal ini, proxy lokal tidak dapat terhubung ke server proxy tunneling yang aman. Jika Anda menggunakan klien dalam mode sumber, Anda mungkin melihat pesan galat berikut:

```
Invalid access token: The access token was previously used and cannot be used again
```

Kesalahan terjadi karena token akses klien (CAT) hanya dapat digunakan sekali oleh proxy lokal, dan kemudian menjadi tidak valid. Untuk mengatasi kesalahan ini, putar token akses klien dalam SOURCE mode untuk menghasilkan CAT baru untuk sumbernya. Untuk contoh yang menunjukkan cara memutar CAT sumber, lihat [Putar contoh CAT sumber](#).

## Kesalahan ketidakcocokan token klien

### Note

Menggunakan token klien untuk menggunakan kembali CAT tidak disarankan. Kami menyarankan Anda menggunakan `RotateTunnelAccessToken` API sebagai gantinya untuk memutar token akses klien untuk menyambung kembali ke terowongan.

Jika Anda menggunakan token klien, Anda dapat menggunakan kembali CAT untuk menyambung kembali ke terowongan. Untuk menggunakan kembali CAT, Anda harus memberikan token klien dengan CAT saat pertama kali Anda terhubung ke tunneling yang aman. Tunneling aman menyimpan token klien sehingga untuk upaya koneksi berikutnya menggunakan token yang sama, token klien juga harus disediakan. Untuk informasi selengkapnya tentang penggunaan token klien, lihat [implementasi referensi proxy lokal di GitHub](#).

Saat menggunakan token klien, jika Anda menggunakan klien dalam mode sumber, Anda mungkin melihat kesalahan berikut:

```
Invalid client token: The provided client token does not match the client token that was previously set.
```

Kesalahan terjadi karena token klien yang disediakan tidak cocok dengan token klien yang disediakan dengan CAT saat mengakses terowongan. Untuk mengatasi kesalahan ini, putar CAT dalam SOURCE mode untuk menghasilkan CAT baru untuk sumbernya. Berikut ini menunjukkan contoh:

## Putar contoh CAT sumber

Berikut ini menunjukkan contoh cara menjalankan RotateTunnelAccessToken API dalam SOURCE mode untuk menghasilkan CAT baru untuk sumbernya:

```
aws iotsecuretunneling rotate-tunnel-access-token \
 --region <region> \
 --tunnel-id <tunnel-id> \
 --client-mode SOURCE
```

Menjalankan perintah ini menghasilkan token akses sumber baru dan mengembalikan ARN terowongan Anda.

```
{
 "sourceAccessToken": "<source-access-token>",
 "tunnelArn": "arn:aws:iot:<region>:<account-id>tunnel/<tunnel-id>"
}
```

Anda sekarang dapat menggunakan token sumber baru untuk menghubungkan proxy lokal dalam mode sumber.

```
export AWSIOT_TUNNEL_ACCESS_TOKEN=<source-access-token>
./localproxy -r <region> -s <port>
```

Berikut ini menunjukkan contoh output menjalankan proxy lokal:

```
...
[info] Starting proxy in source mode
...
[info] Successfully established websocket connection with proxy server ...
[info] Listening for new connection on port <port>
...
```

## Masalah konektivitas perangkat jarak jauh

Saat menggunakan tunneling AWS IoT aman, perangkat mungkin terputus secara tidak terduga bahkan jika terowongan terbuka. Untuk mengidentifikasi apakah perangkat masih terhubung ke terowongan, Anda dapat menggunakan [DescribeTunnel](#) API atau terowongan [deskripsikan](#). AWS CLI



Perangkat dapat terputus karena berbagai alasan. Untuk mengatasi masalah konektivitas, Anda dapat memutar CAT di tujuan jika perangkat terputus karena kemungkinan alasan berikut:

- CAT di tempat tujuan menjadi tidak valid.
- Token tidak dikirim ke perangkat melalui topik MQTT yang dicadangkan terowongan aman:

```
$aws/things/<thing-name>/tunnels/notify
```

Contoh berikut menunjukkan cara mengatasi masalah ini:

Putar contoh CAT tujuan

Pertimbangkan perangkat jarak jauh, *<RemoteThing1>*. Untuk membuka terowongan untuk hal itu, Anda dapat menggunakan perintah berikut:

```
aws iotsecuretunneling open-tunnel \
 --region <region> \
 --destination-config thingName=<RemoteThing1>,services=SSH
```

Menjalankan perintah ini menghasilkan detail terowongan dan CAT untuk sumber dan tujuan Anda.

```
{
 "sourceAccessToken": "<source-access-token>",
 "destinationAccessToken": "<destination-access-token>",
 "tunnelId": "<tunnel-id>",
 "tunnelArn": "arn:aws:iot:<region>:<account-id>:tunnel/<tunnel-id>"
}
```

Namun, saat Anda menggunakan [DescribeTunnel](#) API, output menunjukkan bahwa perangkat telah terputus, seperti yang diilustrasikan di bawah ini:

```
aws iotsecuretunneling describe-tunnel \
 --tunnel-id <tunnel-id> \
 --region <region>
```

Menjalankan perintah ini menampilkan bahwa perangkat masih tidak terhubung.

```
{
 "tunnel": {
```

```
...
 "destinationConnectionState": {
 "status": "DISCONNECTED"
 },
 ...
}
```

Untuk mengatasi kesalahan ini, jalankan `RotateTunnelAccessToken` API dengan klien dalam `DESTINATION` mode dan konfigurasi untuk tujuan. Menjalankan perintah ini akan mencabut token akses lama, menghasilkan token baru, dan mengirim ulang token ini ke topik 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>
```

Menjalankan perintah ini menghasilkan token akses baru seperti yang ditunjukkan di bawah ini. Token kemudian dikirim ke perangkat untuk terhubung ke terowongan, jika agen perangkat diatur dengan benar.

```
{
 "destinationAccessToken": "destination-access-token",
 "tunnelArn": "arn:aws:iot:region:account-id:tunnel/tunnel-id"
}
```

# Penyediaan perangkat

AWS menyediakan beberapa cara berbeda untuk menyediakan perangkat dan menginstal sertifikat klien unik di dalamnya. Bagian ini menjelaskan setiap cara dan cara memilih yang terbaik untuk solusi IoT Anda. Opsi ini dijelaskan secara rinci dalam whitepaper berjudul [Device Manufacturing and Provisioning with X.509 Certificates](#) in. AWS IoT Core

Pilih opsi yang paling sesuai dengan situasi Anda

- Anda dapat menginstal sertifikat di perangkat IoT sebelum dikirimkan

[Jika Anda dapat menginstal sertifikat klien unik dengan aman di perangkat IoT Anda sebelum dikirimkan untuk digunakan oleh pengguna akhir, Anda ingin menggunakan just-in-time provisioning \(JITP\) atau just-in-time registration \(JITR\).](#)

Menggunakan JITP dan JITR, otoritas sertifikat (CA) yang digunakan untuk menandatangani sertifikat perangkat terdaftar AWS IoT dan dikenali oleh AWS IoT saat perangkat pertama kali terhubung. Perangkat disediakan AWS IoT pada koneksi pertamanya menggunakan detail template penyediaannya.

Untuk informasi lebih lanjut tentang satu hal, JITP, JITR, dan penyediaan massal perangkat yang memiliki sertifikat unik, lihat. [the section called “Penyediaan perangkat yang memiliki sertifikat perangkat”](#)

- Pengguna akhir atau penginstal dapat menggunakan aplikasi untuk menginstal sertifikat di perangkat IoT mereka

Jika Anda tidak dapat menginstal sertifikat klien unik dengan aman di perangkat IoT Anda sebelum dikirimkan ke pengguna akhir, tetapi pengguna akhir atau penginstal dapat menggunakan aplikasi untuk mendaftarkan perangkat dan menginstal sertifikat perangkat unik, Anda ingin menggunakan [penyediaan oleh proses pengguna tepercaya](#).

Menggunakan pengguna tepercaya, seperti pengguna akhir atau penginstal dengan akun yang dikenal, dapat menyederhanakan proses pembuatan perangkat. Alih-alih sertifikat klien yang unik, perangkat memiliki sertifikat sementara yang memungkinkan perangkat terhubung hanya AWS IoT selama 5 menit. Selama jendela 5 menit itu, pengguna tepercaya memperoleh sertifikat klien unik dengan masa pakai yang lebih lama dan menginstalnya di perangkat. Masa pakai sertifikat klaim yang terbatas meminimalkan risiko sertifikat yang dikompromikan.

Untuk informasi selengkapnya, lihat [the section called “Penyediaan oleh pengguna tepercaya”](#).

- Pengguna akhir TIDAK DAPAT menggunakan aplikasi untuk menginstal sertifikat di perangkat IoT mereka

Jika tidak satu pun dari opsi sebelumnya akan berfungsi dalam solusi IoT Anda, [penyediaan melalui proses klaim](#) adalah opsi. Dengan proses ini, perangkat IoT Anda memiliki sertifikat klaim yang dibagikan oleh perangkat lain di armada. Pertama kali perangkat terhubung dengan sertifikat klaim, AWS IoT mendaftarkan perangkat menggunakan templat penyediaannya dan mengeluarkan perangkat sertifikat klien uniknya untuk akses selanjutnya. AWS IoT

Opsi ini memungkinkan penyediaan otomatis perangkat saat terhubung AWS IoT, tetapi dapat menimbulkan risiko yang lebih besar jika terjadi sertifikat klaim yang dikompromikan. Jika sertifikat klaim dikompromikan, Anda dapat menonaktifkan sertifikat. Menonaktifkan sertifikat klaim mencegah semua perangkat dengan sertifikat klaim tersebut terdaftar di masa mendatang. Namun; menonaktifkan sertifikat klaim tidak memblokir perangkat yang telah disediakan.

Untuk informasi selengkapnya, lihat [the section called “Penyediaan dengan klaim”](#).

## Penyediaan perangkat di AWS IoT

Saat menyediakan perangkat AWS IoT, Anda harus membuat sumber daya agar perangkat Anda dan AWS IoT dapat berkomunikasi dengan aman. Sumber daya lain dapat dibuat untuk membantu Anda mengelola armada perangkat Anda. Sumber daya berikut dapat dibuat selama proses penyediaan:

- Sesuatu yang IoT.

Hal-hal IoT adalah entri dalam registri perangkat. AWS IoT Setiap benda memiliki nama dan set atribut yang unik, dan dikaitkan dengan perangkat fisik. Hal-hal dapat didefinisikan menggunakan tipe benda atau dikelompokkan ke dalam kelompok benda. Untuk informasi selengkapnya, lihat [Mengelola perangkat dengan AWS IoT](#).

Meskipun tidak diperlukan, membuat sesuatu memungkinkan untuk mengelola armada perangkat Anda secara lebih efektif dengan mencari perangkat berdasarkan jenis benda, grup benda, dan atribut benda. Untuk informasi selengkapnya, lihat [Pengindeksan armada](#).

**Note**

Untuk mengindeks data status konektivitas Thing Anda, sediakan Thing Anda dan konfigurasi agar nama Thing cocok dengan ID klien yang digunakan pada permintaan Connect.

- Sertifikat X.509.

Perangkat menggunakan sertifikat X.509 untuk melakukan otentikasi timbal balik dengan AWS IoT. Anda dapat mendaftarkan sertifikat yang ada atau telah AWS IoT menghasilkan dan mendaftarkan sertifikat baru untuk Anda. Anda mengaitkan sertifikat dengan perangkat dengan melampirkannya ke benda yang mewakili perangkat. Anda juga harus menyalin sertifikat dan kunci pribadi terkait ke perangkat. Perangkat menyajikan sertifikat saat menghubungkan ke AWS IoT. Untuk informasi selengkapnya, lihat [Autentikasi](#).

- Kebijakan IoT.

Kebijakan IoT menentukan operasi yang dapat dilakukan perangkat. AWS IoT Kebijakan IoT dilampirkan ke sertifikat perangkat. Saat perangkat menunjukkan sertifikat tersebut AWS IoT, maka akan diberikan izin yang ditentukan dalam kebijakan. Untuk informasi selengkapnya, lihat [Otorisasi](#). Setiap perangkat membutuhkan sertifikat untuk berkomunikasi AWS IoT.

AWS IoT mendukung penyediaan armada otomatis menggunakan templat penyediaan. Templat penyediaan menjelaskan sumber daya yang AWS IoT diperlukan untuk menyediakan perangkat Anda. Template berisi variabel yang memungkinkan Anda menggunakan satu template untuk menyediakan beberapa perangkat. Saat menyediakan perangkat, Anda menentukan nilai untuk variabel khusus perangkat menggunakan kamus atau peta. Untuk menyediakan perangkat lain, tentukan nilai baru dalam kamus.

Anda dapat menggunakan penyediaan otomatis apakah perangkat Anda memiliki sertifikat unik (dan kunci pribadi terkait) atau tidak.

## API penyediaan armada

Ada beberapa kategori API yang digunakan dalam penyediaan armada:

- Fungsi bidang kontrol ini membuat dan mengelola templat penyediaan armada dan mengonfigurasi kebijakan pengguna tepercaya.

- [CreateProvisioningTemplate](#)
  - [CreateProvisioningTemplateVersion](#)
  - [DeleteProvisioningTemplate](#)
  - [DeleteProvisioningTemplateVersion](#)
  - [DescribeProvisioningTemplate](#)
  - [DescribeProvisioningTemplateVersion](#)
  - [ListProvisioningTemplate](#)
  - [ListProvisioningTemplateVersions](#)
  - [UpdateProvisioningTemplate](#)
- Pengguna tepercaya dapat menggunakan fungsi bidang kontrol ini untuk menghasilkan klaim orientasi sementara. Klaim sementara ini diteruskan ke perangkat selama konfigurasi Wi-Fi atau metode serupa.
- [CreateProvisioningKlaim](#)
- MQTT API yang digunakan selama proses penyediaan oleh perangkat dengan sertifikat klaim penyediaan yang disematkan di perangkat, atau diteruskan ke perangkat oleh pengguna tepercaya.
- [the section called “CreateCertificateFromCsr”](#)
  - [the section called “CreateKeysAndCertificate”](#)
  - [the section called “RegisterThing”](#)

## Penyediaan perangkat yang tidak memiliki sertifikat perangkat menggunakan penyediaan armada

Dengan menggunakan penyediaan AWS IoT armada, AWS IoT dapat menghasilkan dan mengirimkan sertifikat perangkat dan kunci pribadi dengan aman ke perangkat Anda saat mereka terhubung AWS IoT untuk pertama kalinya. AWS IoT menyediakan sertifikat klien yang ditandatangani oleh otoritas sertifikat Amazon Root (CA).

Ada dua cara untuk menggunakan penyediaan armada:

- [Penyediaan dengan klaim](#)
- [Penyediaan oleh pengguna tepercaya](#)

## Penyediaan dengan klaim

Perangkat dapat diproduksi dengan sertifikat klaim penyediaan dan kunci pribadi (yang merupakan kredensial tujuan khusus) yang tertanam di dalamnya. Jika sertifikat ini terdaftar AWS IoT, layanan dapat menukarnya dengan sertifikat perangkat unik yang dapat digunakan perangkat untuk operasi reguler. Proses ini mencakup langkah-langkah berikut:

Sebelum Anda mengirimkan perangkat

1. Panggilan [CreateProvisioningTemplate](#) untuk membuat template penyediaan. API ini mengembalikan template ARN. Untuk informasi selengkapnya, lihat [Penyediaan perangkat MQTT API](#).

Anda juga dapat membuat template penyediaan armada di konsol. AWS IoT

- a. Dari panel navigasi, pilih Connect, lalu pilih Fleet provisioning templates.
  - b. Pilih Buat template dan ikuti petunjuknya.
2. Buat sertifikat dan kunci pribadi terkait untuk digunakan sebagai penyediaan sertifikat klaim.
  3. Daftarkan sertifikat ini AWS IoT dan kaitkan kebijakan IoT yang membatasi penggunaan sertifikat. Contoh kebijakan IoT berikut membatasi penggunaan sertifikat yang terkait dengan kebijakan ini untuk menyediakan perangkat.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": ["iot:Connect"],
 "Resource": "*"
 },
 {
 "Effect": "Allow",
 "Action": ["iot:Publish","iot:Receive"],
 "Resource": [
 "arn:aws:iot:aws-region:aws-account-id:topic/$aws/certificates/
create/*",
 "arn:aws:iot:aws-region:aws-account-id:topic/$aws/provisioning-
templates/templateName/provision/*"
]
 }
],
}
```

```
{
 "Effect": "Allow",
 "Action": "iot:Subscribe",
 "Resource": [
 "arn:aws:iot:aws-region:aws-account-id:topicfilter/$aws/certificates/create/*",
 "arn:aws:iot:aws-region:aws-account-id:topicfilter/$aws/provisioning-templates/templateName/provision/*"
]
}
```

4. Berikan izin AWS IoT layanan untuk membuat atau memperbarui sumber daya IoT seperti hal-hal dan sertifikat di akun Anda saat menyediakan perangkat. Lakukan ini dengan melampirkan kebijakan AWSIoTThingsRegistration terkelola ke peran IAM (disebut peran penyedia) yang mempercayai kepala layanan. AWS IoT
5. Memproduksi perangkat dengan sertifikat klaim penyedia yang tertanam dengan aman di dalamnya.

Perangkat sekarang siap dikirim ke tempat ia akan diinstal untuk digunakan.

#### Important

Kunci pribadi klaim penyedia harus diamankan setiap saat, termasuk di perangkat. Kami menyarankan Anda menggunakan AWS IoT CloudWatch metrik dan log untuk memantau indikasi penyalahgunaan. Jika Anda mendeteksi penyalahgunaan, matikan sertifikat klaim penyedia sehingga tidak dapat digunakan untuk penyedia perangkat.

Untuk menginisialisasi perangkat untuk digunakan

1. Perangkat menggunakan [AWS IoT SDK Perangkat](#), [SDK Seluler](#), dan [AWS IoT Klien Perangkat](#) untuk menyambung dan mengautentikasi dengan AWS IoT menggunakan sertifikat klaim penyedia yang diinstal pada perangkat.

#### Note

Untuk keamanan, yang `certificateOwnershipToken` dikembalikan [CreateCertificateFromCsr](#) dan [CreateKeysAndCertificate](#)



kedaluwarsa setelah satu jam. [RegisterThing](#) harus dipanggil sebelum `certificateOwnershipToken` kedaluwarsa. Jika sertifikat yang dibuat oleh [CreateCertificateFromCsr](#) atau [CreateKeysAndCertificate](#) belum diaktifkan dan belum dilampirkan pada kebijakan atau sesuatu pada saat token kedaluwarsa, sertifikat akan dihapus. Jika token kedaluwarsa, perangkat dapat menelepon [CreateCertificateFromCsr](#) atau [CreateKeysAndCertificate](#) lagi untuk menghasilkan sertifikat baru.

2. Perangkat memperoleh sertifikat permanen dan kunci pribadi dengan menggunakan salah satu opsi ini. Perangkat akan menggunakan sertifikat dan kunci untuk semua otentikasi future dengan AWS IoT.
  - a. Panggilan [CreateKeysAndCertificate](#) untuk membuat sertifikat baru dan kunci pribadi menggunakan otoritas AWS sertifikat.  
  
Atau
  - b. Panggilan [CreateCertificateFromCsr](#) untuk menghasilkan sertifikat dari permintaan penandatanganan sertifikat yang menjaga kunci pribadinya tetap aman.
3. Dari perangkat, panggil [RegisterThing](#) untuk mendaftarkan perangkat dengan AWS IoT dan membuat sumber daya cloud.

Layanan Fleet Provisioning menggunakan template penyediaan untuk menentukan dan membuat sumber daya cloud seperti IoT. Template dapat menentukan atribut dan grup yang menjadi milik benda itu. Kelompok benda harus ada sebelum hal baru dapat ditambahkan ke mereka.

4. Setelah menyimpan sertifikat permanen pada perangkat, perangkat harus memutuskan sambungan dari sesi yang dimulai dengan sertifikat klaim penyediaan dan menyambung kembali menggunakan sertifikat permanen.

Perangkat sekarang siap untuk berkomunikasi secara normal AWS IoT.

## Penyediaan oleh pengguna tepercaya

Dalam banyak kasus, perangkat terhubung AWS IoT untuk pertama kalinya ketika pengguna tepercaya, seperti pengguna akhir atau teknisi instalasi, menggunakan aplikasi seluler untuk mengonfigurasi perangkat di lokasi yang digunakan.

**⚠ Important**

Anda harus mengelola akses dan izin pengguna tepercaya untuk melakukan prosedur ini. Salah satu cara untuk melakukannya adalah dengan menyediakan dan memelihara akun untuk pengguna tepercaya yang mengautentikasi mereka dan memberi mereka akses ke AWS IoT fitur dan operasi API yang diperlukan untuk melakukan prosedur ini.

Sebelum Anda mengirimkan perangkat

1. *Panggilan `CreateProvisioningTemplate` untuk membuat template penyediaan dan mengembalikan `TemplateArn` dan `TemplateName`-nya.*
2. Buat peran IAM yang digunakan oleh pengguna tepercaya untuk memulai proses penyediaan. Template penyediaan hanya memungkinkan pengguna tersebut untuk menyediakan perangkat. Sebagai contoh:


```
{
 "Effect": "Allow",
 "Action": [
 "iot:CreateProvisioningClaim"
],
 "Resource": [
 "arn:aws:iot:aws-region:aws-account-id:provisioningtemplate/templateName"
]
}
```

3. Berikan izin AWS IoT layanan untuk membuat atau memperbarui sumber daya IoT, seperti hal-hal dan sertifikat di akun Anda saat menyediakan perangkat. Anda melakukan ini dengan melampirkan kebijakan `AWSIoTThingsRegistration` terkelola ke peran IAM (disebut peran penyediaan) yang mempercayai prinsip layanan. AWS IoT
4. Sediakan sarana untuk mengidentifikasi pengguna tepercaya Anda, seperti dengan memberi mereka akun yang dapat mengautentikasi mereka dan mengotorisasi interaksi mereka dengan operasi AWS API yang diperlukan untuk mendaftarkan perangkat mereka.

Untuk menginisialisasi perangkat untuk digunakan


1. Pengguna tepercaya masuk ke aplikasi seluler atau layanan web penyediaan Anda.

2. Aplikasi seluler atau aplikasi web menggunakan peran IAM dan panggilan [CreateProvisioningClaim](#) untuk mendapatkan sertifikat klaim penyediaan sementara dari AWS IoT

 Note

Untuk keamanan, sertifikat klaim penyediaan sementara yang `CreateProvisioningClaim` dikembalikan kedaluwarsa setelah lima menit. Langkah-langkah berikut harus berhasil mengembalikan sertifikat yang valid sebelum sertifikat klaim penyediaan sementara berakhir. Sertifikat klaim penyediaan sementara tidak muncul dalam daftar sertifikat akun Anda.

3. Aplikasi seluler atau aplikasi web memasok sertifikat klaim penyediaan sementara ke perangkat bersama dengan informasi konfigurasi yang diperlukan, seperti kredensi Wi-Fi.
4. Perangkat menggunakan sertifikat klaim penyediaan sementara untuk terhubung AWS IoT menggunakan [AWS IoT SDK Perangkat, SDK Seluler, dan AWS IoT Klien Perangkat](#)
5. Perangkat memperoleh sertifikat permanen dan kunci pribadi dengan menggunakan salah satu opsi ini dalam waktu lima menit setelah terhubung AWS IoT dengan sertifikat klaim penyediaan sementara. Perangkat akan menggunakan sertifikat dan memasukkan opsi ini kembali untuk semua otentikasi future. AWS IoT
  - a. Panggilan [CreateKeysAndCertificate](#) untuk membuat sertifikat baru dan kunci pribadi menggunakan otoritas AWS sertifikat.  
  
Atau
  - b. Panggilan [CreateCertificateFromCsr](#) untuk menghasilkan sertifikat dari permintaan penandatanganan sertifikat yang menjaga kunci pribadinya tetap aman.

 Note

Ingat [CreateKeysAndCertificate](#) atau [CreateCertificateFromCsr](#) harus mengembalikan sertifikat yang valid dalam waktu lima menit setelah terhubung AWS IoT dengan sertifikat klaim penyediaan sementara.

6. Perangkat memanggil [RegisterThing](#) untuk mendaftarkan perangkat dengan AWS IoT dan membuat sumber daya cloud.

Layanan Fleet Provisioning menggunakan template penyediaan untuk menentukan dan membuat sumber daya cloud seperti IoT. Template dapat menentukan atribut dan grup yang menjadi milik benda itu. Kelompok benda harus ada sebelum hal baru dapat ditambahkan ke mereka.

7. Setelah menyimpan sertifikat permanen pada perangkat, perangkat harus memutuskan sambungan dari sesi yang dimulai dengan sertifikat klaim penyediaan sementara dan menyambung kembali menggunakan sertifikat permanen.

Perangkat sekarang siap untuk berkomunikasi secara normal AWS IoT.

## Menggunakan kait pra-penyediaan dengan CLI AWS

Prosedur berikut membuat template penyediaan dengan kait pra-penyediaan. Fungsi Lambda yang digunakan di sini adalah contoh yang dapat dimodifikasi.

Untuk membuat dan menerapkan hook pra-penyediaan ke template penyediaan

1. Buat fungsi Lambda yang memiliki input dan output yang ditentukan. Fungsi Lambda sangat dapat disesuaikan. `allowProvisioning` dan `parameterOverrides` diperlukan untuk membuat kait pra-penyediaan. Untuk informasi selengkapnya tentang membuat fungsi Lambda, lihat [Menggunakan AWS Lambda dengan Antarmuka Baris AWS Perintah](#).

Berikut ini adalah contoh output fungsi Lambda:

```
{
 "allowProvisioning": True,
 "parameterOverrides": {
 "incomingKey0": "incomingValue0",
 "incomingKey1": "incomingValue1"
 }
}
```

2. AWS IoT menggunakan kebijakan berbasis sumber daya untuk memanggil Lambda, jadi Anda harus memberikan izin AWS IoT untuk memanggil fungsi Lambda Anda.

### Important

Pastikan untuk menyertakan `source-arn` atau `source-account` dalam kunci konteks kondisi global dari kebijakan yang dilampirkan pada tindakan Lambda Anda untuk

mencegah manipulasi izin. Untuk informasi selengkapnya tentang langkah ini, lihat [Pencegahan confused deputy lintas layanan](#).

Berikut ini adalah contoh menggunakan [izin tambahan memberikan izin](#) IoT ke Lambda Anda.

```
aws lambda add-permission \
 --function-name myLambdaFunction \
 --statement-id iot-permission \
 --action lambda:InvokeFunction \
 --principal iot.amazonaws.com
```

### 3. [Tambahkan hook pra-penyediaan ke templat menggunakan perintah create-provisioning-template atau update-provisioning-template](#).

Contoh CLI berikut menggunakan [create-provisioning-template](#) untuk membuat template penyediaan yang memiliki kait pra-penyediaan:

```
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
```

Output dari perintah ini terlihat seperti berikut:

```
{
 "templateArn": "arn:aws:iot:us-east-1:1234564789012:provisioningtemplate/myTemplate",
 "defaultVersionId": 1,
 "templateName": myTemplate
}
```

Anda juga dapat memuat parameter dari file alih-alih mengetik semuanya sebagai nilai parameter baris perintah untuk menghemat waktu. Untuk informasi selengkapnya, lihat [Memuat AWS CLI Parameter dari File](#). Berikut ini menunjukkan template parameter dalam format JSON diperluas:

```
{
 "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"
 }
 },
 "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"]}
 }
}

```

Berikut ini menunjukkan pre-provisioning-hook parameter dalam format JSON diperluas:

```

{
 "targetArn" : "arn:aws:lambda:us-
east-1:765219403047:function:pre_provisioning_test",
 "payloadVersion" : "2020-04-01"
}

```

## Penyediaan perangkat yang memiliki sertifikat perangkat

AWS IoT menyediakan tiga cara untuk menyediakan perangkat ketika mereka sudah memiliki sertifikat perangkat (dan kunci pribadi terkait) pada mereka:

- Penyediaan satu hal dengan templat penyediaan. Ini adalah opsi yang baik jika Anda hanya perlu menyediakan perangkat satu per satu.
- Just-in-time provisioning (JITP) dengan template yang menyediakan perangkat saat pertama kali terhubung ke AWS IoT. Ini adalah pilihan yang baik jika Anda perlu mendaftarkan sejumlah besar perangkat, tetapi Anda tidak memiliki informasi tentang mereka yang dapat Anda kumpulkan ke dalam daftar penyediaan massal.
- Pendaftaran massal. Opsi ini memungkinkan Anda untuk menentukan daftar nilai template penyediaan satu hal yang disimpan dalam file dalam bucket S3. Pendekatan ini bekerja dengan

baik jika Anda memiliki sejumlah besar perangkat yang dikenal yang karakteristik yang diinginkan dapat Anda kumpulkan ke dalam daftar.

## Topik

- [Penyediaan satu hal](#)
- [ust-in-time Penyediaan J](#)
- [Registrasi massal](#)

## Penyediaan satu hal

Untuk menyediakan sesuatu, gunakan [RegisterThing](#) API atau perintah `register-thing` CLI. Perintah `register-thing` CLI mengambil argumen berikut:

`--templat-tubuh`

Template penyediaan.

`--parameter`

Daftar pasangan nama-nilai untuk parameter yang digunakan dalam template penyediaan, dalam format JSON (misalnya). `{"ThingName" : "MyProvisionedThing", "CSR" : "csr-text"}`

Lihat [Templat penyediaan](#).

[RegisterThing](#) atau `register-thing` mengembalikan ARN untuk sumber daya dan teks sertifikat yang dibuatnya:

```
{
 "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"
 }
}
```



Jika parameter dihilangkan dari kamus, nilai default digunakan. Jika tidak ada nilai default yang ditentukan, parameter tidak diganti dengan nilai.

## ust-in-time Penyediaan J

Anda dapat menggunakan just-in-time provisioning (JITP) untuk menyediakan perangkat Anda ketika mereka pertama kali mencoba untuk terhubung. AWS IoT Untuk menyediakan perangkat, Anda harus mengaktifkan pendaftaran otomatis dan mengaitkan templat penyediaan dengan sertifikat CA yang digunakan untuk menandatangani sertifikat perangkat. Keberhasilan dan kesalahan penyediaan dicatat seperti di [Metrik penyediaan perangkat](#) Amazon. CloudWatch

### Topik

- [Ikhtisar JITP](#)
- [Daftarkan CA menggunakan templat penyediaan](#)
- [Daftarkan CA menggunakan nama templat penyediaan](#)

### Ikhtisar JITP

Ketika perangkat mencoba untuk terhubung AWS IoT dengan menggunakan sertifikat yang ditandatangani oleh sertifikat CA terdaftar, AWS IoT memuat template dari sertifikat CA dan menggunakannya untuk memanggil [RegisterThing](#). Alur kerja JITP pertama-tama mendaftarkan sertifikat dengan nilai status. PENDING\_ACTIVATION Ketika alur penyediaan perangkat selesai, status sertifikat diubah menjadi. ACTIVE

AWS IoT mendefinisikan parameter berikut yang dapat Anda deklarasikan dan referensi dalam templat penyediaan:

- `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`

Nilai untuk parameter template penyediaan ini terbatas pada apa yang dapat diekstrak JITP dari bidang subjek sertifikat perangkat yang sedang disediakan. Sertifikat harus berisi nilai untuk semua parameter dalam badan template. `AWS::IoT::Certificate::IdParameter` mengacu pada ID yang dihasilkan secara internal, bukan ID yang terkandung dalam sertifikat. Anda bisa mendapatkan nilai ID ini menggunakan `principal()` fungsi di dalam AWS IoT aturan.

### Note

Anda dapat menyediakan perangkat menggunakan fitur AWS IoT Core just-in-time provisioning (JITP) tanpa harus mengirim seluruh rantai kepercayaan pada koneksi pertama perangkat ke. AWS IoT Core Menyajikan sertifikat CA adalah opsional, tetapi perangkat diperlukan untuk mengirim ekstensi [Server Name Indication \(SNI\)](#) ketika terhubung ke. AWS IoT Core

### Contoh badan template

File JSON berikut adalah contoh badan template dari template JITP lengkap.

```
{
 "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\ "]}] }"
 }
}
}
}

```

Templat sampel ini mendeklarasikan nilai untuk parameter

`AWS::IoT::Certificate::CommonName`, `AWS::IoT::Certificate::SerialNumber`, `AWS::IoT::Certificate::Country`, dan `AWS::IoT::Certificate::Id` penyediaan yang diekstraksi dari sertifikat dan digunakan di

bagian tersebut. Resources Alur kerja JITP kemudian menggunakan template ini untuk melakukan tindakan berikut:

- Daftarkan sertifikat dan atur statusnya ke PENDING\_ACTIVE.
- Buat sumber daya satu hal.
- Buat satu sumber kebijakan.
- Lampirkan kebijakan ke sertifikat.
- Lampirkan sertifikat pada objek .
- Perbarui status sertifikat ke AKTIF.

Penyediaan perangkat gagal jika sertifikat tidak memiliki semua properti yang disebutkan di Parameters bagian. templateBody Misalnya, jika `AWS::IoT::Certificate::Country` disertakan dalam templat, tetapi sertifikat tidak memiliki Country properti, penyediaan perangkat akan gagal.

Anda juga dapat menggunakan CloudTrail untuk memecahkan masalah dengan template JITP Anda. Untuk informasi tentang metrik yang dicatat di Amazon CloudWatch, lihat [Metrik penyediaan perangkat](#). Untuk informasi selengkapnya tentang penyediaan templat, lihat Templat [penyediaan](#).

#### Note

Selama proses penyediaan, just-in-time provisioning (JITP) memanggil operasi API bidang kontrol lainnya. AWS IoT Panggilan ini mungkin melebihi [Kuota AWS IoT Pelambatan](#) yang ditetapkan untuk akun Anda dan mengakibatkan panggilan terhambat. Hubungi [AWS Customer Support](#) untuk menaikkan kuota throttling Anda jika perlu.

## Daftarkan CA menggunakan templat penyediaan

Untuk mendaftarkan CA menggunakan templat penyediaan lengkap, ikuti langkah-langkah berikut:

1. Simpan template penyediaan Anda dan informasi ARN peran seperti contoh berikut sebagai file JSON:

```
{
 "templateBody" : "{\r\n \"Parameters\" : {\r\n \"AWS::IoT::Certificate::CommonName\" : {\r\n \"Type\" : \"String\"\r\n }\r\n }\r\n}"
```

```

\n },\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 \"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 \"roleArn\" : \"arn:aws:iam::123456789012:role/JITPRole\"
}

```

Dalam contoh ini, nilai `templateBody` bidang harus berupa objek JSON yang ditentukan sebagai string yang lolos dan hanya dapat menggunakan nilai-nilai dalam daftar [sebelumnya](#). Anda dapat menggunakan berbagai alat untuk membuat output JSON yang diperlukan, seperti `json.dumps` (Python) `JSON.stringify` atau (Node). Nilai `roleArn` bidang harus ARN dari peran yang `AWSIoTThingsRegistration` melekat padanya. Selain itu, template Anda dapat menggunakan yang sudah ada `PolicyName` alih-alih inline `PolicyDocument` dalam contoh.

- Daftarkan sertifikat CA dengan operasi [RegisterCACertificate](#) API atau perintah CLI. [register-ca-certificate](#) Anda akan menentukan direktori template penyediaan dan peran informasi ARN yang Anda simpan di langkah sebelumnya:

Berikut ini menunjukkan contoh cara mendaftarkan sertifikat CA dalam DEFAULT mode menggunakan 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
```

Berikut ini menunjukkan contoh cara mendaftarkan sertifikat CA dalam SNI\_ONLY mode menggunakan 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
```

Untuk informasi selengkapnya, lihat [Mendaftarkan Sertifikat CA Anda](#).

3. (Opsional) Perbarui pengaturan untuk sertifikat CA dengan menggunakan [operasi UpdateCertificate](#) API atau perintah CLI. [update-ca-certificate](#)

Berikut ini menunjukkan contoh cara memperbarui sertifikat CA menggunakan AWS CLI:

```
aws iot update-ca-certificate --certificate-id caCertificateId
--new-auto-registration-status ENABLE --registration-config
file://your-template
```

## Daftarkan CA menggunakan nama templat penyediaan

Untuk mendaftarkan CA menggunakan nama templat penyediaan, ikuti langkah-langkah berikut:

1. Simpan isi template penyediaan Anda sebagai file JSON. Anda dapat menemukan contoh badan template dalam [contoh badan template](#).
2. Untuk membuat template penyediaan, gunakan [CreateProvisioningTemplate](#) API atau perintah CLI [create-provisioning-template](#):

```
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

Untuk just-in-time penyediaan (JITP), Anda harus menentukan jenis templat JITP saat membuat templat penyediaan. Untuk informasi selengkapnya tentang jenis templat, lihat [CreateProvisioningTemplate](#) di Referensi AWS API.

- Untuk mendaftarkan CA dengan nama template, gunakan [RegisterCACertificate](#) API atau perintah CLI: [register-ca-certificate](#)

```
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
```

## Registrasi massal

Anda dapat menggunakan [start-thing-registration-task](#) perintah untuk mendaftarkan hal-hal dalam jumlah besar. Perintah ini mengambil template penyediaan, nama bucket S3, nama kunci, dan ARN peran yang memungkinkan akses ke file di bucket S3. File di bucket S3 berisi nilai yang digunakan untuk mengganti parameter dalam template. File harus berupa file JSON yang dibatasi baris baru. Setiap baris berisi semua nilai parameter untuk mendaftarkan satu perangkat. Sebagai contoh:

```
{"ThingName": "foo", "SerialNumber": "123", "CSR": "csr1"}
{"ThingName": "bar", "SerialNumber": "456", "CSR": "csr2"}
```

Operasi API terkait registrasi massal berikut mungkin berguna:

- [ListThingRegistrationTasks](#): Daftar tugas penyediaan barang massal saat ini.
- [DescribeThingRegistrationTask](#): Memberikan informasi tentang tugas pendaftaran barang massal tertentu.
- [StopThingRegistrationTask](#): Menghentikan tugas pendaftaran barang massal.
- [ListThingRegistrationTaskLaporan](#): Digunakan untuk memeriksa hasil dan kegagalan untuk tugas pendaftaran barang massal.

### Note

- Hanya satu tugas operasi pendaftaran massal yang dapat berjalan pada satu waktu (per akun).
- Operasi registrasi massal memanggil operasi API bidang AWS IoT kontrol lainnya. Panggilan ini mungkin melebihi [Kuota AWS IoT Pelambatan di akun Anda dan menyebabkan kesalahan throttle](#). Hubungi [AWS Customer Support](#) untuk menaikkan kuota AWS IoT throttling Anda, jika perlu.

## Templat penyediaan

Template penyediaan adalah dokumen JSON yang menggunakan parameter untuk menjelaskan sumber daya yang harus digunakan perangkat Anda untuk berinteraksi. AWS IoT Template penyediaan berisi dua bagian: Parameters dan Resources. Ada dua jenis template penyediaan di AWS IoT. Satu digunakan untuk just-in-time penyediaan (JITP) dan registrasi massal, dan yang kedua digunakan untuk penyediaan armada.

### Topik

- [Bagian parameter](#)
- [Bagian sumber daya](#)
- [Contoh template untuk pendaftaran massal](#)
- [Contoh template untuk just-in-time penyediaan \(JITP\)](#)
- [Penyediaan armada](#)

## Bagian parameter

ParametersBagian ini menyatakan parameter yang digunakan di Resources bagian. Setiap parameter mendeklarasikan nama, tipe, dan nilai default opsional. Nilai default digunakan ketika kamus diteruskan dengan template tidak berisi nilai untuk parameter. ParametersBagian dari dokumen template terlihat seperti berikut:

```
{
 "Parameters" : {
 "ThingName" : {
```



```
 "Type" : "String"
 },
 "SerialNumber" : {
 "Type" : "String"
 },
 "Location" : {
 "Type" : "String",
 "Default" : "WA"
 },
 "CSR" : {
 "Type" : "String"
 }
}
}
```

Cuplikan isi template ini mendeklarasikan empat parameter: `ThingName`, `SerialNumber`, `Location` dan `CSR`. Semua parameter ini bertipe `String`. `Location` parameter mendeklarasikan nilai default dari `"WA"`.

## Bagian sumber daya

`Resources` bagian badan templat menyatakan sumber daya yang diperlukan untuk berkomunikasi dengan perangkat Anda AWS IoT: sesuatu, sertifikat, dan satu atau beberapa kebijakan IoT. Setiap sumber daya menentukan nama logis, tipe, dan satu set properti.

Nama logis memungkinkan Anda untuk merujuk ke sumber daya di tempat lain dalam template.

Jenis menentukan jenis sumber daya yang Anda deklarasikan. Jenis yang valid adalah:

- `AWS::IoT::Thing`
- `AWS::IoT::Certificate`
- `AWS::IoT::Policy`

Properti yang Anda tentukan bergantung pada jenis sumber daya yang Anda deklarasikan.

## Sumber daya benda

Sumber daya benda dideklarasikan menggunakan properti berikut:

- `ThingName`: Tali.

- `AttributePayload`: Opsional. Daftar pasangan nama-nilai.
- `ThingTypeName`: Opsional. String untuk jenis hal terkait untuk benda tersebut.
- `ThingGroups`: Opsional. Daftar kelompok tempat benda itu berada.
- `BillingGroup`: Opsional. String untuk nama grup penagihan terkait.
- `PackageVersions`: Opsional. String untuk paket terkait dan nama versi.

## Sumber daya sertifikat

Anda dapat menentukan sertifikat dengan salah satu cara berikut:

- Permintaan penandatanganan sertifikat (CSR).
- ID sertifikat perangkat yang ada. (Hanya ID sertifikat yang dapat digunakan dengan templat penyediaan armada.)
- Sertifikat perangkat yang dibuat dengan sertifikat CA terdaftar AWS IoT. Jika Anda memiliki lebih dari satu sertifikat CA yang terdaftar dengan bidang subjek yang sama, Anda juga harus lulus dalam sertifikat CA yang digunakan untuk menandatangani sertifikat perangkat.

### Note

Saat Anda mendeklarasikan sertifikat dalam templat, gunakan hanya salah satu metode ini. Misalnya, jika Anda menggunakan CSR, Anda juga tidak dapat menentukan ID sertifikat atau sertifikat perangkat. Untuk informasi selengkapnya, lihat [Sertifikat klien X.509](#).

Untuk informasi selengkapnya, lihat [Ikhtisar Sertifikat X.509](#).

Sumber daya sertifikat dideklarasikan menggunakan properti berikut:

- `CertificateSigningRequest`: Tali.
- `CertificateId`: Tali.
- `CertificatePem`: Tali.
- `CACertificatePem`: Tali.
- `Status`: Opsional. String yang bisa ACTIVE atau INACTIVE. Default ke ACTIVE.

Contoh:

- Sertifikat yang ditentukan dengan CSR:

```
{
 "certificate" : {
 "Type" : "AWS::IoT::Certificate",
 "Properties" : {
 "CertificateSigningRequest": {"Ref" : "CSR"},
 "Status" : "ACTIVE"
 }
 }
}
```

- Sertifikat yang ditentukan dengan ID sertifikat yang ada:

```
{
 "certificate" : {
 "Type" : "AWS::IoT::Certificate",
 "Properties" : {
 "CertificateId": {"Ref" : "CertificateId"}
 }
 }
}
```

- Sertifikat yang ditentukan dengan sertifikat yang ada .pem dan sertifikat CA.pem:

```
{
 "certificate" : {
 "Type" : "AWS::IoT::Certificate",
 "Properties" : {
 "CACertificatePem": {"Ref" : "CACertificatePem"},
 "CertificatePem": {"Ref" : "CertificatePem"}
 }
 }
}
```

## Sumber daya kebijakan

Sumber daya kebijakan dideklarasikan menggunakan salah satu properti berikut:

- `PolicyName`: Opsional. String. Default ke hash dokumen kebijakan. Hanya `PolicyName` dapat merujuk AWS IoT kebijakan tetapi tidak kebijakan IAM. Jika Anda menggunakan AWS IoT

kebijakan yang ada, untuk `PolicyName` properti, masukkan nama kebijakan. Jangan sertakan `PolicyDocument` properti.

- `PolicyDocument`: Opsional. Sebuah objek JSON ditentukan sebagai string lolos. Jika tidak `PolicyDocument` disediakan, kebijakan harus sudah dibuat.

#### Note

Jika ada `Policy` bagian, `PolicyName` atau `PolicyDocument`, tetapi tidak keduanya, harus ditentukan.

## Ganti pengaturan

Jika template menentukan sumber daya yang sudah ada, `OverrideSettings` bagian ini memungkinkan Anda menentukan tindakan yang akan diambil:

### DO\_NOTHING

Tinggalkan sumber daya apa adanya.

### REPLACE

Ganti sumber daya dengan sumber daya yang ditentukan dalam template.

### FAIL

Menyebabkan permintaan gagal dengan `aResourceConflictsException`.

### MERGE

Hanya berlaku untuk `ThingGroups` dan `AttributePayload` properti `athing`. Gabungkan atribut yang ada atau keanggotaan grup dari benda tersebut dengan yang ditentukan dalam template.

Ketika Anda mendeklarasikan sumber daya sesuatu, Anda dapat menentukan `OverrideSettings` untuk properti berikut:

- `ATTRIBUTE_PAYLOAD`
- `THING_TYPE_NAME`
- `THING_GROUPS`

Ketika Anda mendeklarasikan sumber daya sertifikat, Anda dapat menentukan `OverrideSettings` untuk properti. Status

`OverrideSetting` tidak tersedia untuk sumber daya kebijakan.

## Contoh sumber daya

Cuplikan template berikut mendeklarasikan sesuatu, sertifikat, dan kebijakan:

```
{
 "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\"] }] }"
 }
 }
 }
}
```

Hal itu dinyatakan dengan:

- Nama logisnya "thing".
- Tipe `AWS::IoT::Thing`.
- Satu set properti benda.

Properti benda termasuk nama benda, satu set atribut, nama tipe benda opsional, dan daftar opsional grup benda yang menjadi milik benda itu.

Parameter direferensikan oleh `{"Ref": "parameter-name"}`. Ketika template dievaluasi, parameter diganti dengan nilai parameter dari kamus yang diteruskan dengan template.

Sertifikat dinyatakan dengan:

- Nama logisnya "certificate".
- Tipe `AWS::IoT::Certificate`.
- Satu set properti.

Properti termasuk CSR untuk sertifikat, dan pengaturan status ke `ACTIVE`. Teks CSR diteruskan sebagai parameter dalam kamus yang diteruskan dengan template.

Kebijakan ini dinyatakan dengan:

- Nama logisnya "policy".
- Tipe `AWS::IoT::Policy`.
- Baik nama kebijakan yang ada atau dokumen kebijakan.

## Contoh template untuk pendaftaran massal

File JSON berikut adalah contoh template penyediaan lengkap yang menentukan sertifikat dengan CSR:

(Nilai `PolicyDocument` bidang harus berupa objek JSON yang ditentukan sebagai string yang lolos.)

```
{
 "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"
 }
 },
 "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\"] }] }"
 }
 }
}
}
}

```

## Contoh template untuk just-in-time penyediaan (JITP)

File JSON berikut adalah contoh template penyediaan lengkap yang menentukan sertifikat yang ada dengan ID sertifikat:

```
{
 "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\"] }] }"
 }
}
}
```

### Important

Anda harus menggunakan `CertificateId` dalam template yang digunakan untuk penyediaan JIT.

Untuk informasi selengkapnya tentang jenis templat penyediaan, lihat [CreateProvisioningTemplate](#) di referensi AWS API.

Untuk informasi selengkapnya tentang cara menggunakan template ini untuk just-in-time penyediaan, lihat: [Penyediaan tepat waktu](#).

## Penyediaan armada

Templat penyediaan armada digunakan oleh AWS IoT untuk mengatur konfigurasi cloud dan perangkat. Template ini menggunakan parameter dan sumber daya yang sama dengan JITP dan templat pendaftaran massal. Untuk informasi selengkapnya, lihat [Templat](#)

[penyediaan](#). Templat penyediaan armada dapat berisi Mapping bagian dan bagian. DeviceConfiguration Anda dapat menggunakan fungsi intrinsik di dalam templat penyediaan armada untuk menghasilkan konfigurasi khusus perangkat. Templat penyediaan armada diberi nama sumber daya dan diidentifikasi oleh ARN (misalnya,). `arn:aws:iot:us-west-2:1234568788:provisioningtemplate/templateName`

## Pemetaan

Bagian Mappings opsional cocok dengan kunci untuk satu set nilai yang sesuai. Misalnya, jika Anda ingin menetapkan nilai berdasarkan AWS Wilayah, Anda dapat membuat pemetaan yang menggunakan Wilayah AWS nama sebagai kunci dan berisi nilai yang ingin Anda tentukan untuk setiap Wilayah tertentu. Anda menggunakan fungsi intrinsik `Fn::FindInMap` untuk mengambil nilai-nilai dalam peta.

Anda tidak dapat menyertakan parameter, parameter semu, atau memanggil fungsi intrinsik di bagian tersebut. Mappings

## Konfigurasi perangkat

Bagian konfigurasi perangkat berisi data arbitrer yang ingin Anda kirim ke perangkat saat penyediaan. Sebagai contoh:

```
{
 "DeviceConfiguration": {
 "Foo": "Bar"
 }
}
```

Jika Anda mengirim pesan ke perangkat menggunakan format payload JavaScript Object Notation (JSON), AWS IoT Core format data ini sebagai JSON. Jika Anda menggunakan format payload Concise Binary Object Representation (CBOR), AWS IoT Core format data ini sebagai CBOR. DeviceConfigurationBagian ini tidak mendukung objek JSON bersarang.

## Fungsi intrinsik

Fungsi intrinsik digunakan di bagian mana pun dari template penyediaan kecuali bagian. Mappings

### `Fn::Join`

Menambahkan satu set nilai ke dalam nilai tunggal, dipisahkan oleh pembatas yang ditentukan. Jika pembatas adalah string kosong, nilainya digabungkan tanpa pembatas.

**⚠ Important**

`Fn::Join` tidak didukung untuk [the section called "Sumber daya kebijakan"](#).

**Fn::Select**

Mengembalikan objek tunggal dari daftar objek dengan indeks.

**⚠ Important**

`Fn::Select` tidak memeriksa `null` nilai atau jika indeks berada di luar batas array. Kedua kondisi menghasilkan kesalahan penyediaan, jadi pastikan Anda memilih nilai indeks yang valid dan daftar berisi nilai non-null.

**Fn::FindInMap**

Mengembalikan nilai yang sesuai dengan kunci dalam peta dua tingkat yang dinyatakan di `Mappings` bagian.

**Fn::Split**

Membagi string ke dalam daftar nilai string sehingga Anda dapat memilih elemen dari daftar string. Anda menentukan pembatas yang menentukan di mana string dibagi (misalnya, koma). Setelah Anda membagi string, gunakan `Fn::Select` untuk memilih elemen.

Misalnya, jika string yang dipisahkan koma dari ID subnet diimpor ke templat tumpukan, Anda dapat membagi string pada setiap koma. Dari daftar ID subnet, gunakan `Fn::Select` untuk menentukan ID subnet untuk sumber daya.

**Fn::Sub**

Mengganti variabel dalam string input dengan nilai yang Anda tentukan. Anda dapat menggunakan fungsi ini untuk membuat perintah atau output yang menyertakan nilai yang tidak tersedia sampai Anda membuat atau memperbarui tumpukan.

**Contoh template untuk penyediaan armada**

```
{
 "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"
 }
 },
},
```

```

 "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

Template penyediaaan yang ada dapat diperbarui untuk menambahkan kait [pra-penyediaan](#).

## Kait pra-penyediaan

AWS merekomendasikan penggunaan fungsi hook pra-penyediaan saat membuat templat penyediaaan untuk memungkinkan lebih banyak kontrol atas perangkat mana dan berapa banyak perangkat yang ada di dalam akun Anda. Kait pra-penyediaan adalah fungsi Lambda yang memvalidasi parameter yang diteruskan dari perangkat sebelum mengizinkan perangkat disediakan. Fungsi Lambda ini harus ada di akun Anda sebelum Anda menyediakan perangkat karena dipanggil setiap kali perangkat mengirimkan permintaan. [the section called “RegisterThing”](#)

### Important

Pastikan untuk menyertakan `source-arn` atau `source-account` dalam kunci konteks kondisi global dari kebijakan yang dilampirkan pada tindakan Lambda Anda untuk mencegah

manipulasi izin. Untuk informasi selengkapnya tentang langkah ini, lihat [Pencegahan confused deputy lintas layanan](#).

Agar perangkat dapat disediakan, fungsi Lambda Anda harus menerima objek input dan mengembalikan objek keluaran yang dijelaskan di bagian ini. Penyediaan berlangsung hanya jika fungsi Lambda mengembalikan objek dengan. "allowProvisioning": True

## Masukan kait pra-penyediaan

AWS IoT mengirimkan objek ini ke fungsi Lambda saat perangkat mendaftar dengan. 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",
 ...
 }
}
```

parametersObjek yang diteruskan ke fungsi Lambda berisi properti dalam parameters argumen yang diteruskan dalam payload [the section called "RegisterThing"](#) permintaan.

## Nilai pengembalian kait pra-penyediaan

Fungsi Lambda harus mengembalikan respons yang menunjukkan apakah fungsi Lambda telah mengizinkan permintaan penyediaan dan nilai properti apa pun untuk diganti.

Berikut ini adalah contoh respons yang berhasil dari fungsi pra-penyediaan.

```
{
 "allowProvisioning": true,
 "parameterOverrides" : {
 "Key": "newCustomValue",
 ...
 }
}
```

```
}
```

"parameterOverrides" nilai akan ditambahkan ke "parameters" parameter payload [the section called "RegisterThing"](#) permintaan.

#### Note

- Jika fungsi Lambda gagal, permintaan penyediaan gagal ACCESS\_DENIED dan kesalahan dicatat ke Log. CloudWatch
- Jika fungsi Lambda tidak ditampilkan "allowProvisioning": "true" dalam respons, permintaan penyediaan gagal. ACCESS\_DENIED
- Fungsi Lambda harus selesai berjalan dan kembali dalam 5 detik, jika tidak permintaan penyediaan gagal.

## Contoh Lambda kait pra-penyediaan

### Python

Contoh hook Lambda pra-penyediaan dengan Python.

```
import json

def pre_provisioning_hook(event, context):
 print(event)

 return {
 'allowProvisioning': True,
 'parameterOverrides': {
 'DeviceLocation': 'Seattle'
 }
 }
```

### Java

Contoh hook Lambda pra-penyediaan di Jawa.

Kelas 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;
 }
}
```

### Permintaan kelas:

```
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;
}
```

Kelas respons:

```
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

Contoh kait pra-penyediaan Lambda di JavaScript

```
exports.handler = function(event, context, callback) {
 console.log(JSON.stringify(event, null, 2));
 var reply = {
 allowProvisioning: true,
 parameterOverrides: {
 DeviceLocation: 'Seattle'
 }
 };
 callback(null, reply);
}
```

# Penandatanganan sertifikat yang dikelola sendiri menggunakan penyedia AWS IoT Core sertifikat

Anda dapat membuat penyedia AWS IoT Core sertifikat untuk menandatangani permintaan penandatanganan sertifikat (CSR) dalam penyediaan AWS IoT armada. Penyedia sertifikat mereferensikan fungsi Lambda dan API [CreateCertificateFromCsrMQTT](#) untuk penyediaan armada. Fungsi Lambda menerima CSR dan mengembalikan sertifikat klien yang ditandatangani.

Bila Anda tidak memiliki penyedia sertifikat dengan Akun AWS, [CreateCertificateFromCsrMQTT API](#) dipanggil dalam penyediaan armada untuk menghasilkan sertifikat dari CSR. Setelah Anda membuat penyedia sertifikat, perilaku API [CreateCertificateFromCsrMQTT akan berubah dan semua panggilan ke API](#) MQTT ini akan memanggil penyedia sertifikat untuk mengeluarkan sertifikat.

Dengan penyedia AWS IoT Core sertifikat, Anda dapat menerapkan solusi yang memanfaatkan otoritas sertifikat swasta (CA) seperti CA tepercaya publik lainnya [AWS Private CA](#), atau Infrastruktur Kunci Publik (PKI) Anda sendiri untuk menandatangani CSR. Selain itu, Anda dapat menggunakan penyedia sertifikat untuk menyesuaikan bidang sertifikat klien Anda seperti periode validitas, algoritma penandatanganan, penerbit, dan ekstensi.

## Important

Anda hanya dapat membuat satu penyedia sertifikat per Akun AWS. Perubahan perilaku penandatanganan berlaku untuk seluruh armada yang memanggil [API CreateCertificateFromCsrMQTT](#) hingga Anda menghapus penyedia sertifikat dari penyedia sertifikat. Akun AWS

Dalam topik ini:

- [Cara kerja penandatanganan sertifikat yang dikelola sendiri dalam penyediaan armada](#)
- [Input fungsi Lambda penyedia sertifikat](#)
- [Nilai pengembalian fungsi Lambda penyedia sertifikat](#)
- [Contoh fungsi Lambda](#)
- [Penandatanganan sertifikat yang dikelola sendiri untuk penyediaan armada](#)
- [AWS CLI perintah untuk penyedia sertifikat](#)

# Cara kerja penandatanganan sertifikat yang dikelola sendiri dalam penyediaan armada

## Konsep utama

Konsep berikut memberikan detail yang dapat membantu Anda memahami cara kerja penandatanganan sertifikat yang dikelola sendiri dalam penyediaan AWS IoT armada. Untuk informasi selengkapnya, lihat [Menyediakan perangkat yang tidak memiliki sertifikat perangkat menggunakan penyediaan armada](#).

## AWS IoT penyediaan armada

Dengan penyediaan AWS IoT armada (kependekan dari penyediaan armada), AWS IoT Core menghasilkan dan mengirimkan sertifikat perangkat dengan aman ke perangkat Anda saat mereka terhubung untuk pertama kalinya. AWS IoT Core Anda dapat menggunakan penyediaan armada untuk menghubungkan perangkat yang tidak memiliki sertifikat perangkat. AWS IoT Core

## Permintaan penandatanganan sertifikat (CSR)

Dalam proses penyediaan armada, perangkat membuat permintaan AWS IoT Core melalui [penyediaan armada](#) MQTT API. Permintaan ini mencakup permintaan penandatanganan sertifikat (CSR), yang akan ditandatangani untuk membuat sertifikat klien.

## AWS penandatanganan sertifikat terkelola dalam penyediaan armada

AWS managed adalah pengaturan default untuk penandatanganan sertifikat dalam penyediaan armada. Dengan penandatanganan sertifikat AWS terkelola, AWS IoT Core akan menandatangani CSR menggunakan CA-nya sendiri.

## Penandatanganan sertifikat yang dikelola sendiri dalam penyediaan armada

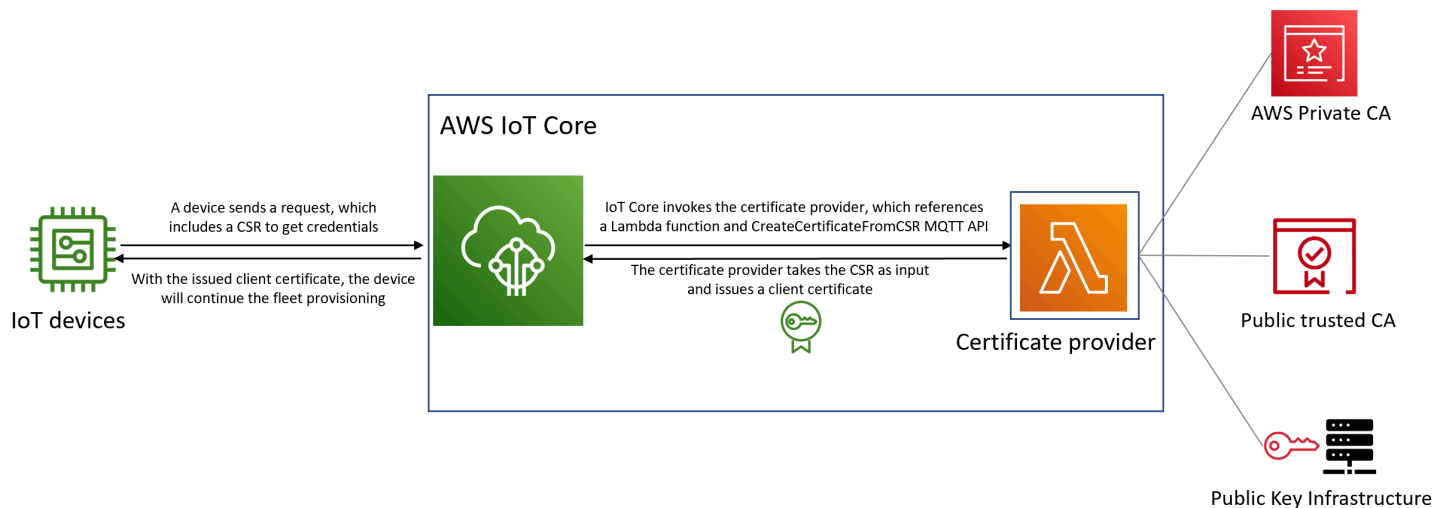
Self-managed adalah opsi lain untuk penandatanganan sertifikat dalam penyediaan armada. Dengan penandatanganan sertifikat yang dikelola sendiri, Anda membuat penyedia AWS IoT Core sertifikat untuk menandatangani CSR. Anda dapat menggunakan penandatanganan sertifikat yang dikelola sendiri untuk menandatangani CSR dengan CA yang dihasilkan oleh AWS Private CA, CA tepercaya publik lainnya, atau Infrastruktur Kunci Publik (PKI) Anda sendiri.

## AWS IoT Core penyedia sertifikat

AWS IoT Core penyedia sertifikat (kependekan dari penyedia sertifikat) adalah sumber daya yang dikelola pelanggan yang digunakan untuk penandatanganan sertifikat yang dikelola sendiri dalam penyediaan armada.

## Diagram

Diagram berikut adalah ilustrasi yang disederhanakan tentang cara kerja penandatanganan sertifikat mandiri dalam penyediaan AWS IoT armada.



- Ketika perangkat IoT baru diproduksi atau diperkenalkan ke armada, diperlukan sertifikat klien untuk mengautentikasi dirinya sendiri. AWS IoT Core
- Sebagai bagian dari proses penyediaan armada, perangkat membuat permintaan AWS IoT Core untuk sertifikat klien melalui [penyediaan armada](#) MQTT API. Permintaan ini mencakup permintaan penandatanganan sertifikat (CSR).
- AWS IoT Core memanggil penyedia sertifikat dan meneruskan CSR sebagai input ke penyedia.
- Penyedia sertifikat mengambil CSR sebagai masukan dan mengeluarkan sertifikat klien.

Untuk penandatanganan sertifikat AWS terkelola, AWS IoT Core tandatangani CSR menggunakan CA-nya sendiri dan mengeluarkan sertifikat klien.

- Dengan sertifikat klien yang dikeluarkan, perangkat akan melanjutkan penyediaan armada dan membuat koneksi yang aman dengannya. AWS IoT Core

## Input fungsi Lambda penyedia sertifikat

AWS IoT Core mengirimkan objek berikut ke fungsi Lambda saat perangkat mendaftar dengannya. Nilai CSR dalam [format Privacy-Enhanced Mail \(PEM\)](#) yang disediakan dalam permintaan. `certificateSigningRequest` `CreateCertificateFromCsr` `principalId` adalah ID dari prinsipal yang digunakan untuk terhubung AWS IoT Core saat membuat

CreateCertificateFromCsr permintaan. `clientId` adalah ID klien yang disetel untuk koneksi MQTT.

```
{
 "certificateSigningRequest": "string",
 "principalId": "string",
 "clientId": "string"
}
```

## Nilai pengembalian fungsi Lambda penyedia sertifikat

Fungsi Lambda harus mengembalikan respons yang berisi nilai `certificatePem`. Berikut ini adalah contoh respons yang sukses. AWS IoT Core akan menggunakan nilai pengembalian (`certificatePem`) untuk membuat sertifikat.

```
{
 "certificatePem": "string"
}
```

Jika pendaftaran berhasil, `CreateCertificateFromCsr` akan mengembalikan yang sama `certificatePem` dalam `CreateCertificateFromCsr` tanggapan. Untuk informasi selengkapnya, lihat contoh payload respons dari [CreateCertificateFromCsr](#)

## Contoh fungsi Lambda

Sebelum membuat penyedia sertifikat, Anda harus membuat fungsi Lambda untuk menandatangani CSR. Berikut ini adalah contoh fungsi Lambda di Python. Fungsi ini memanggil AWS Private CA untuk menandatangani input CSR, menggunakan CA pribadi dan algoritma SHA256WITHRSA penandatanganan. Sertifikat klien yang dikembalikan akan berlaku selama satu tahun. Untuk informasi selengkapnya tentang AWS Private CA dan cara membuat CA pribadi, lihat [Apa itu CA AWS Pribadi?](#) dan [Membuat CA pribadi](#).

```
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

- Sertifikat yang dikembalikan oleh fungsi Lambda harus memiliki nama subjek dan kunci publik yang sama dengan Permintaan Penandatanganan Sertifikat (CSR).
- Fungsi Lambda harus selesai berjalan dalam 5 detik.
- Fungsi Lambda harus sama Akun AWS dan Wilayah sebagai sumber daya penyedia sertifikat.
- Kepala AWS IoT layanan harus diberikan izin pemanggilan ke fungsi Lambda. Untuk menghindari [masalah deputi yang membingungkan](#), kami sarankan Anda mengatur `sourceArn` dan `sourceAccount` untuk izin pemanggilan. Untuk informasi lebih lanjut, lihat [Pencegahan Deputi Bingung Lintas Layanan](#).

Contoh kebijakan berbasis sumber daya berikut untuk Lambda memberikan AWS IoT izin untuk [menjalankan fungsi 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"
 }
 }
 }
]
```

## Penandatanganan sertifikat yang dikelola sendiri untuk penyediaan armada

Anda dapat memilih penandatanganan sertifikat yang dikelola sendiri untuk penyediaan armada menggunakan atau. AWS CLI atau AWS Management Console

### AWS CLI

Untuk memilih penandatanganan sertifikat yang dikelola sendiri, Anda harus membuat penyedia AWS IoT Core sertifikat untuk menandatangani CSR dalam penyediaan armada. AWS IoT Core memanggil penyedia sertifikat, yang mengambil CSR sebagai input dan mengembalikan sertifikat klien. Untuk membuat penyedia sertifikat, gunakan operasi `CreateCertificateProvider` API atau perintah `create-certificate-provider` CLI.

#### Note

Setelah Anda membuat penyedia sertifikat, perilaku [CreateCertificateFromCsrAPI untuk penyediaan armada](#) akan berubah sehingga semua panggilan ke `CreateCertificateFromCsr` akan memanggil penyedia sertifikat untuk membuat

sertifikat. Diperlukan beberapa menit agar perilaku ini berubah setelah penyedia sertifikat dibuat.

```
aws iot create-certificate-provider \
 --certificateProviderName my-certificate-provider \
 --lambdaFunctionArn arn:aws:lambda:us-east-1:123456789012:function:my-
function-1 \
 --accountDefaultForOperations CreateCertificateFromCsr
```

Berikut ini menunjukkan contoh output untuk perintah ini:

```
{
 "certificateProviderName": "my-certificate-provider",
 "certificateProviderArn": "arn:aws:iot:us-east-1:123456789012:certificateprovider:my-
certificate-provider"
}
```

Untuk informasi selengkapnya, lihat [CreateCertificateProvider](#) dari Referensi AWS IoTAPI.

## AWS Management Console

Untuk memilih penandatanganan sertifikat yang dikelola sendiri menggunakan AWS Management Console, ikuti langkah-langkahnya:

1. Pergi ke [AWS IoT konsol](#).
2. Di navigasi kiri, di bawah Keamanan, pilih Penandatanganan sertifikat.
3. Pada halaman Penandatanganan sertifikat, di bawah Detail penandatanganan sertifikat, pilih Edit metode penandatanganan sertifikat.
4. Pada halaman Edit metode penandatanganan sertifikat, di bawah Metode penandatanganan sertifikat, pilih Dikelola sendiri.
5. Di bagian Pengaturan yang dikelola sendiri, masukkan nama untuk penyedia sertifikat, lalu buat atau pilih fungsi Lambda.
6. Pilih Perbarui penandatanganan sertifikat.



## AWS CLI perintah untuk penyedia sertifikat

### Buat penyedia sertifikat

Untuk membuat penyedia sertifikat, gunakan operasi `CreateCertificateProvider` API atau perintah `create-certificate-provider` CLI.

#### Note

Setelah Anda membuat penyedia sertifikat, perilaku [CreateCertificateFromCsrAPI untuk penyediaan armada](#) akan berubah sehingga semua panggilan ke `CreateCertificateFromCsr` akan memanggil penyedia sertifikat untuk membuat sertifikat. Diperlukan beberapa menit agar perilaku ini berubah setelah penyedia sertifikat dibuat.

```
aws iot create-certificate-provider \
 --certificateProviderName my-certificate-provider \
 --lambdaFunctionArn arn:aws:lambda:us-east-1:123456789012:function:my-
function-1 \
 --accountDefaultForOperations CreateCertificateFromCsr
```

Berikut ini menunjukkan contoh output untuk perintah ini:

```
{
 "certificateProviderName": "my-certificate-provider",
 "certificateProviderArn": "arn:aws:iot:us-east-1:123456789012:certificateprovider:my-
certificate-provider"
}
```

Untuk informasi selengkapnya, lihat [CreateCertificateProvider](#) dari Referensi AWS IoTAPI.

### Perbarui penyedia sertifikat

Untuk memperbarui penyedia sertifikat, gunakan operasi `UpdateCertificateProvider` API atau perintah `update-certificate-provider` CLI.

```
aws iot update-certificate-provider \
 --certificateProviderName my-certificate-provider \
 --lambdaFunctionArn arn:aws:lambda:us-east-1:123456789012:function:my-
function-1 \
 --accountDefaultForOperations CreateCertificateFromCsr
```

```
--certificateProviderName my-certificate-provider \
--lambdaFunctionArn arn:aws:lambda:us-east-1:123456789012:function:my-
function-2 \
--accountDefaultForOperations CreateCertificateFromCsr
```

Berikut ini menunjukkan contoh output untuk perintah ini:

```
{
 "certificateProviderName": "my-certificate-provider",
 "certificateProviderArn": "arn:aws:iot:us-east-1:123456789012:certificateprovider:my-
certificate-provider"
}
```

Untuk informasi selengkapnya, lihat [UpdateCertificateProvider](#) dari Referensi AWS IoTAPI.

## Jelaskan penyedia sertifikat

Untuk mendeskripsikan penyedia sertifikat, gunakan operasi DescribeCertificateProvider API atau perintah describe-certificate-provider CLI.

```
aws iot describe-certificate-provider --certificateProviderName my-certificate-provider
```

Berikut ini menunjukkan contoh output untuk perintah ini:

```
{
 "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"
}
```

Untuk informasi selengkapnya, lihat [DescribeCertificateProvider](#) dari Referensi AWS IoTAPI.

## Hapus penyedia sertifikat

Untuk menghapus penyedia sertifikat, gunakan operasi DeleteCertificateProvider API atau perintah delete-certificate-provider CLI. Jika Anda menghapus sumber daya penyedia

sertifikat, perilaku `CreateCertificateFromCsr` akan dilanjutkan, dan AWS IoT akan membuat sertifikat yang ditandatangani oleh AWS IoT dari CSR.

```
aws iot delete-certificate-provider --certificateProviderName my-certificate-provider
```

Perintah ini tidak menghasilkan output apa pun.

Untuk informasi selengkapnya, lihat [DeleteCertificateProvider](#) dari Referensi AWS IoTAPI.

## Daftar penyedia sertifikat

Untuk membuat daftar penyedia sertifikat di dalam Anda Akun AWS, gunakan operasi `ListCertificateProviders` API atau perintah `list-certificate-providers` CLI.

```
aws iot list-certificate-providers
```

Berikut ini menunjukkan contoh output untuk perintah ini:

```
{
 "certificateProviders": [
 {
 "certificateProviderName": "my-certificate-provider",
 "certificateProviderArn": "arn:aws:iot:us-
east-1:123456789012:certificateprovider:my-certificate-provider"
 }
]
}
```

Untuk informasi selengkapnya, lihat [ListCertificateProvider](#) dari Referensi AWS IoTAPI.

## Membuat kebijakan dan peran IAM untuk pengguna yang menginstal perangkat

### Note

Prosedur ini hanya untuk digunakan jika diarahkan oleh AWS IoT konsol.  
Untuk membuka halaman ini dari konsol, buka [buat templat penyediaan baru](#).

## Mengapa ini tidak bisa dilakukan di AWS IoT konsol?

Untuk pengalaman yang paling aman, tindakan IAM dilakukan di konsol IAM. Prosedur di bagian ini memandu Anda melalui langkah-langkah untuk membuat peran dan kebijakan IAM yang diperlukan untuk menggunakan templat penyedia.

## Membuat kebijakan IAM untuk pengguna yang akan menginstal perangkat

Prosedur ini menjelaskan cara membuat kebijakan IAM yang mengizinkan pengguna untuk menginstal perangkat menggunakan templat penyedia.

Saat melakukan prosedur ini, Anda akan beralih antara konsol IAM dan AWS IoT konsol. Sebaiknya buka kedua konsol secara bersamaan saat Anda menyelesaikan prosedur ini.

Untuk membuat kebijakan IAM bagi pengguna yang akan menginstal perangkat

1. Buka [hub Kebijakan di konsol IAM](#).
2. Pilih Buat Kebijakan.
3. Di halaman Buat kebijakan, pilih tab JSON.
4. Beralih ke halaman di AWS IoT konsol tempat Anda memilih Konfigurasi kebijakan dan peran pengguna.
5. Dalam kebijakan penyedia sampel, pilih Salin.
6. Beralih kembali ke konsol IAM.
7. Di editor JSON, tempel kebijakan yang Anda salin dari konsol. AWS IoT Kebijakan ini khusus untuk template yang Anda buat di AWS IoT konsol.
8. Untuk melanjutkan, pilih Selanjutnya: Tanda.
9. Pada halaman Tambah tag (Opsional), pilih Tambah tag untuk setiap tag yang ingin Anda tambahkan ke kebijakan ini. Anda dapat melewati langkah ini jika Anda tidak memiliki tag untuk ditambahkan.
10. Untuk melanjutkan, pilih Selanjutnya: Tinjauan.
11. Pada halaman Kebijakan Tinjauan, lakukan hal berikut:
  - a. Untuk Nama\*, masukkan nama untuk kebijakan yang akan membantu Anda mengingat tujuan kebijakan tersebut.

Perhatikan nama yang Anda berikan pada kebijakan ini karena Anda akan menggunakannya dalam prosedur berikutnya.

- b. Anda dapat memilih untuk memasukkan deskripsi opsional untuk kebijakan yang Anda buat.
  - c. Tinjau sisa kebijakan ini dan tag-nya.
12. Untuk menyelesaikan pembuatan kebijakan baru, pilih Buat kebijakan.

Setelah membuat kebijakan baru, lanjutkan [the section called “Membuat peran IAM untuk pengguna yang akan menginstal perangkat”](#) untuk membuat entri peran pengguna yang akan dilampirkan kebijakan ini.

## Membuat peran IAM untuk pengguna yang akan menginstal perangkat

Langkah-langkah ini menjelaskan cara membuat peran IAM yang mengautentikasi pengguna yang akan menginstal perangkat menggunakan templat penyediaan.

Untuk membuat kebijakan IAM bagi pengguna yang akan menginstal perangkat

1. Buka [hub Peran di konsol IAM](#).
2. Pilih Buat peran.
3. Di Pilih entitas tepercaya, pilih jenis entitas tepercaya yang ingin Anda berikan akses ke templat yang Anda buat.
4. Pilih atau masukkan identifikasi entitas tepercaya yang ingin Anda berikan aksesnya, lalu pilih Berikutnya.
5. Di halaman Tambahkan izin, di Kebijakan izin, di kotak pencarian, masukkan nama kebijakan yang Anda buat di [prosedur sebelumnya](#).
6. Untuk daftar kebijakan, pilih kebijakan yang Anda buat di prosedur sebelumnya, lalu pilih Berikutnya.
7. Di bagian Nama, tinjau, dan buat, lakukan hal berikut:
  - a. Untuk nama Peran, masukkan nama peran yang akan membantu Anda mengingat tujuan peran tersebut.
  - b. Untuk Deskripsi, Anda dapat memilih untuk memasukkan deskripsi opsional Peran. Ini tidak diperlukan untuk melanjutkan.
  - c. Tinjau nilai-nilai di Langkah 1 dan Langkah 2.
  - d. Untuk Tambahkan tag (Opsional), Anda dapat memilih untuk menambahkan tag ke peran ini. Ini tidak diperlukan untuk melanjutkan.
  - e. Verifikasi informasi di halaman ini lengkap dan benar, lalu pilih Buat peran.

Setelah Anda membuat peran baru, kembali ke AWS IoT konsol untuk terus membuat template.

## Memperbarui kebijakan yang ada untuk mengotorisasi templat baru

Langkah-langkah berikut menjelaskan cara menambahkan templat baru ke kebijakan IAM yang mengizinkan pengguna untuk menginstal perangkat menggunakan templat penyedia.

Untuk menambahkan template baru ke kebijakan IAM yang ada

1. Buka [hub Kebijakan di konsol IAM](#).
2. Di kotak pencarian, masukkan nama kebijakan yang akan diperbarui.
3. Untuk daftar di bawah kotak pencarian, temukan kebijakan yang ingin Anda perbarui dan pilih nama kebijakan.
4. Untuk ringkasan Kebijakan, pilih tab JSON, jika panel tersebut belum terlihat.
5. Untuk mengubah dokumen kebijakan, pilih Edit kebijakan.
6. Di editor, pilih tab JSON, jika panel itu belum terlihat.
7. Dalam dokumen kebijakan, temukan pernyataan kebijakan yang berisi `iot:CreateProvisioningClaim` tindakan.

Jika dokumen kebijakan tidak berisi pernyataan kebijakan dengan `iot:CreateProvisioningClaim` tindakan, salin cuplikan pernyataan berikut dan tempel sebagai entri tambahan dalam Statement larik dalam dokumen kebijakan.

### Note

Cuplikan ini harus ditempatkan sebelum `]` karakter penutup dalam array. Statement Anda mungkin perlu menambahkan koma sebelum atau sesudah cuplikan ini untuk memperbaiki kesalahan sintaks apa pun.

```
{
 "Effect": "Allow",
 "Action": [
 "iot:CreateProvisioningClaim"
],
 "Resource": [
 "--PUT YOUR NEW TEMPLATE ARN HERE--"
]
}
```

```
}

```

8. Beralih ke halaman di AWS IoT konsol tempat Anda memilih Ubah izin peran pengguna.
9. Temukan ARN Sumber Daya dari template dan pilih Salin.
10. Beralih kembali ke konsol IAM.
11. Rekatkan Nama Sumber Daya Amazon (ARN) yang disalin di bagian atas daftar ARN template dalam array sehingga Statement ini adalah entri pertama.

Jika ini adalah satu-satunya ARN dalam array, hapus koma di akhir nilai yang baru saja Anda tempelkan.

12. Tinjau pernyataan kebijakan yang diperbarui dan perbaiki kesalahan yang ditunjukkan oleh editor.
13. Untuk menyimpan dokumen kebijakan yang diperbarui, pilih Kebijakan tinjauan.
14. Tinjau kebijakan, lalu pilih Simpan perubahan.
15. Kembali ke AWS IoT konsol.

## Penyediaan perangkat MQTT API

Layanan Penyediaan Armada mendukung operasi API MQTT berikut:

- [the section called "CreateCertificateFromCsr"](#)
- [the section called "CreateKeysAndCertificate"](#)
- [the section called "RegisterThing"](#)

*API ini mendukung buffer respons dalam format Concise Binary Object Representation (CBOR) dan JavaScript Object Notation (JSON), tergantung pada format payload topik.* Untuk kejelasan, contoh respons dan permintaan di bagian ini ditampilkan dalam format JSON.

| <b>format muatan</b> | Jenis data format respons               |
|----------------------|-----------------------------------------|
| cbor                 | Representasi Objek Biner Ringkas (CBOR) |
| json                 | JavaScript Notasi Objek (JSON)          |

### ⚠ Important

Sebelum memublikasikan topik pesan permintaan, berlangganan topik respons untuk menerima respons. Pesan yang digunakan oleh API ini menggunakan protokol publish/subscribe MQTT untuk menyediakan interaksi permintaan dan respons.

Jika Anda tidak berlangganan topik respons sebelum memublikasikan permintaan, Anda mungkin tidak menerima hasil permintaan tersebut.

## CreateCertificateFromCsr

Membuat sertifikat dari permintaan penandatanganan sertifikat (CSR). AWS IoT menyediakan sertifikat klien yang ditandatangani oleh otoritas sertifikat Amazon Root (CA). Sertifikat baru memiliki PENDING\_ACTIVATION status. Saat Anda menelepon RegisterThing untuk menyediakan sesuatu dengan sertifikat ini, status sertifikat berubah menjadi ACTIVE atau INACTIVE seperti yang dijelaskan dalam templat.

Untuk informasi selengkapnya tentang membuat sertifikat klien menggunakan sertifikat Otoritas Sertifikat dan permintaan penandatanganan sertifikat, lihat [Buat sertifikat klien menggunakan sertifikat CA Anda](#).

### 📘 Note

Untuk keamanan, yang certificateOwnershipToken dikembalikan [CreateCertificateFromCsr](#) kedaluwarsa setelah satu jam. [RegisterThing](#) harus dipanggil sebelum certificateOwnershipToken kedaluwarsa. Jika sertifikat yang dibuat oleh [CreateCertificateFromCsr](#) belum diaktifkan dan dilampirkan pada kebijakan atau sesuatu pada saat token kedaluwarsa, sertifikat akan dihapus. Jika token kedaluwarsa, perangkat dapat menelepon [CreateCertificateFromCsr](#) untuk menghasilkan sertifikat baru.

## CreateCertificateFromCsrpermintaan

Publikasikan pesan dengan `$aws/certificates/create-from-csr/payload-format` topik.

`payload-format`

Format payload pesan sebagai cbor atau json.



## CreateCertificateFromCsr minta muatan

```
{
 "certificateSigningRequest": "string"
}
```

### certificateSigningRequest

CSR, dalam format PEM.

## CreateCertificateFromCsr respon

Berlangganan `$aws/certificates/create-from-csr/payload-format/accepted`.

### payload-format

Format payload pesan sebagai cbor atau json.

## CreateCertificateFromCsr muatan respons

```
{
 "certificateOwnershipToken": "string",
 "certificateId": "string",
 "certificatePem": "string"
}
```

### certificateOwnershipToken

Token untuk membuktikan kepemilikan sertifikat selama penyediaan.

### certificateId

ID sertifikat. Operasi manajemen sertifikat hanya mengambil CertificateID.

### certificatePem

Data sertifikat, dalam format PEM.

## CreateCertificateFromCsr kesalahan

Untuk menerima tanggapan kesalahan, berlangganan `$aws/certificates/create-from-csr/payload-format/rejected`.

## payload-format

Format payload pesan sebagai cbor atau json.

### CreateCertificateFromCsr kesalahan muatan

```
{
 "statusCode": int,
 "errorCode": "string",
 "errorMessage": "string"
}
```

#### statusCode

Kode status.

#### errorCode

Kode kesalahan.

#### errorMessage

Pesan kesalahan.

## CreateKeysAndCertificate

Membuat kunci baru dan sertifikat. AWS IoT menyediakan sertifikat klien yang ditandatangani oleh otoritas sertifikat Amazon Root (CA). Sertifikat baru memiliki PENDING\_ACTIVATION status. Saat Anda menelepon RegisterThing untuk menyediakan sesuatu dengan sertifikat ini, status sertifikat berubah menjadi ACTIVE atau INACTIVE seperti yang dijelaskan dalam templat.

### Note

Untuk keamanan, yang `certificateOwnershipToken` dikembalikan [CreateKeysAndCertificate](#) kedaluwarsa setelah satu jam. [RegisterThing](#) harus dipanggil sebelum `certificateOwnershipToken` kedaluwarsa. Jika sertifikat yang dibuat oleh [CreateKeysAndCertificate](#) belum diaktifkan dan dilampirkan pada kebijakan atau sesuatu pada saat token kedaluwarsa, sertifikat akan dihapus. Jika token kedaluwarsa, perangkat dapat menelepon [CreateKeysAndCertificate](#) untuk menghasilkan sertifikat baru.

## CreateKeysAndCertificatepermintaan

Publikasikan pesan \$aws/certificates/create/*payload-format* dengan muatan pesan kosong.

payload-format

Format payload pesan sebagai cbor atau json.

## CreateKeysAndCertificaterespon

Berlangganan \$aws/certificates/create/*payload-format*/accepted.

payload-format

Format payload pesan sebagai cbor atau json.

## CreateKeysAndCertificaterespon

```
{
 "certificateId": "string",
 "certificatePem": "string",
 "privateKey": "string",
 "certificateOwnershipToken": "string"
}
```

certificateId

ID sertifikat.

certificatePem

Data sertifikat, dalam format PEM.

privateKey

Kunci privat.

certificateOwnershipToken

Token untuk membuktikan kepemilikan sertifikat selama penyediaan.

## CreateKeysAndCertificate kesalahan

Untuk menerima tanggapan kesalahan, berlangganan `$aws/certificates/create/payload-format/rejected`.

`payload-format`

Format payload pesan sebagai cbor atau json.

### CreateKeysAndCertificatekesalahan muatan

```
{
 "statusCode": int,
 "errorCode": "string",
 "errorMessage": "string"
}
```

`statusCode`

Kode status.

`errorCode`

Kode kesalahan.

`errorMessage`

Pesan kesalahan.

## RegisterThing

Ketentuan sesuatu menggunakan template yang telah ditentukan sebelumnya.

### RegisterThing permintaan

Publikasikan pesan di `$aws/provisioning-templates/templateName/provision/payload-format`.

`payload-format`

Format payload pesan sebagai cbor atau json.

## templateName

Nama template penyediaan.

## RegisterThing minta muatan

```
{
 "certificateOwnershipToken": "string",
 "parameters": {
 "string": "string",
 ...
 }
}
```

## certificateOwnershipToken

Token untuk membuktikan kepemilikan sertifikat. AWS IoT menghasilkan token saat Anda membuat sertifikat melalui MQTT.

## parameters

Tidak wajib. Pasangan nilai kunci dari perangkat yang digunakan oleh [kait pra-penyediaan untuk mengevaluasi permintaan](#) pendaftaran.

## RegisterThing respon

Berlangganan\$aws/provisioning-templates/*templateName*/provision/*payload-format*/accepted.

## payload-format

Format payload pesan sebagai cbor ataujson.

## templateName

Nama template penyediaan.

## RegisterThing muatan respons

```
{
 "deviceConfiguration": {
```

```
 "string": "string",
 ...
 },
 "thingName": "string"
}
```

## deviceConfiguration

Konfigurasi perangkat didefinisikan dalam template.

## thingName

Nama hal IoT yang dibuat selama penyediaan.

## RegisterThing respon kesalahan

Untuk menerima tanggapan kesalahan, berlangganan `aws/provisioning-templates/templateName/provision/payload-format/rejected`.

## payload-format

Format payload pesan sebagai cbor atau json.

## templateName

Nama template penyediaan.

## RegisterThing muatan respons kesalahan

```
{
 "statusCode": int,
 "errorCode": "string",
 "errorMessage": "string"
}
```

## statusCode

Kode status.

## errorCode

Kode kesalahan.

## errorMessage

Pesan kesalahan.

## Pengindeksan armada

Anda dapat menggunakan pengindeksan armada untuk mengindeks, mencari, dan mengumpulkan data perangkat Anda dari sumber berikut: registri, [AWS IoT Device Shadow AWS IoT](#), [AWS IoT konektivitas](#), [Katalog Paket Perangkat Lunak AWS IoT Manajemen Perangkat Lunak](#), dan pelanggaran. [AWS IoT Device Defender](#) Anda dapat menanyakan sekelompok perangkat, dan statistik agregat pada catatan perangkat yang didasarkan pada kombinasi atribut perangkat yang berbeda, termasuk status, konektivitas, dan pelanggaran perangkat. Dengan pengindeksan armada, Anda dapat mengatur, menyelidiki, dan memecahkan masalah armada perangkat Anda.

Pengindeksan armada menyediakan kemampuan berikut.

## Mengelola pembaruan indeks

Anda dapat mengatur indeks armada untuk mengindeks pembaruan untuk grup hal Anda, pendaftar benda, bayangan perangkat, konektivitas perangkat, dan pelanggaran perangkat. Saat Anda mengaktifkan pengindeksan armada, AWS IoT buat indeks untuk hal-hal atau grup benda Anda. `AWS_Things` adalah indeks yang dibuat untuk semua hal Anda. `AWS_ThingGroups` adalah indeks yang berisi semua kelompok hal Anda. Setelah pengindeksan armada aktif, Anda dapat menjalankan kueri pada indeks Anda. Misalnya, Anda dapat menemukan semua perangkat yang genggam dan memiliki daya tahan baterai lebih dari 70 persen. AWS IoT memperbarui indeks terus menerus dengan data terbaru Anda. Untuk informasi selengkapnya, lihat [Mengelola pengindeksan armada](#).

## Mencari di seluruh sumber data

Anda dapat membuat string kueri berdasarkan [bahasa kueri](#) dan menggunakannya untuk mencari di seluruh sumber data. Anda juga perlu mengonfigurasi sumber data dalam pengaturan pengindeksan armada sehingga konfigurasi pengindeksan berisi sumber data yang ingin Anda cari. String kueri menjelaskan hal-hal yang ingin Anda temukan. Anda dapat membuat kueri dengan menggunakan bidang `AWS` terkelola, bidang khusus, dan atribut apa pun dari sumber data yang diindeks. Untuk informasi selengkapnya tentang sumber data yang mendukung pengindeksan armada, lihat [Mengelola pengindeksan hal](#).

## Kueri untuk data agregat

Anda dapat mencari data agregat dan mengembalikan statistik, persentil, kardinalitas, atau daftar hal-hal dengan kueri penelusuran tentang bidang tertentu. Anda dapat menjalankan agregasi pada



bidang AWS terkelola atau atribut apa pun yang Anda konfigurasi sebagai bidang kustom dalam pengaturan pengindeksan armada. Untuk informasi selengkapnya tentang kueri agregasi, lihat [Menanyakan data agregat](#).

## Memantau data agregat dan membuat alarm dengan menggunakan metrik armada

Anda dapat menggunakan metrik armada untuk mengirim data agregat CloudWatch secara otomatis, menganalisis tren, dan membuat alarm untuk memantau status agregat armada Anda berdasarkan ambang batas yang telah ditentukan sebelumnya. Untuk informasi selengkapnya tentang metrik armada, lihat [Metrik armada](#).

## Mengelola pengindeksan armada

Pengindeksan armada mengelola dua jenis indeks untuk Anda: pengindeksan benda dan pengindeksan grup benda.

### Pengindeksan hal

Indeks yang dibuat untuk semua barang Anda disebut `AWS_Things`. Pengindeksan hal mendukung sumber data berikut: data [AWS IoT registri](#), data [AWS IoT Device Shadow](#), data [AWS IoT konektivitas](#), dan data [AWS IoT Device Defender](#) pelanggaran. Dengan menambahkan sumber data ini ke konfigurasi pengindeksan armada, Anda dapat mencari sesuatu, meminta data agregat, dan membuat grup benda dinamis dan metrik armada berdasarkan kueri penelusuran Anda.

Registry -AWS IoT menyediakan registri yang membantu Anda mengelola berbagai hal. Anda dapat menambahkan data registri ke konfigurasi pengindeksan armada untuk mencari perangkat berdasarkan nama benda, deskripsi, dan atribut registri lainnya. Untuk informasi selengkapnya tentang registri, lihat [Cara mengelola sesuatu dengan registri](#).

Shadow -[Layanan AWS IoT Device Shadow](#) menyediakan bayangan yang membantu Anda menyimpan data status perangkat Anda. Pengindeksan hal mendukung bayangan klasik tanpa nama dan bayangan bernama. Untuk mengindeks bayangan bernama, aktifkan pengaturan bayangan bernama Anda dan tentukan nama bayangan Anda dalam konfigurasi pengindeksan benda. Secara default, Anda dapat menambahkan hingga 10 nama bayangan per Akun AWS. Untuk melihat cara meningkatkan jumlah batas nama bayangan, lihat [AWS IoT Device Management Kuota](#) di Referensi AWS Umum.

Untuk menambahkan bayangan bernama untuk pengindeksan:

- Jika Anda menggunakan [AWS IoT konsol](#), aktifkan pengindeksan Thing, pilih Tambahkan bayangan bernama, dan tambahkan nama bayangan Anda melalui pemilihan bayangan Bernama.
- Jika Anda menggunakan AWS Command Line Interface (AWS CLI), atur `namedShadowIndexingMode` menjadi `ON`, dan tentukan nama bayangan di [IndexingFilter](#). Untuk melihat contoh perintah CLI, lihat [Mengelola pengindeksan hal](#).

#### Important

20 Juli 2022 adalah rilis General Availability (GA) dari integrasi pengindeksan armada Manajemen AWS IoT Perangkat dengan bayangan AWS IoT Core bernama dan AWS IoT Device Defender mendeteksi pelanggaran. Dengan rilis GA ini, Anda dapat mengindeks bayangan bernama tertentu dengan menentukan nama bayangan. Jika Anda menambahkan bayangan bernama untuk pengindeksan selama periode pratinjau publik fitur ini dari 30 November 2021 hingga 19 Juli 2022, kami mendorong Anda untuk mengonfigurasi ulang pengaturan pengindeksan armada Anda dan memilih nama bayangan tertentu untuk mengurangi biaya pengindeksan dan mengoptimalkan kinerja.

Untuk informasi selengkapnya tentang bayangan, lihat [Layanan AWS IoT Device Shadow](#).

Konektivitas -Data konektivitas perangkat membantu Anda mengidentifikasi status koneksi perangkat Anda. Data konektivitas ini didorong oleh peristiwa [siklus hidup](#). Saat klien menghubungkan atau memutuskan sambungan, AWS IoT menerbitkan peristiwa siklus hidup dengan pesan ke topik MQTT. Pesan sambungkan atau putus sambungan dapat berupa daftar elemen JSON yang memberikan rincian status koneksi. Untuk informasi selengkapnya tentang konektivitas perangkat, lihat [Peristiwa siklus hidup](#).

Pelanggaran Device Defender - data AWS IoT Device Defender pelanggaran membantu mengidentifikasi perilaku perangkat anomali terhadap perilaku normal yang Anda tentukan dalam Profil Keamanan. Profil Keamanan berisi serangkaian perilaku perangkat yang diharapkan. Setiap perilaku menggunakan metrik yang menentukan perilaku normal perangkat Anda. Untuk informasi selengkapnya tentang pelanggaran Device Defender, lihat [AWS IoT Device Defender mendeteksi](#).

Untuk informasi selengkapnya, lihat [Mengelola pengindeksan hal](#).

## Pengindeksan kelompok hal

AWS\_ThingGroups adalah indeks yang berisi semua grup barang dan grup penagihan Anda. Anda dapat menggunakan indeks ini untuk mencari grup berdasarkan nama grup, deskripsi, atribut, dan semua nama grup induk.

Untuk informasi selengkapnya, lihat [Mengelola pengindeksan grup hal](#).

## Bidang yang dikelola

Bidang terkelola berisi data yang terkait dengan hal-hal, grup benda, bayangan perangkat, konektivitas perangkat, dan pelanggaran Device Defender. AWS IoT mendefinisikan tipe data di bidang terkelola. Anda menentukan nilai dari setiap bidang terkelola saat Anda membuat AWS IoT sesuatu. Misalnya, nama benda, grup benda, dan deskripsi benda semuanya adalah bidang yang dikelola. Pengindeksan armada mengindeks bidang terkelola berdasarkan mode pengindeksan yang Anda tentukan. Bidang terkelola tidak dapat diubah atau ditampilkan `customFields`. Untuk informasi selengkapnya, lihat [Bidang kustom](#).

Berikut daftar bidang terkelola untuk pengindeksan hal:

- Bidang terkelola untuk registri

```
"managedFields" : [
 {name:thingId, type:String},
 {name:thingName, type:String},
 {name:registry.version, type:Number},
 {name:registry.thingTypeName, type:String},
 {name:registry.thingGroupNames, type:String},
]
```

- Bidang terkelola untuk bayangan klasik yang tidak disebutkan namanya

```
"managedFields" : [
 {name:shadow.version, type:Number},
 {name:shadow.hasDelta, type:Boolean}
]
```

- Bidang terkelola untuk bayangan bernama

```
"managedFields" : [
 {name:shadow.name.shadowName.version, type:Number},
```

```
{name:shadow.name.shadowName.hasDelta, type:Boolean}
]
```

- Bidang terkelola untuk konektivitas benda

```
"managedFields" : [
 {name:connectivity.timestamp, type:Number},
 {name:connectivity.version, type:Number},
 {name:connectivity.connected, type:Boolean},
 {name:connectivity.disconnectReason, type:String}
]
```

- Bidang terkelola untuk 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}
]
```

- Bidang terkelola untuk grup benda

```
"managedFields" : [
 {name:description, type:String},
 {name:parentGroupNames, type:String},
 {name:thingGroupId, type:String},
 {name:thingGroupName, type:String},
 {name:version, type:Number},
]
```

Tabel berikut mencantumkan bidang terkelola yang tidak dapat dicari.

| Sumber data         | Bidang terkelola yang tidak dapat dicari |
|---------------------|------------------------------------------|
| Registri            | registry.version                         |
| Bayangan tanpa nama | shadow.version                           |
| Bayangan bernama    | shadow.name.*.version                    |

|                      |                                          |
|----------------------|------------------------------------------|
| Sumber data          | Bidang terkelola yang tidak dapat dicari |
| Pertahanan Perangkat | <code>deviceDefender.version</code>      |
| Kelompok benda       | <code>version</code>                     |

## Bidang kustom

Anda dapat menggabungkan atribut benda, data Device Shadow, dan data pelanggaran Device Defender dengan membuat bidang khusus untuk mengindeksnya. `customFieldsAtribut` adalah daftar nama bidang dan pasangan tipe data. Anda dapat melakukan kueri agregasi berdasarkan tipe data. Mode pengindeksan yang Anda pilih mempengaruhi bidang dapat ditentukan. `customFields` Misalnya, jika Anda menentukan mode `REGISTRY` pengindeksan, Anda tidak dapat menentukan bidang kustom dari bayangan benda. [Anda dapat menggunakan perintah CLI `update-indexing-configuration` untuk membuat atau memperbarui bidang kustom \(lihat perintah contoh di `Memperbarui contoh konfigurasi pengindeksan`\).](#)

- Nama bidang kustom

Nama bidang khusus untuk atribut grup benda dan benda dimulai dengan `attributes.`, diikuti dengan nama atribut. Jika pengindeksan bayangan tanpa nama aktif, hal-hal dapat memiliki nama bidang khusus yang dimulai dengan `shadow.desired` atau `shadow.reported`, diikuti oleh nama nilai data bayangan yang tidak disebutkan namanya. Jika pengindeksan bayangan bernama aktif, hal-hal dapat memiliki nama bidang khusus yang dimulai dengan `shadow.name.*.desired.` atau `shadow.name.*.reported.`, diikuti oleh nilai data bayangan bernama. Jika pengindeksan pelanggaran Device Defender aktif, hal-hal dapat memiliki nama bidang khusus yang dimulai dengan `deviceDefender.`, diikuti dengan nilai data pelanggaran Device Defender.

Atribut atau nama nilai data yang mengikuti awalan hanya dapat memiliki karakter alfanumerik, `-` (tanda hubung), dan `_` (garis bawah). Itu tidak bisa memiliki spasi.

Jika ada inkonsistensi tipe antara bidang kustom dalam konfigurasi Anda dan nilai yang diindeks, pengindeksan armada mengabaikan nilai yang tidak konsisten untuk kueri agregasi. CloudWatch Log sangat membantu saat memecahkan masalah kueri agregasi. Untuk informasi selengkapnya, lihat [Memecahkan masalah kueri agregasi untuk layanan pengindeksan armada](#).

- Jenis bidang kustom

Jenis bidang kustom memiliki nilai yang didukung berikut: `Number`, `String`, dan `Boolean`.

## Kelola pengindeksan hal

Indeks yang dibuat untuk semua barang Anda adalah `AWS_Things`. Anda dapat mengontrol apa yang akan diindeks dari sumber data berikut: data [AWS IoT registri](#), data [AWS IoT Device Shadow](#), data [AWS IoT konektivitas](#), dan data [AWS IoT Device Defender](#) pelanggaran.

Dalam topik ini:

- [Mengaktifkan pengindeksan hal](#)
- [Menggambarkan indeks sesuatu](#)
- [Menanyakan indeks sesuatu](#)
- [Pembatasan dan batasan](#)
- [Otorisasi](#)

## Mengaktifkan pengindeksan hal

Anda menggunakan perintah CLI [update-indexing-configuration](#) atau operasi Configuration API untuk membuat indeks [UpdateIndexingdan](#) mengontrol konfigurasinya. `AWS_Things` Dengan menggunakan parameter `--thing-indexing-configuration` (`thingIndexingConfiguration`), Anda mengontrol jenis data (misalnya, registri, bayangan, data konektivitas perangkat, dan data pelanggaran Device Defender) yang diindeks.

`--thing-indexing-configuration` Parameter mengambil string dengan struktur berikut:

```
{
 "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"
 }
]
}
```

## Mode pengindeksan hal

Anda dapat menentukan mode pengindeksan hal yang berbeda dalam konfigurasi pengindeksan Anda, tergantung pada sumber data apa yang ingin Anda indeks dan cari perangkat dari:

- `thingIndexingMode`: Kontrol jika registri atau bayangan diindeks. Kapan `thingIndexingMode` diatur menjadi OFF, pengindeksan hal dinonaktifkan.
- `thingConnectivityIndexingMode`: Menentukan apakah data konektivitas hal diindeks.
- `deviceDefenderIndexingMode`: Menentukan apakah data pelanggaran Device Defender diindeks.
- `namedShadowIndexingMode`: Menentukan jika bernama data bayangan diindeks. Untuk memilih bayangan bernama untuk ditambahkan ke konfigurasi pengindeksan armada Anda, setelah `namedShadowIndexingMode` menjadi ON dan tentukan nama bayangan bernama Anda. [filter](#)

Tabel di bawah ini menunjukkan nilai yang valid untuk setiap mode pengindeksan dan sumber data yang diindeks untuk setiap nilai.

| Atribut                       | Nilai valid         | Registri | Bayangan | Konektivitas | Pelanggaran DD | Bernama bayangan |
|-------------------------------|---------------------|----------|----------|--------------|----------------|------------------|
| thingIndexingMode             | MATI                |          |          |              |                |                  |
|                               | REGISTRASI          | ✓        |          |              |                |                  |
|                               | REGISTRY_AND_SHADOW | ✓        | ✓        |              |                |                  |
| thingConnectivityIndexingMode | Tidak ditentukan.   |          |          |              |                |                  |
|                               | MATI                |          |          |              |                |                  |
|                               | STATUS              |          |          | ✓            |                |                  |
| deviceDefenderIndexingMode    | Tidak ditentukan.   |          |          |              |                |                  |
|                               | MATI                |          |          |              |                |                  |
|                               | PELANGGARAN         |          |          |              | ✓              |                  |
| namedShadowIndexingMode       | Tidak ditentukan.   |          |          |              |                |                  |
|                               | MATI                |          |          |              |                |                  |
|                               | PADA                |          |          |              |                | ✓                |

## Bidang terkelola dan bidang kustom

### Bidang yang dikelola

Bidang terkelola berisi data yang terkait dengan hal-hal, grup benda, bayangan perangkat, konektivitas perangkat, dan pelanggaran Device Defender. AWS IoT mendefinisikan tipe data di bidang terkelola. Anda menentukan nilai dari setiap bidang terkelola saat Anda membuat AWS IoT



sesuatu. Misalnya, nama benda, grup benda, dan deskripsi benda semuanya adalah bidang yang dikelola. Pengindeksan armada mengindeks bidang terkelola berdasarkan mode pengindeksan yang Anda tentukan. Bidang terkelola tidak dapat diubah atau ditampilkan `customFields`.

## Bidang kustom

Anda dapat menggabungkan atribut, data Device Shadow, dan data pelanggaran Device Defender dengan membuat bidang khusus untuk mengindeksnya. `customFieldsAtribut` adalah daftar nama bidang dan pasangan tipe data. Anda dapat melakukan kueri agregasi berdasarkan tipe data. Mode pengindeksan yang Anda pilih mempengaruhi bidang dapat ditentukan. `customFields` Misalnya, jika Anda menentukan mode REGISTRY pengindeksan, Anda tidak dapat menentukan bidang kustom dari bayangan benda. [Anda dapat menggunakan perintah CLI `update-indexing-configuration` untuk membuat atau memperbarui bidang kustom \(lihat perintah contoh di `Memperbarui contoh konfigurasi pengindeksan`\)](#). Untuk informasi selengkapnya, lihat [Bidang kustom](#).

## Filter pengindeksan

Filter pengindeksan menyediakan pilihan tambahan untuk bayangan bernama dan data geolokasi.

### **namedShadowNames**

Untuk menambahkan bayangan bernama ke konfigurasi pengindeksan armada Anda, atur `namedShadowIndexingMode` menjadi ON dan tentukan nama bayangan bernama Anda di `namedShadowNames` filter.

### Contoh

```
"filter": {
 "namedShadowNames": ["namedShadow1", "namedShadow2"]
}
```

### **geoLocations**

Untuk menambahkan data geolokasi ke konfigurasi pengindeksan armada Anda:

- Jika data geolokasi Anda disimpan dalam bayangan klasik (tanpa nama), atur `thingIndexingMode` menjadi REGISTRY\_AND\_SHADOW, dan tentukan data geolokasi Anda dalam filter. `geoLocations`

Contoh filter di bawah ini menentukan objek GeoLocation dalam bayangan klasik (tanpa nama):

```
"filter": {
```

```

 "geoLocations": [
 {
 "name": "shadow.reported.location",
 "order": "LonLat"
 }
]
 }

```

- Jika data geolokasi Anda disimpan dalam bayangan bernama, atur `namedShadowIndexingMode` menjadi ON, tambahkan nama bayangan di `namedShadowNames` filter, dan tentukan data geolokasi Anda di `geoLocations` filter.

Contoh filter di bawah ini menentukan objek `GeoLocation` dalam bayangan bernama `()nameShadow1`:

```

"filter": {
 "namedShadowNames": ["namedShadow1"],
 "geoLocations": [
 {
 "name": "shadow.name.namedShadow1.reported.location",
 "order": "LonLat"
 }
]
}

```

Untuk informasi selengkapnya, lihat [IndexingFilter](#) dari Referensi AWS IoT API.

Memperbarui contoh konfigurasi pengindeksan

Untuk memperbarui konfigurasi pengindeksan Anda, gunakan perintah AWS IoT `update-indexing-configuration` CLI. Contoh berikut menunjukkan cara menggunakan `update-indexing-configuration`.

Sintaks pendek:

```

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}]']
```

### Sintaks 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}] }] }'
```

Perintah ini tidak menghasilkan output apa pun.

Untuk memeriksa status indeks benda, jalankan perintah `describe-index` CLI:

```
aws iot describe-index --index-name "AWS_Things"
```

Output dari `describe-index` perintah terlihat seperti berikut:

```
{
 "indexName": "AWS_Things",
 "indexStatus": "ACTIVE",
 "schema": "MULTI_INDEXING_MODE"
}
```

#### Note

Diperlukan waktu sejenak bagi pengindeksan armada untuk memperbarui indeks armada. Kami sarankan menunggu hingga `indexStatus` acara ACTIVE sebelum menggunakannya. Anda dapat memiliki nilai yang berbeda di bidang skema tergantung pada sumber data apa yang telah Anda konfigurasi. Untuk informasi selengkapnya, lihat [Menjelaskan indeks sesuatu](#).

Untuk mendapatkan detail konfigurasi pengindeksan hal Anda, jalankan perintah `get-indexing-configuration` CLI:

```
aws iot get-indexing-configuration
```

Output dari `get-indexing-configuration` perintah terlihat seperti berikut:

```
{
 "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"
}
}
```

Untuk memperbarui bidang khusus, Anda dapat menjalankan `update-indexing-configuration` perintah. Contohnya adalah sebagai berikut:

```
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}]'
```

Perintah ini ditambahkan `shadow.desired.intensity` ke konfigurasi pengindeksan.

#### Note

Memperbarui konfigurasi pengindeksan bidang kustom menimpa semua bidang kustom yang ada. Pastikan untuk menentukan semua bidang khusus saat menelepon `update-indexing-configuration`.

Setelah indeks dibangun kembali, Anda dapat menggunakan kueri agregasi pada bidang yang baru ditambahkan, mencari data registri, data bayangan, dan data status konektivitas benda.

Saat mengubah mode pengindeksan, pastikan semua bidang kustom Anda valid dengan menggunakan mode pengindeksan baru. Misalnya, jika Anda mulai menggunakan `REGISTRY_AND_SHADOW` mode dengan bidang khusus yang disebut `shadow.desired.temperature`, Anda harus menghapus bidang `shadow.desired.temperature` khusus sebelum mengubah mode pengindeksan

menjadi **REGISTRY**. Jika konfigurasi pengindeksan Anda berisi bidang khusus yang tidak diindeks oleh mode pengindeksan, pembaruan gagal.

## Menggambarkan indeks sesuatu

Perintah berikut menunjukkan cara menggunakan perintah `describe-index` CLI untuk mengambil status indeks benda saat ini.

```
aws iot describe-index --index-name "AWS_Things"
```

Respons perintah dapat terlihat seperti berikut:

```
{
 "indexName": "AWS_Things",
 "indexStatus": "BUILDING",
 "schema": "REGISTRY_AND_SHADOW_AND_CONNECTIVITY_STATUS"
}
```

Pertama kali Anda mengindeks armada, AWS IoT membangun indeks Anda. `indexStatus` kapan dalam **BUILDING** keadaan, Anda tidak dapat menanyakan indeks. Indeks `schema` for the things menunjukkan jenis data (**REGISTRY\_AND\_SHADOW\_AND\_CONNECTIVITY\_STATUS**) yang diindeks.

Mengubah konfigurasi indeks Anda menyebabkan indeks dibangun kembali. Selama proses ini, `indexStatus` adalah **REBUILDING**. Anda dapat menjalankan kueri pada data dalam indeks things saat sedang dibangun kembali. Misalnya, jika Anda mengubah konfigurasi indeks dari **REGISTRY** ke **REGISTRY\_AND\_SHADOW** saat indeks sedang dibangun kembali, Anda dapat meminta data registri, termasuk pembaruan terbaru. Namun, Anda tidak dapat menanyakan data bayangan hingga pembangunan kembali selesai. Jumlah waktu yang diperlukan untuk membangun atau membangun kembali indeks tergantung pada jumlah data.

Anda dapat melihat nilai yang berbeda di bidang skema tergantung pada sumber data yang telah Anda konfigurasi. Tabel berikut menunjukkan nilai skema yang berbeda dan deskripsi yang sesuai:

| Skema      | Deskripsi                                               |
|------------|---------------------------------------------------------|
| MATI       | Tidak ada sumber data yang dikonfigurasi atau diindeks. |
| REGISTRASI | Data registri diindeks.                                 |

| Skema                                                              | Deskripsi                                                                                                                              |
|--------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------|
| REGISTRY_AND_SHADOW                                                | Data registri dan data bayangan (klasik) yang tidak disebutkan namanya diindeks.                                                       |
| REGISTRY_AND_CONNECTIVITY                                          | Data registri dan data konektivitas diindeks.                                                                                          |
| REGISTRY_AND_SHADOW_AND_CONNECTIVITY_STATUS_REGISTRY_SHADOW_STATUS | Data registri, data bayangan tanpa nama (klasik), dan data konektivitas diindeks.                                                      |
| MULTI_INDEXING_MODE                                                | Bayangan bernama atau Device Defender data pelanggaran diindeks, selain registri, bayangan tanpa nama (klasik) atau data konektivitas. |

## Menanyakan indeks sesuatu

Gunakan perintah `search-index` CLI untuk menanyakan data dalam indeks.

```
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":"AQFCuvk7zZ3D9p0YMbFCeHbdZ+h=G"
}
```

Dalam respons JSON, "connectivity" (sebagaimana diaktifkan oleh `thingConnectivityIndexingMode=STATUS` pengaturan) memberikan nilai Boolean, stempel waktu, dan `DisconnectReason` yang menunjukkan apakah perangkat terhubung ke AWS IoT Core Perangkat "mything1" terputus (`false`) pada waktu POSIX 1556649874716 karena `CONNECTION_LOST`. Untuk informasi selengkapnya tentang alasan pemutusan sambungan, lihat Peristiwa [siklus hidup](#).

```
"connectivity": {
 "connected":false,
 "timestamp":1556649874716,
 "disconnectReason": "CONNECTION_LOST"
}
```

Perangkat "mything2" terhubung (`true`) pada waktu 1556649855046 POSIX:

```
"connectivity": {
 "connected":true,
 "timestamp":1556649855046
}
```

Stempel waktu diberikan dalam milidetik sejak zaman, jadi 1556649855046 mewakili 18:44:15.046 pada hari Selasa, 30 April 2019 (UTC).

#### Important

Jika perangkat telah terputus selama kurang lebih satu jam, "timestamp" nilai dan "disconnectReason" nilai status konektivitas mungkin hilang.

## Pembatasan dan batasan

Ini adalah batasan dan batasan untuk `AWS_Things`.

### Bidang bayangan dengan tipe kompleks

Bidang bayangan diindeks hanya jika nilai bidang adalah tipe sederhana, seperti objek JSON yang tidak berisi array, atau array yang seluruhnya terdiri dari tipe sederhana. Tipe sederhana berarti string, angka, atau salah satu literal `true` atau `false`. Misalnya, mengingat status bayangan berikut, nilai bidang `"palette"` tidak diindeks karena itu adalah array yang berisi item dari tipe kompleks. Nilai bidang `"colors"` diindeks karena setiap nilai dalam array adalah string.

```
{
 "state": {
 "reported": {
 "switched": "ON",
 "colors": ["RED", "GREEN", "BLUE"],
 "palette": [
 {
 "name": "RED",
 "intensity": 124
 },
 {
 "name": "GREEN",
 "intensity": 68
 },
 {
 "name": "BLUE",
 "intensity": 201
 }
]
 }
 }
}
```

### Nama bidang bayangan bersarang

Nama-nama bidang bayangan bersarang disimpan sebagai string terbatas periode (.). Misalnya, diberikan dokumen bayangan:

```
{
 "state": {
```

```

 "desired": {
 "one": {
 "two": {
 "three": "v2"
 }
 }
 }
 }
}

```

Nama bidang `three` disimpan sebagai `desired.one.two.three`. Jika Anda juga memiliki dokumen bayangan, itu disimpan seperti ini:

```

{
 "state": {
 "desired": {
 "one.two.three": "v2"
 }
 }
}

```

Keduanya cocok dengan kueri untuk `shadow.desired.one.two.three:v2`. Sebagai praktik terbaik, jangan gunakan titik dalam nama bidang bayangan.

### Metadata bayangan

Bidang di bagian metadata bayangan diindeks, tetapi hanya jika bidang yang sesuai di bagian bayangan diindeks. "state" (Pada contoh sebelumnya, "palette" bidang di bagian metadata bayangan juga tidak diindeks.)

### Perangkat yang tidak terdaftar

[Pengeindeksan armada mengindeks status konektivitas untuk perangkat yang `clientId` koneksinya sama dengan hal `thingName` yang terdaftar di Registry.](#)

### Bayangan yang tidak terdaftar

Jika Anda menggunakan [UpdateThingShadow](#) untuk membuat bayangan menggunakan nama benda yang belum terdaftar di AWS IoT akun Anda, bidang dalam bayangan ini tidak diindeks. Ini berlaku untuk bayangan klasik tanpa nama dan bayangan bernama.

## Nilai numerik

Jika ada data registri atau bayangan yang dikenali oleh layanan sebagai nilai numerik, itu diindeks seperti itu. Anda dapat membentuk kueri yang melibatkan rentang dan operator perbandingan pada nilai numerik (misalnya, "attribute.foo<5" atau "shadow.reported.foo:[75 TO 80]"). Untuk dikenali sebagai numerik, nilai data harus berupa nomor JSON tipe literal yang valid. Nilai dapat berupa bilangan bulat dalam rentang  $-2^{53} \dots 2^{53}-1$ , floating point presisi ganda dengan notasi eksponensial opsional, atau bagian dari array yang hanya berisi nilai-nilai ini.

## Nilai nol

Nilai nol tidak diindeks.

## Nilai maksimum

Jumlah maksimum bidang kustom untuk kueri agregasi adalah 5.

Jumlah maksimum persentil yang diminta untuk kueri agregasi adalah 100.

## Otorisasi

Anda dapat menentukan indeks things sebagai Amazon Resource Name (ARN) dalam tindakan AWS IoT kebijakan, sebagai berikut.

| Tindakan          | Sumber Daya                                                                                 |
|-------------------|---------------------------------------------------------------------------------------------|
| iot:SearchIndex   | Indeks ARN (misalnya, <code>arn:aws:iot: <i>your-aws-region</i> :index/AWS_Things</code> ). |
| iot:DescribeIndex | Indeks ARN (misalnya, <code>arn:aws:iot: <i>your-aws-region</i> :index/AWS_Things</code> ). |

### Note

Jika Anda memiliki izin untuk menanyakan indeks armada, Anda dapat mengakses data berbagai hal di seluruh armada.

## Kelola pengindeksan grup hal

`AWS_ThingGroups` adalah indeks yang berisi semua grup barang dan grup penagihan Anda. Anda dapat menggunakan indeks ini untuk mencari grup berdasarkan nama grup, deskripsi, atribut, dan semua nama grup induk.

### Mengaktifkan pengindeksan grup hal

Anda dapat menggunakan `thing-group-indexing-configuration` pengaturan di API [UpdateIndexingKonfigurasi](#) untuk membuat `AWS_ThingGroups` indeks dan mengontrol konfigurasinya. Anda dapat menggunakan [GetIndexingConfiguration API untuk mengambil konfigurasi](#) pengindeksan saat ini.

Untuk memperbarui konfigurasi pengindeksan grup benda, jalankan perintah CLI `update-indexing-configuration`:

```
aws iot update-indexing-configuration --thing-group-indexing-configuration
thingGroupIndexingMode=ON
```

Anda juga dapat memperbarui konfigurasi untuk pengindeksan grup benda dan benda dalam satu perintah, sebagai berikut:

```
aws iot update-indexing-configuration --thing-indexing-configuration
thingIndexingMode=REGISTRY --thing-group-indexing-configuration
thingGroupIndexingMode=ON
```

Berikut ini adalah nilai yang valid untuk `thingGroupIndexingMode`.

#### MATI

Tidak ada indeks pengindeks/hapus.

#### PADA

Buat atau konfigurasikan `AWS_ThingGroups` indeks.

Untuk mengambil konfigurasi pengindeksan grup benda dan benda saat ini, jalankan perintah CLI `get-indexing-configuration`:

```
aws iot get-indexing-configuration
```

Respons perintah terlihat seperti berikut:

```
{
 "thingGroupIndexingConfiguration": {
 "thingGroupIndexingMode": "ON"
 }
}
```

## Menggambarkan indeks grup

Untuk mengambil status `AWS_ThingGroups` indeks saat ini, gunakan perintah `describe-index` CLI:

```
aws iot describe-index --index-name "AWS_ThingGroups"
```

Respons perintah terlihat seperti berikut:

```
{
 "indexStatus": "ACTIVE",
 "indexName": "AWS_ThingGroups",
 "schema": "THING_GROUPS"
}
```

AWS IoT membangun indeks Anda saat pertama kali Anda mengindeks. Anda tidak dapat menanyakan indeks jika `indexStatus` ada `BUILDING`.

## Menanyakan indeks grup benda

Untuk kueri data dalam indeks, gunakan perintah `search-index` CLI:

```
aws iot search-index --index-name "AWS_ThingGroups" --query-string
"thingGroupName:mythinggroup*"
```

## Otorisasi

Anda dapat menentukan indeks grup benda sebagai ARN sumber daya dalam tindakan AWS IoT kebijakan, sebagai berikut.

| Tindakan                       | Sumber Daya                                                                                      |
|--------------------------------|--------------------------------------------------------------------------------------------------|
| <code>iot:SearchIndex</code>   | Indeks ARN (misalnya, <code>arn:aws:iot: <i>your-aws-region</i> :index/AWS_ThingGroups</code> ). |
| <code>iot:DescribeIndex</code> | Indeks ARN (misalnya, <code>arn:aws:iot: <i>your-aws-region</i> :index/AWS_ThingGroups</code> ). |

## Kueri untuk data agregat

AWS IoT menyediakan empat API (`GetStatistics`, `GetCardinality`, `GetPercentiles`, dan `GetBucketsAggregation`) yang memungkinkan Anda mencari data agregat armada perangkat.

### Note

Untuk masalah dengan nilai yang hilang atau tidak terduga untuk API agregasi, baca panduan [pemecahan masalah pengindeksan Armada](#).

## GetStatistics

[GetStatistics](#) API dan perintah `get-statistics` CLI mengembalikan hitungan, rata-rata, jumlah, minimum, maksimum, jumlah kuadrat, varians, dan standar deviasi untuk bidang agregat yang ditentukan.

Perintah `get-statistics` CLI mengambil parameter berikut:

`index-name`

Nama indeks yang akan dicari. Nilai default-nya adalah `AWS_Things`.

`query-string`

Query yang digunakan untuk mencari indeks. Anda dapat menentukan "\*" untuk mendapatkan hitungan semua hal yang diindeks di Anda. Akun AWS



## aggregationField

(Opsional) Bidang untuk agregat. Bidang ini harus berupa bidang terkelola atau kustom yang ditentukan saat Anda menelepon `update-indexing-configuration`. Jika Anda tidak menentukan bidang agregasi, `registry.version` digunakan sebagai bidang agregasi.

## query-version

Versi kueri yang akan digunakan. Nilai default-nya adalah `2017-09-30`.

Jenis bidang agregasi dapat mempengaruhi statistik yang dikembalikan.

## GetStatistics dengan nilai string

Jika Anda menggabungkan pada bidang string, pemanggilan akan `GetStatistics` mengembalikan jumlah perangkat yang memiliki atribut yang cocok dengan kueri. Sebagai contoh:

```
aws iot get-statistics --aggregation-field 'attributes.stringAttribute'
 --query-string '*'
```

Perintah ini mengembalikan jumlah perangkat yang berisi atribut bernama `stringAttribute`:

```
{
 "statistics": {
 "count": 3
 }
}
```

## GetStatistics dengan nilai Boolean

Saat Anda memanggil `GetStatistics` dengan bidang agregasi Boolean:

- **AVERAGE** adalah persentase perangkat yang cocok dengan kueri.
- **MINIMUM** adalah 0 atau 1 sesuai dengan aturan berikut:
  - Jika semua nilai untuk bidang agregasi adalah `false`, **MINIMUM** adalah 0.
  - Jika semua nilai untuk bidang agregasi adalah `true`, **MINIMUM** adalah 1.
  - Jika nilai untuk bidang agregasi adalah campuran dari `false` dan `true`, **MINIMUM** adalah 0.
- **MAKSIMUM** adalah 0 atau 1 sesuai dengan aturan berikut:
  - Jika semua nilai untuk bidang agregasi adalah `false`, **MAXIMUM** adalah 0.

- Jika semua nilai untuk bidang agregasi adalah `true`, `MAXIMUM` adalah 1.
- Jika nilai untuk bidang agregasi adalah campuran dari `false` dan `true`, `MAKSIMUM` adalah 1.
- `SUM` adalah jumlah dari bilangan bulat yang setara dengan nilai Boolean.
- `COUNT` adalah jumlah hal-hal yang cocok dengan kriteria string kueri dan berisi nilai bidang agregasi yang valid.

## GetStatistics dengan nilai numerik

Ketika Anda memanggil `GetStatistics` dan menentukan bidang agregasi jenis `Number`, `GetStatistics` mengembalikan nilai-nilai berikut:

### `count`

Hitungan hal-hal yang cocok dengan kriteria string kueri dan berisi nilai bidang agregasi yang valid.

### `rata-rata`

Rata-rata nilai numerik yang cocok dengan query.

### `sum`

Jumlah nilai numerik yang cocok dengan query.

### `minimum`

Nilai numerik terkecil yang cocok dengan kueri.

### `maksimum`

Nilai numerik terbesar yang cocok dengan kueri.

### `jumlah OfSquares`

Jumlah kuadrat dari nilai numerik yang cocok dengan query.

### `perbedaan`

Varians dari nilai numerik yang cocok dengan query. Varians dari satu set nilai adalah rata-rata kuadrat dari perbedaan setiap nilai dari nilai rata-rata himpunan.

### `stdDeviation`

Standar deviasi dari nilai numerik yang cocok dengan query. Standar deviasi dari satu set nilai adalah ukuran seberapa tersebar nilai-nilai tersebut.

Contoh berikut menunjukkan cara memanggil `get-statistics` dengan bidang kustom numerik.

```
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": 13472.222222222222,
 "stdDeviation": 116.06990230986766
 }
}
```

Untuk bidang agregasi numerik, jika nilai bidang melebihi nilai ganda maksimum, nilai statistik kosong.

## GetCardinality

[GetCardinality](#) API dan perintah `get-cardinality` CLI mengembalikan perkiraan jumlah nilai unik yang cocok dengan kueri. Misalnya, Anda mungkin ingin menemukan jumlah perangkat dengan tingkat baterai kurang dari 50 persen:

```
aws iot get-cardinality --index-name AWS_Things --query-string "batterylevel
> 50" --aggregation-field "shadow.reported.batterylevel"
```

Perintah ini mengembalikan jumlah hal dengan tingkat baterai lebih dari 50 persen:

```
{
 "cardinality": 100
}
```

`cardinality` selalu dikembalikan `get-cardinality` bahkan jika tidak ada bidang yang cocok. Sebagai contoh:

```
aws iot get-cardinality --query-string "thingName:Non-existent*"
```

```
--aggregation-field "attributes.customField_STR"
```

```
{
 "cardinality": 0
}
```

Perintah `get-cardinality` CLI mengambil parameter berikut:

`index-name`

Nama indeks yang akan dicari. Nilai default-nya adalah `AWS_Things`.

`query-string`

Query yang digunakan untuk mencari indeks. Anda dapat menentukan "\*" untuk mendapatkan hitungan semua hal yang diindeks di Anda. Akun AWS

`aggregationField`

Bidang untuk agregat.

`query-version`

Versi kueri yang akan digunakan. Nilai default-nya adalah `2017-09-30`.

## GetPercentiles

[GetPercentiles](#) API dan perintah `get-percentiles` CLI mengelompokkan nilai agregat yang cocok dengan kueri ke dalam pengelompokan persentil. Pengelompokan persentil default adalah: 1,5,25,50,75,95,99, meskipun Anda dapat menentukan sendiri saat menelepon.

`GetPercentiles` Fungsi ini mengembalikan nilai untuk setiap kelompok persentil yang ditentukan (atau pengelompokan persentil default). Grup persentil "1" berisi nilai bidang agregat yang terjadi di sekitar satu persen dari nilai yang cocok dengan kueri. Grup persentil "5" berisi nilai bidang agregat yang terjadi di sekitar lima persen dari nilai yang cocok dengan kueri, dan seterusnya. Hasilnya adalah perkiraan, semakin banyak nilai yang cocok dengan kueri, semakin akurat nilai persentil.

Contoh berikut menunjukkan cara memanggil perintah `get-percentiles` 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
 }
]
}
```

Perintah berikut menunjukkan output yang dikembalikan dari `get-percentiles` saat tidak ada dokumen yang cocok.

```
aws iot get-percentiles --query-string "thingName:Non-existent*"
 --aggregation-field "attributes.customField_NUM"
```

```
{
 "percentiles": []
}
```

Perintah `get-percentile` CLI mengambil parameter berikut:

`index-name`

Nama indeks yang akan dicari. Nilai default-nya adalah `AWS_Things`.

`query-string`

Query yang digunakan untuk mencari indeks. Anda dapat menentukan "\*" untuk mendapatkan hitungan semua hal yang diindeks di Anda. Akun AWS

`aggregationField`

Bidang agregat, yang harus `Number` bertipe.

`query-version`

Versi kueri yang akan digunakan. Nilai default-nya adalah `2017-09-30`.

`percents`

(Opsional) Anda dapat menggunakan parameter ini untuk menentukan pengelompokan persentil kustom.

## GetBucketsAgregasi

API [GetBucketsAgregasi](#) dan perintah `get-buckets-aggregation` CLI mengembalikan daftar bucket dan jumlah total hal yang sesuai dengan kriteria string kueri.

Contoh berikut menunjukkan cara memanggil perintah `get-buckets-aggregation` CLI.

```
aws iot get-buckets-aggregation --query-string '*' --index-name AWS_Things --
aggregation-field 'shadow.reported.batterylevelpercent' --buckets-aggregation-type
'termsAggregation={maxBuckets=5}'
```

Perintah ini mengembalikan yang berikut:

```
{
 "totalCount": 20,
 "buckets": [
 {
 "keyValue": "100",
 "count": 12
 },
 {
 "keyValue": "90",
 "count": 5
 },
 {
 "keyValue": "75",
 "count": 3
 }
]
}
```

Perintah `get-buckets-aggregation` CLI mengambil parameter berikut:

`index-name`

Nama indeks yang akan dicari. Nilai default-nya adalah `AWS_Things`.

`query-string`

Query yang digunakan untuk mencari indeks. Anda dapat menentukan "\*" untuk mendapatkan hitungan semua hal yang diindeks di Anda. Akun AWS

`aggregation-field`

Bidang untuk agregat.

`buckets-aggregation-type`

Kontrol dasar bentuk respons dan jenis agregasi bucket yang akan dilakukan.

## Otorisasi

Anda dapat menentukan indeks grup benda sebagai ARN sumber daya dalam tindakan AWS IoT kebijakan, sebagai berikut.

| Tindakan                       | Sumber Daya                                                                                                                                                          |
|--------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>iot:GetStatistics</code> | Indeks ARN (misalnya, <code>arn:aws:iot:<i>your-aws-region</i>:index/AWS_Things</code> atau <code>arn:aws:iot:<i>your-aws-region</i>:index/AWS_ThingGroups</code> ). |

## Sintaks kueri

Dalam pengindeksan armada, Anda menggunakan sintaks kueri untuk menentukan kueri.

## Fitur yang didukung

Sintaks query mendukung fitur-fitur berikut:

- Syarat dan frasa
- Mencari bidang
- Pencarian awalan
- Rentang pencarian
- Operator Boolean AND, OR, NOT, dan -. Tanda hubung digunakan untuk mengecualikan sesuatu dari hasil pencarian (misalnya, `thingName:(tv* AND -plasma)`).
- Pengelompokan
- Pengelompokan lapangan
- Melarikan diri dari karakter khusus (seperti dengan \)

## Fitur yang tidak didukung

Sintaks kueri tidak mendukung fitur berikut:

- Pencarian wildcard terkemuka (seperti `*xyz`), tetapi mencari `*` cocok dengan semua hal



- Ekspresi reguler
- Meningkatkan
- Peringkat
- Pencarian fuzzy
- Pencarian kedekatan
- Penyortiran
- Agregasi
- Karakter khusus: `@,#,%,\,/,';`, dan `.`. Perhatikan bahwa `,` hanya didukung di geoquery.

## Catatan

Beberapa hal yang perlu diperhatikan tentang bahasa kueri:

- Operator defaultnya adalah AND. Kueri untuk `"thingName:abc thingType:xyz"` setara dengan `"thingName:abc AND thingType:xyz"`.
- Jika bidang tidak ditentukan, AWS IoT cari istilah di semua bidang registri, Device Shadow, dan Device Defender.
- Semua nama bidang peka huruf besar/kecil.
- Pencarian tidak peka huruf besar/kecil. Kata-kata dipisahkan oleh karakter spasi putih seperti yang didefinisikan oleh Java. `Character.isWhitespace(int)`
- Pengindeksan data Device Shadow (bayangan tanpa nama dan bayangan bernama) mencakup bagian yang dilaporkan, diinginkan, delta, dan metadata.
- Versi bayangan perangkat dan registri tidak dapat dicari, tetapi ada dalam respons.
- Jumlah maksimum istilah dalam kueri adalah dua belas.
- Karakter khusus hanya `,` didukung di geoqueries.

## Contoh pertanyaan

Tentukan kueri dalam string kueri menggunakan sintaks kueri. Kueri diteruskan ke [SearchIndexAPI](#). Tabel berikut mencantumkan beberapa contoh string query.

| String kueri                               | Hasil                                                                                                                                                                                                                   |
|--------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| abc                                        | Kueri untuk “abc” di registri apa pun, bayangan (bayangan klasik tanpa nama dan bayangan bernama), atau bidang pelanggaran Device Defender.                                                                             |
| thingName:myThingName                      | Pertanyaan untuk sesuatu dengan nama “sayaThingName”.                                                                                                                                                                   |
| thingName:my*                              | Pertanyaan untuk hal-hal dengan nama yang dimulai dengan “saya”.                                                                                                                                                        |
| thingName:ab?                              | Kueri untuk hal-hal dengan nama yang memiliki “ab” ditambah satu karakter tambahan (misalnya, “aba”, “abb”, “abc”, dan sebagainya).                                                                                     |
| thingTypeName:aa                           | Kueri untuk hal-hal yang terkait dengan tipe “aa”.                                                                                                                                                                      |
| thingGroupNames:a                          | Kueri untuk hal-hal dengan nama grup hal induk “a”.                                                                                                                                                                     |
| thingGroupNames:a*                         | Kueri untuk hal-hal dengan nama grup hal induk yang cocok dengan pola “a*”.                                                                                                                                             |
| attributes.myAttribute:75                  | Kueri untuk hal-hal dengan atribut bernama “myAttribute” yang memiliki nilai 75.                                                                                                                                        |
| attributes.myAttribute:[75 TO 80]          | Kueri untuk hal-hal dengan atribut bernama “myAttribute” yang memiliki nilai yang termasuk dalam rentang numerik (75—80, inklusif).                                                                                     |
| attributes.myAttribute:{75 TO 80}          | Kueri untuk hal-hal dengan atribut bernama “MyAttribute” yang memiliki nilai yang termasuk dalam rentang numerik (>75 dan <=80).                                                                                        |
| attributes.serialNumber:["abcd" TO "abcf"] | Kueri untuk hal-hal dengan atribut bernama “SerialNumber” yang memiliki nilai dalam rentang string alfanumerik. Kueri ini mengembalikan hal-hal dengan atribut “serialNumber” dengan nilai “abcd”, “abce”, atau “abcf”. |

| String kueri                                                                                  | Hasil                                                                                                                                                                                                                                                                                                                                                |
|-----------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>attributes.myAttribute:i*t</code>                                                       | Kueri untuk hal-hal dengan atribut bernama "myAttribute" di mana nilainya adalah 'i', diikuti oleh sejumlah karakter, diikuti oleh 't'.                                                                                                                                                                                                              |
| <code>attributes.attr1:abc AND attributes.attr2&lt;5 NOT attributes.attr3&gt;10</code>        | Kueri untuk hal-hal yang menggabungkan istilah menggunakan ekspresi Boolean. Kueri ini mengembalikan hal-hal yang memiliki atribut bernama "attr1" dengan nilai "abc", atribut bernama "attr2" yang kurang dari 5, dan atribut bernama "attr3" yang tidak lebih besar dari 10.                                                                       |
| <code>shadow.hasDelta:true</code>                                                             | Kueri untuk hal-hal dengan bayangan tanpa nama yang memiliki elemen delta.                                                                                                                                                                                                                                                                           |
| <code>NOT attributes.model:legacy</code>                                                      | Kueri untuk hal-hal di mana atribut bernama "model" bukan "warisan".                                                                                                                                                                                                                                                                                 |
| <code>shadow.reported.stats.battery:{70 TO 100} (v2 OR v3) NOT attributes.model:legacy</code> | Kueri untuk hal-hal dengan yang berikut: <ul style="list-style-type: none"> <li>• <code>stats.battery</code> Atribut bayangan benda memiliki nilai antara 70 dan 100.</li> <li>• Teks "v2" atau "v3" muncul dalam nama benda, nama jenis, atau nilai atribut.</li> <li>• <code>model</code> Atribut benda itu tidak disetel ke "warisan".</li> </ul> |
| <code>shadow.reported.myvalues:2</code>                                                       | Kueri untuk hal-hal di mana <code>myvalues</code> array di bagian bayangan yang dilaporkan berisi nilai 2.                                                                                                                                                                                                                                           |
| <code>shadow.reported.location:* NOT shadow.desired.stats.battery:*</code>                    | Kueri untuk hal-hal dengan yang berikut: <ul style="list-style-type: none"> <li>• <code>location</code> Atribut ada di <code>reported</code> bagian bayangan.</li> <li>• <code>stats.battery</code> Atribut tidak ada di <code>desired</code> bagian bayangan.</li> </ul>                                                                            |
| <code>shadow.name.&lt;shadowName&gt;.hasDelta:true</code>                                     | Kueri untuk hal-hal yang memiliki bayangan dengan nama yang diberikan dan juga elemen delta.                                                                                                                                                                                                                                                         |

| String kueri                                                                                            | Hasil                                                                                                                                                                          |
|---------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>shadow.name.&lt;shadowName&gt;.desired.filament:*</code>                                          | Kueri untuk hal-hal yang memiliki bayangan dengan nama yang diberikan dan juga properti filamen yang diinginkan.                                                               |
| <code>shadow.name.&lt;shadowName&gt;.reported.location:*</code>                                         | Kueri untuk hal-hal yang memiliki bayangan dengan nama yang diberikan dan di mana <code>location</code> atribut ada di bagian yang dilaporkan bayangan bernama.                |
| <code>connectivity.connected:true</code>                                                                | Kueri untuk semua perangkat yang terhubung.                                                                                                                                    |
| <code>connectivity.connected:false</code>                                                               | Kueri untuk semua perangkat yang terputus.                                                                                                                                     |
| <code>connectivity.connected:true AND connectivity.timestamp : [1557651600000 TO 1557867600000]</code>  | Kueri untuk semua perangkat yang terhubung dengan stempel waktu sambungan $\geq 1557651600000$ dan $\leq 1557867600000$ . Stempel waktu diberikan dalam milidetik sejak zaman. |
| <code>connectivity.connected:false AND connectivity.timestamp : [1557651600000 TO 1557867600000]</code> | Kueri untuk semua perangkat yang terputus dengan stempel waktu pemutus $\geq 1557651600000$ dan $\leq 1557867600000$ . Stempel waktu diberikan dalam milidetik sejak zaman.    |
| <code>connectivity.connected:true AND connectivity.timestamp &gt; 1557651600000</code>                  | Kueri untuk semua perangkat yang terhubung dengan stempel waktu sambungan $> 1557651600000$ . Stempel waktu diberikan dalam milidetik sejak zaman.                             |
| <code>connectivity.connected:*</code>                                                                   | Kueri untuk semua perangkat dengan informasi konektivitas hadir.                                                                                                               |
| <code>connectivity.disconnectReason:*</code>                                                            | Kueri untuk semua perangkat dengan konektivitas <code>DisconnectAson</code> hadir.                                                                                             |

| String kueri                                                                                                     | Hasil                                                                                                                                                                                                                                         |
|------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>connectivity.disconnectReason:CLIENT_INITIATED_DISCONNECT</code>                                           | Kueri untuk semua perangkat terputus karena CLIENT_INITIATED_DISCONNECT.                                                                                                                                                                      |
| <code>deviceDefender.violationCount:[0 TO 100]</code>                                                            | Kueri untuk hal-hal dengan pelanggaran Device Defender menghitung nilai yang termasuk dalam rentang numerik (0-100, inklusif).                                                                                                                |
| <code>deviceDefender.&lt;device-SecurityProfile&gt;.disconnectBehavior.inViolation:true</code>                   | Pertanyaan untuk hal-hal yang melanggar perilaku <code>disconnectBehavior</code> sebagaimana didefinisikan dalam profil <code>device-SecurityProfile</code> keamanan. Perhatikan bahwa <code>InViolation:false</code> bukan kueri yang valid. |
| <code>deviceDefender.&lt;device-SecurityProfile&gt;.disconnectBehavior.lastViolationValue.number&gt;2</code>     | Kueri untuk hal-hal yang melanggar perilaku <code>disconnectBehavior</code> sebagaimana didefinisikan dalam perangkat profil keamanan- <code>SecurityProfile</code> dengan nilai peristiwa pelanggaran terakhir lebih besar dari 2.           |
| <code>deviceDefender.&lt;device-SecurityProfile&gt;.disconnectBehavior.lastViolationTime&gt;1634227200000</code> | Kueri untuk hal-hal yang melanggar perilaku <code>disconnectBehavior</code> sebagaimana didefinisikan dalam perangkat profil keamanan- <code>SecurityProfile</code> dengan peristiwa pelanggaran terakhir setelah waktu yang ditentukan.      |
| <code>shadow.name.gps-tracker.reported.coordinates:geo_distance,47.6204,-122.3491,15.5km</code>                  | Kueri untuk hal-hal yang berada dalam jarak radial 15,5 km dari koordinat 47.6204, -122.3491. String kueri ini berlaku ketika data lokasi Anda disimpan dalam bayangan bernama.                                                               |
| <code>shadow.reported.coordinates:geo_distance,47.6204,-122.3491,15.5km</code>                                   | Kueri untuk hal-hal yang berada dalam jarak radial 15,5 km dari koordinat 47.6204, -122.3491. String kueri ini berlaku saat data lokasi Anda disimpan dalam bayangan klasik.                                                                  |

## Contoh kueri grup hal

Kueri ditentukan dalam string kueri menggunakan sintaks kueri dan diteruskan ke API.

[SearchIndex](#) Tabel berikut mencantumkan beberapa contoh string query.

| String kueri                                           | Hasil                                                                                                                                                                                                                         |
|--------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>abc</code>                                       | Kueri untuk "abc" di bidang apa pun.                                                                                                                                                                                          |
| <code>thingGroupName:myGroupThingName</code>           | Kueri untuk grup benda dengan nama "GroupThingName saya".                                                                                                                                                                     |
| <code>thingGroupName:my*</code>                        | Kueri untuk grup benda dengan nama yang dimulai dengan "saya".                                                                                                                                                                |
| <code>thingGroupName:ab?</code>                        | Kueri untuk grup benda dengan nama yang memiliki "ab" ditambah satu karakter tambahan (misalnya: "aba", "abb", "abc", dan sebagainya).                                                                                        |
| <code>attributes.myAttribute:75</code>                 | Kueri untuk grup benda dengan atribut bernama "myAttribute" yang memiliki nilai 75.                                                                                                                                           |
| <code>attributes.myAttribute:[75 TO 80]</code>         | Kueri untuk grup benda dengan atribut bernama "myAttribute" yang nilainya berada dalam rentang numerik (75—80, inklusif).                                                                                                     |
| <code>attributes.myAttribute:[75 TO 80]</code>         | Kueri untuk grup benda dengan atribut bernama "MyAttribute" yang nilainya berada dalam rentang numerik (>75 dan <=80).                                                                                                        |
| <code>attributes.myAttribute:["abcd" TO "abcf"]</code> | Kueri untuk grup benda dengan atribut bernama "myAttribute" yang nilainya berada dalam rentang string alfanumerik. Kueri ini mengembalikan grup benda dengan atribut "serialNumber" dengan nilai "abcd", "abce", atau "abcf". |
| <code>attributes.myAttribute:i*t</code>                | Kueri untuk grup benda dengan atribut bernama "myAttribute" yang nilainya 'i', diikuti oleh sejumlah karakter, diikuti oleh 't'.                                                                                              |

| String kueri                                                                           | Hasil                                                                                                                                                                                                                                                                                |
|----------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>attributes.attr1:abc AND attributes.attr2&lt;5 NOT attributes.attr3&gt;10</code> | Kueri untuk grup benda yang menggabungkan istilah menggunakan ekspresi Boolean. Kueri ini mengembalikan grup benda yang memiliki atribut bernama "attr1" dengan nilai "abc", atribut bernama "attr2" yang kurang dari 5, dan atribut bernama "attr3" yang tidak lebih besar dari 10. |
| <code>NOT attributes.myAttribute:cde</code>                                            | Kueri untuk grup benda di mana atribut bernama "myAttribute" bukan "cde".                                                                                                                                                                                                            |
| <code>parentGroupNames:(myParentThingGroupName)</code>                                 | Kueri untuk grup benda yang nama grup induknya cocok dengan "saya ParentThingGroupName".                                                                                                                                                                                             |
| <code>parentGroupNames:(myParentThingGroupName OR myRootThingGroupName)</code>         | Kueri untuk grup benda yang nama grup induknya cocok dengan "my ParentThingGroupName" atau "my RootThingGroupName".                                                                                                                                                                  |
| <code>parentGroupNames:(myParentThingGroupNa*)</code>                                  | Kueri untuk grup benda yang nama grup induknya dimulai dengan "saya ParentThingGroupNa".                                                                                                                                                                                             |

## Pengindeksan data lokasi

Anda dapat menggunakan [pengindeksan AWS IoT armada untuk mengindeks](#) data lokasi terkirim terakhir perangkat Anda dan mencari perangkat menggunakan geoquery. Fitur ini menyelesaikan pemantauan perangkat dan kasus penggunaan manajemen seperti pelacakan lokasi dan pencarian kedekatan. [Pengindeksan lokasi bekerja mirip dengan fitur pengindeksan armada lainnya, dan dengan konfigurasi tambahan untuk ditentukan dalam pengindeksan hal Anda.](#)

[Kasus penggunaan umum meliputi: pencarian dan agregat perangkat yang terletak dalam batas geografis yang diinginkan, dapatkan wawasan spesifik lokasi menggunakan istilah kueri yang terkait dengan metadata perangkat dan status dari sumber data yang diindeks, memberikan tampilan terperinci seperti memfilter hasil ke area geografis tertentu untuk mengurangi kelambatan rendering dalam peta pemantauan armada Anda dan melacak lokasi perangkat yang terakhir dilaporkan, dan mengidentifikasi perangkat yang berada di luar batas yang diinginkan dan menghasilkan alarm menggunakan armada meta rik.](#) Untuk memulai pengindeksan lokasi dan geoquery, lihat. [???](#)

## Format data yang didukung

AWS IoT pengindeksan armada mendukung format data lokasi berikut:

1. Representasi teks terkenal dari sistem referensi koordinat

String yang mengikuti [informasi Geografis - Representasi teks terkenal dari format sistem referensi koordinat](#). Contohnya bisa jadi "POINT(long lat)".

2. String yang mewakili koordinat

String dengan format "latitude, longitude" atau "longitude, latitude". Jika Anda menggunakan "longitude, latitude", Anda juga harus menentukan `orderGeoLocations`. Contohnya bisa jadi "41.12, -71.34".

3. Objek kunci lat (lintang), lon (bujur)

Format ini berlaku untuk bayangan klasik dan bernama bayangan. Kunci yang didukung: `lat`, `latitude`, `lon`, `long`, `longitude`. Contohnya bisa jadi {"lat": 41.12, "lon": -71.34}.

4. Array yang mewakili koordinat

Array dengan format `[lat, lon]` atau `[lon, lat]`. Jika Anda menggunakan format `[lon, lat]`, yang sama dengan koordinat di [GeoJSON](#) (berlaku untuk bayangan klasik dan bayangan bernama), Anda juga harus menentukan `orderGeoLocations`

Contohnya dapat berupa:

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

## Cara mengindeks data lokasi

Langkah-langkah berikut menunjukkan cara memperbarui konfigurasi pengindeksan untuk data lokasi Anda dan menggunakan geoquery untuk mencari perangkat.

1. Ketahui di mana data lokasi Anda disimpan

Pengindeksan armada saat ini mendukung pengindeksan data lokasi yang disimpan dalam bayangan klasik atau bayangan bernama.

2. Gunakan format data lokasi yang didukung

Pastikan format data lokasi Anda mengikuti salah satu [format data yang didukung](#).

3. Perbarui konfigurasi pengindeksan

Pada kebutuhan minimum, aktifkan konfigurasi pengindeksan hal (registri). Anda juga harus mengaktifkan pengindeksan pada bayangan klasik atau bayangan bernama yang berisi data lokasi Anda. Saat memperbarui pengindeksan hal Anda, Anda harus menyertakan data lokasi Anda dalam konfigurasi pengindeksan.

4. Membuat dan menjalankan geoqueries

Bergantung pada kasus penggunaan Anda, buat geoquery dan jalankan untuk mencari perangkat. [Geoquery yang Anda tulis harus mengikuti sintaks Query](#). Anda dapat menemukan beberapa contoh di [di???](#).

## Perbarui konfigurasi pengindeksan hal

Untuk mengindeks data lokasi, Anda harus memperbarui konfigurasi pengindeksan dan menyertakan data lokasi Anda. Tergantung di mana data lokasi Anda disimpan, ikuti langkah-langkah untuk memperbarui konfigurasi pengindeksan Anda:

## Data lokasi disimpan dalam bayangan klasik

Jika data lokasi Anda disimpan dalam bayangan klasik, Anda harus `thingIndexingMode` menyetel menjadi `REGISTRY_AND_SHADOW` dan menentukan data lokasi Anda di `geoLocations` bidang (`namedanorder`) di [filter](#).

Dalam contoh konfigurasi pengindeksan hal berikut, Anda menentukan jalur data lokasi `shadow.reported.coordinates` sebagai `name` dan `LonLat` sebagai `order`.

```
{
 "thingIndexingMode": "REGISTRY_AND_SHADOW",
 "filter": {
 "geoLocations": [
 {
 "name": "shadow.reported.coordinates",
 "order": "LonLat"
 }
]
 }
}
```

- `thingIndexingMode`

Mode pengindeksan mengontrol jika registri atau bayangan diindeks. Kapan `thingIndexingMode` diatur menjadi `OFF`, pengindeksan hal dinonaktifkan.

Untuk mengindeks data lokasi yang disimpan dalam bayangan klasik, Anda `thingIndexingMode` harus mengatur menjadi `REGISTRY_AND_SHADOW`. Untuk informasi selengkapnya, lihat [???](#).

- `filter`

Filter pengindeksan menyediakan pilihan tambahan untuk bayangan bernama dan data geolokasi. Untuk informasi selengkapnya, lihat [???](#).

- `geoLocations`

Daftar target geolokasi yang Anda pilih untuk diindeks. Jumlah maksimum target geolokasi default untuk pengindeksan adalah 1. Untuk meningkatkan batas, lihat [AWS IoT Device Management Kuota](#).

- `name`

Nama bidang target geolokasi. Contoh nilai name dapat berupa jalur data lokasi bayangan Anda: `shadow.reported.coordinates`.

- **order**

Urutan bidang target geolokasi. Nilai yang valid: `LatLon` dan `LonLat`. `LatLon` berarti garis lintang dan bujur. `LonLat` berarti bujur dan lintang. Bidang ini bersifat opsional. Nilai default-nya adalah `LatLon`.

Data lokasi disimpan dalam bayangan bernama

Jika data lokasi Anda disimpan dalam bayangan bernama, atur `namedShadowIndexingMode` menjadi `ON`, tambahkan nama bayangan bernama Anda ke `namedShadowNames` bidang di [filter](#), dan tentukan jalur data lokasi Anda di `geoLocations` bidang di [filter](#).

Dalam contoh konfigurasi pengindeksan hal berikut, Anda menentukan jalur data lokasi `shadow.name.namedShadow1.reported.coordinates` sebagai name dan `LonLat` sebagai `order`.

```
{
 "thingIndexingMode": "REGISTRY",
 "namedShadowIndexingMode": "ON",
 "filter": {
 "namedShadowNames": [
 "namedShadow1"
],
 "geoLocations": [
 {
 "name": "shadow.name.namedShadow1.reported.coordinates",
 "order": "LonLat"
 }
]
 }
}
```

- **thingIndexingMode**

Mode pengindeksan mengontrol jika registri atau bayangan diindeks. Kapan `thingIndexingMode` diatur menjadi `OFF`, pengindeksan hal dinonaktifkan.

Untuk mengindeks data lokasi yang disimpan dalam bayangan bernama, Anda `thingIndexingMode` harus mengatur menjadi `REGISTRY` (atau `REGISTRY_AND_SHADOW`). Untuk informasi selengkapnya, lihat [???](#).

- `filter`

Filter pengindeksan menyediakan pilihan tambahan untuk bayangan bernama dan data geolokasi. Untuk informasi selengkapnya, lihat [???](#).

- `geoLocations`

Daftar target geolokasi yang Anda pilih untuk diindeks. Jumlah maksimum target geolokasi default untuk pengindeksan adalah 1. Untuk meningkatkan batas, lihat [AWS IoT Device Management Kuota](#).

- `name`

Nama bidang target geolokasi. Contoh nilai `name` dapat berupa jalur data lokasi bayangan Anda: `shadow.name.namedShadow1.reported.coordinates`.

- `order`

Urutan bidang target geolokasi. Nilai yang valid: `LatLon` dan `LonLat`. `LatLon` berarti garis lintang dan bujur. `LonLat` berarti bujur dan lintang. Bidang ini bersifat opsional. Nilai default-nya adalah `LatLon`.

## Contoh geoqueries

Setelah Anda menyelesaikan konfigurasi pengindeksan untuk data lokasi Anda, jalankan geoquery untuk mencari perangkat. Anda juga dapat menggabungkan geoquery Anda dengan string kueri lainnya. Untuk informasi selengkapnya, lihat [???](#) dan [???](#).

### Contoh kueri 1

Contoh ini mengasumsikan data lokasi disimpan dalam bayangan `gps-tracker` bernama. Output dari perintah ini adalah daftar perangkat yang berada dalam jarak radial 15,5 km dari titik pusat dengan koordinat (47.6204, -122.3491).

```
aws iot search-index --query-string \
"shadow.name.gps-tracker.reported.coordinates:geo_distance,47.6204,-122.3491,15.5km"
```

## Contoh kueri 2

Contoh ini mengasumsikan data lokasi disimpan dalam bayangan klasik. Output dari perintah ini adalah daftar perangkat yang berada dalam jarak radial 15,5 km dari titik pusat dengan koordinat (47.6204, -122.3491).

```
aws iot search-index --query-string \
"shadow.reported.coordinates:geo_distance,47.6204,-122.3491,15.5km"
```

## Contoh kueri 3

Contoh ini mengasumsikan data lokasi disimpan dalam bayangan klasik. Output dari perintah ini adalah daftar perangkat yang tidak terhubung dan di luar jarak radial 15,5 km dari titik pusat dengan koordinat (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 memulai

Tutorial ini menunjukkan cara menggunakan pengindeksan [armada untuk mengindeks data lokasi Anda](#). Untuk mempermudah, Anda membuat sesuatu untuk mewakili perangkat Anda dan menyimpan data lokasi dalam bayangan bernama, memperbarui konfigurasi pengindeksan hal untuk pengindeksan lokasi, dan menjalankan contoh geoquery untuk mencari perangkat dalam batas radial.

Tutorial ini membutuhkan waktu sekitar 15 menit untuk menyelesaikannya.

Dalam topik ini:

- [Prasyarat](#)
- [Ciptakan benda dan bayangan](#)
- [Perbarui konfigurasi pengindeksan hal](#)
- [Jalankan geoquery](#)

## Prasyarat

- Instal versi terbaru [AWS CLI](#).

- [Biasakan diri Anda dengan pengindeksan Lokasi dan geoquery, Kelola pengindeksan hal, dan sintaks Kueri.](#)

## Ciptakan benda dan bayangan

Anda membuat sesuatu untuk mewakili perangkat Anda, dan bayangan bernama untuk menyimpan data lokasinya (koordinat 47.61564, -122.33584).

1. Jalankan perintah berikut untuk membuat benda Anda yang mewakili sepeda Anda bernama Bike-1. Untuk informasi selengkapnya tentang cara membuat sesuatu menggunakan AWS CLI, lihat [create-thing](#) dari Referensi. AWS CLI

```
aws iot create-thing --thing-name "Bike-1" \
--attribute-payload '{"attributes": {"model": "OEM-2302-12", "battery": "35",
"acqDate": "06/09/23"}}'
```

Output dari perintah ini dapat terlihat seperti berikut:

```
{
 "thingName": "Bike-1",
 "thingArn": "arn:aws:iot:us-east-1:123456789012:thing/Bike-1",
 "thingId": "df9cf01d-b0c8-48fe-a2e2-e16cff6b23df"
}
```

2. Jalankan perintah berikut untuk membuat bayangan bernama untuk menyimpan data lokasi Sepeda-1 (koordinat 47.61564, -122.33584). Untuk informasi selengkapnya tentang cara membuat bayangan bernama menggunakan AWS CLI, lihat [update-thing-shadow](#) dari Referensi. AWS CLI

```
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" \

```

Perintah ini tidak menghasilkan output apa pun. Untuk melihat bayangan bernama yang Anda buat, Anda dapat menjalankan perintah CLI [list-named-shadows-for-thing](#).

```
aws iot-data list-named-shadows-for-thing --thing-name Bike-1
```

Output dari perintah ini dapat terlihat seperti berikut:

```
{
 "results": [
 "Bike1-shadow"
],
 "timestamp": 1699574309
}
```

## Perbarui konfigurasi pengindeksan hal

Untuk mengindeks data lokasi Anda, Anda harus memperbarui konfigurasi pengindeksan hal Anda untuk menyertakan data lokasi. Karena data lokasi Anda disimpan dalam bayangan bernama dalam tutorial ini, atur `thingIndexingMode` menjadi `REGISTRY` (pada persyaratan minimum), atur `namedShadowIndexingMode` menjadi `ON`, dan tambahkan data lokasi Anda ke konfigurasi. Dalam contoh ini, Anda harus menambahkan nama bayangan bernama Anda dan jalur data lokasi bayangan ke `filter`.

1. Jalankan perintah untuk memperbarui konfigurasi pengindeksan Anda untuk pengindeksan lokasi.

```
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"}] } }
```

Perintah tidak menghasilkan output apa pun. Anda mungkin perlu menunggu beberapa saat hingga pembaruan selesai. Untuk memeriksa status, jalankan perintah [CLI deskripsi-indeks](#). Jika Anda melihat `indexStatus show:ACTIVE`, pembaruan pengindeksan hal Anda selesai.

2. Jalankan perintah untuk memverifikasi konfigurasi pengindeksan Anda. Langkah ini bersifat opsional.

```
aws iot get-indexing-configuration
```

Outputnya bisa terlihat seperti berikut:

```
{
 "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"
}
}
```

## Jalankan geoquery

Sekarang Anda telah memperbarui konfigurasi pengindeksan hal Anda untuk memasukkan data lokasi. Cobalah untuk membuat beberapa geoquery dan menjalankannya untuk melihat apakah Anda bisa mendapatkan hasil pencarian yang Anda inginkan. Geoquery harus mengikuti [sintaks Query](#). Anda dapat menemukan beberapa contoh geoquery yang berguna di [???](#)

Dalam contoh perintah berikut, Anda menggunakan geoquery `shadow.name.Bike1-shadow.reported.coordinates:geo_distance,47.6204,-122.3491,15.5km` untuk mencari perangkat yang berada dalam jarak radial 15,5 km dari titik pusat dengan koordinat (47.6204, -122.3491).

```
aws iot search-index --query-string "shadow.name.Bike1-
shadow.reported.coordinates:geo_distance,47.6204,-122.3491,15.5km"
```

Karena Anda memiliki perangkat yang terletak di koordinat "lat": 47.6153, "lon": -122.3333, yang berada dalam jarak 15,5 km dari titik pusat, Anda harus dapat melihat perangkat ini (Sepeda-1) dalam output. Outputnya bisa terlihat seperti berikut:

```
{
 "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}"
 }
]
}
```

Untuk informasi selengkapnya, lihat [???](#).

## Metrik armada

Metrik armada adalah fitur [pengindeksan armada](#), layanan terkelola yang memungkinkan Anda mengindeks, mencari, dan mengumpulkan data perangkat Anda. AWS IoT Anda dapat menggunakan metrik armada, untuk memantau status agregat perangkat armada Anda [CloudWatch](#) dari waktu ke waktu, termasuk meninjau tingkat pemutusan perangkat armada Anda atau perubahan level baterai rata-rata pada periode tertentu.

Dengan menggunakan metrik armada, Anda dapat membuat [kueri agregasi](#) yang hasilnya terus dipancarkan [CloudWatch](#) sebagai metrik untuk menganalisis tren dan membuat alarm. Untuk tugas pemantauan, Anda dapat menentukan kueri agregasi dari berbagai jenis agregasi (Statistik, Kardinalitas, dan Persentil). Anda dapat menyimpan semua kueri agregasi untuk membuat metrik armada untuk digunakan kembali di masa mendatang.

## Tutorial memulai

Dalam tutorial ini, Anda membuat [metrik armada](#) untuk memantau suhu sensor Anda untuk mendeteksi anomali potensial. Saat membuat metrik armada, Anda menentukan [kueri agregasi](#) yang mendeteksi jumlah sensor dengan suhu melebihi 80 derajat Fahrenheit. Anda menentukan kueri yang akan dijalankan setiap 60 detik dan hasil kueri dipancarkan CloudWatch, di mana Anda dapat melihat jumlah sensor yang memiliki potensi risiko suhu tinggi, dan menyetel alarm. Untuk menyelesaikan tutorial ini, Anda akan menggunakan [AWS CLI](#).

Dalam tutorial ini, Anda akan belajar cara:

- [Mengatur](#)
- [Buat metrik armada](#)
- [Lihat metrik di CloudWatch](#)
- [Bersihkan sumber daya](#)

Tutorial ini membutuhkan waktu sekitar 15 menit untuk menyelesaikannya.

### Prasyarat

- Instal versi terbaru [AWS CLI](#)
- Biasakan diri Anda dengan [Query untuk data agregat](#)
- Biasakan diri Anda dengan [Menggunakan CloudWatch metrik Amazon](#)

### Penyiapan

Untuk menggunakan metrik armada, aktifkan pengindeksan armada. Untuk mengaktifkan pengindeksan armada untuk grup benda atau benda Anda dengan sumber data tertentu dan konfigurasi terkait, ikuti instruksi dalam Mengelola pengindeksan [hal dan Mengelola pengindeksan grup \[hal\]\(#\)](#).

Untuk mengatur

1. Jalankan perintah berikut untuk mengaktifkan pengindeksan armada dan tentukan sumber data yang akan dicari.

```
aws iot update-indexing-configuration \
```

```
--thing-indexing-configuration
"thingIndexingMode=REGISTRY_AND_SHADOW,customFields=[{name=attributes.temperature,type=Number},
{name=attributes.rackId,type=String},
{name=attributes.stateNormal,type=Boolean}],thingConnectivityIndexingMode=STATUS" \
```

Contoh perintah CLI sebelumnya memungkinkan pengindeksan armada untuk mendukung pencarian data registri, data bayangan, dan status konektivitas benda menggunakan indeks. `AWS_Things`

Perubahan konfigurasi dapat memakan waktu beberapa menit untuk diselesaikan. Verifikasi bahwa pengindeksan armada Anda diaktifkan sebelum Anda membuat metrik armada.

Untuk memeriksa apakah pengindeksan armada Anda telah diaktifkan, jalankan perintah CLI berikut:

```
aws --region us-east-1 iot describe-index --index-name "AWS_Things"
```

Untuk informasi selengkapnya, lihat [Mengaktifkan pengindeksan hal](#).

2. Jalankan skrip bash berikut untuk membuat sepuluh hal dan menggambarkannya.

```
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
```

Skrip ini menciptakan sepuluh hal untuk mewakili sepuluh sensor. Setiap hal memiliki atribut `temperature`, `rackId`, dan `stateNormal` seperti yang dijelaskan dalam tabel berikut:

| Atribut     | Jenis data | Deskripsi                                  |
|-------------|------------|--------------------------------------------|
| temperature | Angka      | Nilai suhu di Fahrenheit                   |
| rackId      | String     | ID rak server yang berisi sensor           |
| stateNormal | Boolean    | Apakah nilai suhu sensor normal atau tidak |

Output dari skrip ini berisi sepuluh file JSON. Salah satu file JSON terlihat seperti berikut:

```
{
 "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"
}
```

Untuk informasi selengkapnya, lihat [Membuat sesuatu](#).

## Buat metrik armada

Untuk membuat metrik armada

1. Jalankan perintah berikut untuk membuat metrik armada bernama *High\_Temp\_FM*. Anda membuat metrik armada untuk memantau jumlah sensor dengan suhu melebihi 80 derajat Fahrenheit. 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
```

--metrik-nama

Tipe data: string. `--metric-name`Parameter menentukan nama metrik armada. Dalam contoh ini, Anda membuat metrik armada bernama High\_Temp\_FM.

--kueri-string

Tipe data: string. `--query-string`Parameter menentukan string query. Dalam contoh ini, string kueri berarti menanyakan semua hal dengan nama yang dimulai dengan TempSensor dan dengan suhu lebih tinggi dari 80 derajat Fahrenheit. Untuk informasi selengkapnya, lihat [Sintaks kueri](#).

--periode

Tipe data: bilangan bulat. `--period`Parameter menentukan waktu untuk mengambil data agregat dalam hitungan detik. Dalam contoh ini, Anda menentukan bahwa metrik armada yang Anda buat mengambil data agregat setiap 60 detik.

--agregasi-bidang


Tipe data: string. `--aggregation-field`Parameter menentukan atribut untuk mengevaluasi. Dalam contoh ini, atribut suhu harus dievaluasi.

--agregasi-tipe

`--aggregation-type`Parameter menentukan ringkasan statistik untuk ditampilkan dalam metrik armada. Untuk tugas pemantauan, Anda dapat menyesuaikan properti kueri agregasi untuk berbagai jenis agregasi (Statistik, Kardinalitas, dan Persentil). Dalam contoh ini, Anda menentukan hitungan untuk jenis agregasi dan Statistik untuk mengembalikan jumlah perangkat yang memiliki atribut yang cocok dengan kueri, dengan kata lain, untuk mengembalikan jumlah perangkat dengan nama yang dimulai dengan TempSensor dan dengan suhu lebih tinggi dari 80 derajat Fahrenheit. Untuk informasi selengkapnya, lihat [Menanyakan data agregat](#).

Output dari perintah ini terlihat seperti berikut:

```
{
 "metricArn": "arn:aws:iot:region:111122223333:fleetmetric/high_temp_FM",
 "metricName": "high_temp_FM"
}
```

 Note

Diperlukan waktu beberapa saat agar titik data ditampilkan CloudWatch.

Untuk mempelajari lebih lanjut tentang cara membuat metrik armada, baca [Mengelola metrik armada](#).

Jika Anda tidak dapat membuat metrik armada, baca Metrik [armada pemecahan masalah](#).

2. (Opsional) Jalankan perintah berikut untuk menjelaskan metrik armada Anda bernama High\_Temp\_FM:

```
aws iot describe-fleet-metric --metric-name "high_temp_FM"
```

Output dari perintah ini terlihat seperti berikut:

```
{
 "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"
}
```

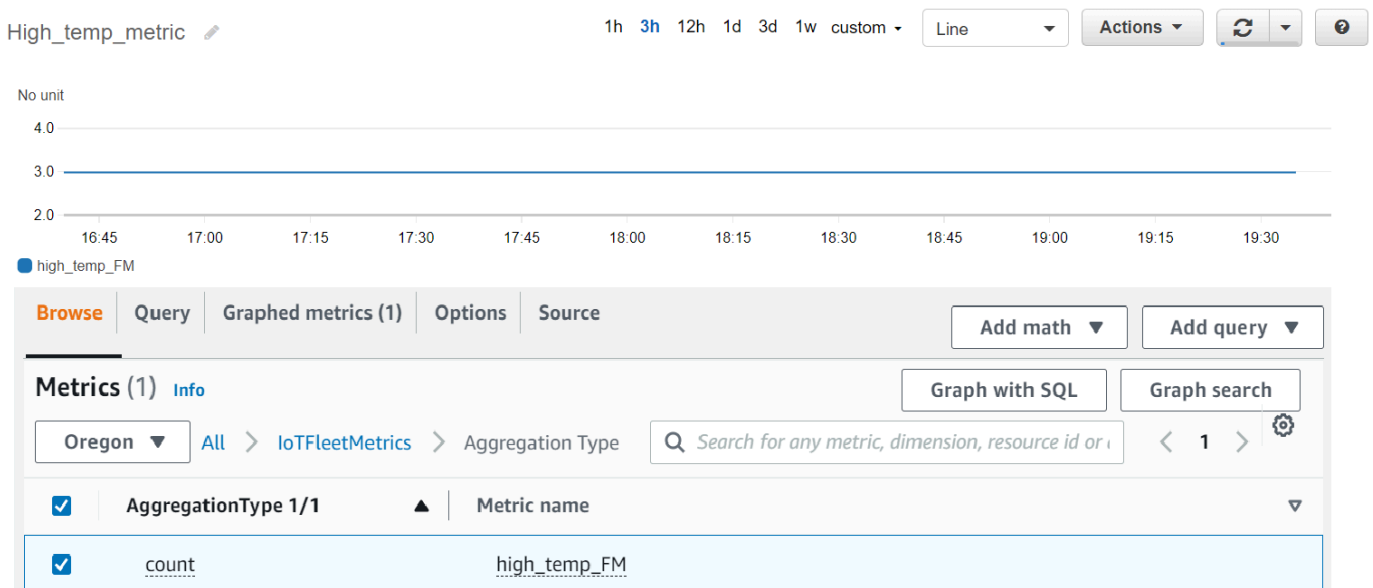
## Lihat metrik armada di CloudWatch

Setelah membuat metrik armada, Anda dapat melihat data metrik di CloudWatch. Dalam tutorial ini, Anda akan melihat metrik yang menunjukkan jumlah sensor dengan nama yang dimulai dengan TempSensordan dengan suhu lebih tinggi dari 80 derajat Fahrenheit.

Untuk melihat titik data di CloudWatch

1. Buka CloudWatch konsol di <https://console.aws.amazon.com/cloudwatch/>.
2. Pada CloudWatch menu di panel kiri, pilih Metrik untuk memperluas submenu dan kemudian pilih Semua metrik. Ini membuka halaman dengan bagian atas untuk menampilkan grafik dan bagian bawah berisi empat bagian tab.
3. Bagian tab pertama Semua metrik mencantumkan semua metrik yang dapat Anda lihat dalam grup, pilih IoT. FleetMetrics Ini berisi semua metrik armada Anda.
4. Pada bagian Jenis agregasi pada tab Semua metrik, pilih Jenis agregasi untuk melihat semua metrik armada yang Anda buat.
5. Pilih metrik armada untuk menampilkan grafik di sebelah kiri bagian Jenis agregasi. Anda akan melihat *jumlah* nilai di sebelah kiri nama Metrik Anda, dan ini adalah nilai dari jenis agregasi yang Anda tentukan di bagian [Buat metrik armada](#) dari tutorial ini.
6. Pilih tab kedua bernama Metrik grafik di sebelah kanan tab Semua metrik untuk melihat metrik armada yang Anda pilih dari langkah sebelumnya.

Anda harus dapat melihat grafik yang menampilkan jumlah sensor dengan suhu lebih tinggi dari 80 derajat Fahrenheit seperti berikut ini:





**Note**

Atribut Periode secara CloudWatch default hingga 5 menit. Ini adalah interval waktu antara titik data yang ditampilkan di CloudWatch. Anda dapat mengubah pengaturan Periode berdasarkan kebutuhan Anda.

7. (Opsional) Anda dapat mengatur alarm metrik.
  1. Pada CloudWatch menu di panel kiri, pilih Alarm untuk memperluas submenu dan kemudian pilih Semua alarm.
  2. Pada halaman Alarm, pilih Buat alarm di sudut kanan atas. Ikuti instruksi Buat alarm di konsol untuk membuat alarm sesuai kebutuhan. Untuk informasi selengkapnya, lihat [Menggunakan CloudWatch alarm Amazon](#).

Untuk mempelajari lebih lanjut, baca [Menggunakan CloudWatch metrik Amazon](#).

Jika Anda tidak dapat melihat titik data CloudWatch, baca Metrik [armada pemecahan masalah](#).

## Hapus

Untuk menghapus metrik armada

Anda menggunakan perintah delete-fleet-metric CLI untuk menghapus metrik armada.

Untuk menghapus metrik armada bernama High\_Temp\_FM, jalankan perintah berikut.

```
aws iot delete-fleet-metric --metric-name "high_temp_FM"
```

Untuk membersihkan hal-hal

Anda menggunakan perintah delete-thing CLI untuk menghapus sesuatu.

Untuk menghapus sepuluh hal yang Anda buat, jalankan skrip berikut:

```
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
```

Untuk membersihkan metrik di CloudWatch

CloudWatch tidak mendukung penghapusan metrik. Metrik kedaluwarsa berdasarkan jadwal retensi mereka. Untuk mempelajari lebih lanjut, [Menggunakan CloudWatch metrik Amazon](#).

## Mengelola metrik armada

Topik ini menunjukkan cara menggunakan AWS IoT konsol dan AWS CLI mengelola metrik armada Anda.

Topik

- [Mengelola metrik armada \(Konsol\)](#)
- [Mengelola metrik armada \(CLI\)](#)
- [Otorisasi penandaan sumber daya IoT](#)

### Mengelola metrik armada (Konsol)

Bagian berikut menunjukkan cara menggunakan AWS IoT konsol untuk mengelola metrik armada Anda. Pastikan Anda telah mengaktifkan pengindeksan armada dengan sumber dan konfigurasi data terkait sebelum membuat metrik armada.

Aktifkan pengindeksan armada

Jika Anda telah mengaktifkan pengindeksan armada, lewati bagian ini.

Jika Anda belum mengaktifkan pengindeksan armada, ikuti petunjuk ini.

1. Buka AWS IoT konsol Anda di <https://console.aws.amazon.com/iot/>.
2. Pada AWS IoT menu, pilih Pengaturan.
3. Untuk melihat pengaturan terperinci, pada halaman Pengaturan, gulir ke bawah ke bagian pengindeksan Armada.
4. Untuk memperbarui pengaturan pengindeksan armada, di sebelah kanan bagian pengindeksan Armada, pilih Kelola pengindeksan.
5. Pada halaman Kelola pengindeksan armada, perbarui pengaturan pengindeksan armada berdasarkan kebutuhan Anda.

- Konfigurasi

Untuk mengaktifkan pengindeksan hal, aktifkan pengindeksan Thing, lalu pilih sumber data yang ingin Anda indeks.

Untuk mengaktifkan pengindeksan grup hal, aktifkan pengindeksan grup Thing.

- Bidang kustom untuk agregasi - opsional

Bidang kustom adalah daftar nama bidang dan pasangan jenis bidang.

Untuk menambahkan pasangan bidang kustom, pilih Tambahkan bidang baru. Masukkan nama bidang khusus seperti `attributes.temperature`, lalu pilih jenis bidang dari menu Jenis bidang. Perhatikan bahwa nama bidang kustom dimulai dengan `attributes.` dan akan disimpan sebagai atribut untuk menjalankan kueri [agregasi hal](#).

Untuk memperbarui dan menyimpan pengaturan, pilih Perbarui.

## Buat metrik armada

1. Buka AWS IoT konsol Anda di <https://console.aws.amazon.com/iot/>.

2. Pada AWS IoT menu, pilih Kelola, lalu pilih Metrik Armada.

3. Pada halaman Metrik Armada, pilih Buat metrik armada dan selesaikan langkah pembuatannya.

4. Pada langkah 1 Konfigurasi metrik armada

- Di bagian Kueri, masukkan string kueri untuk menentukan hal-hal atau kelompok hal yang ingin Anda lakukan pencarian agregat. String query terdiri dari atribut dan nilai. Untuk Properti, pilih atribut yang Anda inginkan, atau, jika tidak muncul dalam daftar, masukkan atribut di bidang. Masukkan nilai setelahnya: `.` Sebuah string query contoh bisa `thingName:TempSensor*`. Untuk setiap string kueri yang Anda masukkan, tekan enter di keyboard Anda. Jika Anda memasukkan beberapa string kueri, tentukan hubungannya dengan memilih `dan`, `atau`, dan `tidak`, atau `atau tidak` di antara mereka.
- Di properti Laporan, pilih Nama indeks, Jenis agregasi, dan bidang Agregasi dari daftar masing-masing. Selanjutnya, pilih data yang ingin Anda agregat Pilih data, di mana Anda dapat memilih beberapa nilai data.
- Pilih Berikutnya.

5. Pada langkah 2 Tentukan properti metrik armada

- Di bidang nama metrik Armada, masukkan nama untuk metrik armada yang Anda buat.

- Di Deskripsi - bidang opsional, masukkan deskripsi untuk metrik armada yang Anda buat. Bidang ini bersifat opsional.
  - Di bidang Jam dan Menit, masukkan waktu (seberapa sering) metrik armada yang Anda inginkan untuk memancarkan data. CloudWatch
  - Pilih Berikutnya.
6. Pada langkah 3 Tinjau dan buat
- Tinjau pengaturan langkah 1 dan langkah 2. Untuk mengedit pengaturan, pilih Edit.
  - Pilih Buat metrik armada.

Setelah pembuatan berhasil, metrik armada terdaftar di halaman metrik Armada.

### Memperbarui metrik armada

1. Pada halaman metrik Armada, pilih metrik armada yang ingin Anda perbarui.
2. Pada halaman Detail metrik armada, pilih Edit. Ini membuka langkah-langkah pembuatan di mana Anda dapat memperbarui metrik armada Anda di salah satu dari tiga langkah.
3. Setelah Anda selesai memperbarui metrik armada, pilih Perbarui metrik armada.

### Hapus metrik armada

1. Pada halaman metrik Armada, pilih metrik armada yang ingin Anda hapus.
2. Pada halaman berikutnya yang menampilkan detail metrik armada Anda, pilih Hapus.
3. Di kotak dialog, masukkan nama metrik armada Anda untuk mengonfirmasi penghapusan.
4. Pilih Hapus. Langkah ini menghapus metrik armada Anda secara permanen.

### Mengelola metrik armada (CLI)

Bagian berikut menunjukkan cara menggunakan metrik AWS CLI untuk mengelola armada Anda. Pastikan Anda telah mengaktifkan pengindeksan armada dengan sumber dan konfigurasi data terkait sebelum membuat metrik armada. Untuk mengaktifkan pengindeksan armada untuk hal-hal atau grup benda Anda, ikuti instruksi dalam [Mengelola pengindeksan hal atau Mengelola pengindeksan grup hal](#).

### Buat metrik armada

Anda dapat menggunakan perintah `create-fleet-metric` CLI untuk membuat metrik armada.

```
aws iot create-fleet-metric --metric-name "YourFleetMetricName" --query-string "*" --period 60 --aggregation-field "registry.version" --aggregation-type name=Statistics,values=sum
```

Output dari perintah ini berisi nama dan Amazon Resource Name (ARN) dari metrik armada Anda. Outputnya terlihat seperti berikut:

```
{
 "metricArn": "arn:aws:iot:us-east-1:111122223333:fleetmetric/YourFleetMetricName",
 "metricName": "YourFleetMetricName"
}
```

## Daftar metrik armada

Anda dapat menggunakan perintah `list-fleet-metric` CLI untuk mencantumkan semua metrik armada di akun Anda.

```
aws iot list-fleet-metrics
```

Output dari perintah ini berisi semua metrik armada Anda. Outputnya terlihat seperti berikut:

```
{
 "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"
 }
]
}
```

## Jelaskan metrik armada

Anda dapat menggunakan perintah `describe-fleet-metric` CLI untuk menampilkan informasi lebih rinci tentang metrik armada.

```
aws iot describe-fleet-metric --metric-name "YourFleetMetricName"
```

Output dari perintah berisi informasi rinci tentang metrik armada yang ditentukan. Outputnya terlihat seperti berikut:

```
{
 "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"
}
```

## Memperbarui metrik armada

Anda dapat menggunakan perintah `update-fleet-metric` CLI untuk memperbarui metrik armada.

```
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
```

`update-fleet-metric` Perintah tidak menghasilkan output apa pun. Anda dapat menggunakan perintah `describe-fleet-metric` CLI untuk melihat hasilnya.

```
{
 "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"
}
```

## Hapus metrik armada

Gunakan perintah `delete-fleet-metric` CLI untuk menghapus metrik armada.

```
aws iot delete-fleet-metric --metric-name "YourFleetMetricName"
```

Perintah ini tidak menghasilkan output apa pun jika penghapusan berhasil atau jika Anda menentukan metrik armada yang tidak ada.

Untuk informasi selengkapnya, lihat [Memecahkan masalah metrik armada](#).

## Otorisasi penandaan sumber daya IoT

Untuk kontrol yang lebih baik atas metrik armada yang dapat Anda buat, modifikasi, atau gunakan, Anda dapat melampirkan tag ke metrik armada.

Untuk menandai metrik armada yang Anda buat dengan menggunakan AWS Management Console atau AWS CLI, Anda harus menyertakan `iot:TagResource` tindakan dalam kebijakan IAM untuk memberikan izin kepada pengguna. Jika kebijakan IAM Anda tidak disertakan `iot:TagResource`, tindakan apa pun untuk membuat metrik armada dengan tag akan menampilkan `AccessDeniedException` kesalahan.

Untuk informasi umum tentang menandai sumber daya Anda, lihat [Menandai sumber daya Anda AWS IoT](#).

## Contoh kebijakan IAM

Lihat contoh kebijakan IAM berikut yang memberikan izin penandaan saat Anda membuat metrik armada:

```
{
 "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/*"
]
 }
]
}
```

Untuk informasi lebih lanjut, lihat [Tindakan, sumber daya, kunci syarat untuk AWS IoT](#).



# Pengiriman file berbasis MQTT

Salah satu opsi yang dapat Anda gunakan untuk mengelola file dan mentransfernya ke AWS IoT perangkat di armada Anda adalah pengiriman file berbasis MQTT. Dengan fitur ini di AWS Cloud Anda dapat membuat aliran yang berisi banyak file, Anda dapat memperbarui data aliran (daftar file dan deskripsi), mendapatkan data aliran, dan banyak lagi. AWS IoT Pengiriman file berbasis MQTT dapat mentransfer data dalam blok kecil ke perangkat IoT Anda, menggunakan protokol MQTT dengan dukungan untuk pesan permintaan dan respons di JSON atau CBOR.

Untuk informasi selengkapnya tentang cara mentransfer data ke dan dari perangkat IoT menggunakan AWS IoT, lihat. [Menghubungkan perangkat ke AWS IoT](#)

Topik

- [Apa itu aliran?](#)
- [Mengelola aliran di AWS Cloud](#)
- [Menggunakan pengiriman AWS IoT file berbasis MQTT di perangkat](#)
- [Contoh kasus penggunaan di FreeRTOS OTA](#)

## Apa itu aliran?

Dalam AWS IoT, aliran adalah sumber daya yang dapat dialamatkan secara publik yang merupakan abstraksi untuk daftar file yang dapat ditransfer ke perangkat IoT. Aliran tipikal berisi informasi berikut:

- Nama Sumber Daya Amazon (ARN) yang secara unik mengidentifikasi aliran pada waktu tertentu. ARN ini memiliki pola. `arn:partition:iot:region:account-ID:stream/stream ID`
- ID aliran yang mengidentifikasi aliran Anda dan digunakan (dan biasanya diperlukan) dalam perintah AWS Command Line Interface (AWS CLI) atau SDK.
- Deskripsi aliran yang memberikan deskripsi sumber daya aliran.
- Versi aliran yang mengidentifikasi versi tertentu dari aliran. Karena data streaming dapat dimodifikasi segera sebelum perangkat memulai transfer data, versi streaming dapat digunakan oleh perangkat untuk menegakkan pemeriksaan konsistensi.
- Daftar file yang dapat ditransfer ke perangkat. Untuk setiap file dalam daftar, aliran merekam ID file, ukuran file, dan informasi alamat file, yang terdiri dari, misalnya, nama bucket Amazon S3, kunci objek, dan versi objek.

- Peran AWS Identity and Access Management (IAM) yang memberikan pengiriman file AWS IoT berbasis MQTT izin untuk membaca file aliran yang disimpan dalam penyimpanan data.

AWS IoT Pengiriman file berbasis MQTT menyediakan fungsionalitas berikut sehingga perangkat dapat mentransfer data dari Cloud: AWS

- Transfer data menggunakan protokol MQTT.
- Support untuk format JSON atau CBOR.
- Kemampuan untuk mendeskripsikan aliran ([DescribeStreamAPI](#)) untuk mendapatkan daftar file aliran, versi streaming, dan informasi terkait.
- Kemampuan untuk mengirim data dalam blok kecil ([GetStreamAPI](#)) sehingga perangkat dengan kendala perangkat keras dapat menerima blok.
- Support untuk ukuran blok dinamis per permintaan, untuk mendukung perangkat yang memiliki kapasitas memori berbeda.
- Optimalisasi untuk permintaan streaming bersamaan ketika beberapa perangkat meminta blok data dari file streaming yang sama.
- Amazon S3 sebagai penyimpanan data untuk file streaming.
- Support untuk penerbitan log transfer data dari pengiriman file AWS IoT berbasis MQTT ke CloudWatch

[Untuk kuota pengiriman file berbasis MQTT, lihat Service Quotas di AWS IoT Core Referensi Umum AWS](#)

## Mengelola aliran di AWS Cloud

AWS IoT menyediakan AWS SDK dan AWS CLI perintah yang dapat Anda gunakan untuk mengelola aliran di AWS Cloud. Anda dapat menggunakan perintah ini untuk melakukan hal berikut:

- Buat aliran. [CLI/SDK](#)
- Jelaskan aliran untuk mendapatkan informasinya. [CLI/SDK](#)
- Daftar aliran di Akun AWS. [CLI/SDK](#)
- Perbarui daftar file atau deskripsi aliran dalam aliran. [CLI/SDK](#)
- Hapus aliran. [CLI/SDK](#)

**Note**

Pada saat ini, aliran tidak terlihat di. AWS Management Console Anda harus menggunakan AWS CLI atau AWS SDK untuk mengelola aliran masuk AWS IoT. Selain itu, [Embedded C SDK](#) adalah satu-satunya SDK yang mendukung transfer file berbasis MQTT.

Sebelum Anda menggunakan pengiriman file AWS IoT berbasis MQTT dari perangkat Anda, Anda harus memastikan kondisi berikut terpenuhi untuk perangkat Anda seperti yang ditunjukkan pada bagian berikutnya:

- Kebijakan yang mencerminkan izin yang benar yang diperlukan untuk mentransmisikan data melalui MQTT.
- Perangkat Anda dapat terhubung ke AWS IoT Device Gateway.
- Pernyataan kebijakan yang menyatakan Anda dapat menandai sumber daya. Jika `CreateStream` dipanggil dengan tag, maka `iot:TagResource` diperlukan.

Sebelum Anda menggunakan pengiriman file AWS IoT berbasis MQTT dari perangkat Anda, Anda harus mengikuti langkah-langkah di bagian berikutnya untuk memastikan bahwa perangkat Anda diotorisasi dengan benar dan dapat terhubung ke Device Gateway. AWS IoT

## Berikan izin ke perangkat Anda

Anda dapat mengikuti langkah-langkah di [Membuat AWS IoT kebijakan](#) untuk membuat kebijakan perangkat atau menggunakan kebijakan perangkat yang ada. Lampirkan kebijakan ke sertifikat yang terkait dengan perangkat Anda dan tambahkan izin berikut ke kebijakan perangkat.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": ["iot:Connect"],
 "Resource": [
 "arn:partition:iot:region:accountID:client/
 ${iot:Connection.Thing.ThingName}"
]
 },
 {
 "Effect": "Allow",
```

```

 "Action": ["iot:Receive", "iot:Publish"],
 "Resource": [
 "arn:partition:iot:region:accountID:topic/$aws/things/
 ${iot:Connection.Thing.ThingName}/streams/*"
]
 },
 {
 "Effect": "Allow",
 "Action": "iot:Subscribe",
 "Resource": [
 "arn:partition:iot:region:accountID:topicfilter/$aws/things/
 ${iot:Connection.Thing.ThingName}/streams/*"
]
 }
]
}

```

## Hubungkan perangkat Anda ke AWS IoT

Perangkat yang menggunakan pengiriman file AWS IoT berbasis MQTT diperlukan untuk terhubung dengannya. AWS IoT AWS IoT Pengiriman file berbasis MQTT terintegrasi dengan AWS IoT di AWS Cloud, sehingga perangkat Anda harus langsung terhubung ke titik [akhir Data Plane](#). AWS IoT

### Note

Titik akhir bidang AWS IoT data khusus untuk Akun AWS dan Wilayah. Anda harus menggunakan titik akhir untuk Akun AWS dan Wilayah tempat perangkat Anda terdaftar. AWS IoT

Untuk informasi selengkapnya, lihat [Menghubungkan ke AWS IoT Core](#).

## TagResource Pemakaian

Tindakan CreateStream API membuat aliran untuk mengirimkan satu atau lebih file besar dalam potongan melalui MQTT.

Panggilan CreateStream API yang berhasil memerlukan izin berikut:

- `iot:CreateStream`
- `iot:TagResource(CreateStreamjika dengan tag)`

Kebijakan yang mendukung dua izin tersebut ditunjukkan di bawah ini:

```
{
 "Version": "2012-10-17",
 "Statement": {
 "Action": ["iot:CreateStream", "iot:TagResource"],
 "Effect": "Allow",
 "Resource": "arn:partition:iot:region:accountID:stream/streamId",
 }
}
```

Tindakan pernyataan `iot:TagResource` kebijakan diperlukan untuk memastikan pengguna tidak dapat membuat atau memperbarui tag pada sumber daya tanpa izin yang tepat. Tanpa tindakan pernyataan kebijakan khusus dari `iot:TagResource`, panggilan `CreateStream` API akan mengembalikan `AccessDeniedException` jika permintaan dilengkapi dengan tag.

Untuk informasi lebih lanjut, lihat tautan berikut:

- [CreateStream](#)
- [TagResource](#)
- [Tag](#)

## Menggunakan pengiriman AWS IoT file berbasis MQTT di perangkat

Untuk memulai proses transfer data, perangkat harus menerima kumpulan data awal, yang mencakup ID aliran minimal. Anda dapat menggunakan [Tugas](#) untuk menjadwalkan tugas transfer data untuk perangkat Anda dengan menyertakan kumpulan data awal dalam dokumen pekerjaan. Ketika perangkat menerima kumpulan data awal, perangkat tersebut kemudian harus memulai interaksi dengan pengiriman file berbasis AWS IoT MQTT. Untuk bertukar data dengan pengiriman file AWS IoT berbasis MQTT, perangkat harus:

- Gunakan protokol MQTT untuk berlangganan. [Topik pengiriman file berbasis MQTT](#)
- Kirim permintaan dan kemudian tunggu untuk menerima tanggapan menggunakan pesan MQTT.

Anda dapat secara opsional menyertakan ID file streaming dan versi aliran dalam kumpulan data awal. Mengirim ID file streaming ke perangkat dapat menyederhanakan pemrograman firmware/

perangkat lunak perangkat, karena menghilangkan kebutuhan untuk membuat `DescribeStream` permintaan dari perangkat untuk mendapatkan ID ini. Perangkat dapat menentukan versi streaming dalam `GetStream` permintaan untuk menerapkan pemeriksaan konsistensi jika aliran telah diperbarui secara tidak terduga.

## Gunakan `DescribeStream` untuk mendapatkan data streaming

AWS IoT Pengiriman file berbasis MQTT menyediakan `DescribeStream` API untuk mengirim data aliran ke perangkat. Data aliran yang dikembalikan oleh API ini mencakup ID aliran, versi aliran, deskripsi aliran, dan daftar file aliran, yang masing-masing memiliki ID file dan ukuran file dalam byte. Dengan informasi ini, perangkat dapat memilih file arbitrer untuk memulai proses transfer data.

### Note

Anda tidak perlu menggunakan `DescribeStream` API jika perangkat Anda menerima semua ID file streaming yang diperlukan dalam kumpulan data awal.

Ikuti langkah-langkah ini untuk membuat `DescribeStream` permintaan.

1. Berlangganan filter topik “diterima”`$aws/things/ThingName/streams/StreamId/description/json`.
2. Berlangganan filter topik “ditolak”`$aws/things/ThingName/streams/StreamId/rejected/json`.
3. Publikasikan `DescribeStream` permintaan dengan mengirim pesan ke `$aws/things/ThingName/streams/StreamId/describe/json`.
4. Jika permintaan diterima, perangkat Anda menerima `DescribeStream` respons pada filter topik “diterima”.
5. Jika permintaan ditolak, perangkat Anda menerima respons kesalahan pada filter topik “ditolak”.

### Note

Jika Anda mengganti `json` dengan `cbor` topik dan filter topik yang ditampilkan, perangkat Anda menerima pesan dalam format CBOR, yang lebih ringkas daripada JSON.

## DescribeStream permintaan

DescribeStreamPermintaan khas di JSON terlihat seperti contoh berikut.

```
{
 "c": "ec944cfb-1e3c-49ac-97de-9dc4aaad0039"
}
```

- (Opsional) `c` adalah bidang token klien.

Token klien tidak boleh lebih dari 64 byte. Token klien yang lebih panjang dari 64 byte menyebabkan respons kesalahan dan pesan `InvalidRequest` kesalahan.

## DescribeStream respon

DescribeStreamRespons di JSON terlihat seperti contoh berikut.

```
{
 "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` adalah bidang token klien. Ini dikembalikan jika diberikan dalam `DescribeStream` permintaan. Gunakan token klien untuk mengaitkan respons dengan permintaannya.
- `s` adalah versi aliran sebagai bilangan bulat. Anda dapat menggunakan versi ini untuk melakukan pemeriksaan konsistensi dengan `GetStream` permintaan Anda.
- `r` berisi daftar file dalam aliran.
  - `f` adalah ID file aliran sebagai bilangan bulat.

- “z” adalah ukuran file stream dalam jumlah byte.
- “d” berisi deskripsi aliran.

## Dapatkan blok data dari file streaming

Anda dapat menggunakan `GetStream` API sehingga perangkat dapat menerima file streaming dalam blok data kecil, sehingga dapat digunakan oleh perangkat yang memiliki kendala dalam memproses ukuran blok besar. Untuk menerima seluruh file data, perangkat mungkin perlu mengirim atau menerima beberapa permintaan dan tanggapan sampai semua blok data diterima dan diproses.

### GetStream permintaan

Ikuti langkah-langkah ini untuk membuat `GetStream` permintaan.

1. Berlangganan filter topik “diterima”`$aws/things/ThingName/streams/StreamId/data/json`.
2. Berlangganan filter topik “ditolak”`$aws/things/ThingName/streams/StreamId/rejected/json`.
3. Publikasikan `GetStream` permintaan ke topik`$aws/things/ThingName/streams/StreamId/get/json`.
4. Jika permintaan diterima, perangkat Anda akan menerima satu atau lebih `GetStream` tanggapan pada filter topik “diterima”. Setiap pesan respons berisi informasi dasar dan muatan data untuk satu blok.
5. Ulangi langkah 3 dan 4 untuk menerima semua blok data. Anda harus mengulangi langkah-langkah ini jika jumlah data yang diminta lebih besar dari 128 KB. Anda harus memprogram perangkat Anda untuk menggunakan beberapa `GetStream` permintaan untuk menerima semua data yang diminta.
6. Jika permintaan ditolak, perangkat Anda akan menerima respons kesalahan pada filter topik “ditolak”.

#### Note

- Jika Anda mengganti “json” dengan “cbor” dalam topik dan filter topik yang ditampilkan, perangkat Anda akan menerima pesan dalam format CBOR, yang lebih ringkas daripada JSON.



- AWS IoT Pengiriman file berbasis MQTT membatasi ukuran blok hingga 128 KB. Jika Anda membuat permintaan untuk blok yang lebih dari 128 KB, permintaan akan gagal.
- Anda dapat membuat permintaan untuk beberapa blok yang ukuran totalnya lebih besar dari 128 KB (misalnya, jika Anda membuat permintaan untuk 5 blok masing-masing 32 KB dengan total 160 KB data). Dalam hal ini, permintaan tidak gagal, tetapi perangkat Anda harus membuat beberapa permintaan untuk menerima semua data yang diminta. Layanan akan mengirim blok tambahan saat perangkat Anda membuat permintaan tambahan. Kami menyarankan Anda melanjutkan dengan permintaan baru hanya setelah tanggapan sebelumnya telah diterima dan diproses dengan benar.
- Terlepas dari ukuran total data yang diminta, Anda harus memprogram perangkat Anda untuk memulai percobaan ulang ketika blok tidak diterima, atau tidak diterima dengan benar.

GetStreamPermintaan khas di JSON terlihat seperti contoh berikut.

```
{
 "c": "1bb8aaa1-5c18-4d21-80c2-0b44fee10380",
 "s": 1,
 "f": 0,
 "l": 4096,
 "o": 2,
 "n": 100,
 "b": "...
}
```

- [opsional] c "" adalah bidang token klien.

Token klien tidak boleh lebih dari 64 byte. Token klien yang lebih panjang dari 64 byte menyebabkan respons kesalahan dan pesan `InvalidRequest` kesalahan.

- [opsional] s "" adalah bidang versi aliran (bilangan bulat).

Pengiriman file berbasis MQTT menerapkan pemeriksaan konsistensi berdasarkan versi yang diminta ini dan versi streaming terbaru di cloud. Jika versi streaming yang dikirim dari perangkat dalam `GetStream` permintaan tidak cocok dengan versi streaming terbaru di cloud, layanan akan mengirimkan respons kesalahan dan pesan `VersionMismatch` kesalahan. Biasanya, perangkat menerima versi aliran yang diharapkan (terbaru) dalam kumpulan data awal atau sebagai respons terhadap `DescribeStream`.

- “f” adalah ID file aliran (bilangan bulat dalam kisaran 0 hingga 255).

ID file stream diperlukan saat Anda membuat atau memperbarui aliran menggunakan AWS CLI atau SDK. Jika perangkat meminta file streaming dengan ID yang tidak ada, layanan akan mengirimkan respons kesalahan dan pesan `ResourceNotFound` kesalahan.

- “l” adalah ukuran blok data dalam byte (bilangan bulat dalam kisaran 256 hingga 131.072).

Lihat [Membangun bitmap untuk permintaan GetStream](#) petunjuk tentang cara menggunakan bidang bitmap untuk menentukan bagian file aliran mana yang akan dikembalikan dalam `GetStream` respons. Jika perangkat menentukan ukuran blok yang berada di luar jangkauan, layanan akan mengirimkan respons kesalahan dan pesan `BlockSizeOutOfBounds` kesalahan.

- [opsional] o "" adalah offset blok dalam file aliran (bilangan bulat dalam kisaran 0 hingga 98.304).

Lihat [Membangun bitmap untuk permintaan GetStream](#) petunjuk tentang cara menggunakan bidang bitmap untuk menentukan bagian file aliran mana yang akan dikembalikan dalam `GetStream` respons. Nilai maksimum 98.304 didasarkan pada batas ukuran file aliran 24 MB dan 256 byte untuk ukuran blok minimum. Defaultnya adalah 0 jika tidak ditentukan.

- [opsional] n "" adalah jumlah blok yang diminta (bilangan bulat dalam kisaran 0 hingga 98.304).

Bidang “n” menentukan baik (1) jumlah blok yang diminta, atau (2) ketika bidang bitmap (“b”) digunakan, batas jumlah blok yang akan dikembalikan oleh permintaan bitmap. Penggunaan kedua ini opsional. Jika tidak didefinisikan, defaultnya ke 131072/. *DataBlockSize*

- [opsional] b "" adalah bitmap yang mewakili blok yang diminta.

Menggunakan bitmap, perangkat Anda dapat meminta blok non-berturut-turut, yang membuat penanganan percobaan ulang setelah kesalahan menjadi lebih nyaman. Lihat [Membangun bitmap untuk permintaan GetStream](#) petunjuk tentang cara menggunakan bidang bitmap untuk menentukan bagian file aliran mana yang akan dikembalikan `GetStream` sebagai respons. Untuk bidang ini, ubah bitmap menjadi string yang mewakili nilai bitmap dalam notasi heksadesimal. Bitmap harus kurang dari 12.288 byte.

#### Important

Entah n "" atau b "" harus ditentukan. Jika tidak satu pun dari mereka ditentukan, `GetStream` permintaan mungkin tidak valid ketika ukuran file kurang dari 131072 byte (128 KB).

## GetStream respon

GetStreamRespon di JSON terlihat seperti contoh ini untuk setiap blok data yang diminta.

```
{
 "c": "1bb8aaa1-5c18-4d21-80c2-0b44fee10380",
 "f": 0,
 "l": 4096,
 "i": 2,
 "p": "..."}
}
```

- “c” adalah bidang token klien. Ini dikembalikan jika diberikan dalam GetStream permintaan. Gunakan token klien untuk mengaitkan respons dengan permintaannya.
- “f” adalah ID dari file aliran tempat muatan blok data saat ini berada.
- “l” adalah ukuran payload blok data dalam byte.
- “i” adalah ID dari blok data yang terdapat dalam payload. Blok data diberi nomor mulai dari 0.
- “p” berisi payload blok data. Bidang ini adalah string, yang mewakili nilai blok data dalam pengkodean [Base64](#).

## Membangun bitmap untuk permintaan GetStream

Anda dapat menggunakan bidang bitmap (b) dalam GetStream permintaan untuk mendapatkan blok non-berturut-turut dari file streaming. Ini membantu perangkat dengan kapasitas RAM terbatas menangani masalah pengiriman jaringan. Perangkat hanya dapat meminta blok-blok yang tidak diterima atau tidak diterima dengan benar. Bitmap menentukan blok file stream mana yang akan dikembalikan. Untuk setiap bit, yang diatur ke 1 di bitmap, blok yang sesuai dari file aliran akan dikembalikan.

Berikut adalah contoh cara menentukan bitmap dan bidang pendukungnya dalam GetStream permintaan. Misalnya, Anda ingin menerima file streaming dalam potongan 256 byte (ukuran blok). Pikirkan setiap blok 256 byte memiliki nomor yang menentukan posisinya dalam file, mulai dari 0. Jadi blok 0 adalah blok pertama 256 byte dalam file, blok 1 adalah yang kedua, dan seterusnya. Anda ingin meminta blok 20, 21, 24 dan 43 dari file.

## Blok offset

Karena blok pertama adalah nomor 20, tentukan offset (bidango) sebagai 20 untuk menghemat ruang di bitmap.

## Jumlah blok

Untuk memastikan bahwa perangkat Anda tidak menerima lebih banyak blok daripada yang dapat ditangani dengan sumber daya memori terbatas, Anda dapat menentukan jumlah maksimum blok yang harus dikembalikan dalam setiap pesan yang dikirim oleh pengiriman file berbasis MQTT. Perhatikan bahwa nilai ini diabaikan jika bitmap itu sendiri menentukan kurang dari jumlah blok ini, atau jika itu akan membuat ukuran total pesan respons yang dikirim oleh pengiriman file berbasis MQTT lebih besar dari batas layanan 128 KB per permintaan. `GetStream`

## Blokir bitmap

Bitmap itu sendiri adalah array byte yang tidak ditandatangani yang dinyatakan dalam notasi heksadesimal, dan termasuk dalam `GetStream` permintaan sebagai representasi string dari nomor tersebut. Tetapi untuk membangun string ini, mari kita mulai dengan memikirkan bitmap sebagai urutan panjang bit (bilangan biner). Jika sedikit dalam urutan ini diatur ke 1, blok yang sesuai dari file aliran akan dikirim kembali ke perangkat. Sebagai contoh, kita ingin menerima blok 20, 21, 24, dan 43, jadi kita harus mengatur bit 20, 21, 24, dan 43 di bitmap kita. Kita dapat menggunakan blok offset untuk menghemat ruang, jadi setelah kita mengurangi offset dari setiap nomor blok, kita ingin mengatur bit 0, 1, 4, dan 23, seperti contoh berikut.

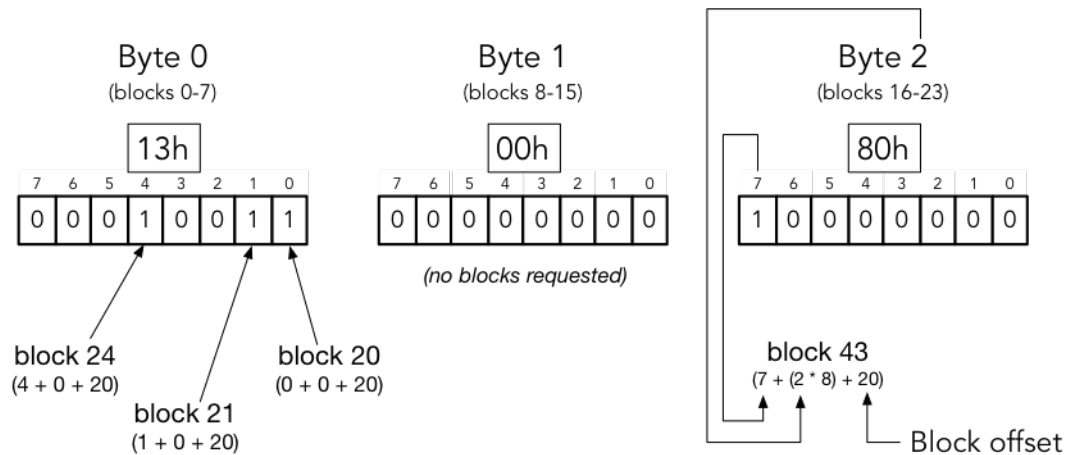
```
1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
```

Mengambil satu byte (8 bit) pada satu waktu, ini secara konvensional ditulis sebagai: "0b00010011", "0b000000", dan "0b10000000". Bit 0 muncul dalam representasi biner kami di akhir byte pertama, dan bit 23 di awal yang terakhir. Ini bisa membingungkan kecuali Anda tahu konvensi. Byte pertama berisi bit 7-0 (dalam urutan itu), byte kedua berisi bit 15-8, byte ketiga berisi bit 23-16, dan seterusnya. Dalam notasi heksadesimal, ini dikonversi menjadi "0x130080".

### Tip

Anda dapat mengonversi biner standar menjadi notasi heksadesimal. Ambil empat digit biner sekaligus dan ubah ini menjadi ekuivalen heksadesimalnya. Misalnya, "0001" menjadi "1", "0011" menjadi "3" dan seterusnya.

## Block bitmap breakdown



$$\text{block number} = (\text{bit position} + (\text{byte offset} * 8) + \text{base offset})$$

Menyatukan ini semua, JSON untuk GetStream permintaan kami terlihat seperti berikut.

```
{
 "c" : "1",
 "s" : 1,
 "l" : 256,
 "f" : 1,
 "o" : 20,
 "n" : 32,
 "b" : "130080"
}
```

- "c" adalah bidang token klien.
- "s" adalah versi streaming yang diharapkan.
- "l" adalah ukuran payload blok data dalam byte.
- "f" adalah ID dari indeks file sumber.
- "o" adalah offset blok.
- "n" adalah jumlah blok.
- "b" adalah bitmap BlockID yang hilang mulai dari offset. Nilai ini harus berdasarkan 64-dikodekan.

## Menangani kesalahan dari pengiriman AWS IoT file berbasis MQTT

Respons kesalahan yang dikirim ke perangkat untuk keduanya `DescribeStream` dan `GetStream` API berisi token klien, kode kesalahan, dan pesan kesalahan. Respons kesalahan tipikal terlihat seperti contoh berikut.

```
{
 "o": "BlockSizeOutOfBounds",
 "m": "The block size is out of bounds",
 "c": "1bb8aaa1-5c18-4d21-80c2-0b44fee10380"
}
```

- “o” adalah kode kesalahan yang menunjukkan alasan terjadinya kesalahan. Lihat kode kesalahan nanti di bagian ini untuk detail selengkapnya.
- “m” adalah pesan kesalahan yang berisi rincian kesalahan.
- “c” adalah bidang token klien. Ini dapat dikembalikan jika diberikan dalam `DescribeStream` permintaan. Anda dapat menggunakan token klien untuk mengaitkan respons dengan permintaannya.

Bidang token klien tidak selalu disertakan dalam respons kesalahan. Ketika token klien yang diberikan dalam permintaan tidak valid atau salah bentuk, itu tidak dikembalikan dalam respons kesalahan.

### Note

Untuk kompatibilitas mundur, bidang dalam respons kesalahan mungkin dalam bentuk yang tidak disingkat. Misalnya, kode kesalahan mungkin ditunjuk oleh bidang “kode” atau “o” dan bidang token klien dapat ditunjuk oleh bidang “ClientToken” atau “c”. Kami menyarankan Anda menggunakan formulir singkatan yang ditunjukkan di atas.

### InvalidTopic

Topik MQTT dari pesan aliran tidak valid.

### InvalidJson

Permintaan Stream bukan dokumen JSON yang valid.

## InvalidCbor

Permintaan Stream bukan dokumen CBOR yang valid.

## InvalidRequest

Permintaan umumnya diidentifikasi sebagai cacat. Untuk informasi selengkapnya, lihat pesan kesalahan.

## Tidak sah

Permintaan tidak diizinkan untuk mengakses file data streaming di media penyimpanan, seperti Amazon S3. Untuk informasi selengkapnya, lihat pesan kesalahan.

## BlockSizeOutOfBounds

Ukuran blok di luar batas. [Lihat bagian "Pengiriman File Berbasis MQTT" di Service Quotas.AWS IoT Core](#)

## OffsetOutOfBounds

Offset di luar batas. [Lihat bagian "Pengiriman File Berbasis MQTT" di Service Quotas.AWS IoT Core](#)

## BlockCountLimitExceeded

Jumlah blok permintaan berada di luar batas. [Lihat bagian "Pengiriman File Berbasis MQTT" di Service Quotas.AWS IoT Core](#)

## BlockBitmapLimitExceeded

Ukuran bitmap permintaan di luar batas. [Lihat bagian "Pengiriman File Berbasis MQTT" di Service Quotas.AWS IoT Core](#)

## ResourceNotFound

Aliran, file, versi file, atau blok yang diminta tidak ditemukan. Lihat pesan kesalahan untuk detail selengkapnya.

## VersionMismatch

Versi streaming dalam permintaan tidak cocok dengan versi streaming di fitur pengiriman file berbasis MQTT. Ini menunjukkan bahwa data aliran telah dimodifikasi sejak versi streaming awalnya diterima oleh perangkat.

## E TagMismatch

S3 ETag dalam aliran tidak cocok dengan ETag dari versi objek S3 terbaru.

## InternalError

Terjadi kesalahan internal dalam pengiriman file berbasis MQTT.

## Contoh kasus penggunaan di FreeRTOS OTA

Agen FreeRTOS OTA over-the-air () AWS IoT menggunakan pengiriman file berbasis MQTT untuk mentransfer gambar firmware FreeRTOS ke perangkat FreeRTOS. Untuk mengirim kumpulan data awal ke perangkat, ia menggunakan layanan AWS IoT Job untuk menjadwalkan pekerjaan pembaruan OTA ke perangkat FreeRTOS.

Untuk implementasi referensi klien pengiriman file berbasis MQTT, lihat kode agen FreeRTOS [OTA dalam dokumentasi FreeRTOS](#).



# Penasihat Perangkat

[Device Advisor](#) adalah kemampuan pengujian berbasis cloud yang dikelola sepenuhnya untuk memvalidasi perangkat IoT selama pengembangan perangkat lunak perangkat. Device Advisor menyediakan pengujian bawaan yang dapat Anda gunakan untuk memvalidasi perangkat IoT untuk konektivitas yang andal dan aman AWS IoT Core dengan, sebelum menerapkan perangkat ke produksi. [Pengujian bawaan Device Advisor membantu Anda memvalidasi perangkat lunak perangkat Anda terhadap praktik terbaik untuk penggunaan Pekerjaan TLS, MQTT, Device Shadow, dan IoT](#). Anda juga dapat mengunduh laporan kualifikasi yang ditandatangani untuk dikirimkan ke Jaringan AWS Mitra agar perangkat Anda memenuhi syarat untuk [Katalog Perangkat AWS Mitra](#) tanpa perlu mengirim perangkat Anda dan menunggu perangkat Anda diuji.

## Note

Device Advisor didukung di wilayah us-east-1, us-west-2, ap-northeast-1, dan eu-west-1. Device Advisor mendukung perangkat dan klien yang menggunakan protokol MQTT dan MQTT over WebSocket Secure (WSS) untuk mempublikasikan dan berlangganan pesan. Semua protokol mendukung IPv4 dan IPv6. Device Advisor mendukung sertifikat server RSA.

Perangkat apa pun yang telah dibangun untuk terhubung AWS IoT Core dapat memanfaatkan Device Advisor. Anda dapat mengakses Device Advisor dari [AWS IoT konsol](#), atau dengan menggunakan AWS CLI atau SDK. Saat Anda siap untuk menguji perangkat Anda, daftarkan AWS IoT Core dan konfigurasi perangkat lunak perangkat dengan titik akhir Device Advisor. Kemudian pilih pengujian bawaan, konfigurasi, jalankan pengujian di perangkat Anda, dan dapatkan hasil pengujian beserta log terperinci atau laporan kualifikasi.

Device Advisor adalah titik akhir pengujian di cloud. AWS Anda dapat menguji perangkat dengan mengonfigurasinya agar tersambung ke titik akhir pengujian yang disediakan oleh Device Advisor. Setelah perangkat dikonfigurasi untuk menyambung ke titik akhir pengujian, Anda dapat mengunjungi konsol Device Advisor atau menggunakan AWS SDK untuk memilih pengujian yang ingin dijalankan di perangkat. Device Advisor kemudian mengelola siklus hidup penuh pengujian, termasuk penyediaan sumber daya, penjadwalan proses pengujian, mengelola mesin status, merekam perilaku perangkat, mencatat hasil, dan memberikan hasil akhir dalam bentuk laporan pengujian.

## Protokol TLS

Protokol Transport Layer Security (TLS) digunakan untuk mengenkripsi data rahasia melalui jaringan yang tidak aman seperti internet. Protokol TLS adalah penerus protokol Secure Sockets Layer (SSL).

Device Advisor mendukung protokol TLS berikut:

- TLS 1.3 (disarankan)
- TLS 1.2

Protokol, pemetaan port, dan otentikasi

Protokol komunikasi perangkat digunakan oleh perangkat atau klien untuk terhubung ke broker pesan dengan menggunakan titik akhir perangkat. Tabel berikut mencantumkan protokol yang didukung oleh titik akhir Device Advisor serta metode otentikasi serta port yang digunakan.

Protokol, otentikasi, dan pemetaan port

| Protokol                | Operasi yang didukung         | Autentikasi               | Port | Nama protokol ALPN |
|-------------------------|-------------------------------|---------------------------|------|--------------------|
| MQTT lebih<br>WebSocket | Publikasikan,<br>Berlangganan | Tanda Tangan<br>Versi 4   | 443  | N/A                |
| MQTT                    | Publikasikan,<br>Berlangganan | Sertifikat klien<br>X.509 | 8883 | x-amzn-mq<br>tt-ca |
| MQTT                    | Publikasikan,<br>Berlangganan | Sertifikat klien<br>X.509 | 443  | N/A                |

Bab ini berisi bagian-bagian berikut:

- [Menyiapkan](#)
- [Memulai dengan Device Advisor di konsol](#)
- [Alur kerja Device Advisor](#)
- [Alur kerja konsol terperinci Device Advisor](#)
- [Durasi panjang menguji alur kerja konsol](#)
- [Titik akhir VPC Penasihat Perangkat \( \)AWS PrivateLink](#)
- [Kasus uji Device Advisor](#)

# Menyiapkan

Sebelum Anda menggunakan Device Advisor untuk pertama kalinya, selesaikan tugas-tugas berikut:

## Buat hal IoT

Pertama, buat IoT dan lampirkan sertifikat ke benda itu. Untuk tutorial tentang cara membuat sesuatu, lihat [Create a thing object](#).


## Membuat peran IAM untuk digunakan sebagai peran perangkat

### Note

Anda dapat dengan cepat membuat peran perangkat dengan konsol Device Advisor. Untuk mempelajari cara mengatur peran perangkat dengan konsol Device Advisor, lihat [Memulai Device Advisor di konsol](#).


1. Buka [AWS Identity and Access Management konsol](#) dan masuk ke yang Akun AWS Anda gunakan untuk pengujian Device Advisor.
2. Di panel navigasi kiri, pilih Kebijakan.
3. Pilih Buat kebijakan.
4. Di bawah Create policy, lakukan hal berikut:
  - a. Untuk Layanan, pilih IoT.
  - b. Di bawah Tindakan, lakukan salah satu hal berikut:
    - (Disarankan) Pilih tindakan berdasarkan kebijakan yang dilampirkan pada hal IoT atau sertifikat yang Anda buat di bagian sebelumnya.
    - Cari tindakan berikut di kotak tindakan Filter dan pilih:
      - Connect
      - Publish
      - Subscribe
      - Receive
      - RetainPublish

- c. Di bawah Sumber Daya, batasi sumber daya klien, topik, dan topik. Membatasi sumber daya ini adalah praktik terbaik keamanan. Untuk membatasi sumber daya, lakukan hal berikut:
  - i. Pilih Tentukan ARN sumber daya klien untuk tindakan Connect.
  - ii. Pilih Add ARN, lalu lakukan salah satu hal berikut:

 Note


ClientID adalah ID klien MQTT yang digunakan perangkat Anda untuk berinteraksi dengan Device Advisor.

- Tentukan Region, AccountID, dan clientID di editor ARN visual.
  - Masukkan Nama Sumber Daya Amazon (ARN) secara manual dari topik IoT yang ingin Anda jalankan dengan kasus pengujian.
- iii. Pilih Tambahkan.
  - iv. Pilih Tentukan sumber topik ARN untuk Menerima dan satu tindakan lagi.
  - v. Pilih Add ARN, lalu lakukan salah satu hal berikut:

 Note


Nama topik adalah topik MQTT tempat perangkat Anda mempublikasikan pesan.

- Tentukan nama Wilayah, AccountID, dan Topik di editor ARN visual.
  - Masukkan ARN topik IoT secara manual yang ingin Anda jalankan dengan kasus pengujian.
- vi. Pilih Tambahkan.
  - vii. Pilih Tentukan sumber daya TopicFilter ARN untuk tindakan Berlangganan.
  - viii. Pilih Add ARN, lalu lakukan salah satu hal berikut:

 Note

Nama topik adalah topik MQTT tempat perangkat Anda berlangganan.

- Tentukan nama Wilayah, AccountID, dan Topik di editor ARN visual.
  - Masukkan ARN topik IoT secara manual yang ingin Anda jalankan dengan kasus pengujian.
- ix. Pilih Tambahkan.
5. Pilih Berikutnya: Tanda.
  6. Pilih Berikutnya: Tinjauan.
  7. Di bawah Kebijakan peninjauan, masukkan Nama untuk kebijakan Anda.
  8. Pilih Buat kebijakan.
  9. Di panel navigasi kiri, Pilih Peran.
  10. Pilih Buat Peran.
  11. Di bawah Pilih entitas tepercaya, pilih Kebijakan kepercayaan khusus.
  12. Masukkan kebijakan kepercayaan berikut ke dalam kotak Kebijakan kepercayaan kustom. Untuk melindungi dari masalah deputi yang membingungkan, tambahkan kunci konteks kondisi global [aws:SourceArn](#) dan [aws:SourceAccount](#) ke kebijakan.

 Important

Anda `aws:SourceArn` harus mematuhi format :

`arn:aws:iotdeviceadvisor:region:account-id:*` . Pastikan yang *region* cocok dengan AWS IoT Wilayah Anda dan *account-id* cocok dengan ID akun pelanggan Anda. Untuk informasi lebih lanjut, lihat [Pencegahan Deputi Bingung Lintas Layanan](#).

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Sid": "AllowAwsIoTCoreDeviceAdvisor",
 "Effect": "Allow",
 "Principal": {
 "Service": "iotdeviceadvisor.amazonaws.com"
 },
 "Action": "sts:AssumeRole",
 "Condition": {
```

```
 "StringEquals": {
 "aws:SourceAccount": "111122223333"
 },
 "ArnLike": {
 "aws:SourceArn":
"arn:aws:iotdeviceadvisor:*:111122223333:suitedefinition/*"
 }
 }
}
]
```

13. Pilih Berikutnya.
14. Pilih kebijakan yang Anda buat di Langkah 4.
15. (Opsional) Di bawah Setel batas izin, pilih Gunakan batas izin untuk mengontrol izin peran maksimum, lalu pilih kebijakan yang Anda buat.
16. Pilih Berikutnya.
17. Masukkan nama Peran dan deskripsi Peran.
18. Pilih Buat peran.

## Membuat kebijakan yang dikelola khusus untuk pengguna IAM untuk menggunakan Device Advisor

1. Arahkan ke konsol IAM di <https://console.aws.amazon.com/iam/>. Jika diminta, masukkan AWS kredensial Anda untuk masuk.
2. Di panel navigasi di sebelah kiri, pilih Kebijakan.
3. Pilih Buat Kebijakan, lalu pilih tab JSON.
4. Tambahkan izin yang diperlukan untuk menggunakan Device Advisor. Dokumen kebijakan dapat ditemukan di topik [Praktik terbaik keamanan](#).
5. Pilih Tinjau Kebijakan.
6. Masukkan Nama dan Deskripsi.
7. Pilih Buat Kebijakan.

## Buat pengguna IAM untuk menggunakan Device Advisor

### Note

Sebaiknya Anda membuat pengguna IAM untuk digunakan saat menjalankan pengujian Device Advisor. Menjalankan pengujian Device Advisor dari pengguna admin dapat menimbulkan risiko keamanan dan tidak disarankan.

1. Arahkan ke konsol IAM di <https://console.aws.amazon.com/iam/> Jika diminta, masukkan AWS kredensial Anda untuk masuk.
2. Di panel navigasi kiri, Pilih Pengguna.
3. Pilih Tambah Pengguna.
4. Masukkan nama pengguna.
5. Pengguna membutuhkan akses terprogram jika mereka ingin berinteraksi dengan AWS luar. AWS Management Console Cara untuk memberikan akses terprogram tergantung pada jenis pengguna yang mengakses AWS.

Untuk memberi pengguna akses terprogram, pilih salah satu opsi berikut.

| Pengguna mana yang membutuhkan akses terprogram?                          | Untuk                                                                                                      | Oleh                                                                                                                                                                                                                                                                        |
|---------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Identitas tenaga kerja<br>(Pengguna yang dikelola di Pusat Identitas IAM) | Gunakan kredensial sementara untuk menandatangani permintaan terprogram ke AWS CLI, AWS SDK, atau API. AWS | Mengikuti petunjuk untuk antarmuka yang ingin Anda gunakan. <ul style="list-style-type: none"> <li>• Untuk AWS CLI, lihat <a href="#">Mengkonfigurasi yang akan AWS CLI digunakan AWS IAM Identity Center</a> dalam Panduan AWS Command Line Interface Pengguna.</li> </ul> |

| Pegguna mana yang membutuhkan akses terprogram? | Untuk                                                                                                                                       | Oleh                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|-------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                                 |                                                                                                                                             | <ul style="list-style-type: none"> <li>• Untuk AWS SDK, alat, dan AWS API, lihat <a href="#">autentikasi Pusat Identitas IAM</a> di Panduan Referensi AWS SDK dan Alat.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                       |
| IAM                                             | Gunakan kredensial sementara untuk menandatangani permintaan terprogram ke AWS CLI, AWS SDK, atau API. AWS                                  | Mengikuti petunjuk dalam <a href="#">Menggunakan kredensial sementara dengan AWS sumber daya</a> di Panduan Pengguna IAM.                                                                                                                                                                                                                                                                                                                                                                                                                                |
| IAM                                             | (Tidak direkomendasikan)<br>Gunakan kredensial jangka panjang untuk menandatangani permintaan terprogram ke AWS CLI, AWS SDK, atau API. AWS | <p>Mengikuti petunjuk untuk antarmuka yang ingin Anda gunakan.</p> <ul style="list-style-type: none"> <li>• Untuk mengetahui AWS CLI, lihat <a href="#">Mengautentikasi menggunakan kredensial pengguna IAM</a> di Panduan Pengguna. AWS Command Line Interface</li> <li>• Untuk AWS SDK dan alat bantu, lihat <a href="#">Mengautentikasi menggunakan kredensial jangka panjang</a> di Panduan Referensi AWS SDK dan Alat.</li> <li>• Untuk AWS API, lihat <a href="#">Mengelola kunci akses untuk pengguna IAM</a> di Panduan Pengguna IAM.</li> </ul> |

## 6. Pilih Selanjutnya: Izin.



7. Untuk memberikan akses, tambahkan izin ke pengguna, grup, atau peran Anda:
  - Pengguna dan grup di AWS IAM Identity Center:

Buat rangkaian izin. Ikuti petunjuk dalam [Buat set izin](#) dalam Panduan Pengguna AWS IAM Identity Center .
  - Pengguna yang dikelola di IAM melalui penyedia identitas:

Buat peran untuk federasi identitas. Ikuti petunjuk dalam [Membuat peran untuk penyedia identitas pihak ketiga \(federasi\)](#) di Panduan Pengguna IAM.
  - Pengguna IAM:
    - Buat peran yang dapat diambil pengguna Anda. Ikuti petunjuk dalam [Membuat peran untuk pengguna IAM](#) dalam Panduan Pengguna IAM.
    - (Tidak disarankan) Pasang kebijakan langsung ke pengguna atau tambahkan pengguna ke grup pengguna. Ikuti petunjuk di [Menambahkan izin ke pengguna \(konsol\)](#) dalam Panduan Pengguna IAM.
8. Masukkan nama kebijakan yang dikelola khusus yang Anda buat di kotak pencarian. Kemudian, pilih kotak centang untuk Nama kebijakan.
9. Pilih Berikutnya: Tanda.
10. Pilih Berikutnya: Tinjauan.
11. Pilih Create user (Buat pengguna).
12. Pilih Tutup.

Device Advisor memerlukan akses ke AWS sumber daya Anda (barang, sertifikat, dan titik akhir) atas nama Anda. Pengguna IAM Anda harus memiliki izin yang diperlukan. Device Advisor juga akan menerbitkan log ke Amazon CloudWatch jika Anda melampirkan kebijakan izin yang diperlukan untuk pengguna IAM Anda.

## Konfigurasi perangkat Anda

Device Advisor menggunakan ekstensi TLS indikasi nama server (SNI) untuk menerapkan konfigurasi TLS. Perangkat harus menggunakan ekstensi ini saat terhubung dan meneruskan nama server yang identik dengan titik akhir pengujian Device Advisor.

Device Advisor memungkinkan koneksi TLS saat tes dalam keadaan. Running Ini menyangkal koneksi TLS sebelum dan sesudah setiap uji coba. Untuk alasan ini, kami menyarankan Anda

menggunakan mekanisme coba lagi koneksi perangkat untuk pengalaman pengujian otomatis sepenuhnya dengan Device Advisor. Anda dapat menjalankan rangkaian pengujian yang mencakup lebih dari satu kasus uji, seperti TLS connect, MQTT connect, dan MQTT publish. Jika Anda menjalankan beberapa kasus pengujian, sebaiknya perangkat Anda mencoba menyambung ke titik akhir pengujian kami setiap lima detik. Anda kemudian dapat mengotomatiskan menjalankan beberapa kasus uji secara berurutan.

#### Note

Untuk menyiapkan perangkat lunak perangkat Anda untuk pengujian, kami sarankan Anda menggunakan SDK yang dapat terhubung AWS IoT Core. Anda kemudian harus memperbarui SDK dengan titik akhir pengujian Device Advisor yang disediakan untuk Anda. Akun AWS

Device Advisor mendukung dua jenis endpoint: Account-level dan Device-level endpoint. Pilih titik akhir yang paling sesuai dengan kasus penggunaan Anda. Untuk menjalankan beberapa rangkaian pengujian secara bersamaan untuk perangkat yang berbeda, gunakan titik akhir tingkat Perangkat.

Jalankan perintah berikut untuk mendapatkan titik akhir tingkat Perangkat:

Untuk pelanggan MQTT yang menggunakan sertifikat klien X.509:

```
aws iotdeviceadvisor get-endpoint --thing-arn your-thing-arn
```

atau

```
aws iotdeviceadvisor get-endpoint --certificate-arn your-certificate-arn
```

Untuk MQTT atas WebSocket pelanggan yang menggunakan Signature Versi 4:

```
aws iotdeviceadvisor get-endpoint --device-role-arn your-device-role-arn --
authentication-method SignatureVersion4
```

Untuk menjalankan satu rangkaian pengujian pada satu waktu, pilih titik akhir tingkat Akun. Jalankan perintah berikut untuk mendapatkan titik akhir Account-level:

```
aws iotdeviceadvisor get-endpoint
```

## Memulai dengan Device Advisor di konsol

Tutorial ini membantu Anda memulai AWS IoT Core Device Advisor di konsol. Device Advisor menawarkan fitur-fitur seperti tes yang diperlukan dan laporan kualifikasi yang ditandatangani. Anda dapat menggunakan pengujian dan laporan ini untuk memenuhi syarat dan daftar perangkat di [Katalog Perangkat AWS Mitra](#) sebagaimana dirinci dalam [program AWS IoT Core kualifikasi](#).

Untuk informasi selengkapnya tentang menggunakan Device Advisor, lihat [Alur kerja Device Advisor](#) dan [Alur kerja konsol terperinci Device Advisor](#).

Untuk menyelesaikan tutorial ini, ikuti langkah-langkah yang diuraikan dalam [Menyiapkan](#).

### Note

Device Advisor didukung sebagai berikut: Wilayah AWS

- AS Timur (Virginia Utara)
- AS Barat (Oregon)
- Asia Pasifik (Tokyo)
- Eropa (Irlandia)

### Memulai

1. Di panel navigasi [AWS IoT konsol](#) di bawah Uji, pilih Device Advisor. Kemudian, pilih tombol Mulai walkthrough di konsol.

The screenshot shows the AWS IoT Core console interface. On the left, a navigation sidebar is visible with the 'Test' section expanded, and 'Device Advisor' highlighted with a red border. The main content area displays the 'Device Advisor' overview page, which includes a header with the title 'Device Advisor Fully managed test capability for IoT devices', a brief description, and a 'Getting started' section with a 'Start walkthrough' button. Below this is a 'How it works' diagram illustrating the process: an IoT Device connects to a test suite, and the user gets results and logs from the Device Advisor.

- Halaman Memulai dengan Device Advisor memberikan ikhtisar langkah-langkah yang diperlukan untuk membuat rangkaian pengujian dan menjalankan pengujian terhadap perangkat Anda. Anda juga dapat menemukan titik akhir pengujian Device Advisor untuk akun Anda di sini. Anda harus mengonfigurasi firmware atau perangkat lunak pada perangkat yang digunakan untuk pengujian agar terhubung ke titik akhir pengujian ini.

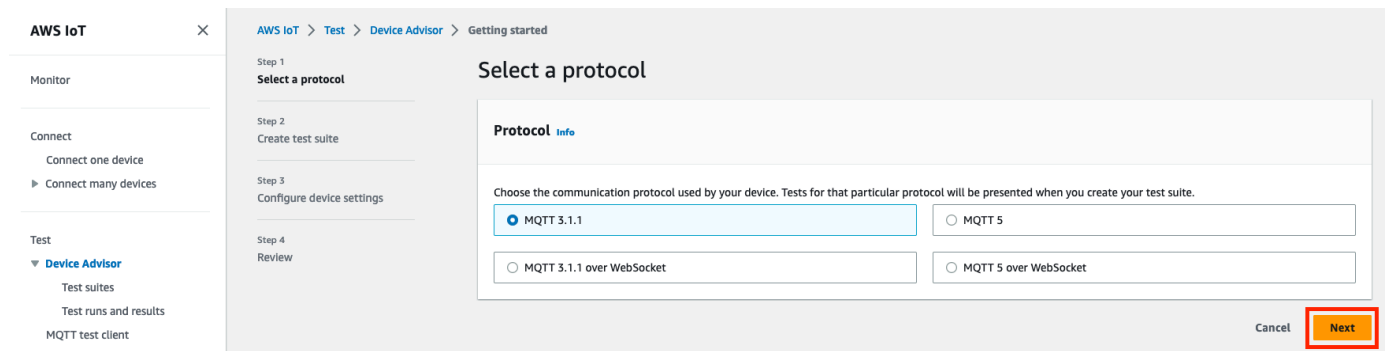
Untuk menyelesaikan tutorial ini, pertama [buat sesuatu dan sertifikat](#). Setelah Anda meninjau informasi di bawah Cara kerjanya, pilih Berikutnya.

The screenshot shows the 'Getting started' page for Device Advisor in the AWS IoT Core console. The page is titled 'Getting started' and contains a 'How it works' section. This section is divided into three steps:
 

- Step 1: Select a protocol**: Select a communication protocol used by your device. Tests for that particular protocol will be presented when you create your test suite.
- Step 2: Create a test suite**: Create a test suite with at least one test group and one test. You can make your own test suite from tests that verify your devices can reliably and securely connect to AWS IoT. You will specify the test settings that allow Device Advisor to work with your particular device. [Learn more about test suites](#)
- Step 3: Configure device settings**: Configure device settings to test. Device Advisor will verify that the device can securely and reliably connect to, interact with and receive updates from AWS IoT. You can get detailed logs to troubleshoot device issues.

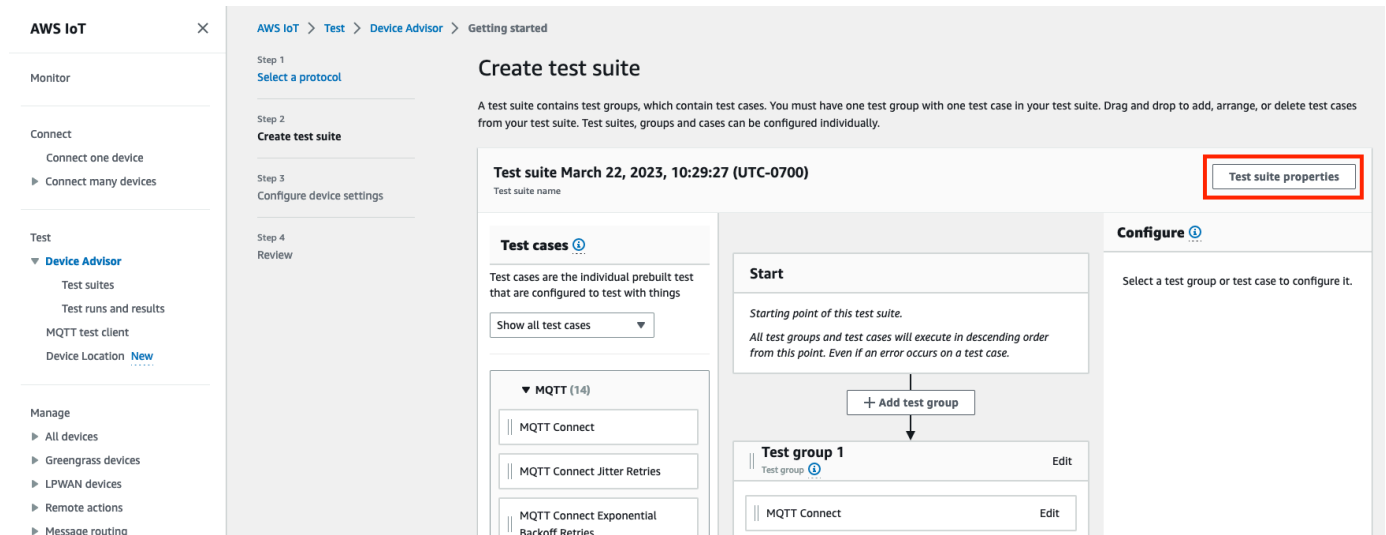
 At the bottom right of the page, there are 'Cancel' and 'Next' buttons, with the 'Next' button highlighted in a red box.

- Pada Langkah 1: Pilih protokol, pilih protokol dari opsi yang tercantum. Lalu, pilih Selanjutnya.



4. Pada Langkah 2, Anda membuat dan mengonfigurasi rangkaian pengujian khusus. Rangkaian pengujian khusus harus memiliki setidaknya satu kelompok uji, dan setiap kelompok uji harus memiliki setidaknya satu kasus uji. Kami telah menambahkan kasus uji MQTT Connect agar Anda dapat memulai.

Pilih properti Test suite.



Menyediakan properti rangkaian pengujian saat Anda membuat rangkaian pengujian. Anda dapat mengonfigurasi properti tingkat suite berikut:

- Nama suite uji: Masukkan nama untuk rangkaian pengujian Anda.
- Batas waktu (opsional): Batas waktu (dalam detik) untuk setiap kasus uji dalam rangkaian pengujian saat ini. Jika Anda tidak menentukan nilai batas waktu, nilai default akan digunakan.
- Tag (opsional): Tambahkan tag ke rangkaian pengujian.

Setelah selesai, pilih Perbarui properti.

**Test suite properties** ✕

**Test suite name**  
Specify a name for this test suite that you can search.

Device Advisor Demo Suite

**Timeout - optional**  
Optional, in seconds. Maximum time Device Advisor waits for a device to respond before tests fail.

300

No tags associated with the resource.

Add new tag

You can add up to 50 more tags.

Cancel **Update properties**

5. (Opsional) Untuk memperbarui konfigurasi grup rangkaian pengujian, pilih tombol Edit di sebelah nama grup pengujian.
- Nama: Masukkan nama kustom untuk grup test suite.
  - Batas waktu (opsional): Batas waktu (dalam detik) untuk setiap kasus uji dalam rangkaian pengujian saat ini. Jika Anda tidak menentukan nilai batas waktu, nilai default akan digunakan.

Setelah selesai, pilih Selesai untuk melanjutkan.

**AWS IoT** ×

AWS IoT > Test > Device Advisor > Getting started

Step 1  
Select a protocol

Step 2  
**Create test suite**

Step 3  
Configure device settings

Step 4  
Review

### Create test suite

A test suite contains test groups, which contain test cases. You must have one test group with one test case in your test suite. Drag and drop to add, arrange, or delete test cases from your test suite. Test suites, groups and cases can be configured individually.

**Device Advisor Demo Suite**  
Test suite name

**Test cases**  
Test cases are the individual prebuilt test that are configured to test with things

Show all test cases

MQTT (14)

- MQTT Connect
- MQTT Connect Jitter Retries
- MQTT Connect Exponential Backoff Retries
- MQTT Reconnect Backoff Retries On Server Disconnect

**Start**  
Starting point of this test suite.  
All test groups and test cases will execute in descending order from this point. Even if an error occurs on a test case.

+ Add test group

**Test group 1**  
Test group

MQTT Connect

**Configure**  
Test group 1

Name  
Specify a name for this test group.  
Test group 1

Timeout - optional  
Optional, in seconds. Maximum time Device Advisor waits for a device to respond before tests fail.  
value

Done

Cancel Delete

6. (Opsional) Untuk memperbarui konfigurasi kasus uji untuk kasus uji, pilih tombol Edit di sebelah nama kasus uji.

- Nama: Masukkan nama kustom untuk grup test suite.
- Batas waktu (opsional): Batas waktu (dalam detik) untuk kasus uji yang dipilih. Jika Anda tidak menentukan nilai batas waktu, nilai default akan digunakan.

Setelah selesai, pilih Selesai untuk melanjutkan.

**AWS IoT** ☰

AWS IoT > Test > Device Advisor > Getting started

Step 1  
Select an IoT thing or certificate

Step 2  
**Create test suite**

Step 3  
Select a device role

Step 4  
Review

### Create test suite

A test suite contains test groups, which contain test cases. You must have one test group with one test case in your test suite. Drag and drop to add, arrange, or delete test cases from your test suite. Test suites, groups and cases can be configured individually.

**Device Advisor demo suite**  
Test suite name

**Test cases**  
Test cases are the individual prebuilt test that are configured to test with things

Show all test cases

MQTT (14)

- MQTT Connect
- MQTT Connect Jitter Retries
- MQTT Connect Exponential Backoff Retries
- MQTT Reconnect Backoff Retries On Server Disconnect
- MQTT Reconnect Backoff Retries On Unstable Connection
- MQTT Subscribe

**Start**  
Starting point of this test suite.  
All test groups and test cases will execute in descending order from this point. Even if an error occurs on a test case.

+ Add test group

**Test group 1**  
Test group

MQTT Connect

**Configure**  
MQTT Connect

Name  
Specify a name for this test group.  
MQTT Connect

Timeout - optional  
Optional, in seconds. Maximum time Device Advisor waits for a device to respond before tests fail.  
value

Done

Cancel Delete

When the tests in this group are completed, testing will continue with the next group.

7. (Opsional) Untuk menambahkan lebih banyak grup pengujian ke rangkaian pengujian, pilih Tambahkan grup pengujian, lalu ikuti petunjuk di Langkah 5.
8. (Opsional) Untuk menambahkan lebih banyak kasus uji, seret kasus uji di bagian Kasus uji ke salah satu grup pengujian Anda.

9. Anda dapat mengubah urutan grup pengujian dan kasus uji Anda. Untuk membuat perubahan, seret kasus uji yang terdaftar ke atas atau ke bawah daftar. Device Advisor menjalankan pengujian sesuai urutan yang Anda cantumkan.

Setelah mengonfigurasi rangkaian pengujian, pilih Berikutnya.

10. Pada Langkah 3, pilih AWS IoT sesuatu atau sertifikat untuk diuji menggunakan Device Advisor. Jika Anda tidak memiliki barang atau sertifikat yang ada, lihat [Menyiapkan](#).



11. Anda dapat mengonfigurasi peran perangkat yang digunakan Device Advisor untuk melakukan tindakan AWS IoT MQTT atas nama perangkat pengujian Anda. Hanya untuk kasus uji MQTT Connect, tindakan Connect dipilih secara otomatis. Ini karena peran perangkat memerlukan izin ini untuk menjalankan rangkaian pengujian. Untuk kasus uji lainnya, tindakan yang sesuai dipilih.

Berikan nilai sumber daya untuk setiap tindakan yang dipilih. Misalnya, untuk tindakan Connect, berikan ID klien yang digunakan perangkat Anda untuk menyambung ke titik akhir Device Advisor. Anda dapat memberikan beberapa nilai dengan nilai terpisah koma, dan nilai awalan dengan karakter wildcard (\*). Misalnya, untuk memberikan izin untuk mempublikasikan topik apa pun yang dimulai dengan `MyTopic`, masukkan **MyTopic\*** sebagai nilai sumber daya.

**AWS IoT** ×

Monitor

Connect

Connect one device

▶ Connect many devices

Test

▼ **Device Advisor**

Test suites

Test runs and results

MQTT test client

Device Location [New](#)

Manage

▼ All devices

Things

Thing groups

Thing types

Fleet metrics

▶ Greengrass devices

▶ LPWAN devices

**Select a device role**

Device role [Info](#)

AWS IoT Core Device Advisor requires permission to perform AWS IoT MQTT actions on behalf of your test device.

Create new role  
Create and use a new device role

Select an existing role  
Use an existing device role

Role name

DeviceAdvisorServiceRole

Permissions [Info](#)

Choose which actions and the associated resources for AWS IoT Core Device Advisor to access using this role. You can enter a specific resource or resource prefix. To enter multiple values for a resource, use commas to separate the values. [Learn more](#)

| Action                                      | Resource type | Resource                                                                |
|---------------------------------------------|---------------|-------------------------------------------------------------------------|
| <input checked="" type="checkbox"/> Connect | Clientid      | MyClient                                                                |
| <input type="checkbox"/> Publish            | Topic         | Specify topics to publish to, e.g. MyTopic, MyTopic*                    |
| <input type="checkbox"/> Subscribe          | TopicFilter   | Specify topic filters to subscribe to, e.g. MyTopic, MyTopic*           |
| <input type="checkbox"/> Receive            | Topic         | Specify topics to receive from e.g. MyTopic, MyTopic*                   |
| <input type="checkbox"/> RetainPublish      | Topic         | Specify topics to publish a retained message to, e.g. MyTopic, MyTopic* |

Untuk menggunakan peran perangkat yang dibuat sebelumnya dari [Menyiapkan](#), pilih Pilih peran yang ada. Kemudian pilih peran perangkat Anda di bawah Pilih peran.

Device Location [New](#)

Manage

▼ All devices

Things

Thing groups

Thing types

Fleet metrics

▶ Greengrass devices

▶ LPWAN devices

**Select a device role**

Device role [Info](#)

AWS IoT Core Device Advisor requires permission to perform AWS IoT MQTT actions on behalf of your test device.

Create new role  
Create and use a new device role

Select an existing role  
Use an existing device role

Select role

DeviceAdvisorServiceRole

Konfigurasi peran perangkat Anda dengan salah satu dari dua opsi yang disediakan, lalu pilih Berikutnya.

12. Di bagian titik akhir Uji, pilih titik akhir yang paling sesuai dengan kasus penggunaan Anda. Untuk menjalankan beberapa rangkaian pengujian secara bersamaan dengan yang sama Akun

AWS, pilih Titik akhir tingkat perangkat. Untuk menjalankan satu rangkaian pengujian pada satu waktu, pilih Titik akhir tingkat Akun.

The screenshot shows the AWS IoT Core console interface. On the left is a navigation menu with categories like 'Remote actions', 'Message routing', 'Security', and 'Fleet Hub'. The main content area is titled 'Test endpoint' and contains instructions to choose between 'Account-level endpoint' and 'Device-level endpoint'. The 'Account-level endpoint' is selected. Below the instructions, there is a text field containing a redacted endpoint URL ending in 'amazonaws.com'. At the bottom right, there are three buttons: 'Cancel', 'Previous', and 'Next'. The 'Next' button is highlighted with a red border.

- Langkah 4 menunjukkan ikhtisar perangkat uji yang dipilih, titik akhir pengujian, rangkaian pengujian, dan peran perangkat uji yang dikonfigurasi. Untuk membuat perubahan pada bagian, pilih tombol Edit untuk bagian yang ingin Anda edit. Setelah mengonfirmasi konfigurasi pengujian, pilih Jalankan untuk membuat rangkaian pengujian dan menjalankan pengujian.

#### Note

Untuk hasil terbaik, Anda dapat menghubungkan perangkat pengujian yang dipilih ke titik akhir pengujian Device Advisor sebelum memulai rangkaian pengujian. Kami menyarankan Anda memiliki mekanisme yang dibuat untuk perangkat Anda untuk mencoba menghubungkan ke titik akhir pengujian kami setiap lima detik hingga satu hingga dua menit.

**AWS IoT** ×

Monitor

Connect

- Connect one device
- Connect many devices

Test

- Device Advisor**
  - Test suites
  - Test runs and results
  - MQTT test client
  - Device Location [New](#)

Manage

- All devices
- Greengrass devices
- LPWAN devices
- Remote actions
- Message routing
- Retained messages
- Security
- Fleet Hub

Device Software

- Billing groups
- Settings
- Feature spotlight
- Documentation [↗](#)

←

- All devices
- Greengrass devices
- LPWAN devices
- Remote actions
- Message routing
- Retained messages
- Security
- Fleet Hub

Device Software

- Billing groups
- Settings
- Feature spotlight
- Documentation [↗](#)

🔄 New console experience  
Tell us what you think

AWS IoT > Test > Device Advisor > Getting started

Step 1  
[Select a protocol](#)

### Review

Step 1: Select a protocol Edit

**Test suite type**

|                   |            |
|-------------------|------------|
| Test suite type   | Protocol   |
| Custom test suite | MQTT 3.1.1 |

Step 2: Create test suite Edit

**Test suite details**

|                           |               |                   |
|---------------------------|---------------|-------------------|
| Test suite name           | Suite version | Test type         |
| Device Advisor Demo Suite | v1            | Custom test suite |

**Start**

Starting point of this test suite.

↓

**Test group 1**

MQTT Connect

When the tests in this group are completed, testing will continue with the next group.

↓

**End**

End point of this test suite.

Step 3: Configure device settings Edit

**Device role details**

|                  |                          |
|------------------|--------------------------|
| Device           | Thing name               |
| MyThing          | MyThing                  |
| Thing ID         | Thing ARN                |
| [Redacted]       | [Redacted]               |
| Device role type | Device role name         |
| Create new role  | DeviceAdvisorServiceRole |

**Test endpoint**

[Redacted] amazonaws.com [↗](#)

Cancel Previous Run

14. Di panel navigasi di bawah Uji, pilih Device Advisor, lalu pilih Uji berjalan dan hasil. Pilih test suite run untuk melihat detail dan log run nya.

The screenshot displays the AWS IoT Device Advisor interface. On the left is a navigation sidebar with sections for Monitor, Connect, Test, and Manage. The main content area shows a breadcrumb trail: AWS IoT > Device Advisor > Test suites > Device Advisor Demo Suite > March 22, 2023, 11:20:48 (UTC-0700). A notification banner at the top prompts to connect a device. Below this, the test suite details are shown for 'March 22, 2023, 11:20:48 (UTC-0700)'. A 'Summary' table lists the device as 'MyThing', protocol as 'MQTT 3.1.1', and suite version as 'v1'. The status is 'In Progress'. A table below shows 'Test group 1 (1)' with one test, 'MQTT Connect', also in progress. At the bottom, there is a 'Tags (0)' section indicating no tags are associated with the resource.

15. Untuk mengakses CloudWatch log Amazon untuk menjalankan suite:

- Pilih log Test suite untuk melihat CloudWatch log untuk menjalankan rangkaian pengujian.
- Pilih Log kasus uji untuk kasus uji apa pun untuk melihat log khusus kasus CloudWatch uji.

16. Berdasarkan hasil pengujian Anda, [pecahkan masalah](#) perangkat Anda hingga semua pengujian lulus.

## Alur kerja Device Advisor

Tutorial ini menjelaskan cara membuat rangkaian pengujian khusus dan menjalankan pengujian terhadap perangkat yang ingin Anda uji di konsol. Setelah tes selesai, Anda dapat melihat hasil tes dan log terperinci.

### Prasyarat

Sebelum Anda memulai tutorial ini, selesaikan langkah-langkah yang diuraikan.

[Menyiapkan](#)

### Buat definisi rangkaian pengujian

Pertama, [instal AWS SDK](#).

## rootGroupsintaks

Grup root adalah string JSON yang menentukan kasus pengujian mana yang akan disertakan dalam rangkaian pengujian Anda. Ini juga menentukan konfigurasi yang diperlukan untuk kasus uji tersebut. Gunakan grup root untuk menyusun dan memesan rangkaian pengujian Anda berdasarkan kebutuhan Anda. Hirarki rangkaian pengujian adalah:

```
test suite # test group(s) # test case(s)
```

Test suite harus memiliki setidaknya satu kelompok uji, dan setiap kelompok uji harus memiliki setidaknya satu kasus uji. Device Advisor menjalankan pengujian dalam urutan di mana Anda menentukan grup pengujian dan kasus pengujian.

Setiap kelompok akar mengikuti struktur dasar ini:

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

Di grup root, Anda menentukan rangkaian pengujian dengan `name`, `configuration`, dan `tests` yang dikandung grup. `tests` Kelompok ini berisi definisi tes individu. Anda mendefinisikan setiap pengujian dengan `name`, `configuration`, dan `test` blok yang mendefinisikan kasus uji untuk pengujian itu. Akhirnya, setiap kasus uji didefinisikan dengan `id` dan `version`.

Untuk informasi tentang cara menggunakan "version" bidang "id" dan untuk setiap kasus uji (testblok), lihat [Kasus uji Device Advisor](#). Bagian itu juga berisi informasi tentang configuration pengaturan yang tersedia.

Blok berikut adalah contoh konfigurasi grup root. Konfigurasi ini menentukan kasus uji MQTT Connect Happy Case dan MQTT Connect Exponential Backoff Retries, bersama dengan deskripsi bidang konfigurasi.

```
{
 "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
 }
 }
]
 }
]
}
```

Anda harus menyediakan konfigurasi grup root saat membuat definisi rangkaian pengujian. Simpan `suiteDefinitionId` yang dikembalikan dalam objek respon. Anda dapat menggunakan ID ini untuk mengambil informasi definisi rangkaian pengujian dan menjalankan rangkaian pengujian Anda.

Berikut adalah contoh Java SDK:

```
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()
)
```

## Dapatkan definisi test suite

Setelah membuat definisi rangkaian pengujian, Anda menerima objek respons operasi `CreateSuiteDefinition` API. `suiteDefinitionId`

Ketika operasi mengembalikansuiteDefinitionId, Anda mungkin melihat id bidang baru dalam setiap grup dan definisi kasus uji dalam grup root. Anda dapat menggunakan ID ini untuk menjalankan subset definisi rangkaian pengujian Anda.

Contoh SDK Java:

```
response = iotDeviceAdvisorClient.GetSuiteDefinition(
 GetSuiteDefinitionRequest.builder()
 .suiteDefinitionId("your-suite-definition-id")
 .build()
)
```

)

## Dapatkan titik akhir pengujian

Gunakan operasi `GetEndpoint` API untuk mendapatkan titik akhir pengujian yang digunakan oleh perangkat Anda. Pilih titik akhir yang paling sesuai dengan pengujian Anda. Untuk menjalankan beberapa rangkaian pengujian secara bersamaan, gunakan titik akhir tingkat Perangkat dengan menyediakan, `certificateArn` `deviceRoleArn` Untuk menjalankan rangkaian pengujian tunggal, jangan berikan argumen ke `GetEndpoint` operasi untuk memilih titik akhir tingkat Akun.

Contoh 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())
```

## Memulai rangkaian uji coba

Setelah Anda membuat definisi rangkaian pengujian dan mengonfigurasi perangkat pengujian Anda untuk terhubung ke titik akhir pengujian Device Advisor, jalankan rangkaian pengujian Anda dengan API `StartSuiteRun`

Untuk pelanggan MQTT, gunakan salah satu `certificateArn` atau `thingArn` untuk menjalankan rangkaian pengujian. Jika keduanya dikonfigurasi, sertifikat digunakan jika itu milik benda tersebut.

Untuk MQTT over WebSocket customer, gunakan `deviceRoleArn` untuk menjalankan test suite. Jika peran yang ditentukan berbeda dari peran yang ditentukan dalam definisi rangkaian pengujian, peran yang ditentukan akan mengesampingkan peran yang ditentukan.

Untuk `parallelRun()`, gunakan `true` jika Anda menggunakan titik akhir tingkat Perangkat untuk menjalankan beberapa rangkaian pengujian secara paralel menggunakan satu. Akun AWS

Contoh 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()
```

Simpan `suiteRunId` dari respon. Anda akan menggunakan ini untuk mengambil hasil test suite run ini.

## Menjalankan test suite

Setelah Anda memulai test suite run, Anda dapat memeriksa kemajuan dan hasilnya dengan `GetSuiteRun` API.

Contoh SDK:

```
// Using the SDK, call the GetSuiteRun API.

response = iotDeviceAdvisorClient.GetSuiteRun(
 GetSuiteRunRequest.builder()
 .suiteDefinitionId("your-suite-definition-id")
 .suiteRunId("your-suite-run-id")
).build())
```

## Hentikan uji coba yang dijalankan

Untuk menghentikan rangkaian pengujian yang masih dalam proses, Anda dapat memanggil operasi `StopSuiteRun` API. Setelah Anda memanggil `StopSuiteRun` operasi, layanan memulai proses pembersihan. Saat layanan menjalankan proses pembersihan, rangkaian pengujian menjalankan pembaruan status ke `Stopping`. Proses pembersihan bisa memakan waktu beberapa menit. Setelah proses selesai, rangkaian pengujian menjalankan pembaruan status ke `Stopped`. Setelah test run benar-benar berhenti, Anda n memulai test suite run lainnya. Anda dapat memeriksa status suite run secara berkala menggunakan operasi `GetSuiteRun` API, seperti yang ditunjukkan pada bagian sebelumnya.

## Contoh SDK:

```
// Using the SDK, call the StopSuiteRun API.

response = iotDeviceAdvisorClient.StopSuiteRun(
 StopSuiteRun.builder()
 .suiteDefinitionId("your-suite-definition-id")
 .suiteRunId("your-suite-run-id")
 .build())
```

## Dapatkan laporan kualifikasi untuk menjalankan rangkaian tes kualifikasi yang sukses

Jika Anda menjalankan rangkaian pengujian kualifikasi yang berhasil diselesaikan, Anda dapat mengambil laporan kualifikasi dengan operasi `GetSuiteRunReport` API. Anda menggunakan laporan kualifikasi ini untuk memenuhi syarat perangkat Anda dengan program AWS IoT Core kualifikasi. Untuk menentukan apakah rangkaian pengujian Anda adalah rangkaian pengujian kualifikasi, periksa apakah `intendedForQualification` parameter disetel ke `true`. Setelah Anda memanggil operasi `GetSuiteRunReport` API, Anda dapat mengunduh laporan dari URL yang dikembalikan hingga 90 detik. Jika lebih dari 90 detik berlalu dari waktu sebelumnya Anda memanggil `GetSuiteRunReport` operasi, panggil operasi lagi untuk mengambil URL baru yang valid.

## Contoh SDK:

```
// Using the SDK, call the getSuiteRunReport API.

response = iotDeviceAdvisorClient.getSuiteRunReport(
 GetSuiteRunReportRequest.builder()
 .suiteDefinitionId("your-suite-definition-id")
 .suiteRunId("your-suite-run-id")
 .build()
)
```

## Alur kerja konsol terperinci Device Advisor

Dalam tutorial ini, Anda akan membuat rangkaian pengujian khusus dan menjalankan pengujian terhadap perangkat yang ingin Anda uji di konsol. Setelah tes selesai, Anda dapat melihat hasil tes dan log terperinci.

## Tutorial

- [Prasyarat](#)
- [Buat definisi rangkaian pengujian](#)
- [Memulai rangkaian uji coba](#)
- [Hentikan uji coba yang dijalankan \(opsional\)](#)
- [Lihat detail dan log test suite run](#)
- [Unduh laporan AWS IoT kualifikasi](#)

## Prasyarat

Untuk menyelesaikan tutorial ini, Anda perlu [membuat sesuatu dan sertifikat](#).

## Buat definisi rangkaian pengujian

1. Di [AWS IoT konsol](#), di panel navigasi, perluas Test, Device Advisor, lalu pilih Test suites.

The screenshot shows the AWS IoT console interface. On the left, the navigation sidebar is visible with 'Test suites' highlighted under the 'Device Advisor' section. The main content area displays three options for creating test suites: 'AWS IoT Core qualification test suite', 'Long duration test suite', and 'Custom test suite'. Below this, there is a table for 'Test suites (0)' with columns for Name, Test Type, Protocol, and Date created. The table is currently empty, and a 'Create test suite' button is visible at the bottom.

Pilih Create Test Suite.

2. Pilih salah satu Use the AWS Qualification test suite atau Create a new test suite.

Untuk protokol, pilih MQTT 3.1.1 atau MQTT 5.

The screenshot shows the 'Create test suite' wizard in the AWS IoT Core console. The left sidebar contains navigation menus for 'Monitor', 'Connect', 'Test', 'Manage', 'Device Software', 'Billing groups', 'Settings', 'Feature spotlight', and 'Documentation'. The main content area is titled 'Create test suite' and includes a progress indicator with four steps: Step 1 (Create test suite), Step 2 (Configure test suite), Step 3 (Select a device role), and Step 4 (Review). The 'Choose test suite type' section offers three options: 'AWS IoT Core qualification test suite' (selected), 'Long duration test suite', and 'Custom test suite'. The 'Protocol' section offers 'MQTT 3.1.1' (selected) and 'MQTT 5'. 'Cancel' and 'Next' buttons are located at the bottom right of the main content area.

Pilih Use the AWS Qualification test suite untuk memenuhi syarat dan daftar perangkat Anda ke Katalog Perangkat AWS Mitra. Dengan memilih opsi ini, kasus uji yang diperlukan untuk kualifikasi perangkat Anda ke program AWS IoT Core kualifikasi telah dipilih sebelumnya. Kelompok uji dan kasus uji tidak dapat ditambahkan atau dihapus. Anda masih perlu mengkonfigurasi properti test suite.

Pilih Create a new test suite untuk membuat dan mengkonfigurasi rangkaian pengujian kustom. Sebaiknya mulai dengan opsi ini untuk pengujian awal dan pemecahan masalah. Rangkaian pengujian khusus harus memiliki setidaknya satu kelompok uji, dan setiap kelompok uji harus memiliki setidaknya satu kasus uji. Untuk tujuan tutorial ini, kita akan memilih opsi ini dan memilih Berikutnya.

**AWS IoT** ×

AWS IoT > Test > Device Advisor > Create test suite

Step 1  
Create test suite

Step 2  
**Configure test suite**

Step 3  
Select a device role

Step 4  
Review

### Configure test suite

A test suite contains test groups, which contain test cases. You must have one test group with one test case in your test suite. Drag and drop to add, arrange, or delete test cases from your test suite. Test suites, groups and cases can be configured individually.

**Test suite December 22, 2022, 11:24:37 (UTC-0800)**  
Test suite name

Test suite properties

**Test cases**

Test cases are the individual prebuilt test that are configured to test with things

Show all test cases

- MQTT (14)
  - MQTT Connect
  - MQTT Connect Jitter Retries
  - MQTT Connect Exponential Backoff Retries
  - MQTT Reconnect Backoff Retries On Server Disconnect
  - MQTT Reconnect Backoff Retries On Unstable Connection

**Start**

Starting point of this test suite.

All test groups and test cases will execute in descending order from this point. Even if an error occurs on a test case.

+ Add test group

**Test group 1**  
Test group

No test cases have been added to this test group.

When the tests in this group are completed, testing will continue with the next group.

**Configure**

Select a test group or test case to configure it.

### 3. Pilih properti Test suite. Anda harus membuat properti test suite saat membuat test suite.

**AWS IoT** ×

AWS IoT > Test > Device Advisor > Create test suite

Step 1  
Create test suite

Step 2  
**Configure test suite**

Step 3  
Select a device role

Step 4  
Review

### Configure test suite

A test suite contains test groups, which contain test cases. You must have one test group with one test case in your test suite. Drag and drop to add, arrange, or delete test cases from your test suite. Test suites, groups and cases can be configured individually.

**Test suite December 22, 2022, 11:24:37 (UTC-0800)**  
Test suite name

Test suite properties

**Test cases**

Test cases are the individual prebuilt test that are configured to test with things

Show all test cases

- MQTT (14)
  - MQTT Connect
  - MQTT Connect Jitter Retries
  - MQTT Connect Exponential Backoff Retries
  - MQTT Reconnect Backoff Retries On Server Disconnect
  - MQTT Reconnect Backoff Retries On Unstable Connection

**Start**

Starting point of this test suite.

All test groups and test cases will execute in descending order from this point. Even if an error occurs on a test case.

+ Add test group

**Test group 1**  
Test group

No test cases have been added to this test group.

When the tests in this group are completed, testing will continue with the next group.

**Configure**

Select a test group or test case to configure it.

Di bawah properti Test suite, isi yang berikut ini:

- Nama suite uji: Anda dapat membuat suite dengan nama khusus.

- Batas waktu (opsional): Batas waktu dalam hitungan detik untuk setiap kasus uji dalam rangkaian pengujian saat ini. Jika Anda tidak menentukan nilai batas waktu, nilai default akan digunakan.
- Tag (opsional): Tambahkan tag ke rangkaian pengujian.

Test suite properties

Test suite name  
Specify a name for this test suite that you can search.

Device Advisor demo suite

Timeout - *optional*  
Optional, in seconds. Maximum time Device Advisor waits for a device to respond before tests fail.

Key Value - *optional*

Enter key Enter value Remove

Custom tag key  
Add new tag

You can add up to 49 more tags.

Cancel Update properties

Setelah selesai, pilih Perbarui properti.

4. Untuk mengubah konfigurasi tingkat grup, di bawah `Test group 1`, pilih Edit. Kemudian, masukkan Nama untuk memberi grup nama khusus.

Secara opsional, Anda juga dapat memasukkan nilai Timeout dalam hitungan detik di bawah grup pengujian yang dipilih. Jika Anda tidak menentukan nilai batas waktu, nilai default akan digunakan.

**Configure test suite**

A test suite contains test groups, which contain test cases. You must have one test group with one test case in your test suite. Drag and drop to add, arrange, or delete test cases from your test suite. Test suites, groups and cases can be configured individually.

**Device Advisor demo suite**  
Test suite name

**Test cases**  
Test cases are the individual prebuilt test that are configured to test with things.

Show all test cases

MQTT (14)

- MQTT Connect
- MQTT Connect Jitter Retries
- MQTT Connect Exponential Backoff Retries
- MQTT Reconnect Backoff Retries On

**Test group 1**  
Test group

Start

Starting point of this test suite.

All test groups and test cases will execute in descending order from this point. Even if an error occurs on a test case.

+ Add test group

Test group 1

No test cases have been added to this test group.

**Configure**

Test group 1

Name  
Specify a name for this test group.

Test group 1

Timeout - optional  
Optional, in seconds. Maximum time Device Advisor waits for a device to respond before tests fail.

value

Done

Cancel Delete

Pilih Selesai.

## 5. Seret salah satu kasus uji yang tersedia dari Kasus uji ke dalam grup uji.

**Configure test suite**

A test suite contains test groups, which contain test cases. You must have one test group with one test case in your test suite. Drag and drop to add, arrange, or delete test cases from your test suite. Test suites, groups and cases can be configured individually.

**Device Advisor demo suite**  
Test suite name

**Test cases**  
Test cases are the individual prebuilt test that are configured to test with things.

Show all test cases

MQTT (14)

- MQTT Connect
- MQTT Connect Jitter Retries
- MQTT Connect Exponential Backoff Retries
- MQTT Reconnect Backoff Retries On

**Test group 1**  
Test group

Start

Starting point of this test suite.

All test groups and test cases will execute in descending order from this point. Even if an error occurs on a test case.

+ Add test group

Test group 1

MQTT Connect

**Configure**

Select a test group or test case to configure it.

## 6. Untuk mengubah konfigurasi tingkat kasus uji untuk kasus uji yang ditambahkan ke grup pengujian, pilih Edit. Kemudian, masukkan Nama untuk memberi grup nama khusus.

Secara opsional, Anda juga dapat memasukkan nilai Timeout dalam hitungan detik di bawah grup pengujian yang dipilih. Jika Anda tidak menentukan nilai batas waktu, nilai default akan digunakan.

The screenshot displays the 'Configure test suite' interface in AWS IoT Core. It includes a sidebar with navigation steps, a main configuration area for a 'Device Advisor demo suite', and a 'Configure' dialog for a specific test case. The dialog shows the test case name as 'MQTT Connect' and a 'Timeout - optional' field with a 'value' placeholder.

Pilih Selesai.

#### Note

Untuk menambahkan lebih banyak grup pengujian ke rangkaian pengujian, pilih Tambahkan grup pengujian. Ikuti langkah-langkah sebelumnya untuk membuat dan mengonfigurasi lebih banyak grup pengujian, atau untuk menambahkan lebih banyak kasus uji ke satu atau beberapa grup pengujian. Kelompok uji dan kasus uji dapat disusun ulang dengan memilih dan menyeret kasus uji ke posisi yang diinginkan. Device Advisor menjalankan pengujian dalam urutan di mana Anda menentukan grup pengujian dan kasus pengujian.

7. Pilih Berikutnya.
8. Pada Langkah 3, konfigurasi peran perangkat yang akan digunakan Device Advisor untuk melakukan tindakan AWS IoT MQTT atas nama perangkat pengujian Anda.

Jika Anda memilih kasus uji MQTT Connect hanya di Langkah 2, tindakan Connect akan dicentang secara otomatis karena izin tersebut diperlukan pada peran perangkat untuk menjalankan rangkaian pengujian ini. Jika Anda memilih kasus uji lainnya, tindakan yang diperlukan terkait akan diperiksa. Pastikan bahwa nilai nilai sumber daya untuk setiap tindakan disediakan. Misalnya, untuk tindakan Connect, berikan id klien yang akan terhubung dengan perangkat Anda ke titik akhir Device Advisor. Anda dapat memberikan beberapa nilai dengan menggunakan koma untuk memisahkan nilai, dan Anda dapat memberikan nilai



awalan menggunakan karakter wildcard (\*) juga. Misalnya, untuk memberikan izin untuk mempublikasikan topik apa pun yang dimulai MyTopic, Anda dapat memberikan "MyTopic\*" sebagai nilai sumber daya.

The screenshot shows the 'Select a device role' step in the AWS IoT Core console. The breadcrumb trail is 'AWS IoT > Test > Device Advisor > Create test suite'. The left sidebar shows the progress: Step 1 (Create test suite), Step 2 (Configure test suite), Step 3 (Select a device role), and Step 4 (Review). The main content area is titled 'Select a device role' and includes a 'Device role info' section with a note: 'AWS IoT Core Device Advisor requires permission to perform AWS IoT MQTT actions on behalf of your test device.' There are two radio buttons: 'Create new role' (selected) and 'Select an existing role'. Below this is a text input for 'Role name' containing 'MyDeviceAdvisorDeviceRole'. The 'Permissions' section has a note: 'Choose which actions and the associated resources for AWS IoT Core Device Advisor to access using this role. You can enter a specific resource or resource prefix. To enter multiple values for a resource, use commas to separate the values. Learn more'. A table lists actions and resources:

| Action                                      | Resource type | Resource                                                                |
|---------------------------------------------|---------------|-------------------------------------------------------------------------|
| <input checked="" type="checkbox"/> Connect | Clientid      | MyClient                                                                |
| <input type="checkbox"/> Publish            | Topic         | Specify topics to publish to, e.g. MyTopic, MyTopic*                    |
| <input type="checkbox"/> Subscribe          | TopicFilter   | Specify topic filters to subscribe to, e.g. MyTopic, MyTopic*           |
| <input type="checkbox"/> Receive            | Topic         | Specify topics to receive from e.g. MyTopic, MyTopic*                   |
| <input type="checkbox"/> RetainPublish      | Topic         | Specify topics to publish a retained message to, e.g. MyTopic, MyTopic* |

At the bottom right, there are 'Cancel', 'Previous', and 'Next' buttons.

Jika Anda telah membuat peran perangkat sebelumnya dan ingin menggunakan peran itu, pilih Pilih peran yang ada dan pilih peran perangkat Anda di bawah Pilih peran.

The screenshot shows the 'Select a device role' step in the AWS IoT Core console. The breadcrumb trail is 'AWS IoT > Test > Device Advisor > Create test suite'. The left sidebar shows the progress: Step 1 (Create test suite), Step 2 (Configure test suite), Step 3 (Select a device role), and Step 4 (Review). The main content area is titled 'Select a device role' and includes a 'Device role info' section with a note: 'AWS IoT Core Device Advisor requires permission to perform AWS IoT MQTT actions on behalf of your test device.' There are two radio buttons: 'Create new role' and 'Select an existing role' (selected). Below this is a dropdown menu labeled 'Select role' with the placeholder text 'Select a device role'. At the bottom right, there are 'Cancel', 'Previous', and 'Next' buttons.

Konfigurasi peran perangkat Anda menggunakan salah satu dari dua opsi yang disediakan dan pilih Berikutnya.

9. Pada Langkah 4, pastikan konfigurasi yang disediakan di setiap langkah akurat. Untuk mengedit konfigurasi yang disediakan untuk langkah tertentu, pilih Edit untuk langkah yang sesuai.

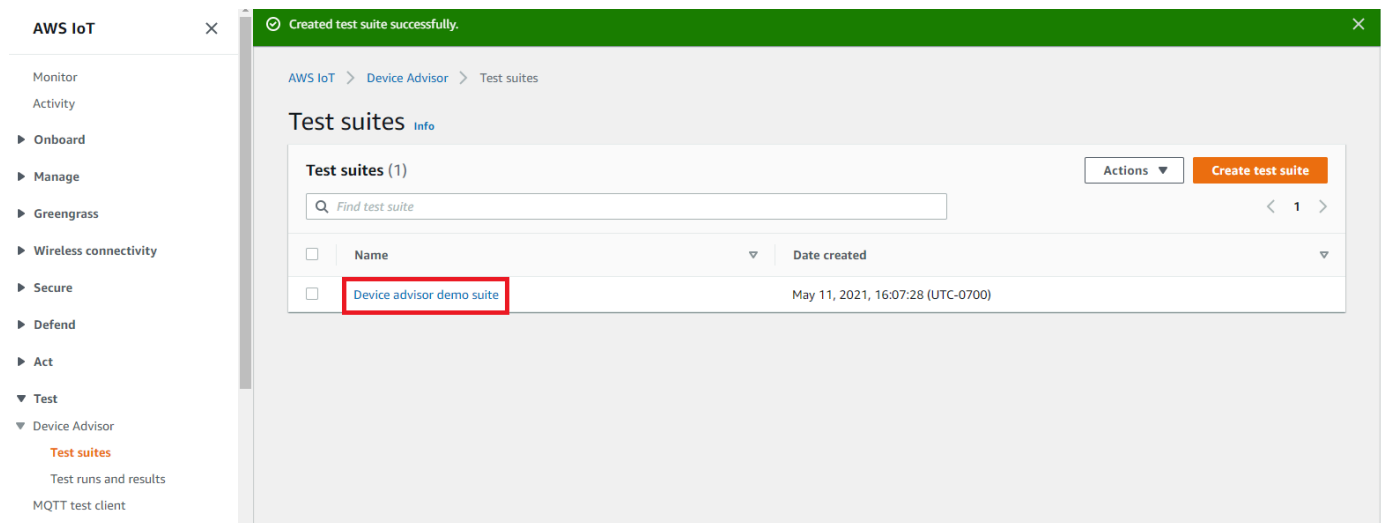
Setelah Anda memverifikasi konfigurasi, pilih Create test suite.

Rangkaian pengujian harus berhasil dibuat dan Anda akan diarahkan ke halaman suite Uji tempat Anda dapat melihat semua rangkaian pengujian yang telah dibuat.

Jika pembuatan rangkaian pengujian gagal, pastikan rangkaian pengujian, grup pengujian, kasus pengujian, dan peran perangkat telah dikonfigurasi sesuai dengan instruksi sebelumnya.

## Memulai rangkaian uji coba

1. Di [AWS IoT konsol](#), di panel navigasi, perluas Test, Device Advisor, lalu pilih Test suites.
2. Pilih rangkaian pengujian yang ingin Anda lihat detail rangkaian pengujianya.



Halaman detail rangkaian pengujian menampilkan semua informasi yang terkait dengan rangkaian pengujian.

3. Pilih Tindakan, lalu Jalankan rangkaian pengujian.

AWS IoT > Device Advisor > Test suites > Device Advisor demo suite

**Device Advisor demo suite**

Actions ▲

- Run test suite
- Edit
- Delete

**Test suite details**

|                     |                                                   |                                |
|---------------------|---------------------------------------------------|--------------------------------|
| Suite version<br>v1 | Created<br>November 05, 2021, 13:28:08 (UTC-0400) | Test type<br>Custom test suite |
|---------------------|---------------------------------------------------|--------------------------------|

**Activity Log**

< 1 >

| Timestamp                | Test suite version | Status | Passed | Failed | Duration |
|--------------------------|--------------------|--------|--------|--------|----------|
| No test suite activities |                    |        |        |        |          |

▼ **Test suite summary**  
A summary of the tests to be run in the test suite, organized by groups.

Start

4. Di bawah konfigurasi Jalankan, Anda harus memilih AWS IoT sesuatu atau sertifikat untuk diuji menggunakan Device Advisor. Jika Anda tidak memiliki barang atau sertifikat yang ada, pertama-tama [buat AWS IoT Core sumber daya](#).

Di bagian titik akhir Uji, pilih titik akhir yang paling sesuai dengan kasus Anda. Jika Anda berencana untuk menjalankan beberapa rangkaian pengujian secara bersamaan menggunakan AWS akun yang sama di masa mendatang, pilih Titik akhir tingkat perangkat. Jika tidak, jika Anda berencana untuk hanya menjalankan satu rangkaian pengujian pada satu waktu, pilih Titik akhir tingkat Akun.

Konfigurasi perangkat pengujian Anda dengan titik akhir pengujian Device Advisor yang dipilih.

Setelah Anda memilih sesuatu atau sertifikat dan memilih titik akhir Device Advisor, pilih Jalankan pengujian.

**Run configuration**

**Select test devices**

Select the IoT thing/certificate to test using the test suite. If not listed below, you must first create a thing/certificate registered with IoT Core before you can run the test suite.

Things Choose a thing for this test suite. To create a new thing, go to [IoT Things](#).

Certificates Choose a certificate for this test suite. To create a new certificate, go to [IoT Certificates](#).

**Things (1)**

Filter things

| Name    | Type |
|---------|------|
| MyThing |      |

**Test endpoint**

Choose the endpoint that best fits your situation. If you want to simultaneously run multiple test suites then use 'Device-level endpoint'; if you want to run only one test suite at a time then choose the 'Account-level endpoint'.

Account-level endpoint Using this endpoint, you can only run one test suite at a time.

Device-level endpoint Using this endpoint, you can run multiple test suites simultaneously.

Copy and paste this endpoint to your test device.  
 t86dcb4139e4919y9tzu6gamma.us-west-2.advisor.iot.aws.dev

**Tags - optional**

A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

No tags associated with the resource.

You can add up to 50 more tags.

## 5. Pilih Buka hasil di spanduk atas untuk melihat detail uji coba.

**'Device Advisor demo suite' is in progress with 'MyThing'.**

[AWS IoT](#) > [Device Advisor](#) > [Test suites](#) > Device Advisor demo suite

**Device Advisor demo suite**

**Test suite details**

|                     |                                                   |                                |
|---------------------|---------------------------------------------------|--------------------------------|
| Suite version<br>v1 | Created<br>November 05, 2021, 13:40:33 (UTC-0400) | Test type<br>Custom test suite |
|---------------------|---------------------------------------------------|--------------------------------|

**Activity Log**

| Timestamp                              | Test suite version | Status               | Passed | Failed | Duration |
|----------------------------------------|--------------------|----------------------|--------|--------|----------|
| November 05, 2021, 13:53:23 (UTC-0400) | v1                 | <span>Pending</span> | -      | -      | -        |

## Hentikan uji coba yang dijalankan (opsional)

1. Di [AWS IoT konsol](#), di panel navigasi, perluas Uji, Penasihat Perangkat, lalu pilih Uji berjalan dan hasil.
2. Pilih rangkaian pengujian yang sedang berlangsung yang ingin Anda hentikan.

**Test runs and results**

**Summary**

|                                                                         |                                                                                     |                                                                         |
|-------------------------------------------------------------------------|-------------------------------------------------------------------------------------|-------------------------------------------------------------------------|
| Number of IoT things available<br>1<br><a href="#">Go to IoT things</a> | Number of IoT certificates available<br>6<br><a href="#">Go to IoT certificates</a> | Number of test suites running<br>1<br><a href="#">Go to test suites</a> |
|-------------------------------------------------------------------------|-------------------------------------------------------------------------------------|-------------------------------------------------------------------------|

**Results of test runs (in progress and completed)**

| Name                      | Timestamp                              | Test suite version | Status      | Passed | Failed | Duration |
|---------------------------|----------------------------------------|--------------------|-------------|--------|--------|----------|
| Device Advisor demo suite | December 07, 2020, 11:16:46 (UTC-0800) | v1                 | In Progress | -      | -      | -        |

### 3. Pilih Actions, lalu Stop test suite.

**Test suites**

Connect your device now  
Connect your device to the Device Advisor test endpoint - xxxxxxxxx.deviceadvisor.iot.us-east-1.amazonaws.com now to validate your device for MQTT Connect. For more information, refer to [Configure your test device](#).

May 11, 2021, 16:15:43 (UTC-0700)

Activity log details

|                   |                     |                                              |                       |
|-------------------|---------------------|----------------------------------------------|-----------------------|
| Device<br>MyThing | Suite version<br>v1 | Created<br>May 11, 2021, 16:15:43 (UTC-0700) | Status<br>In Progress |
|-------------------|---------------------|----------------------------------------------|-----------------------|

Test group 1 (1) In Progress

| Test         | Result      | System message | Logs |
|--------------|-------------|----------------|------|
| MQTT Connect | In Progress |                |      |

Tags - optional  
A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

No tags associated with the resource.

[Add new tag](#)  
You can add up to 50 more tags.

Actions

- Run test suite
- Stop test suite

4. Proses pembersihan akan memakan waktu beberapa menit untuk diselesaikan. Sementara proses pembersihan berjalan, status uji coba akan STOPPING. Tunggu hingga proses pembersihan selesai dan status rangkaian pengujian berubah menjadi STOPPED status sebelum memulai rangkaian baru.

AWS IoT > Device Advisor > Test suites > Device advisor demo suite > May 11, 2021, 16:15:43 (UTC-0700)

May 11, 2021, 16:15:43 (UTC-0700) [Test suite log](#) [Actions](#)

### Activity log details

|         |               |                                   |         |
|---------|---------------|-----------------------------------|---------|
| Device  | Suite version | Created                           | Status  |
| MyThing | v1            | May 11, 2021, 16:15:43 (UTC-0700) | Stopped |

▼ Test group 1 (1) Stopped

| Test         | Result  | System message  | Logs                          |
|--------------|---------|-----------------|-------------------------------|
| MQTT Connect | Stopped | No issues found | <a href="#">Test case log</a> |

### Tags - optional

A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

No tags associated with the resource.

[Add new tag](#)

You can add up to 50 more tags.

[Cancel](#) [Save changes](#)

## Lihat detail dan log test suite run

1. Di [AWS IoT konsol](#), di panel navigasi, perluas Uji, Penasihat Perangkat, lalu pilih Uji berjalan dan hasil.

Halaman ini menampilkan:

- Jumlah hal IoT
  - Jumlah sertifikat IoT
  - Jumlah rangkaian pengujian yang sedang berjalan
  - Semua rangkaian pengujian berjalan yang telah dibuat
2. Pilih rangkaian pengujian yang ingin Anda lihat detail dan log run.

The screenshot shows the AWS IoT Core console interface for 'Test runs and results'. The left sidebar contains navigation options like Monitor, Activity, Onboard, Manage, Greenpress, Secure, Defend, Act, Test, and Device Advisor. The main content area shows a summary with three metrics: Number of IoT things available (1), Number of IoT certificates available (6), and Number of test suites running (1). Below this is a table titled 'Results of test runs (in progress and completed)'. The table has columns for Name, Timestamp, Test suite version, Status, Passed, Failed, and Duration. One row is visible: 'Device Advisor demo suite' with a timestamp of 'December 07, 2020, 11:16:46 (UTC-0800)', version 'v1', and status 'In Progress'. The 'Device Advisor demo suite' text in the table is highlighted with a red box.

Halaman ringkasan run menampilkan status rangkaian pengujian yang dijalankan saat ini. Halaman ini otomatis menyegarkan setiap 10 detik. Kami menyarankan Anda memiliki mekanisme yang dibuat untuk perangkat Anda untuk mencoba menghubungkan ke titik akhir pengujian kami setiap lima detik selama satu hingga dua menit. Kemudian Anda dapat menjalankan beberapa kasus uji secara berurutan secara otomatis.

The screenshot shows the 'Activity log details' page in the AWS IoT Core console. The breadcrumb trail is 'AWS IoT > Device Advisor > Test suites > Device advisor demo suite > December 07, 2020, 17:05:38 (UTC-0800)'. The page title is 'December 07, 2020, 17:05:38 (UTC-0800)'. Below the title, there are buttons for 'Test suite log' and 'Actions'. The main content area is titled 'Activity log details' and shows a table with columns for Device, Suite version, Created, and Status. The device is 'MyThing', the suite version is 'v1', and the status is 'Passed'. Below this is a section for 'Test group 1 (1)' with a 'Passed' status. A table below shows the test results for 'MQTT Connect', with a 'Passed' result, 'No issues found' system message, and a link to 'Test case log'. At the bottom, there is a 'Tags - optional' section with an 'Add new tag' button and a note that up to 50 tags can be added.

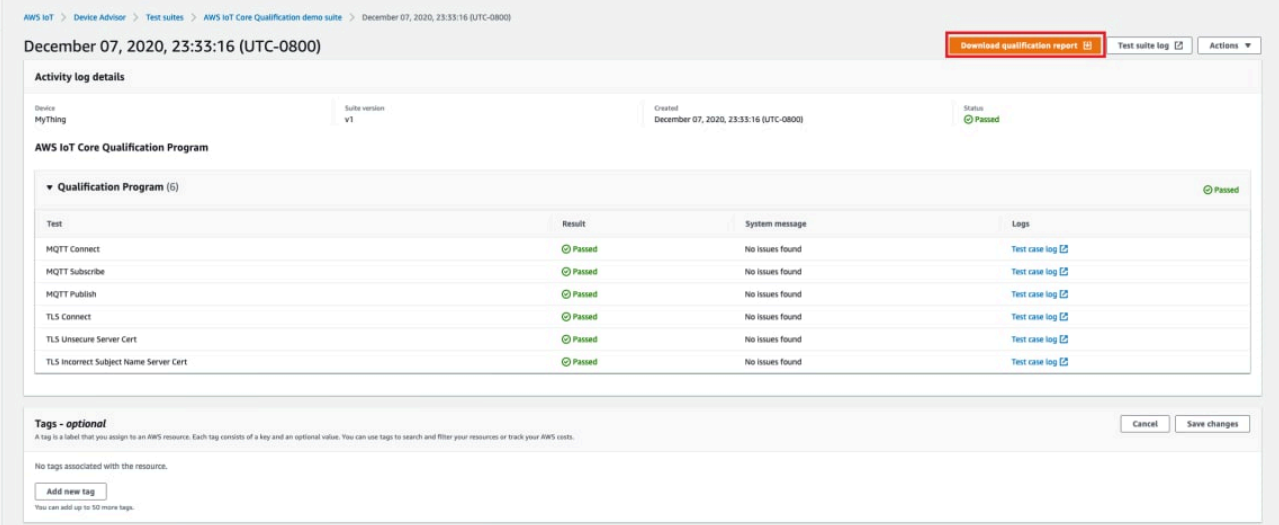
3. Untuk mengakses CloudWatch log untuk menjalankan rangkaian pengujian, pilih Log rangkaian pengujian.

Untuk mengakses CloudWatch log untuk kasus uji apa pun, pilih Log kasus uji.

4. Berdasarkan hasil pengujian Anda, [pecahkan masalah](#) perangkat Anda hingga semua pengujian lulus.

## Unduh laporan AWS IoT kualifikasi

Jika Anda memilih opsi Gunakan rangkaian pengujian AWS IoT Kualifikasi saat membuat rangkaian pengujian dan dapat menjalankan rangkaian tes kualifikasi, Anda dapat mengunduh laporan kualifikasi dengan memilih Unduh laporan kualifikasi di halaman ringkasan uji coba.



The screenshot shows the AWS IoT console interface for a qualification test. The main content area displays the following information:

- Activity log details:** Device: MyThing, Suite version: v1, Created: December 07, 2020, 23:33:16 (UTC-0800), Status: Passed.
- AWS IoT Core Qualification Program:** Qualification Program (6) - Passed.
- Test Results Table:**

| Test                                   | Result | System message  | Logs                          |
|----------------------------------------|--------|-----------------|-------------------------------|
| MQTT Connect                           | Passed | No issues found | <a href="#">Text case log</a> |
| MQTT Subscribe                         | Passed | No issues found | <a href="#">Text case log</a> |
| MQTT Publish                           | Passed | No issues found | <a href="#">Text case log</a> |
| TLS Connect                            | Passed | No issues found | <a href="#">Text case log</a> |
| TLS Unsecure Server Cert               | Passed | No issues found | <a href="#">Text case log</a> |
| TLS Incorrect Subject Name Server Cert | Passed | No issues found | <a href="#">Text case log</a> |

At the bottom of the console, there is a 'Tags - optional' section with an 'Add new tag' button and a note: 'No tags associated with the resource. You can add up to 50 more tags.'

## Durasi panjang menguji alur kerja konsol

Tutorial ini membantu Anda memulai dengan tes durasi panjang pada Device Advisor menggunakan konsol. Untuk menyelesaikan tutorial, ikuti langkah-langkah di [Menyiapkan](#).

1. Di panel navigasi [AWS IoT konsol](#), perluas Test, lalu Device Advisor, lalu Test suite. Pada halaman, pilih Buat rangkaian pengujian durasi panjang.



2. Pada halaman Create test suite, pilih Long duration test suite dan pilih Next.

Untuk protokol, pilih MQTT 3.1.1 atau MQTT 5.

3. Lakukan hal berikut pada halaman Configure test suite:

- a. Perbarui bidang Test suite name.
- b. Perbarui bidang nama grup Uji.

- c. Pilih Operasi perangkat yang dapat dilakukan perangkat. Ini akan memilih tes yang akan dijalankan.
- d. Pilih opsi Pengaturan.

4. (Opsional) Masukkan jumlah waktu maksimum Device Advisor harus menunggu pengujian dasar selesai. Pilih Simpan.

5. Lakukan hal berikut di bagian Tes lanjutan dan Pengaturan tambahan.

- Pilih atau batalkan pilihan Tes lanjutan yang ingin Anda jalankan sebagai bagian dari pengujian ini.
- Edit konfigurasi untuk pengujian bila berlaku.
- Konfigurasi waktu eksekusi tambahan di bawah bagian Pengaturan tambahan.
- Pilih Berikutnya untuk melakukan langkah selanjutnya.

The screenshot shows the AWS IoT Core Device Advisor console. The left sidebar contains navigation options like Monitor, Connect, Test, Manage, and Device Software. The main content area is titled 'Basic tests' and 'Advanced tests'. The 'Advanced tests' section contains a table with the following data:

| Test case                           | Description                        | Configure |
|-------------------------------------|------------------------------------|-----------|
| <input checked="" type="checkbox"/> | Return PUBACK on Qos1 subscription | -         |
| <input checked="" type="checkbox"/> | Receive large payload              | Edit      |
| <input checked="" type="checkbox"/> | Persistent session                 | -         |
| <input checked="" type="checkbox"/> | Keep Alive                         | -         |
| <input checked="" type="checkbox"/> | Intermittent connectivity          | -         |
| <input checked="" type="checkbox"/> | Reconnect backoff                  | Edit      |
| <input checked="" type="checkbox"/> | Long server disconnect             | Edit      |

Below the table is the 'Additional settings' section, which includes a text input field for 'Additional execution time - Optional'. At the bottom right, there are 'Cancel', 'Previous', and 'Next' buttons.

- Pada langkah ini, Buat peran baru atau Pilih peran yang ada. Lihat [Membuat peran IAM untuk digunakan sebagai peran perangkat](#) untuk detail.

**AWS IoT** ×

AWS IoT > Test > Device Advisor > Create test suite

Step 1  
Create test suite

Step 2  
Configure test suite

Step 3  
**Select a device role**

Step 4  
Review

### Select a device role

**Device role** Info  
AWS IoT Core Device Advisor requires permission to perform AWS IoT MQTT actions on behalf of your test device.

Create new role  
Create and use a new device role

Select an existing role  
Use an existing device role

Role name  
DeviceAdvisorServiceRole-lhqPgx83

**Permissions**  
Choose which actions and the associated resources for AWS IoT Core Device Advisor to access using this role. You can enter a specific resource or resource prefix. To enter multiple values for a resource, use commas to separate the values. [Learn more](#)

| Action                                        | Resource type | Resource                                              |
|-----------------------------------------------|---------------|-------------------------------------------------------|
| <input checked="" type="checkbox"/> Connect   | Clientid      | myClientid                                            |
| <input checked="" type="checkbox"/> Publish   | Topic         | MyTopic                                               |
| <input checked="" type="checkbox"/> Subscribe | TopicFilter   | MyTopic                                               |
| <input type="checkbox"/> Receive              | Topic         | Specify topics to receive from e.g. MyTopic, MyTopic* |

Cancel Previous **Next**

7. Tinjau semua konfigurasi yang dibuat hingga langkah ini dan pilih Create test suite.

**AWS IoT** ×

AWS IoT > Test > Device Advisor > Create test suite

Step 1  
Create test suite

Step 2  
Configure test suite

Step 3  
Select a device role

Step 4  
**Review**

### Review

Step 1: Test suite type Edit

**Test suite type details**

|                                  |                        |
|----------------------------------|------------------------|
| Test suite type<br>Long duration | Protocol<br>MQTT 3.1.1 |
|----------------------------------|------------------------|

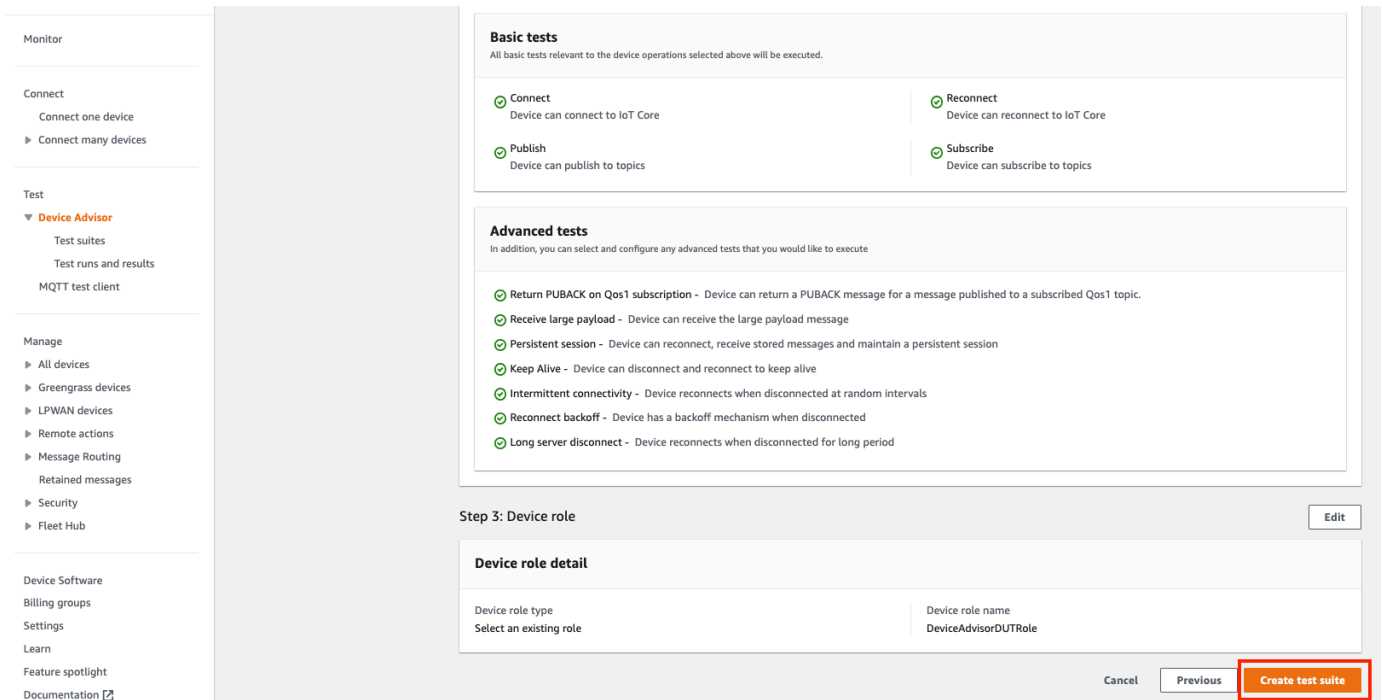
Step 2: Test suite Edit

**Test suite details**

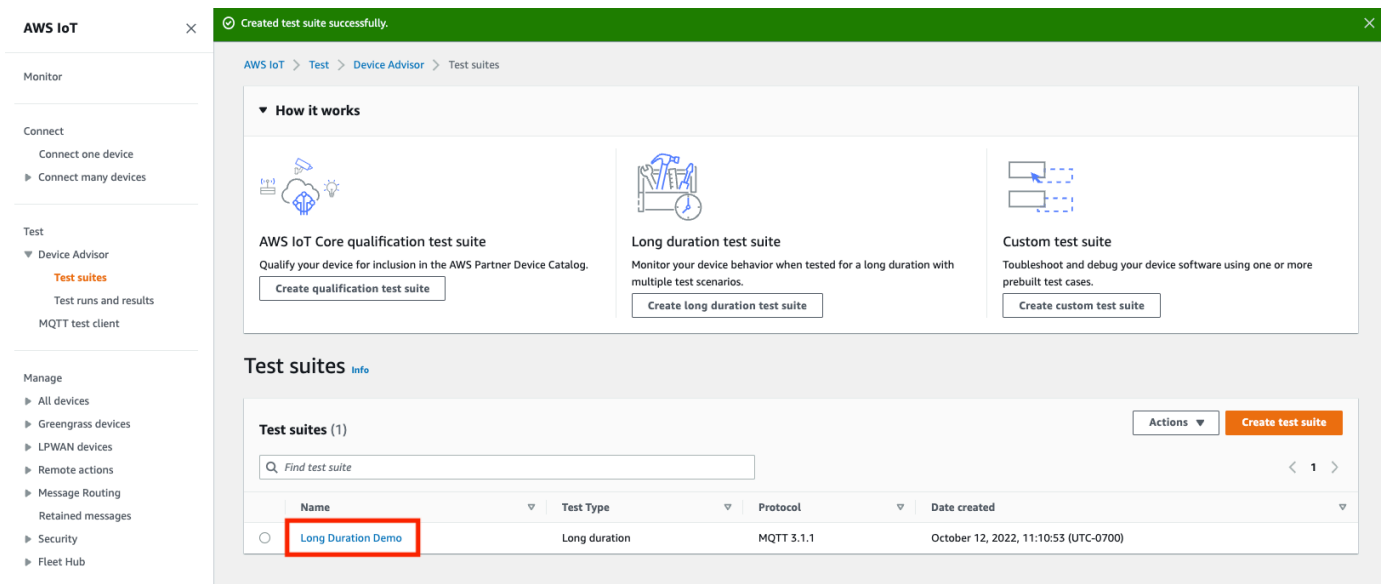
|                                       |                                    |
|---------------------------------------|------------------------------------|
| Test suite name<br>Long Duration Demo | Test group name<br>MQTT Test Group |
|---------------------------------------|------------------------------------|

Device operations  
CONNECT, PUBLISH, SUBSCRIBE

**Basic tests**



8. Rangkaian pengujian yang dibuat berada di bawah bagian Test suites. Pilih suite untuk melihat detail.



9. Untuk menjalankan rangkaian pengujian yang dibuat, pilih Tindakan lalu Jalankan rangkaian pengujian.

**Long Duration Demo**

**Test suite details**

Suite definition ARN  
arn:aws:iotdeviceadvisor:ap-northeast-1:507237901444:suitedefinition/jl17u6vutzki

Suite version  
v1

Created  
October 12, 2022, 11:10:53 (UTC-0700)

Test type  
Long duration

**Activity Log**

| Timestamp                | Test suite version | Status | Passed | Failed | Duration |
|--------------------------|--------------------|--------|--------|--------|----------|
| No test suite activities |                    |        |        |        |          |

**Test suite summary**  
A summary of the tests to be run in the test suite, organized by groups.

**Test suite details**

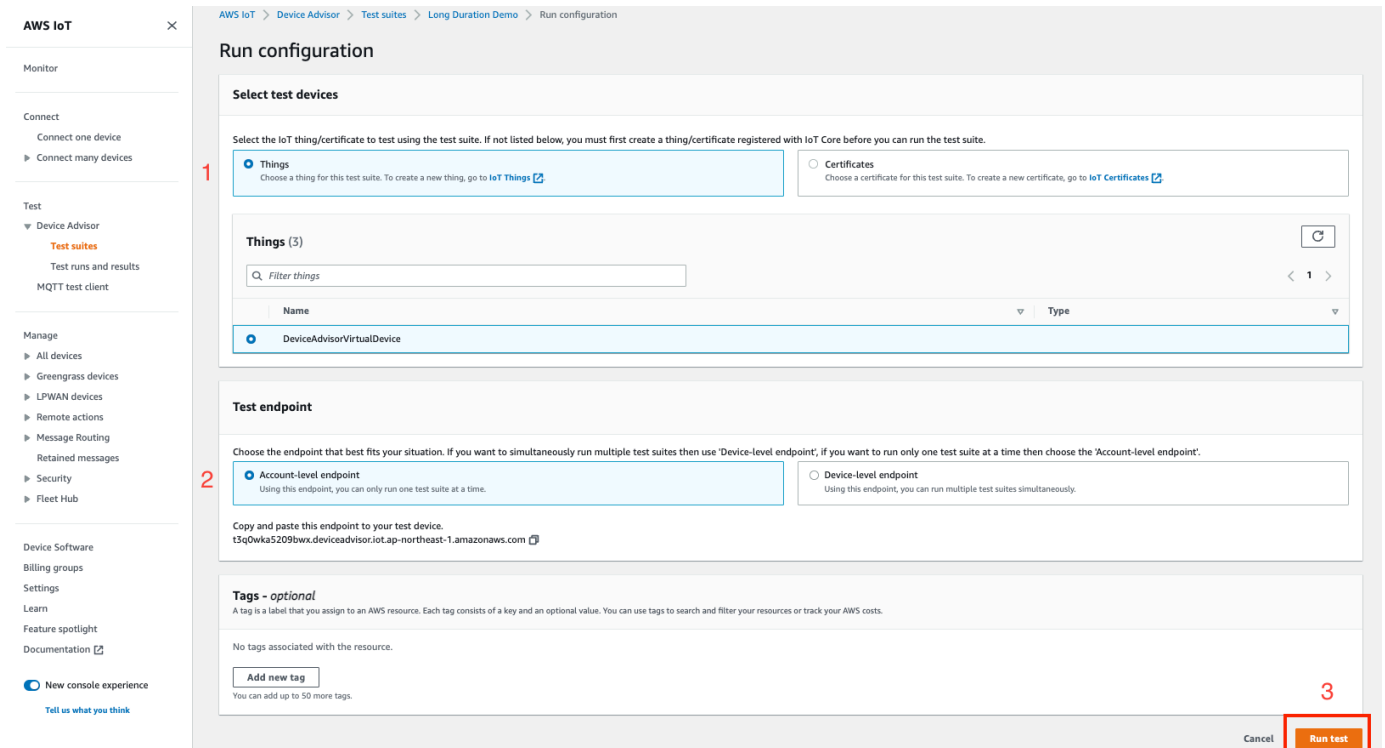
Test suite name  
Long Duration Demo

Test group name  
MQTT Test Group

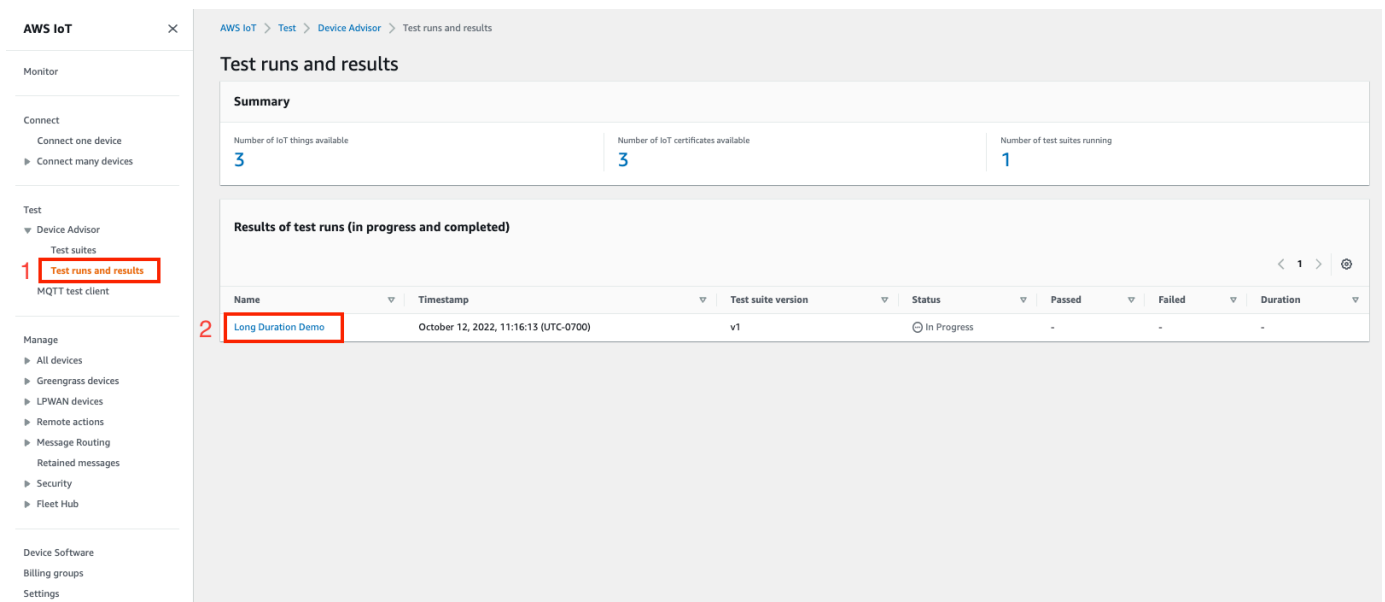
Device operations  
CONNECT, PUBLISH, SUBSCRIBE

10. Pilih opsi konfigurasi di halaman konfigurasi Jalankan.

- Pilih Things atau Certificate untuk menjalankan tes.
- Pilih titik akhir tingkat Akun atau titik akhir tingkat Perangkat.
- Pilih Jalankan tes untuk menjalankan tes.



11. Untuk melihat hasil rangkaian pengujian yang dijalankan, pilih Uji berjalan dan hasil di panel navigasi kiri. Pilih rangkaian pengujian yang dijalankan untuk melihat detail hasil.



12. Langkah sebelumnya memunculkan halaman ringkasan pengujian. Semua detail uji coba ditampilkan di halaman ini. Saat konsol meminta untuk memulai koneksi perangkat, sambungkan perangkat Anda ke titik akhir yang disediakan. Kemajuan tes terlihat di halaman ini.

13. Tes durasi panjang memberikan ringkasan log Uji tambahan di panel samping yang menampilkan semua peristiwa penting yang terjadi antara perangkat dan broker dalam waktu dekat. Untuk melihat log detail yang lebih mendalam, klik Log kasus uji.



## Titik akhir VPC Penasihat Perangkat ()AWS PrivateLink

Anda dapat membuat koneksi pribadi antara VPC Anda dan titik akhir AWS IoT Core Device Advisor pengujian (bidang data) dengan membuat titik akhir VPC antarmuka. Anda dapat menggunakan endpoint ini untuk memvalidasi AWS IoT perangkat untuk konektivitas yang andal dan aman AWS IoT Core sebelum menerapkan perangkat ke produksi. [Pengujian bawaan Device Advisor membantu Anda memvalidasi perangkat lunak perangkat Anda terhadap praktik terbaik untuk penggunaan TLS, MQTT, Device Shadow, dan Pekerjaan.AWS IoT](#)

[AWS PrivateLink](#) memberi daya pada titik akhir antarmuka yang digunakan dengan perangkat IoT Anda. Layanan ini membantu Anda mengakses titik akhir AWS IoT Core Device Advisor pengujian secara pribadi tanpa gateway internet, perangkat NAT, koneksi VPN, atau koneksi. AWS Direct Connect Instans di VPC Anda yang mengirim paket TCP dan MQTT tidak memerlukan alamat IP publik untuk berkomunikasi dengan titik akhir pengujian. AWS IoT Core Device Advisor Lalu lintas antara VPC Anda dan AWS IoT Core Device Advisor tidak pergi. AWS Cloud Setiap komunikasi TLS dan MQTT antara perangkat IoT dan kasus uji Device Advisor tetap berada dalam sumber daya di Anda. Akun AWS

Setiap titik akhir antarmuka diwakili oleh satu atau lebih [antarmuka jaringan elastis](#) dalam subnet Anda.

Untuk mempelajari selengkapnya tentang menggunakan titik akhir VPC antarmuka, lihat [Titik akhir VPC Antarmuka \(\) di AWS PrivateLink Panduan](#) Pengguna VPC Amazon.

## Pertimbangan untuk titik akhir AWS IoT Core Device Advisor VPC

Tinjau [properti dan batasan titik akhir antarmuka](#) di Panduan Pengguna Amazon VPC sebelum menyiapkan titik akhir VPC antarmuka. Pertimbangkan hal-hal berikut sebelum Anda melanjutkan:

- AWS IoT Core Device Advisor saat ini mendukung panggilan ke titik akhir pengujian Device Advisor (bidang data) dari VPC Anda. Broker pesan menggunakan komunikasi pesawat data untuk mengirim dan menerima data. Ia melakukan ini dengan bantuan paket TLS dan MQTT. Titik akhir VPC untuk AWS IoT Core Device Advisor menghubungkan AWS IoT perangkat Anda ke titik akhir pengujian Device Advisor. [Tindakan API bidang kontrol](#) tidak digunakan oleh titik akhir VPC ini. Untuk membuat atau menjalankan rangkaian pengujian atau API bidang kontrol lainnya, gunakan konsol, AWS SDK, atau Antarmuka Baris AWS Perintah melalui internet publik.
- Berikut ini Wilayah AWS mendukung titik akhir VPC untuk: AWS IoT Core Device Advisor
  - AS Timur (Virginia Utara)

- AS Barat (Oregon)
- Asia Pasifik (Tokyo)
- Eropa (Irlandia)
- Device Advisor mendukung MQTT dengan sertifikat klien X.509 dan sertifikat server RSA.
- [Kebijakan titik akhir VPC](#) tidak didukung saat ini.
- [Periksa prasyarat titik akhir VPC untuk petunjuk tentang cara membuat sumber daya yang menghubungkan titik akhir VPC](#). Anda harus membuat VPC dan subnet pribadi untuk menggunakan titik akhir AWS IoT Core Device Advisor VPC.
- Ada kuota pada AWS PrivateLink sumber daya Anda. Untuk informasi lebih lanjut, lihat [AWS PrivateLink kuota](#).
- Titik akhir VPC hanya mendukung lalu lintas IPv4.

## Buat titik akhir VPC antarmuka untuk AWS IoT Core Device Advisor

Untuk memulai dengan titik akhir VPC, [buat titik akhir VPC antarmuka](#). Selanjutnya, pilih AWS IoT Core Device Advisor sebagai Layanan AWS. Jika Anda menggunakan AWS CLI, hubungi [describe-vpc-endpoint-services](#) untuk mengonfirmasi bahwa AWS IoT Core Device Advisor ada di Availability Zone di Anda Wilayah AWS. Konfirmasikan bahwa grup keamanan yang terpasang pada titik akhir memungkinkan [komunikasi protokol TCP](#) untuk lalu lintas MQTT dan TLS. Misalnya, di Wilayah AS Timur (Virginia N.), gunakan perintah berikut:

```
aws ec2 describe-vpc-endpoint-services --service-name com.amazonaws.us-east-1.deviceadvisor.iot
```

Anda dapat membuat titik akhir VPC untuk AWS IoT Core menggunakan nama layanan berikut:

- com.amazonaws.wilayah.deviceadvisor.iot

Secara default, DNS pribadi dihidupkan untuk titik akhir. Ini memastikan bahwa penggunaan titik akhir pengujian default tetap berada dalam subnet pribadi Anda. Untuk mendapatkan titik akhir tingkat akun atau perangkat Anda, gunakan konsol, AWS CLI atau AWS SDK. Misalnya, jika Anda menjalankan [get-endpoint](#) dalam subnet publik atau di internet publik, Anda bisa mendapatkan endpoint dan menggunakannya untuk terhubung ke Device Advisor. Untuk informasi selengkapnya, lihat [Mengakses layanan melalui titik akhir antarmuka](#) dalam Panduan Pengguna Amazon VPC.

Untuk menghubungkan klien MQTT ke antarmuka titik akhir VPC, AWS PrivateLink layanan membuat catatan DNS di zona host pribadi yang dilampirkan ke VPC Anda. Catatan DNS ini mengarahkan permintaan AWS IoT perangkat ke titik akhir VPC.

## Mengontrol akses ke AWS IoT Core Device Advisor lebih dari titik akhir VPC

[Anda dapat membatasi akses perangkat ke AWS IoT Core Device Advisor dan mengizinkan akses hanya melalui titik akhir VPC dengan menggunakan tombol konteks kondisi VPC.](#) AWS IoT Core mendukung kunci konteks terkait VPC berikut:

- [SourceVpc](#)
- [SourceVpce](#)
- [VPCSourceIp](#)

### Note

AWS IoT Core Device Advisor tidak mendukung [kebijakan titik akhir VPC](#) saat ini.

Kebijakan berikut memberikan izin untuk terhubung AWS IoT Core Device Advisor menggunakan ID klien yang cocok dengan nama benda. Ini juga menerbitkan ke topik apa pun yang diawali dengan nama benda. Kebijakan ini bergantung pada perangkat yang terhubung ke titik akhir VPC dengan ID titik akhir VPC tertentu. Kebijakan ini menolak upaya koneksi ke titik akhir AWS IoT Core Device Advisor pengujian publik Anda.

```
{
 "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}/*"
]
}
]
```

## Kasus uji Device Advisor

Device Advisor menyediakan pengujian bawaan dalam enam kategori.

- [TLS](#)
- [MQTT](#)
- [Bayangan](#)
- [Eksekusi Job](#)
- [Izin dan kebijakan](#)
- [Tes durasi panjang](#)

## Kasus uji Penasihat Perangkat agar memenuhi syarat untuk Program Kualifikasi AWS Perangkat.

Perangkat Anda harus lulus tes berikut agar memenuhi syarat sesuai dengan [Program Kualifikasi AWS Perangkat](#).

**Note**

Ini adalah daftar tes kualifikasi yang direvisi.

- [TLS Connect](#) (“TLS Connect”)
- [Sertifikat Server Nama Subjek Salah TLS](#) (“Nama Umum Subjek Salah (CN) /Nama Alternatif Subjek (SAN)”)
- [Sertifikat Server Tidak Aman TLS](#) (“Tidak Ditandatangani Oleh CA yang Diakui”)
- [TLS Device Support untuk AWS IoT Cipher Suites](#) (“TLS Device Support AWS IoT for recommended Cipher Suites”)
- [TLS Menerima Fragmen Ukuran Maksimum](#) (“TLS Menerima Fragmen Ukuran Maksimum”)
- Sertifikat Server [Kedaluwarsa TLS](#) (“Sertifikat server kedaluwarsa”)
- [Sertifikat Server Ukuran Besar TLS](#) (“Sertifikat Server Ukuran Besar TLS”)
- [MQTT Connect](#) (“Perangkat kirim CONNECT AWS IoT Core ke (Happy case)”)
- [MQTT Berlangganan](#) (“Bisa Berlangganan (Happy Case)”)
- [MQTT Publikasikan](#) (“QoS0 (Happy Case)”)
- [MQTT Connect Jitter Retries](#) (“Perangkat menghubungkan percobaan ulang dengan jitter backoff - Tidak ada respons CONNACK”)

## TLS

Gunakan tes ini untuk menentukan apakah protokol keamanan lapisan transport (TLS) antara perangkat Anda dan AWS IoT aman.

**Note**

Device Advisor sekarang mendukung TLS 1.3.

## Jalan Bahagia

### TLS Connect

Memvalidasi jika perangkat yang diuji dapat menyelesaikan jabat tangan TLS. AWS IoT Tes ini tidak memvalidasi implementasi MQTT perangkat klien.

## Example Definisi kasus uji API:

### Note

EXECUTION\_TIMEOUT memiliki nilai default 5 menit. Untuk hasil terbaik, kami merekomendasikan nilai batas waktu 2 menit.

```
"tests":[
 {
 "name":"my_tls_connect_test",
 "configuration": {
 // optional:
 "EXECUTION_TIMEOUT":"300", //in seconds
 },
 "test":{
 "id":"TLS_Connect",
 "version":"0.0.0"
 }
 }
]
```

## Example Output kasus uji:

- Lulus — Perangkat yang diuji menyelesaikan jabat tangan TLS dengan. AWS IoT
- Lulus dengan peringatan — Perangkat yang diuji menyelesaikan jabat tangan TLS dengan AWS IoT, tetapi ada pesan peringatan TLS dari perangkat atau. AWS IoT
- Gagal - Perangkat yang diuji gagal menyelesaikan jabat tangan TLS AWS IoT karena kesalahan jabat tangan.

## TLS Menerima Fragmen Ukuran Maksimum

Kasus uji ini memvalidasi bahwa perangkat Anda dapat menerima dan memproses fragmen ukuran maksimum TLS. Perangkat uji Anda harus berlangganan topik yang telah dikonfigurasi sebelumnya dengan QoS 1 untuk menerima muatan besar. Anda dapat menyesuaikan payload dengan konfigurasi `{payload}`.

## Example Definisi kasus uji API:

### Note

EXECUTION\_TIMEOUT memiliki nilai default 5 menit. Untuk hasil terbaik, kami merekomendasikan nilai batas waktu 2 menit.

```
"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"
 }
 }
]
```

## Cipher Suite

TLS Device Support untuk Cipher Suites yang AWS IoT direkomendasikan

[Memvalidasi bahwa cipher suite dalam pesan TLS Client Hello dari perangkat yang diuji berisi cipher suite yang direkomendasikan.AWS IoT](#) Ini memberikan lebih banyak wawasan tentang cipher suite yang didukung oleh perangkat.

## Example Definisi kasus uji API:

### Note

EXECUTION\_TIMEOUT memiliki nilai default 5 menit. Kami merekomendasikan nilai batas waktu 2 menit.

```
"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 Output kasus uji:

- Lulus — Perangkat di bawah rangkaian sandi pengujian berisi setidaknya satu dari rangkaian AWS IoT sandi yang direkomendasikan dan tidak berisi rangkaian sandi yang tidak didukung.
- Lulus dengan peringatan - Suite cipher perangkat berisi setidaknya satu AWS IoT cipher suite tetapi:
  1. Itu tidak mengandung suite cipher yang direkomendasikan
  2. Ini berisi cipher suite yang tidak didukung oleh. AWS IoT

Kami menyarankan Anda memverifikasi bahwa setiap cipher suite yang tidak didukung aman.

- Gagal — Perangkat di bawah rangkaian cipher pengujian tidak berisi rangkaian cipher yang AWS IoT didukung.

## Sertifikat Server Ukuran Lebih Besar

### Sertifikat Server Ukuran Besar TLS

Validasi di perangkat Anda dapat menyelesaikan jabat tangan TLS AWS IoT saat menerima dan memproses sertifikat server ukuran yang lebih besar. Ukuran sertifikat server (dalam byte) yang digunakan oleh pengujian ini lebih besar daripada yang saat ini digunakan dalam kasus uji TLS Connect dan IoT Core sebesar 20. Selama kasus pengujian ini, AWS IoT uji ruang buffer perangkat Anda untuk TLS. Jika ruang buffer cukup besar, jabat tangan TLS akan hilang tanpa kesalahan. Pengujian ini tidak memvalidasi implementasi MQTT perangkat. Kasus uji ds setelah proses jabat tangan TLS selesai.



## Example Definisi kasus uji API:

### Note

EXECUTION\_TIMEOUT memiliki nilai default 5 menit. Untuk hasil terbaik, kami merekomendasikan nilai batas waktu 2 menit. Jika kasus pengujian ini gagal tetapi kasus uji TLS Connect lolos, kami sarankan Anda meningkatkan batas ruang buffer perangkat untuk TLS Meningkatkan batas ruang buffer memastikan bahwa perangkat Anda dapat memproses sertifikat server ukuran yang lebih besar jika ukurannya bertambah.

```
"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 Output kasus uji:

- Lulus — Perangkat yang diuji menyelesaikan jabat tangan TLS dengan AWS IoT
- Lulus dengan peringatan — Perangkat yang diuji menyelesaikan jabat tangan TLS dengan AWS IoT, tetapi ada pesan peringatan TLS baik dari perangkat atau AWS IoT
- Gagal — Perangkat yang diuji gagal menyelesaikan jabat tangan TLS AWS IoT karena kesalahan selama proses jabat tangan.

## Sertifikat Server TLS Tidak Aman

### Tidak Ditandatangani Oleh CA yang Diakui

Memvalidasi bahwa perangkat yang sedang diuji menutup sambungan jika disajikan dengan sertifikat server tanpa tanda tangan yang valid dari ATS CA. Perangkat hanya boleh terhubung ke titik akhir yang menampilkan sertifikat yang valid.

Example Definisi kasus uji API:

#### Note

EXECUTION\_TIMEOUT memiliki nilai default 5 menit. Kami merekomendasikan nilai batas waktu 2 menit.

```
"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 Output kasus uji:

- Lulus — Perangkat yang diuji menutup koneksi.
- Gagal - Perangkat yang diuji menyelesaikan jabat tangan TLS dengan. AWS IoT

### TLS Sertifikat Server Nama Subjek Salah/Nama Umum Subjek Salah (CN) /Nama Alternatif Subjek (SAN)

Memvalidasi bahwa perangkat yang sedang diuji menutup koneksi jika disajikan dengan sertifikat server untuk nama domain yang berbeda dari yang diminta.

### Example Definisi kasus uji API:

 Note

EXECUTION\_TIMEOUT memiliki nilai default 5 menit. Kami merekomendasikan nilai batas waktu 2 menit.

```
"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 Output kasus uji:


- Lulus — Perangkat yang diuji menutup koneksi.
- Gagal - Perangkat yang diuji menyelesaikan jabat tangan TLS dengan. AWS IoT

## Sertifikat Server Kedaluwarsa TLS

### Sertifikat server kedaluwarsa

Memvalidasi bahwa perangkat yang sedang diuji menutup koneksi jika disajikan dengan sertifikat server yang kedaluwarsa.

### Example Definisi kasus uji API:

 Note

EXECUTION\_TIMEOUT memiliki nilai default 5 menit. Kami merekomendasikan nilai batas waktu 2 menit.

```

"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 Output kasus uji:

- Lulus — Perangkat yang sedang diuji menolak untuk menyelesaikan jabat tangan TLS. AWS IoT Perangkat mengirimkan pesan peringatan TLS sebelum menutup koneksi.
- Lulus dengan peringatan — Perangkat yang diuji menolak untuk menyelesaikan jabat tangan TLS. AWS IoT Namun, itu tidak mengirim pesan peringatan TLS sebelum menutup koneksi.
- Gagal - Perangkat yang diuji melengkap jabat tangan TLS dengan. AWS IoT

## MQTT

### CONNECT, DISCONNECT, dan RECONNECT

“Perangkat mengirim CONNECT ke AWS IoT Core (Happy case)”

Memvalidasi bahwa perangkat yang sedang diuji mengirimkan permintaan CONNECT.

Definisi kasus uji API:

#### Note

EXECUTION\_TIMEOUT memiliki nilai default 5 menit. Kami merekomendasikan nilai batas waktu 2 menit.

```

"tests":[

```

```

{
 "name": "my_mqtt_connect_test",
 "configuration": {
 // optional:
 "EXECUTION_TIMEOUT": "300", // in seconds
 },
 "test": {
 "id": "MQTT_Connect",
 "version": "0.0.0"
 }
}
]

```

“Perangkat dapat mengembalikan PUBACK ke topik arbitrer untuk QoS1”

Kasus uji ini akan memeriksa apakah perangkat (klien) dapat mengembalikan pesan PUBACK jika menerima pesan publikasi dari broker setelah berlangganan topik dengan QoS1.

Konten payload dan ukuran payload dapat dikonfigurasi untuk kasus uji ini. Jika ukuran payload dikonfigurasi, Device Advisor akan menimpa nilai untuk konten payload, dan mengirim payload yang telah ditentukan ke perangkat dengan ukuran yang diinginkan. Ukuran muatan adalah nilai antara 0 hingga 128 dan tidak boleh melebihi 128 KB. AWS IoT Core menolak mempublikasikan dan menghubungkan permintaan yang lebih besar dari 128 KB, seperti yang terlihat di [broker AWS IoT Core pesan dan batas protokol dan halaman kuota](#).

Definisi kasus uji API:

#### Note

EXECUTION\_TIMEOUT memiliki nilai default 5 menit. Kami merekomendasikan nilai batas waktu 2 menit. PAYLOAD\_SIZE dapat dikonfigurasi ke nilai antara 0 dan 128 kilobyte. Mendefinisikan ukuran payload akan menggantikan konten payload karena Device Advisor akan mengirimkan payload yang telah ditentukan sebelumnya dengan ukuran yang diberikan kembali ke perangkat.

```

"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"
 }
}
]

```

“Perangkat menghubungkan percobaan ulang dengan jitter backoff - Tidak ada respons CONNACK”

Memvalidasi bahwa perangkat yang diuji menggunakan backoff jitter yang tepat saat terhubung kembali dengan broker setidaknya lima kali. Broker mencatat stempel waktu perangkat di bawah permintaan CONNECT pengujian, melakukan validasi paket, berhenti sejenak tanpa mengirim CONNACK ke perangkat yang sedang diuji, dan menunggu perangkat yang diuji untuk mengirim ulang permintaan. Upaya koneksi keenam diizinkan untuk melewati dan CONNACK diizinkan mengalir kembali ke perangkat yang diuji.

Proses sebelumnya dilakukan lagi. Secara total, kasus uji ini mengharuskan perangkat untuk terhubung setidaknya 12 kali secara total. Stempel waktu yang dikumpulkan digunakan untuk memvalidasi bahwa jitter backoff digunakan oleh perangkat yang diuji. Jika perangkat yang diuji memiliki penundaan backoff eksponensial yang ketat, kasus uji ini akan lulus dengan peringatan.

Kami merekomendasikan implementasi mekanisme [Exponential Backoff And Jitter](#) pada perangkat yang diuji untuk lulus kasus uji ini.

Definisi kasus uji API:

#### Note

EXECUTION\_TIMEOUT memiliki nilai default 5 menit. Kami merekomendasikan nilai batas waktu 4 menit.

```

"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"
 }
}
]


```

“Perangkat menghubungkan percobaan ulang dengan backoff eksponensial - Tidak ada respons CONNACK”

Memvalidasi bahwa perangkat yang diuji menggunakan backoff eksponensial yang tepat saat terhubung kembali dengan broker setidaknya lima kali. Broker mencatat stempel waktu perangkat di bawah permintaan CONNECT pengujian, melakukan validasi paket, berhenti sejenak tanpa mengirim CONNACK ke perangkat klien, dan menunggu perangkat yang diuji untuk mengirim ulang permintaan. Stempel waktu yang dikumpulkan digunakan untuk memvalidasi bahwa backoff eksponensial digunakan oleh perangkat yang diuji.

Kami merekomendasikan implementasi mekanisme [Exponential Backoff And Jitter](#) pada perangkat yang diuji untuk lulus kasus uji ini.

Definisi kasus uji API:

 Note

EXECUTION\_TIMEOUT memiliki nilai default 5 menit. Kami merekomendasikan nilai batas waktu 4 menit.

```

"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"
 }
 }
]

```

```
}
]
```

### “Perangkat terhubung kembali dengan jitter backoff - Setelah server terputus”

Memvalidasi jika perangkat yang diuji menggunakan jitter dan backoff yang diperlukan saat menyambung kembali setelah terputus dari server. Device Advisor memutus perangkat dari server setidaknya lima kali dan mengamati perilaku perangkat untuk penyambungan ulang MQTT. Device Advisor mencatat stempel waktu permintaan CONNECT untuk perangkat yang sedang diuji, melakukan validasi paket, berhenti sejenak tanpa mengirim CONNACK ke perangkat klien, dan menunggu perangkat yang diuji untuk mengirim ulang permintaan. Stempel waktu yang dikumpulkan digunakan untuk memvalidasi bahwa perangkat yang diuji menggunakan jitter dan backoff saat menyambung kembali. Jika perangkat yang diuji memiliki backoff eksponensial yang ketat atau tidak menerapkan mekanisme backoff jitter yang tepat, kasus uji ini akan lulus dengan peringatan. Jika perangkat yang diuji telah menerapkan backoff linier atau mekanisme backoff konstan, pengujian akan gagal.

Untuk lulus kasus uji ini, kami sarankan untuk menerapkan mekanisme [Exponential Backoff And Jitter](#) pada perangkat yang diuji.

Definisi kasus uji API:

#### Note

EXECUTION\_TIMEOUT memiliki nilai default 5 menit. Kami merekomendasikan nilai batas waktu 4 menit.

Jumlah upaya koneksi ulang untuk memvalidasi backoff dapat diubah dengan menentukan RECONNECTION\_ATTEMPTS. Jumlahnya harus antara 5 dan 10. Nilai bawaannya adalah 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"
 }
}
]
```

### “Perangkat terhubung kembali dengan jitter backoff - Pada koneksi yang tidak stabil”

Memvalidasi jika perangkat yang diuji menggunakan jitter dan backoff yang diperlukan saat menyambung kembali pada koneksi yang tidak stabil. Device Advisor memutus perangkat dari server setelah lima koneksi berhasil, dan mengamati perilaku perangkat untuk koneksi ulang MQTT. Device Advisor mencatat stempel waktu permintaan CONNECT untuk perangkat yang sedang diuji, melakukan validasi paket, mengirim kembali CONNACK, memutuskan sambungan, mencatat stempel waktu pemutusan, dan menunggu perangkat yang diuji untuk mengirim ulang permintaan. Stempel waktu yang dikumpulkan digunakan untuk memvalidasi bahwa perangkat yang diuji menggunakan jitter dan backoff saat menyambung kembali setelah koneksi berhasil tetapi tidak stabil. Jika perangkat yang diuji memiliki backoff eksponensial yang ketat atau tidak menerapkan mekanisme backoff jitter yang tepat, kasus uji ini akan lulus dengan peringatan. Jika perangkat yang diuji telah menerapkan backoff linier atau mekanisme backoff konstan, pengujian akan gagal.

Untuk lulus kasus uji ini, kami sarankan untuk menerapkan mekanisme [Exponential Backoff And Jitter](#) pada perangkat yang diuji.

Definisi kasus uji API:

#### Note

EXECUTION\_TIMEOUT memiliki nilai default 5 menit. Kami merekomendasikan nilai batas waktu 4 menit.

Jumlah upaya koneksi ulang untuk memvalidasi backoff dapat diubah dengan menentukan RECONNECTION\_ATTEMPTS Jumlahnya harus antara 5 dan 10. Nilai bawaannya adalah 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"
 }
}
]

```

## Publikasikan

### “QoS0 (Kasus Bahagia)”

Memvalidasi bahwa perangkat yang sedang diuji menerbitkan pesan dengan QoS0 atau QoS1. Anda juga dapat memvalidasi topik pesan dan payload dengan menentukan nilai topik dan payload dalam pengaturan pengujian.

#### Note

EXECUTION\_TIMEOUT memiliki nilai default 5 menit. Kami merekomendasikan nilai batas waktu 2 menit.

```

"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"
 }
 }
]

```

## “QoS1 terbitkan coba lagi - Tanpa PUBACK”

Memvalidasi bahwa perangkat yang diuji menerbitkan kembali pesan yang dikirim dengan QoS1, jika broker tidak mengirim PUBACK. Anda juga dapat memvalidasi topik pesan dengan menentukan topik ini di pengaturan pengujian. Perangkat klien tidak boleh memutuskan sambungan sebelum memublikasikan ulang pesan. Tes ini juga memvalidasi bahwa pesan yang diterbitkan ulang memiliki pengenal paket yang sama dengan aslinya. Selama eksekusi pengujian, jika perangkat kehilangan koneksi dan menyambung kembali, kasus uji akan diatur ulang tanpa gagal dan perangkat harus melakukan langkah-langkah kasus uji lagi.

Definisi kasus uji API:

### Note

EXECUTION\_TIMEOUT memiliki nilai default 5 menit. Disarankan setidaknya selama 4 menit.

```
"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"
 }
 }
]
```

## “Publikasikan pesan yang disimpan”

Memvalidasi bahwa perangkat yang sedang diuji menerbitkan pesan dengan `retainFlag` disetel ke `true`. Anda dapat memvalidasi topik dan payload pesan dengan menyetel nilai topik dan payload dalam setelan pengujian. Jika yang `retainFlag` dikirim dalam paket PUBLISH tidak disetel ke `true`, kasus uji akan gagal.

## Definisi kasus uji API:

### Note

EXECUTION\_TIMEOUT memiliki nilai default 5 menit. Kami merekomendasikan nilai batas waktu 2 menit. Untuk menjalankan kasus uji ini, tambahkan `iot:RetainPublish` tindakan dalam [peran perangkat](#) Anda.

```
"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"
 }
 }
]
```

## “Publikasikan dengan Properti Pengguna”

Memvalidasi bahwa perangkat yang sedang diuji menerbitkan pesan dengan properti pengguna yang benar. Anda dapat memvalidasi properti pengguna dengan menyetel pasangan nama-nilai dalam pengaturan pengujian. Jika properti pengguna tidak disediakan atau tidak cocok, kasus uji gagal.

## Definisi kasus uji API:

### Note

Ini adalah kasus uji MQTT5 saja.

EXECUTION\_TIMEOUT memiliki nilai default 5 menit. Kami merekomendasikan nilai batas waktu 2 menit.

```
"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"
 }
 }
]
```

## Langganan

### “Bisa Berlangganan (Happy Case)”

Memvalidasi bahwa perangkat yang sedang diuji berlangganan topik MQTT. Anda juga dapat memvalidasi topik yang dilanggan perangkat yang sedang diuji dengan menentukan topik ini di setelan pengujian.

Definisi kasus uji API:

#### Note

EXECUTION\_TIMEOUT memiliki nilai default 5 menit. Kami merekomendasikan nilai batas waktu 2 menit.

```
"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"
 }
 }
]

```

### “Berlangganan Coba Lagi - Tanpa SUBACK”

Memvalidasi bahwa perangkat yang sedang diuji mencoba ulang langganan yang gagal ke topik MQTT. Server kemudian menunggu dan tidak mengirim SUBACK. Jika perangkat klien tidak mencoba lagi langganan, pengujian gagal. Perangkat klien harus mencoba lagi langganan yang gagal dengan Id paket yang sama. Anda juga dapat memvalidasi topik yang dilanggan perangkat yang sedang diuji dengan menentukan topik ini di setelan pengujian. Selama eksekusi pengujian, jika perangkat kehilangan koneksi dan menyambung kembali, kasus uji akan diatur ulang tanpa gagal dan perangkat harus melakukan langkah-langkah kasus uji lagi.

Definisi kasus uji API:

#### Note

EXECUTION\_TIMEOUT memiliki nilai default 5 menit. Kami merekomendasikan nilai batas waktu 4 menit.

```

"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"
 }
}
]

```

## Jaga-Hidup

### “Mqtt Tanpa Ack” PingResp

Kasus uji ini memvalidasi jika perangkat yang diuji terputus saat tidak menerima respons ping. Sebagai bagian dari kasus uji ini, Device Advisor memblokir tanggapan yang dikirim dari AWS IoT Core permintaan publikasi, berlangganan, dan ping. Ini juga memvalidasi jika perangkat yang diuji memutuskan koneksi MQTT.

Definisi kasus uji API:

#### Note

EXECUTION\_TIMEOUT memiliki nilai default 5 menit. Kami merekomendasikan batas waktu yang lebih besar dari 1,5 kali keepAliveTime nilainya. Maksimum keepAliveTime harus tidak lebih dari 230 detik untuk tes ini.

```

"tests": [
 {
 "name": "Mqtt No Ack PingResp",
 "configuration":
 //optional:
 "EXECUTION_TIMEOUT": "306", // in seconds
 },
 "test": {
 "id": "MQTT_No_Ack_PingResp",
 "version": "0.0.0"
 }
}
]

```

## Sesi Persisten

### “Sesi Persisten (Happy Case)”

Kasus uji ini memvalidasi perilaku perangkat saat terputus dari sesi persisten. Kasus uji memeriksa apakah perangkat dapat menyambung kembali, melanjutkan langganan ke topik pemicunya tanpa berlangganan ulang secara eksplisit, menerima pesan yang disimpan dalam topik, dan bekerja seperti yang diharapkan selama sesi persisten. Ketika kasus uji ini berlalu, ini menunjukkan bahwa perangkat klien dapat mempertahankan sesi persisten dengan AWS IoT Core broker dengan cara yang diharapkan. Untuk informasi selengkapnya tentang Sesi AWS IoT Persisten, lihat [Menggunakan sesi persisten MQTT](#).

Dalam kasus pengujian ini, perangkat klien diharapkan untuk TERHUBUNG dengan flag sesi bersih yang disetel ke false, lalu berlangganan topik pemicu. AWS IoT Core Setelah berlangganan berhasil, perangkat akan terputus oleh AWS IoT Core Device Advisor. Saat perangkat dalam keadaan terputus, muatan pesan QoS 1 akan disimpan dalam topik itu. Device Advisor kemudian akan mengizinkan perangkat klien untuk terhubung kembali dengan titik akhir pengujian. Pada titik ini, karena ada sesi persisten, perangkat klien diharapkan untuk melanjutkan langganan topiknya tanpa mengirim paket SUBSCRIBE tambahan dan menerima pesan QoS 1 dari broker. Setelah menyambung kembali, jika perangkat klien berlangganan kembali ke topik pemicunya lagi dengan mengirimkan paket SUBSCRIBE tambahan dan/atau jika klien gagal menerima pesan yang disimpan dari topik pemicu, kasus uji akan gagal.

Definisi kasus uji API:

#### Note

EXECUTION\_TIMEOUT memiliki nilai default 5 menit. Kami merekomendasikan nilai batas waktu minimal 4 menit. Pada koneksi pertama, perangkat klien perlu secara eksplisit berlangganan TRIGGER\_TOPIC yang tidak berlangganan sebelumnya. Untuk lulus test case, perangkat klien harus berhasil berlangganan TRIGGER\_TOPIC dengan QoS 1. Setelah terhubung kembali, perangkat klien diharapkan memahami bahwa ada sesi persisten aktif; jadi harus menerima pesan tersimpan yang dikirim oleh topik pemicu dan mengembalikan PUBACK untuk pesan tertentu.

```
"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"
}
}
```

### “Sesi Persisten - Kedaluwarsa Sesi”

Kasus uji ini membantu memvalidasi perilaku perangkat saat perangkat yang terputus tersambung kembali ke sesi persisten yang kedaluwarsa. Setelah sesi berakhir, kami mengharapkan perangkat untuk berlangganan kembali ke topik yang sebelumnya berlangganan dengan secara eksplisit mengirimkan paket SUBSCRIBE baru.

Selama koneksi pertama, kami mengharapkan perangkat uji untuk TERHUBUNG dengan broker AWS IoT, karena CleanSession benderanya disetel ke false untuk memulai sesi persisten. Perangkat kemudian harus berlangganan topik pemicu. Kemudian perangkat diputuskan oleh AWS IoT Core Device Advisor, setelah berlangganan berhasil dan memulai sesi persisten. Setelah pemutusan sambungan, AWS IoT Core Device Advisor memungkinkan perangkat uji untuk terhubung kembali dengan titik akhir pengujian. Pada titik ini, ketika perangkat uji mengirim paket CONNECT lain, AWS IoT Core Device Advisor mengirimkan kembali paket CONNACK yang menunjukkan bahwa sesi persisten telah kedaluwarsa. Perangkat uji perlu menafsirkan paket ini dengan benar, dan diharapkan untuk berlangganan kembali ke topik pemicu yang sama lagi saat sesi persisten dihentikan. Jika perangkat uji tidak berlangganan kembali ke pemicu topiknya lagi, kasus uji gagal. Agar tes lulus, perangkat perlu memahami bahwa sesi persisten telah berakhir, dan mengirim kembali paket SUBSCRIBE baru untuk topik pemicu yang sama di koneksi kedua.

Jika kasus uji ini lolos untuk perangkat uji, ini menunjukkan bahwa perangkat dapat menangani koneksi ulang saat berakhirnya sesi persisten dengan cara yang diharapkan.

## Definisi kasus uji API:

### Note

EXECUTION\_TIMEOUT memiliki nilai default 5 menit. Kami merekomendasikan nilai batas waktu minimal 4 menit. Perangkat uji perlu secara eksplisit berlangganan TRIGGER\_TOPIC, yang sebelumnya tidak berlangganan. Untuk lulus test case, perangkat uji harus mengirim paket CONNECT dengan CleanSession flag disetel ke false, dan berhasil berlangganan topik pemicu dengan QoS 1. Setelah koneksi berhasil, AWS IoT Core Device Advisor memutus perangkat. Setelah pemutusan, AWS IoT Core Device Advisor memungkinkan perangkat untuk terhubung kembali, dan perangkat diharapkan untuk berlangganan ulang yang sama TRIGGER\_TOPIC karena AWS IoT Core Device Advisor akan menghentikan sesi persisten.

```
"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"
 }
 }
]
```

## Bayangan

Gunakan pengujian ini untuk memverifikasi perangkat Anda yang sedang diuji, gunakan layanan AWS IoT Device Shadow dengan benar. Untuk informasi selengkapnya, lihat [AWS IoT Layanan Device Shadow](#). Jika kasus pengujian ini dikonfigurasi dalam rangkaian pengujian Anda, maka penyediaan sesuatu diperlukan saat memulai suite run.

MQTT over tidak WebSocket didukung saat ini.

## Publikasikan

“Perangkat menerbitkan status setelah terhubung (Happy case)”

Memvalidasi jika perangkat dapat mempublikasikan statusnya setelah terhubung ke AWS IoT Core

Definisi kasus uji API:

### Note

EXECUTION\_TIMEOUT memiliki nilai default 5 menit. Kami merekomendasikan nilai batas waktu 2 menit.

```
"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 dapat disediakan untuk validasi tambahan pada status bayangan persis perangkat Anda, setelah terhubung. Secara default, kasus uji ini memvalidasi status penerbitan perangkat Anda.

Jika tidak *SHADOW\_NAME* disediakan, kasus uji akan mencari pesan yang dipublikasikan ke awalan topik tipe bayangan Unnamed (klasik) secara default. Berikan nama bayangan jika perangkat Anda menggunakan tipe bayangan bernama. Lihat [Menggunakan bayangan di perangkat](#) untuk informasi selengkapnya.

## Perbarui

“Pembaruan perangkat melaporkan status ke status yang diinginkan (Happy case)”

Memvalidasi jika perangkat Anda membaca semua pesan pembaruan yang diterima dan menyinkronkan status perangkat agar sesuai dengan properti status yang diinginkan. Perangkat Anda harus mempublikasikan status terbaru yang dilaporkan setelah disinkronkan. Jika perangkat Anda sudah memiliki bayangan yang ada sebelum menjalankan pengujian, pastikan status yang diinginkan dikonfigurasi untuk kasus uji dan status laporan yang ada belum cocok. Anda dapat mengidentifikasi pesan pembaruan Shadow yang dikirim oleh Device Advisor dengan melihat ClientToken bidang di dokumen Shadow sebagaimana DeviceAdvisorShadowTestCaseSetup adanya.

Definisi kasus uji API:

### Note

EXECUTION\_TIMEOUT memiliki nilai default 5 menit. Kami merekomendasikan nilai batas waktu 2 menit.

```
"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 harus memiliki setidaknya satu atribut dan nilai terkait.

Jika tidak SHADOW\_NAME disediakan, maka kasus uji mencari pesan yang dipublikasikan ke awalan topik dari tipe bayangan Unnamed (klasik) secara default. Berikan nama bayangan jika perangkat Anda menggunakan tipe bayangan bernama. Lihat [Menggunakan bayangan di perangkat](#) untuk informasi selengkapnya.

## Eksekusi Job

“Perangkat dapat menyelesaikan eksekusi pekerjaan”

Kasus uji ini membantu Anda memvalidasi jika perangkat Anda dapat menerima pembaruan menggunakan AWS IoT Pekerjaan, dan mempublikasikan status pembaruan yang berhasil. Untuk informasi selengkapnya tentang AWS IoT Pekerjaan, lihat [Pekerjaan](#).

Agar berhasil menjalankan kasus uji ini, ada dua AWS topik cadangan yang Anda perlukan untuk memberikan [Peran Perangkat](#) Anda. Untuk berlangganan pesan terkait aktivitas pekerjaan, gunakan topik notifikasi dan beri tahu berikutnya. Peran perangkat Anda harus memberikan tindakan PUBLISH untuk topik berikut:

- \$aws/things/thingName /jobs/jobId/dapatkan
- \$ aws/things/ ThingName /jobs/jobId/update

Disarankan untuk memberikan tindakan SUBSCRIBE dan RECEIVE untuk topik-topik berikut:

- \$aws/things/thingName /jobs/dapatkan/diterima
- \$aws/things/thingName /jobs/ jobId/dapatkan/ditolak
- \$aws/things/ ThingName /jobs/ jobId/update/diterima
- \$aws/things/ ThingName /jobs/ jobId/update/ditolak

Disarankan untuk memberikan tindakan BERLANGGANAN untuk topik berikut:

- \$aws/things/ ThingName /jobs/notify-next

Untuk informasi selengkapnya tentang topik yang dicadangkan ini, lihat topik yang dicadangkan untuk [AWS IoT Pekerjaan](#).

MQTT over tidak WebSocket didukung saat ini.

Definisi kasus uji API:

**Note**

EXECUTION\_TIMEOUT memiliki nilai default 5 menit. Kami merekomendasikan nilai batas waktu 3 menit. Bergantung pada dokumen AWS IoT Job atau sumber yang disediakan, sesuaikan nilai batas waktu (misalnya, jika pekerjaan akan memakan waktu lama untuk dijalankan, tentukan nilai batas waktu yang lebih lama untuk kasus uji). Untuk menjalankan pengujian, diperlukan dokumen AWS IoT Job yang valid atau ID pekerjaan yang sudah ada. Dokumen AWS IoT Job dapat disediakan sebagai dokumen JSON atau tautan S3. Jika dokumen pekerjaan disediakan, memberikan ID pekerjaan adalah opsional. Jika ID lowongan diberikan, Device Advisor akan menggunakan ID tersebut saat membuat AWS IoT Job atas nama Anda. Jika dokumen pekerjaan tidak disediakan, Anda dapat memberikan ID yang ada di wilayah yang sama dengan saat Anda menjalankan kasus uji. Dalam hal ini, Device Advisor akan menggunakan AWS IoT Job tersebut saat menjalankan test case.

```
"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"
 }
}
]

```

Untuk informasi selengkapnya tentang membuat dan menggunakan dokumen pekerjaan, lihat [dokumen pekerjaan](#).

## Izin dan kebijakan

Anda dapat menggunakan pengujian berikut untuk menentukan apakah kebijakan yang dilampirkan pada sertifikat perangkat Anda mengikuti praktik terbaik standar.

MQTT over tidak WebSocket didukung saat ini.

“Kebijakan terlampir sertifikat perangkat tidak mengandung wildcard”

Memvalidasi jika kebijakan izin yang terkait dengan perangkat mengikuti praktik terbaik dan tidak memberi perangkat izin lebih dari yang diperlukan.

Definisi kasus uji API:

### Note

EXECUTION\_TIMEOUT memiliki nilai default 1 menit. Sebaiknya atur batas waktu minimal 30 detik.

```

"tests": [
 {

```

```
[
 {
 "name": "my_security_device_policies",
 "configuration": {
 // optional:
 "EXECUTION_TIMEOUT": "60" // in seconds
 },
 "test": {
 "id": "Security_Device_Policies",
 "version": "0.0.0"
 }
 }
]
```

## Tes durasi panjang

Tes durasi panjang adalah rangkaian pengujian baru yang memantau perilaku perangkat saat beroperasi dalam jangka waktu yang lebih lama. Dibandingkan dengan menjalankan pengujian individual yang berfokus pada perilaku tertentu perangkat, tes durasi panjang memeriksa perilaku perangkat dalam berbagai skenario dunia nyata selama masa pakai perangkat. Device Advisor mengatur pengujian dalam urutan yang paling efisien. Pengujian menghasilkan hasil dan log, termasuk log ringkasan dengan metrik berguna tentang kinerja perangkat saat diuji.

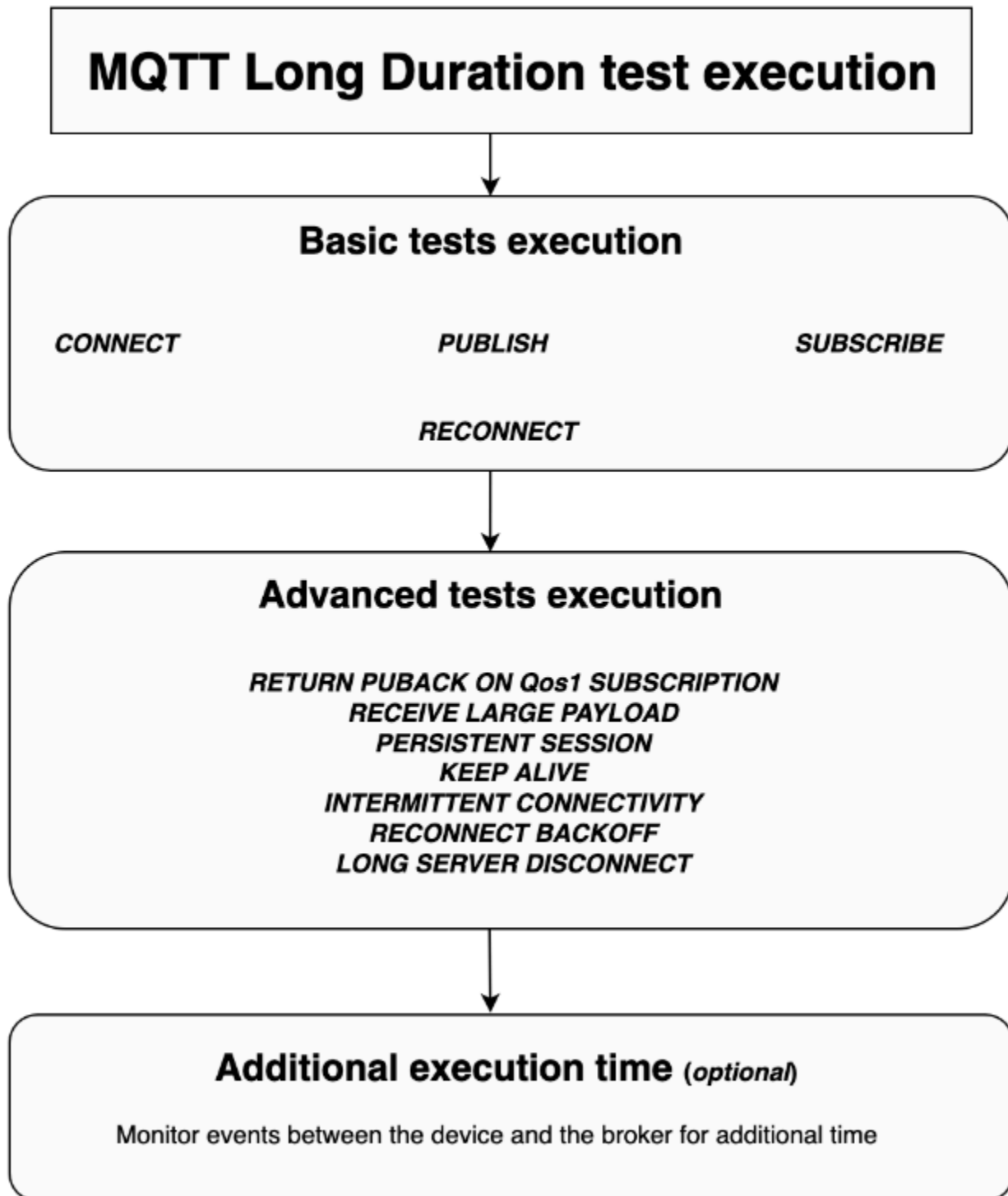
## Kasus uji durasi panjang MQTT

Dalam kasus uji durasi panjang MQTT, perilaku perangkat awalnya diamati dalam skenario kasus bahagia seperti MQTT Connect, Subscribe, Publish, dan Reconnect. Kemudian, perangkat diamati dalam beberapa skenario kegagalan kompleks seperti MQTT Reconnect Backoff, Long Server Disconnect, dan Intermittent Connectivity.

## Alur eksekusi kasus uji durasi panjang MQTT

Ada tiga fase dalam pelaksanaan kasus uji durasi panjang MQTT:





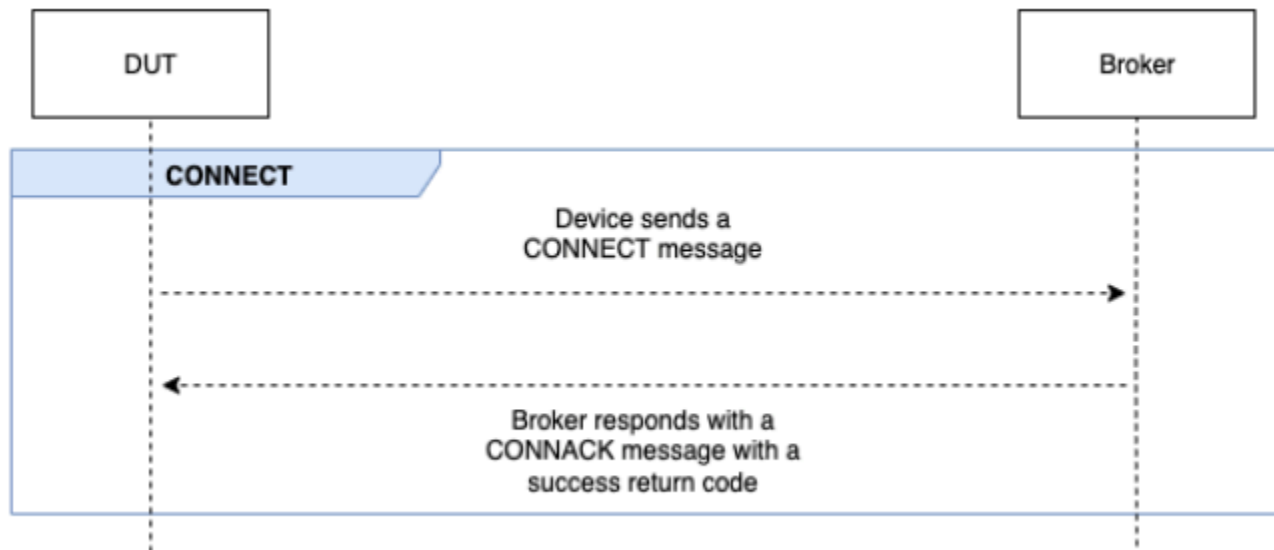
## Eksekusi tes dasar

Pada fase ini, test case menjalankan tes sederhana secara paralel. Pengujian memvalidasi jika perangkat memiliki operasi yang dipilih dalam konfigurasi.

Serangkaian tes dasar dapat mencakup yang berikut, berdasarkan operasi yang dipilih:

### MENGHUBUNG

Skenario ini memvalidasi jika perangkat mampu membuat koneksi yang sukses dengan broker.

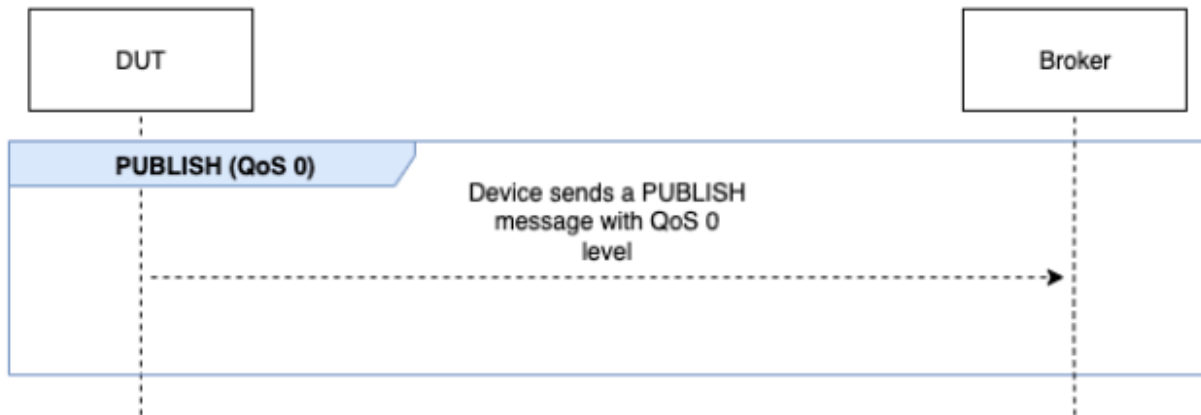


### MENERBITKAN

Skenario ini memvalidasi jika perangkat berhasil menerbitkan terhadap broker.

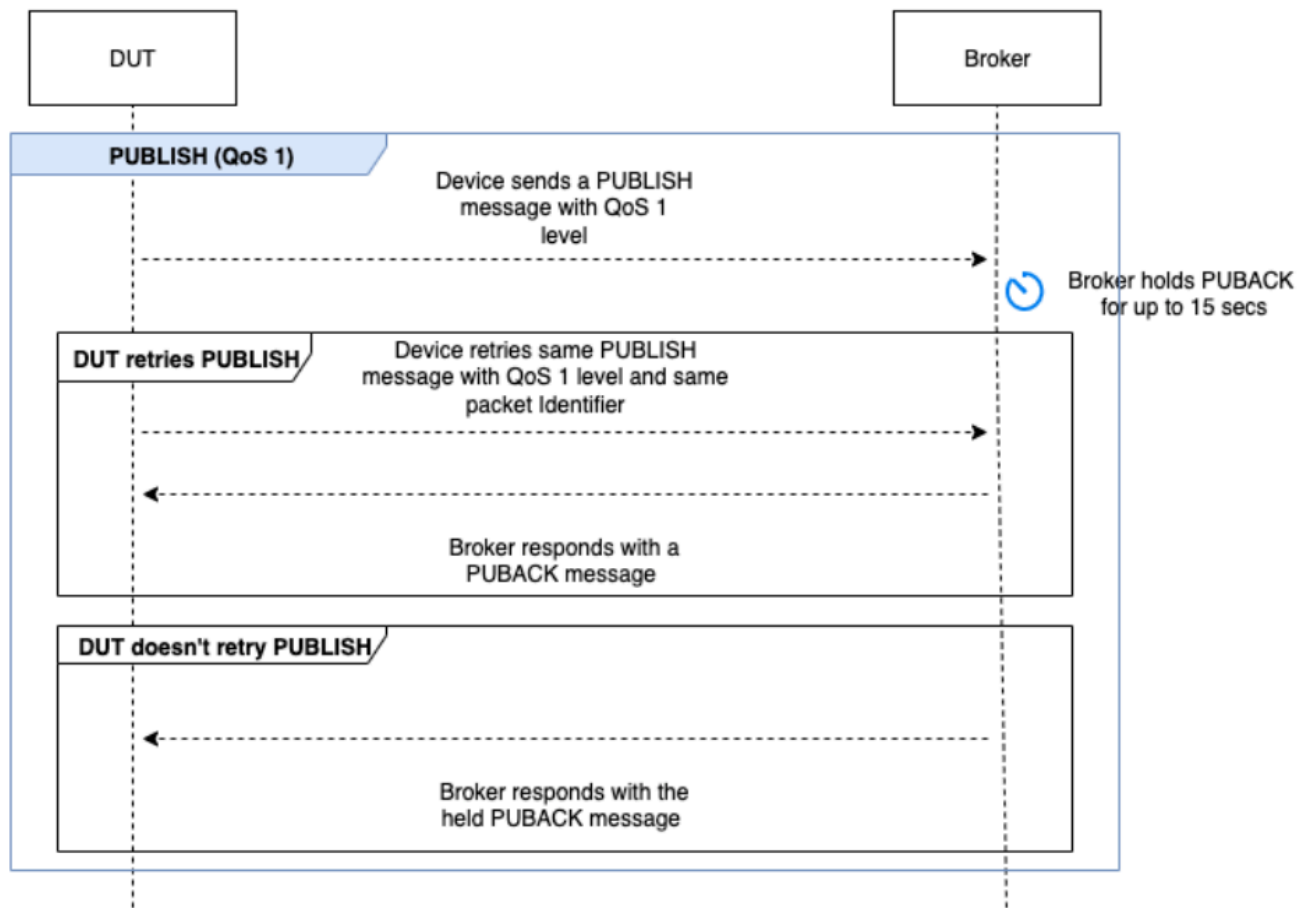
### QoS 0

Kasus uji ini memvalidasi jika perangkat berhasil mengirim PUBLISH pesan ke broker selama publikasi dengan QoS 0. Tes tidak menunggu PUBACK pesan diterima oleh perangkat.



## QoS 1

Dalam kasus uji ini, perangkat diharapkan mengirim dua PUBLISH pesan ke broker dengan QoS 1. Setelah PUBLISH pesan pertama, broker menunggu hingga 15 detik sebelum merespons. Perangkat harus mencoba lagi PUBLISH pesan asli dengan pengenal paket yang sama dalam jendela 15 detik. Jika ya, broker merespons dengan PUBACK pesan dan tes memvalidasi. Jika perangkat tidak mencoba lagi PUBLISH, yang asli PUBACK dikirim ke perangkat dan pengujian ditandai sebagai Lulus dengan peringatan, bersama dengan pesan sistem. Selama eksekusi pengujian, jika perangkat kehilangan koneksi dan menyambung kembali, skenario pengujian akan diatur ulang tanpa gagal dan perangkat harus melakukan langkah-langkah skenario pengujian lagi.

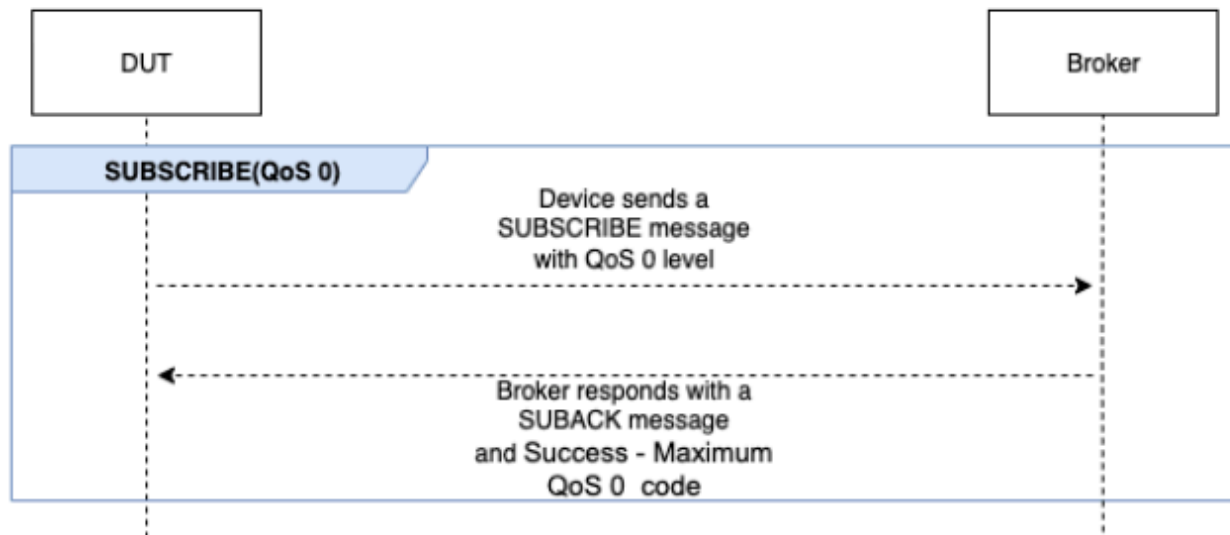


## BERLANGGANAN

Skenario ini memvalidasi jika perangkat berhasil berlangganan broker.

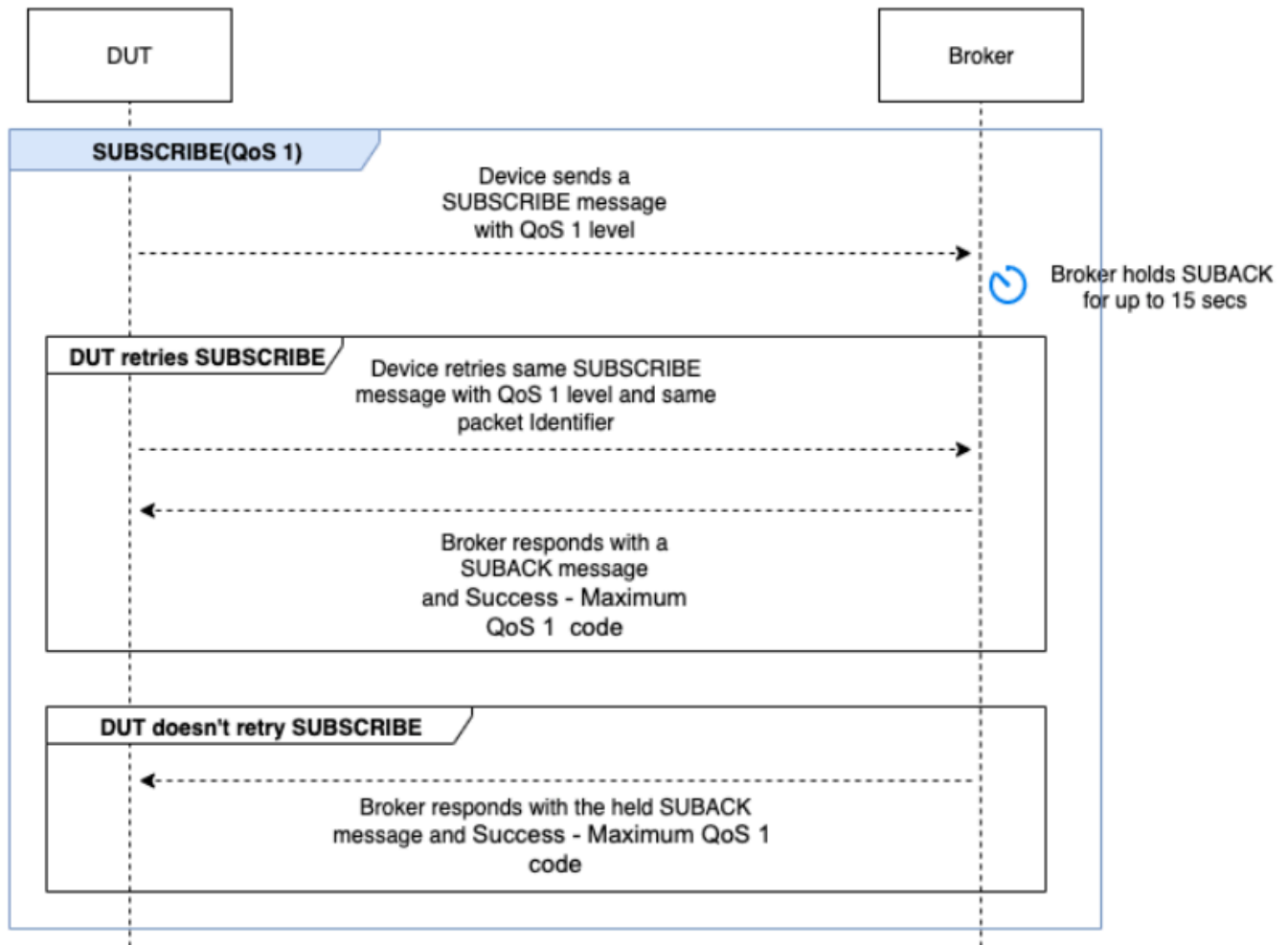
### QoS 0

Kasus uji ini memvalidasi jika perangkat berhasil mengirim SUBSCRIBE pesan ke broker selama berlangganan dengan QoS 0. Tes tidak menunggu perangkat menerima pesan SUBACK.



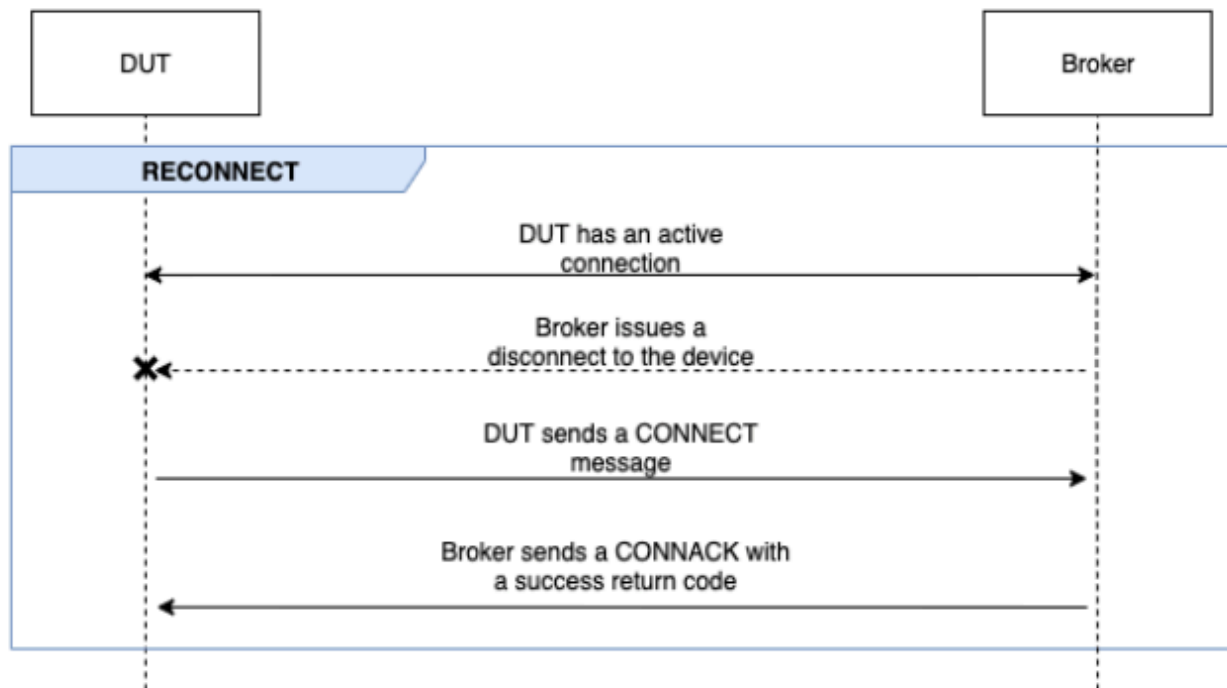
## QoS 1

Dalam kasus uji ini, perangkat diharapkan mengirim dua SUBSCRIBE pesan ke broker dengan QoS 1. Setelah SUBSCRIBE pesan pertama, broker menunggu hingga 15 detik sebelum merespons. Perangkat harus mencoba lagi SUBSCRIBE pesan asli dengan pengenal paket yang sama dalam jendela 15 detik. Jika ya, broker merespons dengan SUBACK pesan dan tes memvalidasi. Jika perangkat tidak mencoba lagi SUBSCRIBE, yang asli SUBACK dikirim ke perangkat dan pengujian ditandai sebagai Lulus dengan peringatan, bersama dengan pesan sistem. Selama eksekusi pengujian, jika perangkat kehilangan koneksi dan menyambung kembali, skenario pengujian akan diatur ulang tanpa gagal dan perangkat harus melakukan langkah-langkah skenario pengujian lagi.



## SAMBUNGAN KEMBALI

Skenario ini memvalidasi jika perangkat berhasil terhubung kembali dengan broker setelah perangkat terputus dari koneksi yang berhasil. Device Advisor tidak akan memutuskan sambungan perangkat jika terhubung lebih dari satu kali sebelumnya selama rangkaian pengujian. Sebaliknya, itu akan menandai tes sebagai Lulus.



### Eksekusi tes lanjutan

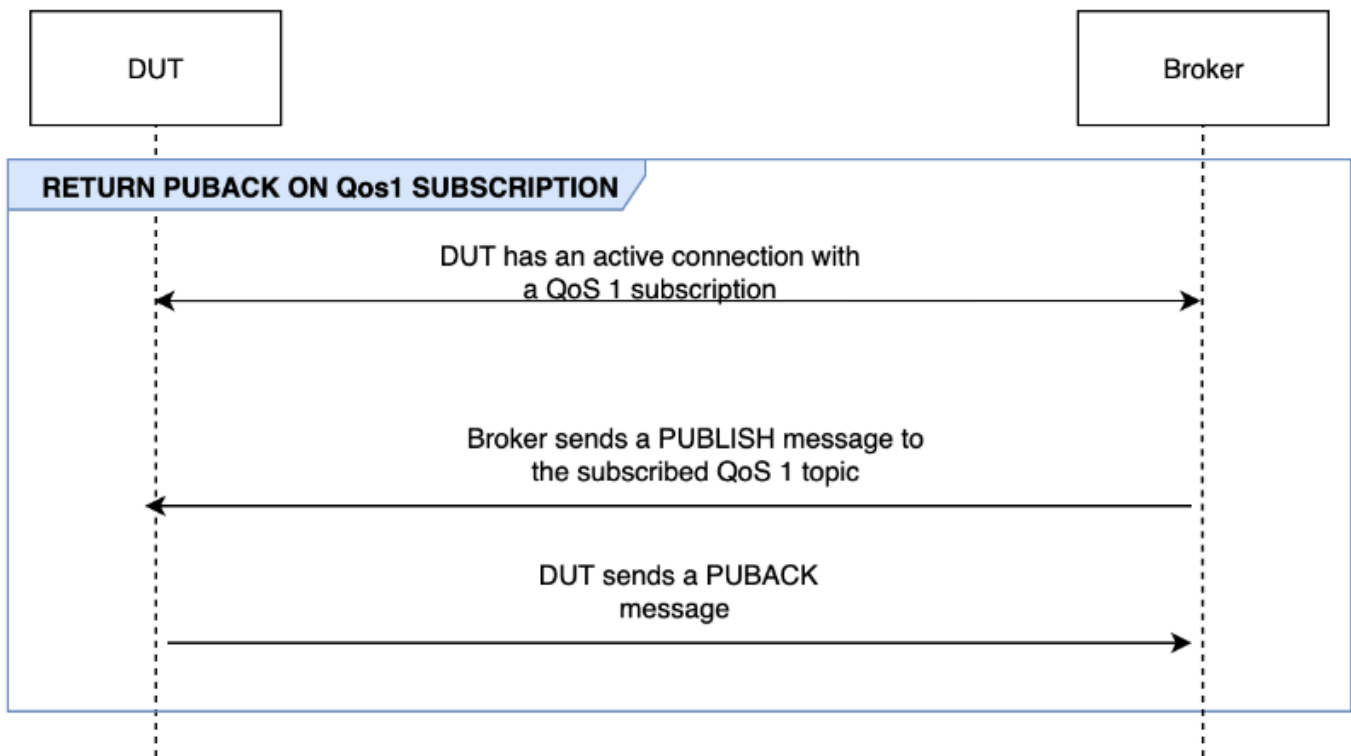
Pada fase ini, kasus uji menjalankan pengujian yang lebih kompleks secara serial untuk memvalidasi jika perangkat mengikuti praktik terbaik. Tes lanjutan ini tersedia untuk dipilih dan dapat dipilih jika tidak diperlukan. Setiap tes lanjutan memiliki nilai batas waktu sendiri berdasarkan apa yang dituntut skenario.

### KEMBALIKAN PUBACK PADA LANGGANAN QoS 1

#### Note

Pilih skenario ini hanya jika perangkat Anda mampu melakukan langganan QoS 1.

Skenario ini memvalidasi jika, setelah perangkat berlangganan topik dan menerima PUBLISH pesan dari broker, ia mengembalikan pesan PUBACK.



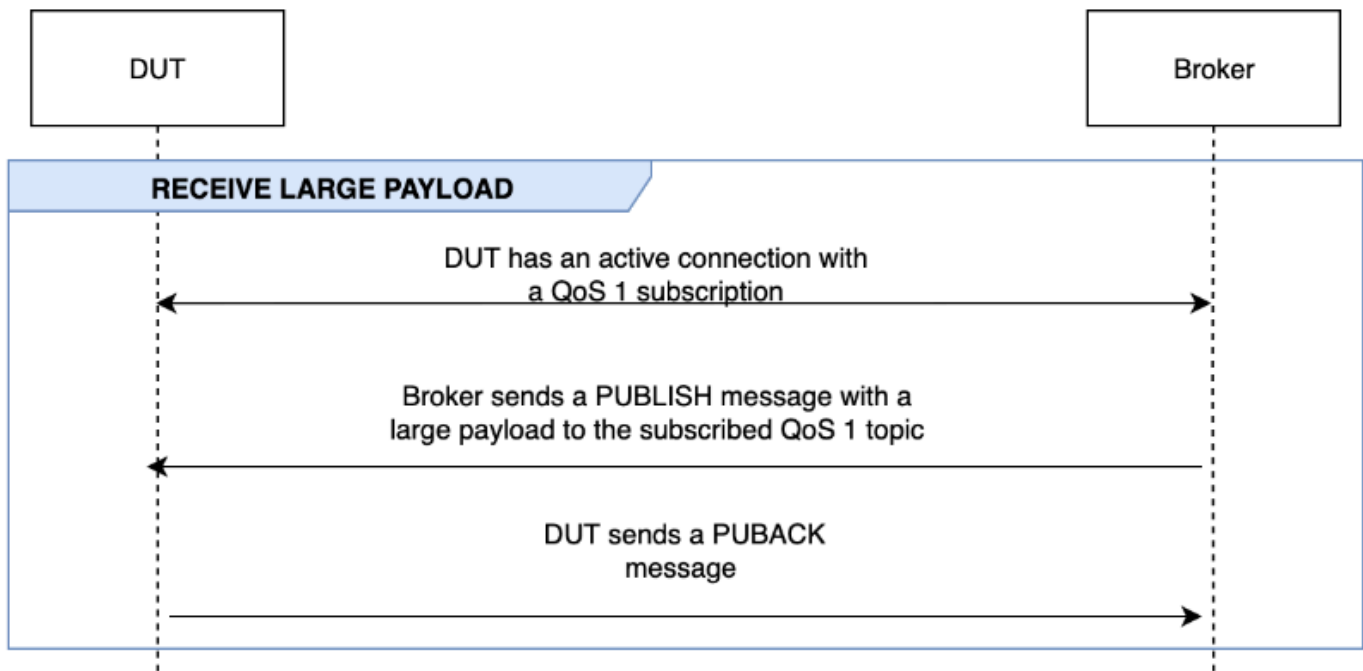
## MENERIMA MUATAN BESAR

### **i** Note

Pilih skenario ini hanya jika perangkat Anda mampu melakukan langganan QoS 1.

Skenario ini memvalidasi jika perangkat merespons dengan PUBACK pesan setelah menerima PUBLISH pesan dari broker untuk topik QoS 1 dengan muatan besar. Format payload yang diharapkan dapat dikonfigurasi menggunakan LONG\_PAYLOAD\_FORMAT opsi.





## SESI PERSISTEN

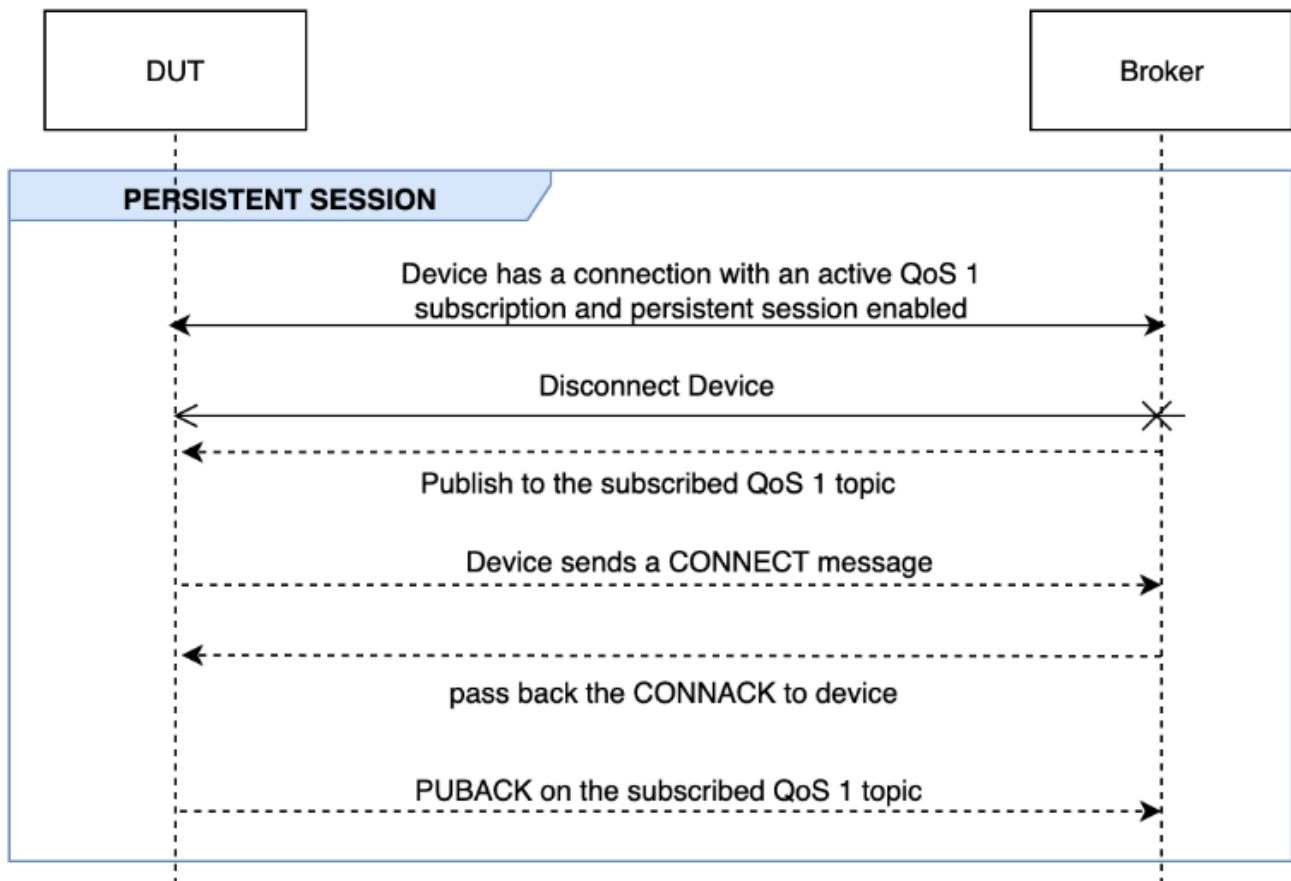
### **i** Note

Pilih skenario ini hanya jika perangkat Anda mampu melakukan langganan QoS 1 dan dapat mempertahankan sesi persisten.

Skenario ini memvalidasi perilaku perangkat dalam mempertahankan sesi persisten. Tes memvalidasi ketika kondisi berikut terpenuhi:

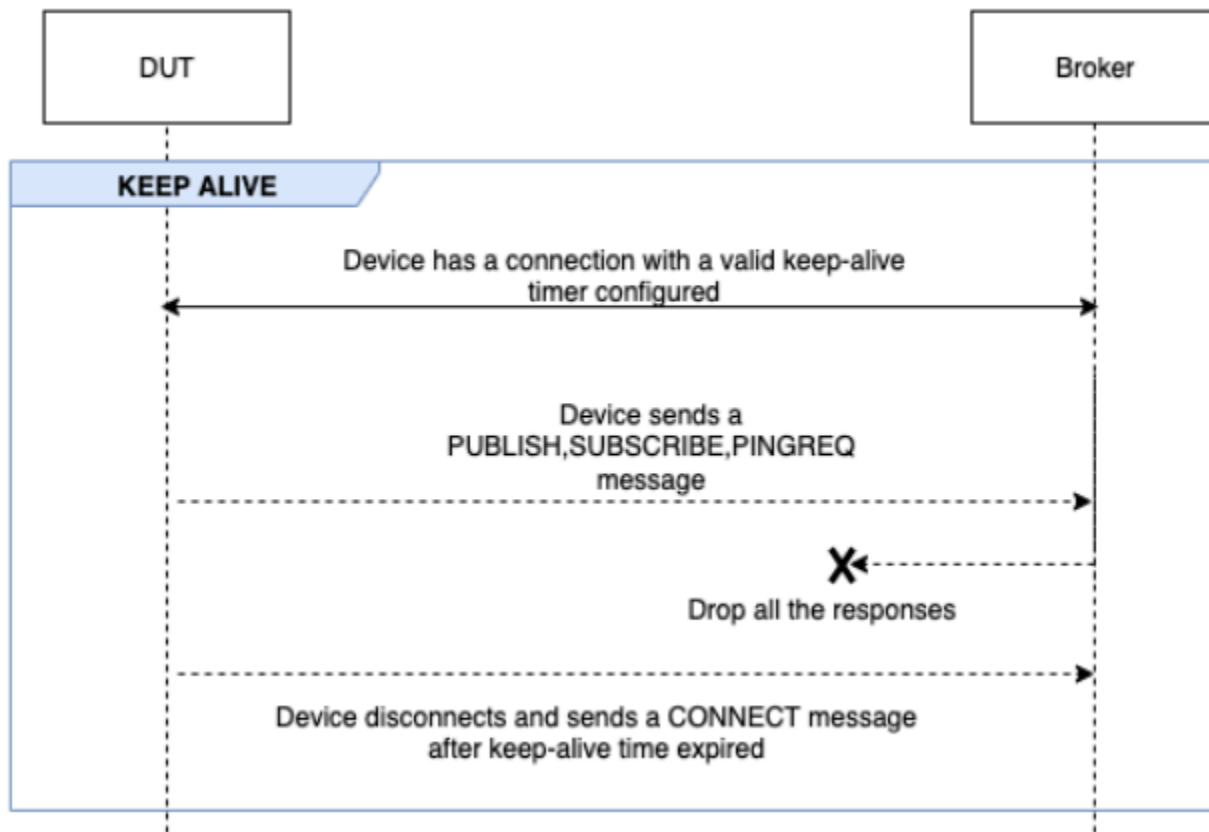
- Perangkat terhubung ke broker dengan langganan QoS 1 aktif dan sesi persisten diaktifkan.
- Perangkat berhasil terputus dari broker selama sesi berlangsung.
- Perangkat terhubung kembali ke broker dan melanjutkan langganan ke topik pemicunya tanpa secara eksplisit berlangganan kembali topik tersebut.
- Perangkat berhasil menerima pesan yang disimpan oleh broker untuk topik berlangganan dan berjalan seperti yang diharapkan.

Untuk informasi selengkapnya tentang Sesi AWS IoT Persisten, lihat [Menggunakan sesi persisten MQTT](#).



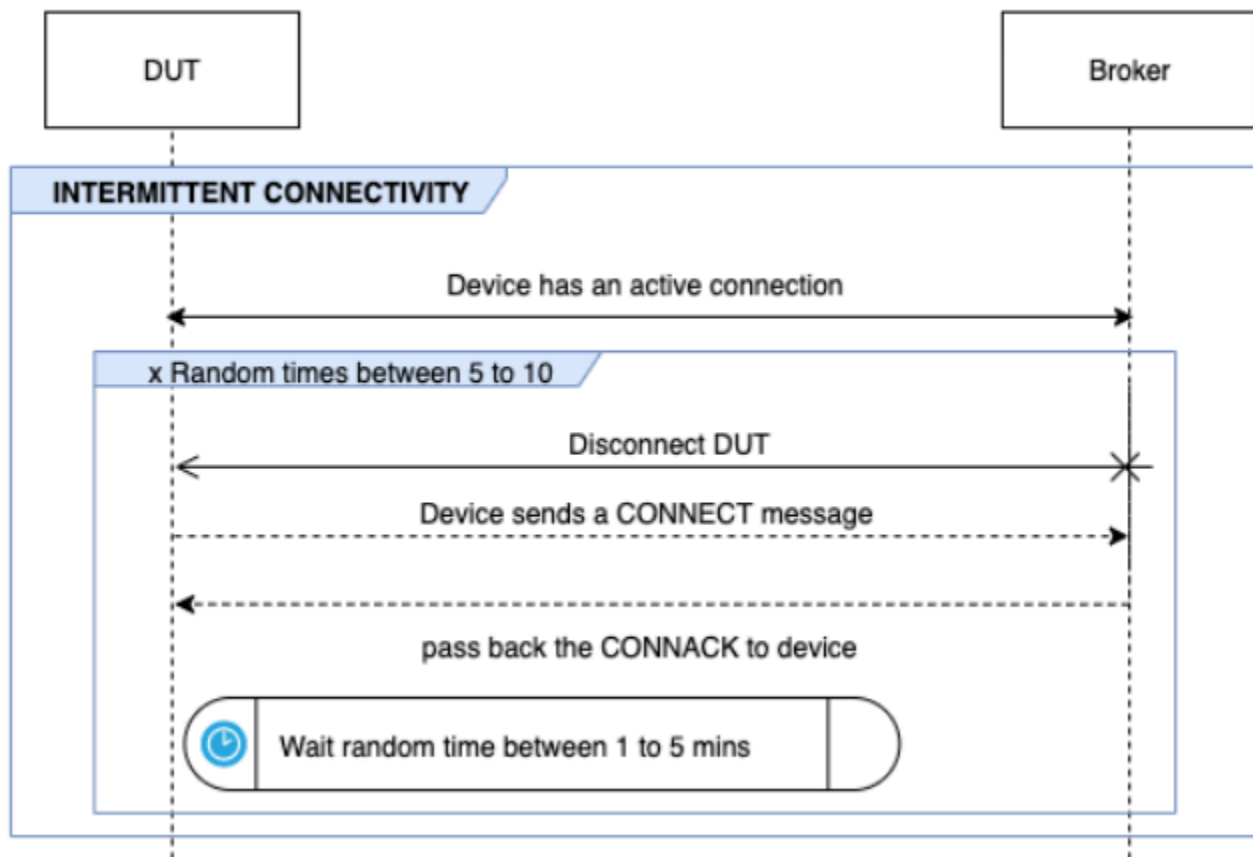
## TETAP HIDUP

Skenario ini memvalidasi jika perangkat berhasil terputus setelah tidak menerima respons ping dari broker. Koneksi harus memiliki timer keep-alive yang valid yang dikonfigurasi. Sebagai bagian dari tes ini, broker memblokir semua tanggapan yang dikirim untuk PUBLISH, SUBSCRIBE, dan PINGREQ pesan. Ini juga memvalidasi jika perangkat yang diuji memutuskan koneksi MQTT.



## KONEKTIVITAS INTERMITEN

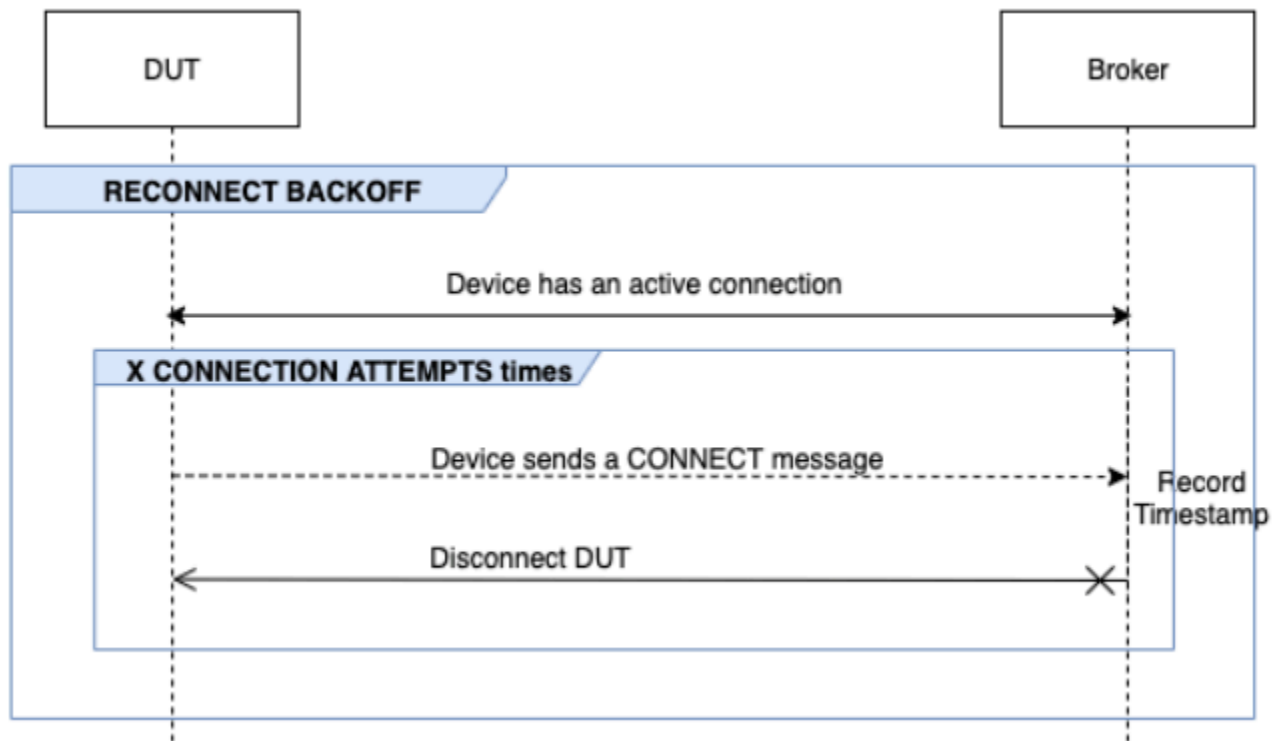
Skenario ini memvalidasi jika perangkat dapat terhubung kembali ke broker setelah broker memutus perangkat pada interval acak untuk jangka waktu acak.



## SAMBUNGAN KEMBALI BACKOFF

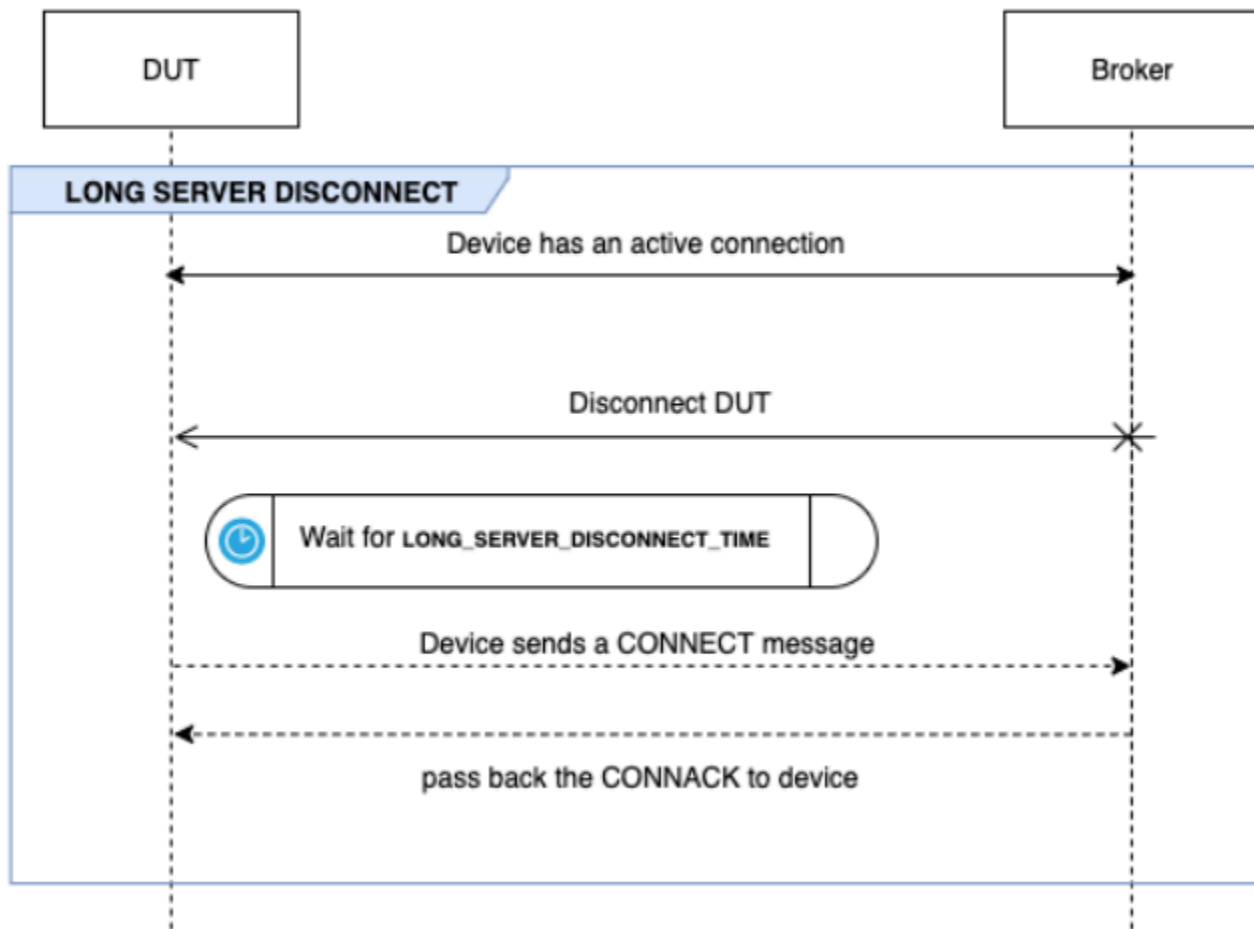
Skenario ini memvalidasi jika perangkat memiliki mekanisme backoff yang diterapkan ketika broker terputus darinya beberapa kali. Device Advisor melaporkan tipe backoff sebagai eksponensial, jitter, linier, atau konstan. Jumlah upaya backoff dapat dikonfigurasi menggunakan opsi. `BACKOFF_CONNECTION_ATTEMPTS` Nilai bawaannya adalah 5. Nilai dapat dikonfigurasi antara 5 dan 10.

Untuk lulus tes ini, kami sarankan untuk menerapkan mekanisme [Exponential Backoff And Jitter](#) pada perangkat yang diuji.



## PEMUTUSAN SERVER PANJANG

Skenario ini memvalidasi jika perangkat berhasil terhubung kembali setelah broker memutus perangkat untuk jangka waktu yang lama (hingga 120 menit). Waktu untuk pemutusan server dapat dikonfigurasi menggunakan `LONG_SERVER_DISCONNECT_TIME` opsi. Nilai defaultnya adalah 120 menit. Nilai ini dapat dikonfigurasi dari 30 hingga 120 menit.



## Waktu eksekusi tambahan

Waktu eksekusi tambahan adalah waktu pengujian menunggu setelah menyelesaikan semua tes di atas dan sebelum mengakhiri kasus uji. Pelanggan menggunakan periode waktu tambahan ini untuk memantau dan mencatat semua komunikasi antara perangkat dan broker. Waktu eksekusi tambahan dapat dikonfigurasi menggunakan `ADDITIONAL_EXECUTION_TIME` opsi. Secara default, opsi ini diatur ke 0 menit dan bisa 0 hingga 120 menit.

## Opsi konfigurasi uji durasi panjang MQTT

Semua opsi konfigurasi yang disediakan untuk uji durasi panjang MQTT adalah opsional. Pilihan berikut tersedia:

## OPERASI

Daftar operasi yang dilakukan perangkat, seperti CONNECT, PUBLISH dan SUBSCRIBE. Kasus uji menjalankan skenario berdasarkan operasi yang ditentukan. Operasi yang tidak ditentukan dianggap valid.

```
{
 "OPERATIONS": ["PUBLISH", "SUBSCRIBE"]
 //by default the test assumes device can CONNECT
}
```

## SKENARIO

Berdasarkan operasi yang dipilih, kasus uji menjalankan skenario untuk memvalidasi perilaku perangkat. Ada dua jenis skenario:

- Skenario Dasar adalah tes sederhana yang memvalidasi jika perangkat dapat melakukan operasi yang dipilih di atas sebagai bagian dari konfigurasi. Ini dipilih sebelumnya berdasarkan operasi yang ditentukan dalam konfigurasi. Tidak ada lagi input yang diperlukan dalam konfigurasi.
- Skenario Lanjutan adalah skenario yang lebih kompleks yang dilakukan terhadap perangkat untuk memvalidasi jika perangkat mengikuti praktik terbaik saat dipenuhi dengan kondisi dunia nyata. Ini opsional dan dapat diteruskan sebagai larik skenario ke input konfigurasi rangkaian pengujian.

```
{
 "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:

Waktu maksimum kasus uji akan menunggu semua tes dasar selesai. Nilai default adalah 60 menit. Nilai ini dapat dikonfigurasi dari 30 hingga 120 menit.

## LONG\_SERVER\_DISCONNECT\_TIME:

Waktu yang dibutuhkan untuk kasus uji untuk memutuskan sambungan dan menyambungkan kembali perangkat selama pengujian Long Server Disconnect. Nilai default adalah 60 menit. Nilai ini dapat dikonfigurasi dari 30 hingga 120 menit.

## TAMBAHAN\_EXECUTION\_TIME:

Mengkonfigurasi opsi ini menyediakan jendela waktu setelah semua tes selesai, untuk memantau peristiwa antara perangkat dan broker. Nilai defaultnya adalah 0 menit. Nilai ini dapat dikonfigurasi dari 0 hingga 120 menit.

## BACKOFF\_CONNECTION\_ATTEMPTS:

Opsi ini mengonfigurasi berapa kali perangkat terputus oleh kasus uji. Ini digunakan oleh tes Reconnect Backoff. Nilai defaultnya adalah 5 percobaan. Nilai ini dapat dikonfigurasi dari 5 hingga 10.

## FORMAT\_LONG\_PAYLOAD\_:

Format payload pesan yang diharapkan perangkat saat kasus uji diterbitkan ke topik QoS 1 yang dilanggan oleh perangkat.

## Definisi kasus uji 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"
 }
}
]
```

## Log ringkasan kasus uji durasi panjang MQTT

Kasus uji durasi panjang MQTT berjalan untuk durasi yang lebih lama daripada kasus uji biasa. Log ringkasan terpisah disediakan, yang mencantumkan peristiwa penting seperti koneksi perangkat, mempublikasikan, dan berlangganan selama dijalankan. Rincian termasuk apa yang diuji, apa yang tidak diuji dan apa yang gagal. Di akhir log, pengujian mencakup ringkasan semua peristiwa yang terjadi selama uji kasus dijalankan. Hal ini mencakup:

- Keep Alive timer dikonfigurasi pada perangkat.
- Bendera sesi persisten dikonfigurasi pada perangkat.
- Jumlah koneksi perangkat selama uji coba.
- Jenis backoff koneksi ulang perangkat, jika divalidasi untuk uji backoff sambungkan kembali.
- Topik yang dipublikasikan perangkat, selama uji kasus dijalankan.
- Topik yang dilangganani perangkat, selama uji kasus dijalankan.

# AWS IoT Device Management Katalog Paket Perangkat Lunak

Dengan AWS IoT Device Management Software Package Catalog, Anda dapat menyimpan inventaris paket perangkat lunak dan versinya. [Anda dapat mengaitkan versi paket dengan hal-hal individual dan grup hal AWS IoT dinamis, dan menerapkannya melalui proses atau AWS IoT pekerjaan internal.](#)

Paket perangkat lunak berisi satu atau lebih versi paket, yang merupakan kumpulan file yang dapat digunakan sebagai satu unit. Versi Package dapat berisi firmware, pembaruan sistem operasi, aplikasi perangkat, konfigurasi, dan patch keamanan. Saat perangkat lunak berkembang dari waktu ke waktu, Anda dapat membuat versi paket baru dan menyebarkannya ke armada Anda.

Hub paket AWS IoT perangkat lunak terletak di dalam AWS IoT Core. Anda dapat menggunakan hub untuk mendaftarkan dan memelihara inventaris dan metadata paket perangkat lunak Anda secara terpusat, yang membuat katalog paket perangkat lunak dan versinya. Anda dapat memilih untuk mengelompokkan perangkat berdasarkan paket perangkat lunak dan versi paket yang digunakan pada perangkat. Fitur ini memberikan kesempatan untuk menyimpan inventaris paket sisi perangkat sebagai perangkat bayangan, asosiasi, dan grup bernama berdasarkan versi, dan memvisualisasikan distribusi versi paket di seluruh armada dengan menggunakan metrik armada.

Jika Anda memiliki sistem penyebaran perangkat lunak internal yang dibuat, Anda dapat terus menggunakan proses itu untuk menyebarkan versi paket Anda. Jika Anda tidak memiliki proses penyebaran yang ditetapkan atau jika Anda mau, sebaiknya gunakan [AWS IoT lowongan](#) untuk menggunakan fitur di Katalog Paket Perangkat Lunak. Untuk informasi lebih lanjut, lihat [Mempersiapkan AWS IoT pekerjaan](#).

Bab ini berisi bagian-bagian berikut:

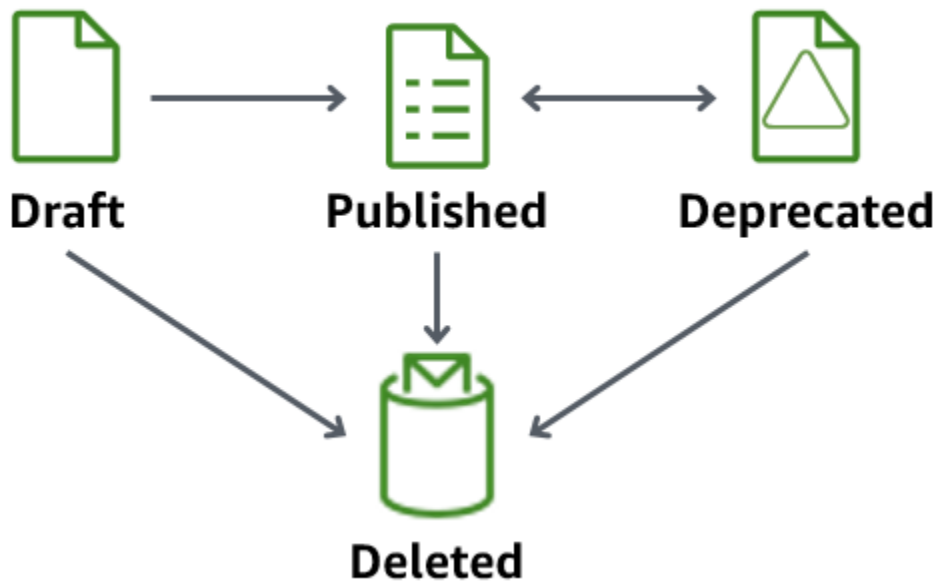
- [Mempersiapkan untuk menggunakan Katalog Paket Perangkat Lunak](#)
- [Mempersiapkan keamanan](#)
- [Mempersiapkan pengindeksan armada](#)
- [Mempersiapkan AWS IoT Pekerjaan](#)
- [Memulai dengan Software Package Catalog](#)

## Mempersiapkan untuk menggunakan Katalog Paket Perangkat Lunak

Bagian berikut memberikan ikhtisar siklus hidup versi paket dan informasi untuk menggunakan Katalog AWS IoT Device Management Paket Perangkat Lunak.

## Siklus hidup versi paket

Versi paket dapat berkembang melalui status siklus hidup berikut: `draft`, `published` dan `deprecated`. Bisa juga `deleted`.



- **Draft**

Saat Anda membuat versi paket, itu dalam `draft` keadaan. Keadaan ini menunjukkan bahwa paket perangkat lunak sedang dipersiapkan atau tidak lengkap.

Sementara versi paket dalam status ini, Anda tidak dapat menerapkannya. Anda dapat mengedit deskripsi, atribut, dan tag versi paket.

Anda dapat mentransisikan versi paket yang berada dalam `draft` status ke `published` atau menjadi `deleted` dengan menggunakan konsol, atau dengan mengeluarkan operasi API [UpdatePackageVersi](#) atau [DeletePackageVersi](#).

- **Diterbitkan**

Ketika versi paket Anda siap untuk digunakan, transisi versi paket ke `published` status. Saat dalam keadaan ini, Anda dapat memilih untuk mengidentifikasi versi paket sebagai versi default dengan mengedit paket perangkat lunak di konsol atau melalui operasi [UpdatePackageAPI](#). Dalam keadaan ini, Anda hanya dapat mengedit deskripsi dan tag.

Anda dapat mentransisikan versi paket yang berada dalam `published` status ke `deprecated` atau menjadi `deleted` dengan menggunakan konsol, atau mengeluarkan operasi API [UpdatePackageVersi](#) atau [DeletePackageVersi](#).

- Usang


Jika versi paket baru tersedia, Anda dapat mentransisikan versi paket sebelumnya ke `deprecated`. Anda masih dapat menerapkan pekerjaan dengan versi paket yang tidak digunakan lagi. Anda juga dapat menamai versi paket usang sebagai versi default, dan hanya mengedit deskripsi dan tag.

Pertimbangkan untuk mentransisikan versi paket ke `deprecated` saat versi sudah usang, tetapi Anda masih memiliki perangkat di lapangan menggunakan versi yang lebih lama atau perlu mempertahankannya karena ketergantungan run-time.

Anda dapat mentransisikan versi paket yang berada dalam `deprecated` status ke `published` atau menjadi `deleted` dengan menggunakan konsol, atau mengeluarkan operasi API [UpdatePackageVersi](#) atau [DeletePackageVersi](#).

- Dihapus

Jika Anda tidak lagi bermaksud menggunakan versi paket, Anda dapat menghapusnya dengan menggunakan konsol atau mengeluarkan operasi API [DeletePackageVersi](#).

 Note

Jika Anda menghapus versi paket saat ada pekerjaan tertunda yang merujuknya, Anda akan menerima pesan kesalahan saat pekerjaan berhasil diselesaikan dan mencoba memperbarui bayangan bernama yang dicadangkan.

Jika versi paket perangkat lunak yang ingin Anda hapus dinamai sebagai versi paket default, Anda harus terlebih dahulu memperbarui paket untuk memberi nama versi lain sebagai default atau membiarkan bidang tidak disebutkan namanya. Anda dapat melakukan ini dengan menggunakan konsol atau operasi API [UpdatePackageVersi](#). (Untuk menghapus versi paket bernama apa pun sebagai default, setel `DefaultVersion` parameter [yang tidak disetel](#) ke `true` saat Anda mengeluarkan operasi [UpdatePackageAPI](#)).

Jika Anda menghapus paket perangkat lunak melalui konsol, itu akan menghapus semua versi paket yang terkait dengan paket itu, kecuali satu dinamai sebagai versi default.

## Konvensi penamaan versi Package

Saat Anda memberi nama versi paket, penting untuk merencanakan dan menerapkan strategi penamaan logis sehingga Anda dan orang lain dapat dengan mudah mengidentifikasi versi paket terbaru dan perkembangan versi. Anda harus memberikan nama versi saat membuat versi paket, tetapi strategi dan formatnya sebagian besar tergantung pada kasus bisnis Anda.

Sebagai praktik terbaik, kami sarankan menggunakan format Versi [SemVer](#) Semantik. Misalnya, 1.2.3 di mana 1 versi utama untuk perubahan yang tidak kompatibel secara fungsional, 2 versi utama untuk perubahan yang kompatibel secara fungsional, dan 3 merupakan versi tambalan (untuk perbaikan bug). Untuk informasi selengkapnya, lihat [Versioning Semantik 2.0.0](#). Untuk informasi selengkapnya tentang persyaratan nama versi paket, lihat [versionName](#) di panduan referensi AWS IoT API.

## Versi default

Menyetel versi sebagai default adalah opsional. Anda dapat menambah atau menghapus versi paket default. Anda juga dapat menerapkan versi paket yang tidak dinamai sebagai versi default.

Saat Anda membuat versi paket, itu ditempatkan dalam draft keadaan dan tidak dapat dinamai sebagai versi default sampai Anda mentransisikan versi paket ke diterbitkan. Software Package Catalog tidak secara otomatis memilih versi sebagai default atau memperbarui versi paket yang lebih baru sebagai default. Anda harus dengan sengaja memberi nama versi paket yang Anda pilih melalui konsol atau dengan mengeluarkan operasi API [UpdatePackageVersi](#).

## Atribut versi

Atribut versi dan nilainya menyimpan informasi penting tentang versi paket Anda. Kami menyarankan Anda menentukan atribut tujuan umum untuk versi paket atau paket. Misalnya, Anda dapat membuat pasangan nama-nilai untuk platform, arsitektur, sistem operasi, tanggal rilis, penulis, atau URL Amazon S3.

Saat Anda membuat AWS IoT pekerjaan dengan dokumen pekerjaan, Anda juga dapat memilih untuk menggunakan variabel substitusi (`$parameter`) yang mengacu pada nilai atribut. Untuk informasi selengkapnya, lihat [Mempersiapkan AWS IoT Pekerjaan](#).

Atribut versi yang digunakan dalam versi paket tidak akan secara otomatis ditambahkan ke bayangan bernama yang dicadangkan dan tidak dapat diindeks atau ditanyakan melalui Pengindeksan Armada secara langsung. Untuk mengindeks atau menanyakan atribut versi paket melalui Fleet Indexing, Anda dapat mengisi atribut `version` dalam bayangan bernama cadangan.

Kami merekomendasikan bahwa parameter atribut versi dalam properti yang dilaporkan perangkat tangkapan bayangan bernama yang dicadangkan, seperti sistem operasi dan waktu instalasi. Mereka juga dapat diindeks dan ditanyakan melalui Pengindeksan Armada.

Atribut versi tidak diperlukan untuk mengikuti konvensi penamaan tertentu. Anda dapat membuat pasangan nama-nilai untuk memenuhi kebutuhan bisnis Anda. Ukuran gabungan semua atribut pada versi paket dibatasi hingga 3KB. Untuk informasi selengkapnya, lihat [Paket perangkat lunak Katalog Paket Perangkat Lunak dan batas versi paket](#).

## Mengaktifkan pengindeksan AWS IoT armada

Anda harus mengaktifkan pengindeksan armada untuk Katalog Paket Perangkat Lunak untuk membuat atau memperbarui paket perangkat lunak dan versi paket. Pengindeksan armada memberikan dukungan yang memungkinkan AWS IoT hal-hal dikelompokkan melalui grup benda dinamis yang difilter berdasarkan versi. Misalnya, pengindeksan armada dapat mengidentifikasi hal-hal yang telah atau tidak memiliki versi paket tertentu yang diinstal, tidak memiliki versi paket yang diinstal, atau cocok dengan pasangan nama-nilai tertentu. Terakhir, pengindeksan armada menyediakan metrik standar dan kustom yang dapat Anda gunakan untuk mendapatkan wawasan tentang keadaan armada Anda. Untuk informasi selengkapnya, lihat [Mempersiapkan pengindeksan armada](#).

### Note

Mengaktifkan pengindeksan armada untuk Katalog Paket Perangkat Lunak menimbulkan biaya layanan standar. Untuk informasi lebih lanjut, lihat [AWS IoT Device Management, Harga](#).

## Cadangan bernama bayangan

Bayangan bernama yang dicadangkan `$package`, mencerminkan keadaan paket perangkat lunak yang diinstal perangkat dan versi paket. Pengindeksan armada menggunakan bayangan bernama cadangan sebagai sumber data untuk membuat metrik standar dan kustom sehingga Anda dapat menanyakan status armada Anda. Untuk informasi selengkapnya, lihat [Mempersiapkan pengindeksan armada](#).

Bayangan bernama cadangan mirip dengan [bayangan bernama](#) dengan pengecualian bahwa namanya telah ditentukan sebelumnya dan Anda tidak dapat mengubahnya. Selain itu, bayangan

bernama yang dicadangkan tidak diperbarui dengan metadata, dan hanya menggunakan kata kunci dan kata kunci `version`. `attributes`

Permintaan pembaruan yang menyertakan kata kunci `description`, seperti, akan menerima respons kesalahan di bawah `rejected` topik. Untuk informasi selengkapnya, lihat [Pesan galat Device Shadow](#).

Ini dapat dibuat saat Anda membuat AWS IoT sesuatu melalui konsol, ketika AWS IoT pekerjaan berhasil menyelesaikan dan memperbarui bayangan, dan jika Anda mengeluarkan operasi [UpdateThingShadow](#) API. Untuk informasi selengkapnya, lihat [UpdateThingShadow](#) di panduan AWS IoT Core pengembang.

### Note

Mengindeks bayangan bernama cadangan tidak dihitung terhadap jumlah bayangan bernama yang dapat diindeks oleh pengindeksan armada. Untuk informasi selengkapnya, lihat [batas pengindeksan AWS IoT Device Management armada dan kuota](#). Selain itu, jika Anda memilih untuk meminta AWS IoT lowongan memperbarui bayangan bernama yang dicadangkan saat pekerjaan berhasil diselesaikan, panggilan API dihitung untuk Device Shadow dan operasi registri Anda dan dapat dikenakan biaya. Untuk informasi selengkapnya, lihat [batas AWS IoT Device Management pekerjaan dan kuota serta](#) tipe data [IndexingFilter](#) API.

## Struktur `$package` bayangan

Bayangan bernama cadangan berisi yang berikut:

```
{
 "state": {
 "reported": {
 "<packageName>": {
 "version": "",
 "attributes": {
 }
 }
 }
 },
 "version" : 1
 "timestamp" : 1672531201
}
```

Properti bayangan diperbarui dengan informasi berikut:

- `<packageName>`: Nama paket perangkat lunak yang diinstal, yang diperbarui dengan parameter [PackageName](#).
- `version`: Nama versi paket yang diinstal, yang diperbarui dengan parameter [versionName](#).
- `attributes`: Metadata opsional disimpan oleh perangkat dan diindeks oleh pengindeksan Armada. Hal ini memungkinkan pelanggan untuk query indeks mereka berdasarkan data yang disimpan.
- `version`: Nomor versi bayangan. Ini secara otomatis bertambah setiap kali bayangan diperbarui dan dimulai pada 1.
- `timestamp`: Menunjukkan kapan bayangan terakhir diperbarui dan direkam dalam [waktu Unix](#).

Untuk informasi selengkapnya tentang format dan perilaku bayangan bernama, lihat [Urutan Layanan AWS IoT Device Shadow pesan](#).

## Menghapus paket perangkat lunak dan versi paketnya

Sebelum Anda menghapus paket perangkat lunak, lakukan hal berikut:

- Konfirmasikan bahwa paket dan versinya tidak digunakan secara aktif.
- Hapus semua versi terkait terlebih dahulu. Jika salah satu versi ditetapkan sebagai versi default, Anda harus menghapus versi default bernama dari paket. Karena menunjuk versi default adalah opsional, tidak ada konflik menghapusnya. Untuk menghapus versi default dari paket perangkat lunak, edit paket melalui konsol atau gunakan operasi [UpdatePackageVersionAPI](#).

Selama tidak ada versi paket default bernama, Anda dapat menggunakan konsol untuk menghapus paket perangkat lunak dan semua versi paketnya juga akan dihapus. Jika Anda menggunakan panggilan API untuk menghapus paket perangkat lunak, Anda harus menghapus versi paket terlebih dahulu dan kemudian paket perangkat lunak.

## Mempersiapkan keamanan

Bagian ini membahas persyaratan keamanan utama untuk Katalog Paket AWS IoT Device Management Perangkat Lunak.



## Otentikasi berbasis sumber daya

Software Package Catalog menggunakan otorisasi berbasis sumber daya untuk memberikan keamanan tambahan saat memperbarui perangkat lunak pada armada Anda. Ini berarti Anda harus membuat kebijakan AWS Identity and Access Management (IAM) yang memberikan hak untuk melakukan `create`, `read`, `update`, `delete`, dan `list` tindakan untuk paket perangkat lunak dan versi paket, dan mereferensikan paket perangkat lunak dan versi paket tertentu yang ingin Anda terapkan di bagian tersebut. Resources Anda juga memerlukan hak-hak ini sehingga Anda dapat memperbarui [bayangan bernama yang dilindungi undang-undang](#). Anda mereferensikan paket perangkat lunak dan versi paket dengan menyertakan Nama Sumber Daya Amazon (ARN) untuk setiap entitas.

### Note

Jika Anda bermaksud kebijakan memberikan hak untuk panggilan API versi paket (seperti [CreatePackageVersi](#), [UpdatePackageVersi](#), [DeletePackageVersi](#)), maka Anda harus menyertakan paket perangkat lunak dan ARN versi paket dalam kebijakan. Jika Anda bermaksud kebijakan untuk memberikan hak untuk panggilan API paket perangkat lunak (seperti [CreatePackage](#), [UpdatePackage](#), dan [DeletePackage](#)) maka Anda harus menyertakan hanya paket perangkat lunak ARN dalam kebijakan.

Struktur paket perangkat lunak dan versi paket ARN sebagai berikut:

- Paket perangkat lunak:  
`arn:aws:iot:<region>:<accountID>:package/<packageName>/package`
- Versi Package: `arn:aws:iot:<region>:<accountID>:package/<packageName>/version/<versionName>`

### Note

Ada hak terkait lainnya yang mungkin Anda sertakan dalam kebijakan ini. Misalnya, Anda mungkin menyertakan ARN untuk `job`, `thinggroup`, dan `jobtemplate`. Untuk informasi selengkapnya dan daftar lengkap opsi kebijakan, lihat [Mengamankan pengguna dan perangkat dengan AWS IoT Lowongan](#).

Misalnya, jika Anda memiliki paket perangkat lunak dan versi paket yang diberi nama sebagai berikut:

- AWS IoT hal: myThing
- Nama Package: samplePackage
- Versi 1.0.0

Kebijakan ini mungkin terlihat seperti contoh berikut:

```
{
 "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 Hak pekerjaan untuk menyebarkan versi paket

Untuk tujuan keamanan, penting bagi Anda untuk memberikan hak untuk menyebarkan paket dan versi paket, dan memberi nama paket dan versi paket tertentu yang diizinkan untuk diterapkan. Untuk melakukannya, Anda membuat peran dan kebijakan IAM yang memberikan izin untuk menerapkan pekerjaan dengan versi paket. Kebijakan harus menentukan versi paket tujuan sebagai sumber daya.

## Kebijakan IAM

Kebijakan IAM memberikan hak untuk membuat pekerjaan yang mencakup paket dan versi yang disebutkan di bagian tersebut `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

Jika Anda ingin menerapkan pekerjaan yang menghapus instalasi paket perangkat lunak dan versi paket, Anda harus mengotorisasi ARN di mana versi paket berada, seperti berikut \$null ini:

```
arn:aws:iot:<regionCode>:111122223333:package/<packageName>/version/$null
```

## AWS IoT Hak pekerjaan untuk memperbarui bayangan bernama yang dilindungi undang-undang

Untuk mengizinkan lowongan memperbarui bayangan nama yang dicadangkan saat pekerjaan berhasil diselesaikan, Anda harus membuat peran dan kebijakan IAM. Ada dua cara Anda dapat

melakukan ini di AWS IoT konsol. Yang pertama adalah ketika Anda membuat paket perangkat lunak di konsol. Jika Anda melihat kotak dialog Aktifkan dependensi untuk manajemen paket, Anda dapat memilih untuk menggunakan peran yang ada atau membuat peran baru. Atau, di AWS IoT konsol, pilih Pengaturan, pilih Kelola pengindeksan, lalu Kelola pengindeksan untuk paket dan versi perangkat.

### Note

Jika Anda memilih agar layanan AWS IoT Job memperbarui bayangan bernama yang dicadangkan saat pekerjaan berhasil diselesaikan, panggilan API akan dihitung untuk Device Shadow dan operasi registri Anda dan dapat dikenakan biaya. Untuk informasi selengkapnya, lihat [harga AWS IoT Core](#).

Saat Anda menggunakan opsi Buat peran, nama peran yang dihasilkan dimulai dengan `aws-iot-role-update-shadows` dan berisi kebijakan berikut:

Menyiapkan peran

Izin

Kebijakan izin memberikan hak untuk melakukan kueri dan memperbarui bayangan benda. `$packageParameter` dalam ARN sumber daya menargetkan bayangan bernama cadangan.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": "iot:DescribeEndpoint",
 "Resource": ""
 },
 {
 "Effect": "Allow",
 "Action": [
 "iot:GetThingShadow",
 "iot:UpdateThingShadow"
],
 "Resource": [
 "arn:aws:iot:<regionCode>:111122223333:thing/<thingName>/$package"
]
 }
]
}
```

```

 }
]
}

```

## Hubungan kepercayaan

Selain kebijakan izin, peran memerlukan hubungan kepercayaan AWS IoT Core sehingga entitas dapat mengambil peran dan memperbarui bayangan bernama yang dicadangkan.

```

{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Principal": {
 "Service": "iot.amazonaws.com"
 },
 "Action": "sts:AssumeRole"
 }
]
}

```

## Menyiapkan kebijakan pengguna

iam: izin PassRole

Terakhir, Anda harus memiliki izin untuk meneruskan peran AWS IoT Core saat Anda memanggil operasi [UpdatePackageConfiguration](#) API.

```

{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "iam:PassRole",
 "iot:UpdatePackageConfiguration"
],
 "Resource": "arn:aws:iam::111122223333:role/<roleName>"
 }
]
}

```

```
}
```

## AWS IoT Izin pekerjaan untuk mengunduh dari Amazon S3

Dokumen pekerjaan disimpan di Amazon S3. Anda merujuk ke file ini ketika Anda mengirim melalui AWS IoT Jobs. Anda harus memberikan AWS IoT Jobs dengan hak untuk men-download file (`s3:GetObject`). Anda juga harus mengatur hubungan kepercayaan antara Amazon S3 dan AWS IoT Jobs. Untuk petunjuk untuk membuat kebijakan ini, lihat [URL yang telah ditandatangani sebelumnya](#) di [Mengelola Pekerjaan](#).

## Mempersiapkan pengindeksan armada

Dengan pengindeksan AWS IoT armada, Anda dapat mencari dan mengumpulkan data dengan menggunakan `shadow () $package` bernama `reserved`. Anda juga dapat mengelompokkan AWS IoT hal-hal dengan menanyakan [grup hal yang dinamis Cadangan bernama bayangan dan dinamis](#). Misalnya, Anda dapat menemukan informasi tentang AWS IoT hal-hal mana yang menggunakan versi paket tertentu, tidak memiliki versi paket tertentu yang diinstal, atau tidak menginstal versi paket apa pun. Anda dapat memperoleh wawasan lebih lanjut dengan menggabungkan atribut. Misalnya, mengidentifikasi hal-hal yang memiliki versi tertentu dan jenis hal tertentu (seperti versi 1.0.0 dan jenis `pump_sensor`). Untuk informasi selengkapnya, lihat [Pengindeksan armada](#).

## Mengatur `$package` bayangan sebagai sumber data

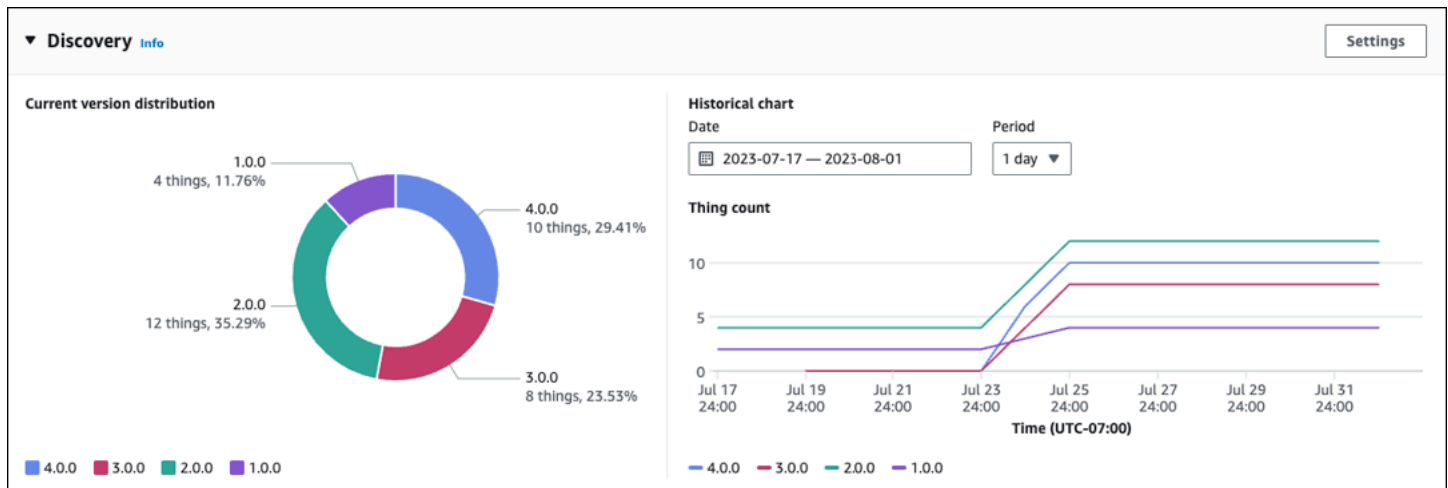
Untuk menggunakan pengindeksan armada dengan Katalog Paket Perangkat Lunak, Anda harus mengaktifkan pengindeksan armada, mengatur bayangan bernama sebagai sumber data, dan mendefinisikan `$package` sebagai filter bayangan bernama. Jika Anda belum mengaktifkan pengindeksan armada, Anda dapat mengaktifkannya dalam proses ini. Dari [AWS IoT Core](#) konsol, buka Pengaturan, pilih Kelola pengindeksan, lalu Tambahkan bayangan bernama, Tambahkan paket dan versi perangkat lunak perangkat, dan Perbarui. Untuk informasi selengkapnya, lihat [Mengelola pengindeksan hal](#).

Sebagai alternatif, Anda dapat mengaktifkan pengindeksan armada saat Anda membuat paket pertama Anda. Saat kotak dialog Aktifkan dependensi untuk manajemen paket muncul, pilih opsi untuk menambahkan paket perangkat lunak dan versi sebagai sumber data ke pengindeksan armada. Dengan memilih opsi ini, Anda juga mengaktifkan pengindeksan armada.

### Note

Mengaktifkan pengindeksan armada untuk Katalog Paket Perangkat Lunak menimbulkan biaya layanan standar. Untuk informasi lebih lanjut, lihat [AWS IoT Device Management, Harga](#).

## Metrik ditampilkan di konsol



Pada halaman detail paket perangkat lunak AWS IoT konsol, panel Discovery menampilkan metrik standar yang dicerna melalui bayangan. \$package

- Bagan distribusi versi saat ini menunjukkan jumlah perangkat dan persentase untuk 10 versi paket terbaru yang terkait dengan AWS IoT sesuatu dari semua perangkat yang terkait dengan paket perangkat lunak ini. Catatan: Jika paket perangkat lunak memiliki lebih banyak versi paket daripada yang berlabel dalam bagan, Anda dapat menemukannya dikelompokkan dalam Lainnya.
- Bagan Historis menunjukkan jumlah perangkat yang terkait dengan versi paket yang dipilih selama periode waktu tertentu. Bagan awalnya kosong sampai Anda memilih hingga 5 versi paket dan menentukan rentang tanggal dan interval waktu. Untuk memilih parameter bagan, pilih Pengaturan. Data yang ditampilkan dalam bagan Historis mungkin berbeda dari bagan distribusi versi saat ini karena perbedaan jumlah versi paket yang ditampilkan dan juga karena Anda dapat memilih versi paket mana yang akan dianalisis dalam bagan Historis. Catatan: Saat Anda memilih versi paket untuk divisualisasikan, versi tersebut dihitung dalam jumlah maksimum batas metrik armada. Untuk informasi selengkapnya, lihat [Batas dan kuota pengindeksan armada](#).

Untuk metode lain untuk mendapatkan wawasan tentang pengumpulan distribusi versi paket, lihat [Mengumpulkan distribusi versi paket melalui `getBucketsAggregation`](#).

## Pola kueri

Pengindeksan armada dengan Katalog Paket Perangkat Lunak menggunakan sebagian besar fitur yang didukung (misalnya, istilah dan frasa dan bidang pencarian) yang merupakan standar untuk pengindeksan armada. Pengecualiannya adalah bahwa `range` kueri `comparison` dan tidak tersedia untuk `version` kunci shadow (`$package`) bernama yang dicadangkan. Namun, kueri ini tersedia untuk `attributes` kuncinya. Untuk informasi selengkapnya, lihat [Sintaks kueri](#).

## Contoh data

Catatan: untuk informasi tentang bayangan bernama yang dicadangkan dan strukturnya, lihat [Bayangan bernama cadangan](#).

Dalam contoh ini, perangkat pertama diberi nama `Anything` dan memiliki paket berikut diinstal:

- Paket perangkat lunak: `SamplePackage`

Versi Package: `1.0.0`

Package ID: `1111`

Bayangan terlihat sebagai berikut:

```
{
 "state": {
 "reported": {
 "SamplePackage": {
 "version": "1.0.0",
 "attributes": {
 "s3UrlForSamplePackage": "https://EXAMPLEBUCKET.s3.us-
west-2.amazonaws.com/exampleCodeFile1",
 "packageID": "1111"
 }
 }
 }
 }
}
```



Perangkat kedua diberi nama `AnotherThing` dan memiliki paket berikut diinstal:

- Paket perangkat lunak: `SamplePackage`

Versi Package: `1.0.0`

Package ID: `1111`

- Paket perangkat lunak: `OtherPackage`

Versi Package: `1.2.5`

Package ID: `2222`

Bayangan terlihat sebagai berikut:

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

## Kueri Sampel

Tabel berikut mencantumkan contoh kueri berdasarkan contoh bayangan perangkat untuk `AnyThing` dan `AnotherThing`. Untuk informasi selengkapnya, lihat [Contoh kueri hal](#).

## Versi terbaru dari AWS IoT Device Tester untuk FreeRTOS

| Informasi yang diminta                                                                    | Kueri                                                                                                                   | Hasil                |
|-------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------|----------------------|
| Hal-hal yang memiliki versi paket tertentu diinstal                                       | <code>shadow.name.\$package.reported.SamplePackage.version:1.0.0</code>                                                 | Anything, OtherThing |
| Hal-hal yang tidak memiliki versi paket tertentu diinstal                                 | <code>NOT shadow.name.\$package.reported.OtherPackage.version:1.2.5</code>                                              | Anything             |
| Perangkat apa pun yang menggunakan versi paket yang ID pakatnya lebih besar dari 1500     | <code>shadow.name.\$package.reported.*.attributes.packageID&gt;1500"</code>                                             | OtherThing           |
| Hal-hal yang memiliki paket tertentu diinstal dan memiliki lebih dari satu paket diinstal | <code>shadow.name.\$package.reported.SamplePackage.version:1.0.0 AND shadow.name.\$package.reported.totalCount:2</code> | OtherThing           |

## Mengumpulkan distribusi versi paket melalui `getBucketsAggregation`

Selain panel Discovery dalam AWS IoT konsol, Anda juga bisa mendapatkan informasi distribusi versi paket dengan menggunakan operasi [GetBucketsAggregation](#) API. Untuk mendapatkan informasi distribusi versi paket, Anda harus melakukan hal berikut:

- Tentukan bidang khusus dalam pengindeksan armada untuk setiap paket perangkat lunak. Catatan: Membuat bidang kustom dihitung terhadap [kuota layanan pengindeksan AWS IoT armada](#).
- Format bidang kustom sebagai berikut:

```
shadow.name.$package.reported.<packageName>.version
```

Untuk informasi selengkapnya, lihat bagian [Bidang kustom](#) dalam pengindeksan AWS IoT armada.

## Mempersiapkan AWS IoT Pekerjaan

AWS IoT Device Management Software Package Catalog memperluas AWS IoT Jobs melalui parameter substitusi, dan integrasi dengan pengindeksan AWS IoT armada, grup benda dinamis, dan AWS IoT benda yang dicadangkan bernama shadow.

### Note

Untuk menggunakan semua fungsionalitas yang ditawarkan Katalog Paket Perangkat Lunak, Anda harus membuat peran dan kebijakan AWS Identity and Access Management (IAM) ini: [Hak AWS IoT pekerjaan untuk menyebarkan versi paket](#) dan [hak AWS IoT Pekerjaan untuk memperbarui bayangan bernama yang dilindungi undang-undang](#). Untuk informasi selengkapnya, lihat [Mempersiapkan keamanan](#).

## Parameter substitusi untuk pekerjaan AWS IoT

Anda dapat menggunakan parameter substitusi sebagai pengganti dalam dokumen pekerjaan Anda AWS IoT . Ketika layanan pekerjaan menemukan parameter substitusi, itu mengarahkan pekerjaan ke atribut versi perangkat lunak bernama untuk nilai parameter. Anda dapat menggunakan proses ini untuk membuat dokumen pekerjaan tunggal dan meneruskan metadata ke dalam pekerjaan melalui atribut tujuan umum. Misalnya, Anda dapat meneruskan URL Amazon Simple Storage Service (Amazon S3), paket perangkat lunak Nama Sumber Daya Amazon (ARN), atau tanda tangan ke dalam dokumen pekerjaan melalui atribut versi paket.

Parameter substitusi harus diformat dalam dokumen pekerjaan sebagai berikut:

```
${aws:iot:package:<packageName>:version:<versionName>:attributes:<anyAttributeName>}
```

Dalam contoh ini, ada paket perangkat lunak bernama `samplePackage`, dan memiliki versi paket bernama `2.1.5` yang memiliki atribut berikut:

- nama:s3URL, nilai: `https://EXAMPLEBUCKET.s3.us-west-2.amazonaws.com/exampleCodeFile`
  - Atribut ini mengidentifikasi lokasi file kode yang disimpan dalam Amazon S3.
- nama:signature, nilai: `aaaaabbbbccccddddddeeeeffffggggghhhhhiiiiijjjj`

- Atribut ini memberikan nilai tanda tangan kode yang dibutuhkan perangkat sebagai tindakan pengamanan. Untuk informasi selengkapnya, lihat [Penandatanganan Kode untuk lowongan kerja](#). Catatan: Atribut ini adalah contoh dan tidak diperlukan sebagai bagian dari Katalog Paket Perangkat Lunak atau pekerjaan.

Untuk `downloads`, parameter dokumen pekerjaan ditulis sebagai berikut:

```
{
 "samplePackage": "${aws:iot:package:samplePackage1:version:2.1.5:attributes:s3URL}"
}
```

Untuk `signature`, parameter dokumen pekerjaan ditulis sebagai berikut:

```
{
 "samplePackage": "${aws:iot:package:samplePackage1:version:2.1.5:attributes:signature}"
}
```

Dokumen pekerjaan lengkap ditulis sebagai berikut:

```
{
 ...
 "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}"
]
 }
}
```

Setelah substitusi dilakukan, dokumen pekerjaan berikut dikerahkan ke perangkat:

```
{
 ...
}
```

```
"Steps": {
 "uninstall": ["samplePackage"],
 "download": [
 {
 "samplePackage": "https://EXAMPLEBUCKET.s3.us-west-2.amazonaws.com/
exampleCodeFile"
 },
],
 "signature": [
 "samplePackage" : "aaaaabbbbccccddddddeeeeffffffggggghhhhhiiiiijjjj"
]
}
}
```

Untuk informasi selengkapnya tentang AWS IoT Pekerjaan, membuat dokumen pekerjaan, dan menerapkan pekerjaan, lihat [Pekerjaan](#).

## Mempersiapkan dokumen pekerjaan dan versi paket untuk penyebaran

Ketika versi paket dibuat, itu dalam draft keadaan untuk menunjukkan bahwa itu sedang dipersiapkan untuk penerapan. Untuk menyiapkan versi paket untuk penyebaran, Anda harus membuat dokumen pekerjaan, menyimpan dokumen di lokasi yang dapat diakses oleh pekerjaan (seperti Amazon S3), dan mengonfirmasi bahwa versi paket memiliki nilai atribut yang Anda inginkan untuk digunakan oleh dokumen pekerjaan. (Catatan: Anda dapat memperbarui atribut untuk versi paket hanya saat berada dalam draft status.)

Bila Anda puas dengan versi paket, publikasikan melalui halaman detail paket perangkat lunak di AWS IoT konsol atau dengan mengeluarkan operasi API [UpdatePackageVersi](#). Anda kemudian dapat mereferensikan versi paket saat Anda membuat pekerjaan baik melalui AWS IoT konsol atau dengan mengeluarkan operasi [CreateJobAPI](#).

## Menamai paket dan versi saat menerapkan

Saat Anda menerapkan AWS IoT pekerjaan, Anda harus memberi nama paket perangkat lunak dan versi paket yang sama yang diberi nama dalam dokumen pekerjaan di job deployment (`destinationPackageVersions`). Jika tidak, Anda akan menerima pesan galat yang menyatakan versi paket yang hilang.

Anda dapat menyertakan paket perangkat lunak tambahan dan versi paket yang tidak disertakan dalam dokumen pekerjaan. Jika Anda melakukan ini, pekerjaan tidak memberikan instruksi kepada

perangkat tentang apa yang harus dilakukan dengan file-file tersebut dan perangkat diharapkan tahu apa yang harus dilakukan. Misalnya, Anda dapat mengirim file tambahan ke perangkat jika berisi data yang mungkin direferensikan perangkat.

## Menargetkan pekerjaan melalui kelompok hal yang AWS IoT dinamis

Software Package Catalog bekerja dengan [pengindeksan armada](#), [AWS IoT pekerjaan](#), dan [grup benda AWS IoT dinamis](#) untuk memfilter dan menargetkan perangkat dalam armada Anda untuk memilih versi paket mana yang akan diterapkan ke perangkat Anda. Anda dapat menjalankan kueri pengindeksan armada berdasarkan informasi paket perangkat Anda saat ini dan menargetkan hal-hal tersebut untuk suatu AWS IoT pekerjaan. Anda juga dapat merilis pembaruan perangkat lunak, tetapi hanya untuk perangkat target yang memenuhi syarat. Misalnya, Anda dapat menentukan bahwa Anda ingin menerapkan konfigurasi hanya ke perangkat yang saat ini menjalankan konfigurasi. `iot-device-client 1.5.09` Untuk informasi selengkapnya, lihat [Membuat grup benda dinamis](#).

## Versi bayangan dan paket bernama yang dicadangkan

Jika dikonfigurasi, AWS IoT Jobs dapat memperbarui sesuatu yang dicadangkan bernama shadow (`$package`) ketika pekerjaan berhasil diselesaikan. Jika Anda melakukannya, Anda tidak perlu mengaitkan versi paket secara manual ke bayangan bernama yang dicadangkan.

Anda dapat memilih untuk mengaitkan atau memperbarui versi paket secara manual ke bayangan bernama yang dicadangkan dalam situasi berikut:

- Anda mendaftarkan sesuatu AWS IoT Core tanpa mengaitkan versi paket yang diinstal.
- AWS IoT Pekerjaan tidak dikonfigurasi untuk memperbarui bayangan bernama yang dicadangkan.
- Anda menggunakan proses internal untuk mengirimkan versi paket ke armada Anda dan proses itu tidak diperbarui AWS IoT Core saat selesai.

### Note

Kami menyarankan Anda menggunakan AWS IoT Jobs untuk memperbarui versi paket di shadow (`$package`) bernama reserved. Memperbarui parameter versi dalam `$package` bayangan melalui proses lain (seperti, panggilan API manual atau terprogram) ketika AWS IoT Jobs juga dikonfigurasi untuk memperbarui bayangan, dapat menyebabkan ketidakkonsistenan antara versi aktual pada perangkat dan versi yang dilaporkan ke bayangan bernama yang dicadangkan.

Anda dapat menambahkan atau memperbarui versi paket ke sesuatu yang dicadangkan bernama shadow (`$package`) melalui konsol atau operasi [UpdateThingShadowAPI](#). Untuk informasi selengkapnya, lihat [Mengaitkan versi paket ke AWS IoT sesuatu](#).

#### Note

Mengaitkan versi paket ke AWS IoT sesuatu tidak secara langsung memperbarui perangkat lunak perangkat. Anda harus menyebarkan versi paket ke perangkat untuk memperbarui perangkat lunak perangkat.

## Menghapus instalasi paket perangkat lunak dan versi paketnya

`$null` adalah placeholder cadangan yang meminta layanan AWS IoT Jobs untuk menghapus paket perangkat lunak dan versi paket yang ada dari bayangan bernama cadangan perangkat. `$package` Untuk informasi selengkapnya, lihat [Cadangan bernama bayangan](#).

Untuk menggunakan fitur ini, ganti nama versi di akhir [tujuan PackageVersion](#) Amazon Resource Name (ARN) dengan `$null`. Setelah itu, Anda harus menginstruksikan layanan Anda untuk menghapus perangkat lunak dari perangkat.

ARN resmi menggunakan format berikut:

```
arn:aws:iot:<regionCode>:111122223333:package/<packageName>/version/$null
```

Misalnya,

```
$ aws iot create-job \
 ... \
 --destinationPackageVersions ["arn:aws:iot:us-east-1:111122223333:package/
samplePackage/version/$null"]
```

## Memulai dengan Software Package Catalog

Anda dapat membangun dan memelihara Katalog Paket AWS IoT Device Management Perangkat Lunak melalui AWS Management Console, operasi AWS IoT Core API, dan AWS Command Line Interface (AWS CLI).

Menggunakan konsol

Untuk menggunakan AWS Management Console, masuk ke AWS akun Anda dan navigasikan ke [AWS IoT Core](#). Di panel navigasi, pilih Paket perangkat lunak. Anda kemudian dapat membuat dan mengelola paket dan versinya dari bagian ini.

Menggunakan operasi API atau CLI

Anda dapat menggunakan operasi AWS IoT Core API untuk membuat dan mengelola fitur Katalog Paket Perangkat Lunak. Untuk informasi selengkapnya, lihat [Referensi AWS IoT API](#) dan [AWS SDK dan Toolkit](#). AWS CLI Perintah juga mengelola katalog Anda. Untuk informasi selengkapnya, lihat Referensi [Perintah AWS IoT CLI](#).

Bab ini berisi bagian-bagian berikut:

- [Membuat paket perangkat lunak dan versi paket](#)
- [Menerapkan versi paket melalui pekerjaan AWS IoT](#)
- [Mengaitkan versi paket ke suatu AWS IoT hal](#)

## Membuat paket perangkat lunak dan versi paket

Anda dapat menggunakan langkah-langkah berikut untuk membuat paket dan hal versi awal melalui AWS Management Console.

Untuk membuat paket perangkat lunak

1. Masuk ke AWS akun Anda dan arahkan ke [AWS IoT konsol](#).
2. Pada panel navigasi, pilih Paket perangkat lunak.
3. Pada halaman paket AWS IoT perangkat lunak, pilih Buat paket. Kotak dialog Aktifkan dependensi untuk manajemen paket muncul.
4. Di bawah Pengindeksan armada, pilih Tambahkan paket dan versi perangkat lunak perangkat. Ini diperlukan untuk Katalog Paket Perangkat Lunak dan menyediakan pengindeksan armada dan metrik tentang armada Anda.
5. [Opsional] Jika Anda ingin AWS IoT lowongan memperbarui bayangan bernama cadangan saat pekerjaan berhasil diselesaikan, pilih Perbarui bayangan otomatis dari pekerjaan. Jika Anda tidak ingin AWS IoT pekerjaan melakukan pembaruan ini, biarkan kotak centang ini tidak dipilih.
6. [Opsional] Untuk memberikan AWS IoT lowongan hak untuk memperbarui bayangan bernama yang dicadangkan, di bawah Pilih peran, pilih Buat peran. Jika Anda tidak ingin AWS IoT lowongan melakukan pembaruan ini, peran ini tidak diperlukan.




7. Buat atau pilih peran.
  - a. Jika Anda tidak memiliki peran untuk tujuan ini: Saat kotak dialog Buat peran muncul, masukkan nama Peran, lalu pilih Buat.
  - b. Jika Anda memiliki peran untuk tujuan ini: Untuk peran Pilih, pilih peran Anda, lalu pastikan kotak centang Lampirkan kebijakan ke peran IAM dipilih.
8. Pilih Konfirmasi. Halaman Buat paket baru muncul.
9. Di bawah Package detail, masukkan nama Package.
10. Di bawah Package description, masukkan informasi untuk membantu Anda mengidentifikasi dan mengelola paket ini.
11. [Opsional] Anda dapat menggunakan tag untuk membantu Anda mengkategorikan dan mengelola paket ini. Untuk menambahkan tag, perluas Tag, pilih Tambahkan tag, dan masukkan pasangan nilai kunci. Anda dapat memasukkan hingga 50 tag. Untuk informasi selengkapnya, lihat [Menandai AWS IoT sumber daya Anda](#).

Untuk menambahkan versi paket saat membuat paket baru

1. Di bawah Versi pertama, masukkan nama Versi.

Sebaiknya gunakan [SemVer format](#) (misalnya, 1.0.0.0) untuk mengidentifikasi versi paket Anda secara unik. Anda juga dapat menggunakan strategi pemformatan berbeda yang lebih sesuai dengan kasus penggunaan Anda. Untuk informasi selengkapnya, lihat [Siklus hidup versi paket](#).

2. Di bawah Deskripsi versi, masukkan informasi yang akan membantu Anda mengidentifikasi dan mengelola versi paket ini.

 Note

Kotak centang Versi default dinonaktifkan karena versi paket dibuat dalam draft keadaan. Anda dapat memberi nama versi default setelah Anda membuat versi paket dan ketika Anda mengubah status menjadipublished. Untuk informasi selengkapnya, lihat [Siklus hidup versi paket](#).

3. [Opsional] Untuk membantu Anda mengelola versi ini atau untuk mengkomunikasikan informasi ke perangkat Anda, masukkan satu atau beberapa pasangan nama-nilai untuk atribut Versi. Pilih

Tambahkan atribut untuk setiap pasangan nama-nilai yang Anda masukkan. Untuk informasi selengkapnya, lihat [Atribut versi](#).

4. [Opsional] Anda dapat menggunakan tag untuk membantu Anda mengkategorikan dan mengelola paket ini. Untuk menambahkan tag, perluas Tag, pilih Tambahkan tag, dan masukkan pasangan nilai kunci. Anda dapat memasukkan hingga 50 tag. Untuk informasi selengkapnya, lihat [Menandai AWS IoT sumber daya Anda](#).
5. Pilih Buat paket. Halaman paket AWS IoT perangkat lunak muncul dan paket Anda tercantum dalam tabel paket.
6. [Opsional] Untuk meninjau informasi tentang paket perangkat lunak dan versi paket yang Anda buat, pilih nama paket Anda. Halaman detail paket muncul.

## Menerapkan versi paket melalui pekerjaan AWS IoT

Anda dapat menggunakan langkah-langkah berikut untuk menyebarkan versi paket melalui AWS Management Console

Prasyarat:

Sebelum memulai, lakukan hal berikut:

- Daftarkan AWS IoT hal-hal dengan AWS IoT Core. Untuk petunjuk arah untuk menambahkan perangkat Anda AWS IoT Core, lihat [Membuat objek benda](#).
- [Opsional] Buat grup AWS IoT benda atau grup benda dinamis untuk menargetkan perangkat yang akan Anda gunakan versi paket. Untuk petunjuk arah untuk membuat grup benda, lihat [Membuat grup benda statis](#). Untuk petunjuk arah untuk membuat grup benda dinamis, lihat [Membuat grup hal dinamis](#).
- Buat paket perangkat lunak dan versi paket. Untuk informasi selengkapnya, lihat [Membuat paket perangkat lunak dan versi paket](#).
- Buat dokumen pekerjaan. Untuk informasi selengkapnya, lihat [Mempersiapkan dokumen pekerjaan dan versi paket untuk penerapan](#).


Untuk menyebarkan pekerjaan AWS IoT

1. Di [AWS IoT konsol](#), pilih Paket perangkat lunak.
2. Pilih paket perangkat lunak yang ingin Anda gunakan. Halaman detail paket perangkat lunak muncul.

3. Pilih versi paket yang ingin Anda terapkan, di bawah Versi, dan pilih Terapkan versi pekerjaan.
4. Jika ini adalah pertama kalinya Anda menerapkan pekerjaan melalui portal ini, kotak dialog yang menjelaskan persyaratan akan muncul. Tinjau informasi dan pilih Akui.
5. Masukkan nama untuk penyebaran atau tinggalkan nama yang dibuat secara otomatis di bidang Nama.
6. [Opsional] Di bidang Deskripsi, masukkan deskripsi yang mengidentifikasi tujuan atau isi penyebaran, atau tinggalkan informasi yang dibuat secara otomatis.

Catatan: Kami menyarankan agar Anda tidak menggunakan informasi identitas pribadi di kolom Nama dan deskripsi Job.

7. [Opsional] Tambahkan tag apa pun untuk dikaitkan dengan pekerjaan ini.
8. Pilih Selanjutnya.
9. Di bawah target Job, pilih hal-hal atau kelompok hal yang harus menerima pekerjaan.
10. Di bidang File Job, tentukan file JSON dokumen pekerjaan.
11. Integrasi Open Jobs dengan layanan Package Catalog.
12. Pilih paket dan versi yang ditentukan dalam dokumen pekerjaan Anda.

 Note

Anda diminta untuk memilih paket dan versi paket yang sama yang ditentukan dalam dokumen pekerjaan. Anda dapat memasukkan lebih banyak, tetapi pekerjaan akan mengeluarkan instruksi hanya untuk paket dan versi yang disertakan dalam dokumen pekerjaan. Untuk informasi selengkapnya, lihat [Penamaan paket dan versi saat menerapkan](#).

13. Pilih Selanjutnya.
14. Pada halaman konfigurasi Job, pilih salah satu jenis pekerjaan berikut di kotak dialog konfigurasi Job:
  - Pekerjaan snapshot: Pekerjaan snapshot selesai ketika selesai dijalankan pada perangkat dan grup target.
  - Pekerjaan berkelanjutan: Pekerjaan berkelanjutan berlaku untuk grup benda dan berjalan di perangkat apa pun yang nantinya Anda tambahkan ke grup target tertentu.
15. Di kotak dialog Konfigurasi tambahan - opsional, tinjau konfigurasi pekerjaan opsional berikut dan buat pilihan Anda sesuai. Untuk informasi selengkapnya, lihat [Peluncuran pekerjaan](#),

[penjadwalan, dan membatalkan konfigurasi serta batas waktu eksekusi Job dan konfigurasi coba lagi.](#)

- Konfigurasi peluncuran
- Konfigurasi penjadwalan
- Konfigurasi batas waktu eksekusi Job
- Konfigurasi coba lagi eksekusi Job
- Batalkan konfigurasi

16. Tinjau pilihan pekerjaan dan kemudian pilih Kirim.

Setelah Anda membuat pekerjaan, konsol menghasilkan tanda tangan JSON dan menempatkannya di dokumen pekerjaan Anda. Anda dapat menggunakan AWS IoT konsol untuk melihat status pekerjaan, atau membatalkan atau menghapus pekerjaan. Untuk mengelola pekerjaan, buka [hub Job konsol](#).

## Mengaitkan versi paket ke suatu AWS IoT hal

Setelah Anda menginstal perangkat lunak pada perangkat Anda, Anda dapat mengaitkan versi paket ke AWS IoT sesuatu yang dicadangkan bernama shadow. Jika AWS IoT pekerjaan telah dikonfigurasi untuk memperbarui bayangan bernama yang dicadangkan setelah pekerjaan diterapkan dan berhasil diselesaikan, Anda tidak perlu menyelesaikan prosedur ini. Untuk informasi selengkapnya, lihat [Cadangan bernama bayangan](#).

Prasyarat:

Sebelum memulai, lakukan hal berikut:

- Buat AWS IoT sesuatu, atau benda, dan bangun telemetry melalui. AWS IoT Core Untuk informasi selengkapnya, lihat [Memulai dengan AWS IoT Core](#).
- Buat paket perangkat lunak dan versi paket. Untuk informasi selengkapnya, lihat [Membuat paket perangkat lunak dan versi paket](#).
- Instal perangkat lunak versi paket pada perangkat.

 Note

Mengaitkan versi paket ke AWS IoT sesuatu tidak memperbarui atau menginstal perangkat lunak pada perangkat fisik. Versi paket harus digunakan ke perangkat.

Untuk mengaitkan versi paket ke suatu AWS IoT hal

1. Pada panel navigasi [AWS IoT konsol](#), perluas menu Semua perangkat dan pilih Things.
2. Identifikasi AWS IoT hal yang ingin Anda perbarui dari daftar dan pilih nama benda untuk menampilkan halaman detailnya.
3. Di bagian Detail, pilih Paket dan versi.
4. Pilih Tambahkan ke paket dan versi.
5. Untuk Pilih paket perangkat, pilih paket perangkat lunak yang Anda inginkan.
6. Untuk Pilih versi, pilih versi perangkat lunak yang Anda inginkan.
7. Pilih Tambahkan paket perangkat.

Paket dan versi muncul di daftar Paket dan versi yang dipilih.

8. Ulangi langkah-langkah ini untuk setiap paket dan versi yang ingin Anda kaitkan dengan hal ini.
9. Setelah selesai, pilih Tambahkan paket dan detail versi. Halaman Detail Thing terbuka dan Anda dapat melihat paket dan versi baru dalam daftar.

# AWS IoT Core Lokasi Perangkat

Sebelum menggunakan fitur Lokasi AWS IoT Core Perangkat, tinjau Syarat dan Ketentuan untuk fitur ini. Perhatikan bahwa AWS dapat mengirimkan parameter permintaan pencarian geolokasi Anda, seperti data lokasi yang digunakan untuk menjalankan pencarian, dan informasi lainnya ke penyedia data pihak ketiga pilihan Anda, yang mungkin berada di luar Wilayah AWS yang Anda gunakan saat ini. Untuk informasi lebih lanjut, lihat [Ketentuan Layanan AWS](#).

Gunakan Lokasi AWS IoT Core Perangkat untuk menguji lokasi perangkat IoT Anda menggunakan pemecah pihak ketiga. Solver adalah algoritma yang disediakan oleh vendor pihak ketiga yang menyelesaikan data pengukuran dan memperkirakan lokasi perangkat Anda. Dengan mengidentifikasi lokasi perangkat Anda, Anda dapat melacak dan men-debug mereka di bidang untuk memecahkan masalah apa pun.

Data pengukuran yang dikumpulkan dari berbagai sumber diselesaikan, dan informasi geolokasi dilaporkan sebagai muatan [GeoJSON](#). Format GeoJSON adalah format yang digunakan untuk menyandikan struktur data geografis. Muatan berisi koordinat lintang dan bujur lokasi perangkat Anda, yang didasarkan pada sistem koordinat [Sistem Geodetik Dunia](#) (WGS84).

## Topik

- [Jenis pengukuran dan pemecah](#)
- [Cara Kerja Lokasi AWS IoT Core Perangkat](#)
- [Cara menggunakan Lokasi AWS IoT Core Perangkat](#)
- [Menyelesaikan lokasi perangkat IoT](#)
- [Menyelesaikan lokasi perangkat menggunakan topik AWS IoT Core MQTT Lokasi Perangkat](#)
- [Pemecah lokasi dan muatan perangkat](#)

## Jenis pengukuran dan pemecah

AWS IoT Core Lokasi Perangkat bermitra dengan vendor pihak ketiga untuk menyelesaikan data pengukuran dan menyediakan perkiraan lokasi perangkat. Tabel berikut menunjukkan jenis pengukuran dan pemecah lokasi pihak ketiga, serta informasi tentang perangkat yang didukung. Untuk informasi tentang perangkat LoRa WAN dan mengonfigurasi lokasi perangkat untuknya, lihat [Mengonfigurasi posisi sumber daya LoRa WAN](#).

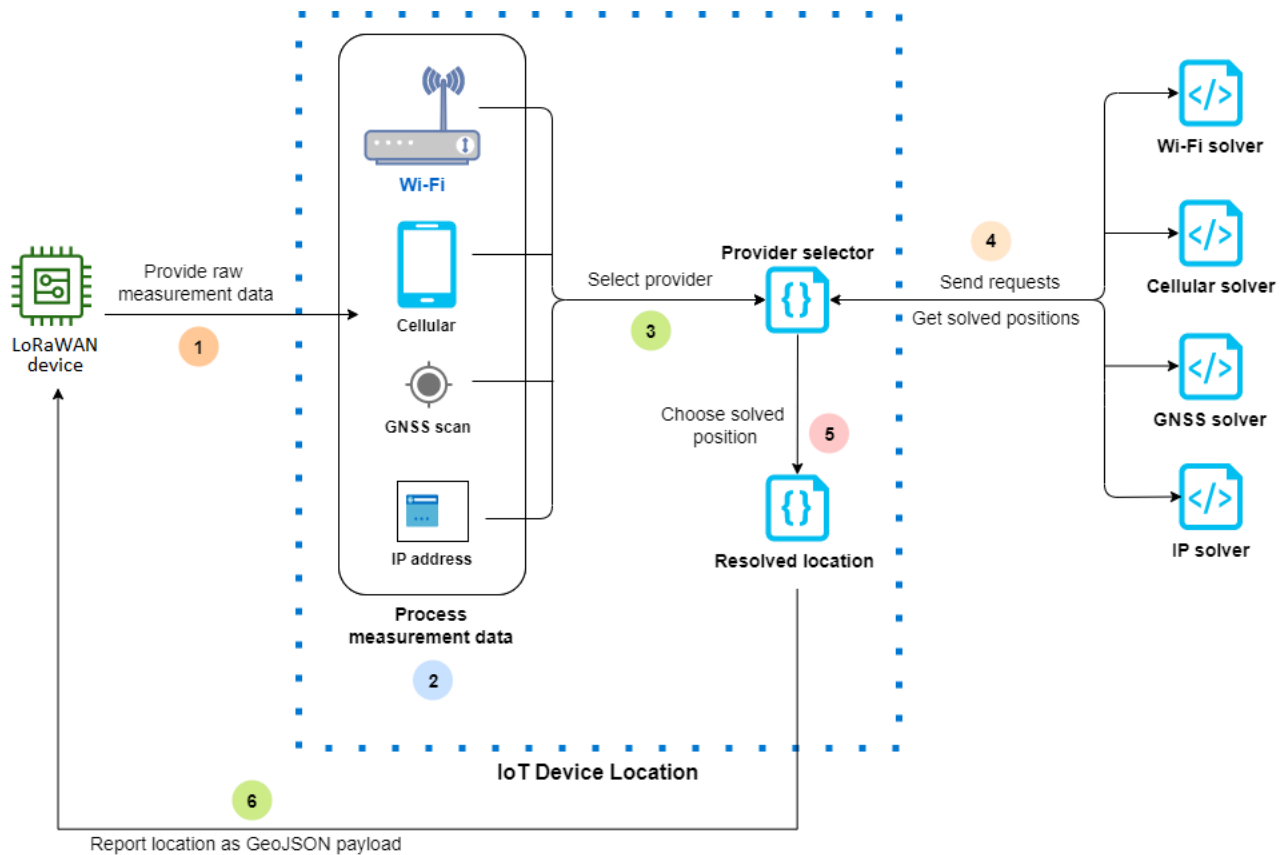
## Jenis pengukuran dan pemecah

| Jenis pengukuran                                                      | Pemecah pihak ketiga          | Perangkat yang didukung                   |
|-----------------------------------------------------------------------|-------------------------------|-------------------------------------------|
| Titik akses Wi-Fi                                                     | Pemecah berbasis Wi-Fi        | Perangkat IoT umum dan perangkat WAN LoRa |
| Menara radio seluler: data GSM, LTE, CDMA, SCDMA, WCMDA, dan TD-SCDMA | Pemecah berbasis seluler      | Perangkat IoT umum dan perangkat WAN LoRa |
| Alamat IP                                                             | Pemecah pencarian terbalik IP | Perangkat IoT umum                        |
| Data pemindaian GNSS (pesan NAV)                                      | Pemecah GNSS                  | Perangkat IoT umum dan perangkat WAN LoRa |

Untuk informasi selengkapnya tentang pemecah lokasi dan contoh yang menunjukkan muatan perangkat untuk berbagai jenis pengukuran, lihat [Pemecah lokasi dan muatan perangkat](#)

## Cara Kerja Lokasi AWS IoT Core Perangkat

Diagram berikut menunjukkan cara Lokasi AWS IoT Core Perangkat mengumpulkan data pengukuran dan menyelesaikan informasi lokasi perangkat Anda.



Langkah-langkah berikut menunjukkan cara kerja Lokasi AWS IoT Core Perangkat.

### 1. Menerima data pengukuran

Data pengukuran mentah yang terkait dengan lokasi perangkat Anda pertama kali dikirim dari perangkat. Data pengukuran ditentukan sebagai muatan JSON.

### 2. Data pengukuran proses

Data pengukuran diproses, dan Lokasi AWS IoT Core Perangkat memilih data pengukuran yang akan digunakan, yang dapat berupa Wi-Fi, seluler, pemindaian GNSS, atau informasi alamat IP.

### 3. Pilih pemecah

Pemecah pihak ketiga dipilih berdasarkan data pengukuran. Misalnya, jika data pengukuran berisi informasi Wi-Fi dan alamat IP, ia memilih pemecah Wi-Fi dan pemecah pencarian terbalik IP.



#### 4. Dapatkan lokasi yang diselesaikan

Permintaan API dikirim ke penyedia pemecah yang meminta penyelesaian lokasi. AWS IoT Core Lokasi Perangkat kemudian mendapatkan perkiraan informasi geolokasi dari pemecah.

#### 5. Pilih lokasi yang diselesaikan

Informasi lokasi yang diselesaikan dan akurasinya dibandingkan, dan Lokasi AWS IoT Core Perangkat memilih hasil geolokasi dengan akurasi tertinggi.

#### 6. Informasi lokasi keluaran

Informasi geolokasi dikirimkan kepada Anda sebagai muatan GeoJSON. Muatan berisi koordinat geografis WGS84, informasi akurasi, tingkat kepercayaan, dan stempel waktu di mana lokasi yang diselesaikan diperoleh.

## Cara menggunakan Lokasi AWS IoT Core Perangkat

Langkah-langkah berikut menunjukkan cara menggunakan Lokasi AWS IoT Core Perangkat.

#### 1. Berikan data pengukuran

Tentukan data pengukuran mentah yang terkait dengan lokasi perangkat Anda sebagai muatan JSON. Untuk mengambil data pengukuran payload, buka log perangkat Anda, atau gunakan CloudWatch Log, dan salin informasi data payload. Payload JSON harus berisi satu atau lebih jenis pengukuran data. Untuk contoh yang menunjukkan format payload untuk berbagai pemecah, lihat [Pemecah lokasi dan muatan perangkat](#)

#### 2. Selesaikan informasi lokasi

Menggunakan halaman [Lokasi Perangkat](#) di AWS IoT konsol atau operasi [GetPositionEstimate](#) API, teruskan data pengukuran muatan dan selesaikan lokasi perangkat. AWS IoT Core Lokasi Perangkat kemudian memilih pemecah dengan akurasi tertinggi dan melaporkan lokasi perangkat. Untuk informasi selengkapnya, lihat [Menyelesaikan lokasi perangkat IoT](#).

#### 3. Salin informasi lokasi

Verifikasi informasi geolokasi yang diselesaikan oleh Lokasi AWS IoT Core Perangkat dan dilaporkan sebagai muatan GeoJSON. Anda dapat menyalin muatan untuk digunakan dengan aplikasi Anda dan Layanan AWS s lainnya. Misalnya, Anda dapat mengirim data lokasi geografis Anda ke Amazon Location Service menggunakan tindakan [Lokasi](#) AWS IoT aturan.

Topik berikut menunjukkan cara menggunakan Lokasi AWS IoT Core Perangkat dan contoh muatan lokasi perangkat.

- [Menyelesaikan lokasi perangkat IoT](#)
- [Pemecah lokasi dan muatan perangkat](#)

## Menyelesaikan lokasi perangkat IoT

Gunakan Lokasi AWS IoT Core Perangkat untuk memecahkan kode data pengukuran dari perangkat Anda, dan menyelesaikan lokasi perangkat menggunakan pemecah pihak ketiga. Lokasi yang diselesaikan dihasilkan sebagai muatan GeoJSON dengan koordinat geografis dan informasi akurasi. Anda dapat menyelesaikan lokasi perangkat Anda dari AWS IoT konsol, AWS IoT Wireless API, atau AWS CLI.

Topik

- [Menyelesaikan lokasi perangkat \(konsol\)](#)
- [Menyelesaikan lokasi perangkat \(API\)](#)
- [Memecahkan masalah kesalahan saat menyelesaikan lokasi](#)

## Menyelesaikan lokasi perangkat (konsol)

Untuk menyelesaikan lokasi perangkat (konsol)

1. Buka halaman [Lokasi Perangkat](#) di AWS IoT konsol.
2. Dapatkan data pengukuran muatan dari log perangkat Anda atau dari CloudWatch Log, dan masukkan di bagian Selesaikan posisi melalui payload.

Kode berikut menunjukkan contoh payload JSON. Muatan berisi data pengukuran seluler dan Wi-Fi. Jika muatan Anda berisi jenis data pengukuran tambahan, pemecah dengan akurasi terbaik akan digunakan. Untuk informasi selengkapnya dan contoh muatan, lihat [the section called “Pemecah lokasi dan muatan perangkat”](#).

### Note

Payload JSON harus berisi setidaknya satu jenis data pengukuran.

```
{
 "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
 }
],
 {
```

```

 "Uarfcnd1": 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. Untuk menyelesaikan informasi lokasi, pilih Selesaikan.

Informasi lokasi adalah jenis gumpalan dan dikembalikan sebagai muatan yang menggunakan format GeoJSON, yang merupakan format yang digunakan untuk pengkodean struktur data geografis. Muatan berisi:

- Koordinat geografis WGS84, yang mencakup informasi lintang dan bujur. Mungkin juga termasuk informasi ketinggian.
- Jenis informasi lokasi yang dilaporkan, seperti Point. [Jenis lokasi titik mewakili lokasi sebagai garis lintang dan bujur WGS84, dikodekan sebagai titik GeoJSON.](#)

- Informasi akurasi horizontal dan vertikal, yang menunjukkan perbedaan, dalam meter, antara informasi lokasi yang diperkirakan oleh pemecah dan lokasi perangkat yang sebenarnya.
- Tingkat kepercayaan, yang menunjukkan ketidakpastian dalam respon estimasi lokasi. Nilai default adalah 0,68, yang menunjukkan probabilitas 68% bahwa lokasi perangkat sebenarnya berada dalam radius ketidakpastian dari perkiraan lokasi.
- Kota, negara bagian, negara, dan kode pos tempat perangkat berada. Informasi ini akan dilaporkan hanya ketika pemecah pencarian balik IP digunakan.
- Informasi stempel waktu, yang sesuai dengan tanggal dan waktu di mana lokasi diselesaikan. Ini menggunakan format timestamp Unix.

Kode berikut menunjukkan contoh payload GeoJSON dikembalikan dengan menyelesaikan lokasi.

#### Note

Jika Lokasi AWS IoT Core Perangkat melaporkan kesalahan saat mencoba menyelesaikan lokasi, Anda dapat memecahkan masalah kesalahan dan menyelesaikan lokasi. Untuk informasi selengkapnya, lihat [Memecahkan masalah kesalahan saat menyelesaikan lokasi](#).

```
{
 "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. Buka bagian Lokasi sumber daya dan verifikasi informasi geolokasi yang dilaporkan oleh Lokasi AWS IoT Core Perangkat. Anda dapat menyalin muatan untuk digunakan dengan aplikasi lain dan Layanan AWS s. Misalnya, Anda dapat menggunakan file [Lokasi](#) untuk mengirim data lokasi geografis Anda ke Amazon Location Service.

## Menyelesaikan lokasi perangkat (API)

Untuk menyelesaikan lokasi perangkat menggunakan AWS IoT Wireless API, gunakan operasi [GetPositionEstimate](#) API atau perintah [get-position-estimate](#) CLI. Tentukan data pengukuran payload sebagai input, dan jalankan operasi API untuk menyelesaikan lokasi perangkat.

### Note

Operasi `GetPositionEstimate` API tidak menyimpan informasi perangkat atau status apa pun dan tidak dapat digunakan untuk mengambil data lokasi historis. Ini melakukan operasi satu kali yang menyelesaikan data pengukuran dan menghasilkan perkiraan lokasi. Untuk mengambil informasi lokasi, Anda harus menentukan informasi payload setiap kali Anda melakukan operasi API ini.

Perintah berikut menunjukkan contoh cara menyelesaikan lokasi menggunakan operasi API ini.

### Note

Saat menjalankan perintah `get-position-estimate` CLI, Anda harus menentukan file JSON output sebagai input pertama. File JSON ini akan menyimpan perkiraan informasi lokasi yang diperoleh sebagai respons dari CLI dalam format GeoJSON. Misalnya, perintah berikut menyimpan informasi lokasi dalam file `locationout.json`.

```
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
```

Contoh ini mencakup titik akses Wi-Fi dan alamat IP sebagai jenis pengukuran. AWS IoT Core Lokasi Perangkat memilih antara pemecah Wi-Fi dan pemecah pencarian balik IP, dan memilih pemecah dengan akurasi yang lebih tinggi.

Lokasi yang diselesaikan dikembalikan sebagai muatan yang menggunakan format GeoJSON, yang merupakan format yang digunakan untuk pengkodean struktur data geografis. Hal ini kemudian disimpan dalam file *locationout.json*. Muatan berisi koordinat lintang dan bujur WGS84, informasi tingkat akurasi dan kepercayaan, tipe data lokasi, dan stempel waktu di mana lokasi diselesaikan.

```
{
 "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"
 }
}
```

## Memecahkan masalah kesalahan saat menyelesaikan lokasi

Ketika Anda mencoba untuk menyelesaikan lokasi, Anda mungkin melihat salah satu kode kesalahan berikut. AWS IoT Core Lokasi Perangkat mungkin menghasilkan kesalahan saat menggunakan operasi `GetPositionEstimate` API, atau merujuk ke nomor baris yang terkait dengan kesalahan di AWS IoT konsol.

- 400 kesalahan

Kesalahan ini menunjukkan bahwa format JSON payload perangkat tidak dapat divalidasi oleh AWS IoT Core Lokasi Perangkat. Kesalahan mungkin terjadi karena:

- Data pengukuran JSON diformat secara tidak benar.

- Muatan hanya berisi informasi stempel waktu.
- Parameter data pengukuran, seperti alamat IP, tidak valid.

Untuk mengatasi kesalahan ini, periksa apakah JSON Anda diformat dengan benar dan berisi data dari satu atau beberapa jenis pengukuran sebagai input. Jika alamat IP tidak valid, untuk informasi tentang bagaimana Anda dapat memberikan alamat IP yang valid untuk mengatasi kesalahan, lihat. [Pemecah pencarian terbalik IP](#)

- 403 kesalahan

Kesalahan ini menunjukkan bahwa Anda tidak memiliki izin untuk melakukan operasi API atau menggunakan AWS IoT konsol untuk mengambil lokasi perangkat. Untuk mengatasi kesalahan ini, verifikasi bahwa Anda memiliki izin yang diperlukan untuk melakukan tindakan ini. Kesalahan ini mungkin terjadi jika AWS Management Console sesi Anda atau token AWS CLI sesi Anda telah kedaluwarsa. Untuk mengatasi kesalahan ini, segarkan token sesi untuk menggunakan AWS CLI, atau keluar dari, AWS Management Console lalu masuk menggunakan kredensial Anda.

- 404 kesalahan

Kesalahan ini menunjukkan bahwa tidak ada informasi lokasi yang ditemukan atau diselesaikan oleh Lokasi AWS IoT Core Perangkat. Kesalahan mungkin terjadi karena kasus-kasus seperti data yang tidak mencukupi dalam input data pengukuran. Sebagai contoh:

- Alamat MAC atau informasi menara seluler tidak cukup.
- Alamat IP tidak tersedia untuk mencari dan mengambil lokasi.
- Payload GNSS tidak cukup.

Untuk mengatasi kesalahan dalam kasus seperti itu, periksa apakah data pengukuran Anda berisi informasi yang cukup yang diperlukan untuk menyelesaikan lokasi perangkat.

- 500 kesalahan

Kesalahan ini menunjukkan bahwa pengecualian server internal terjadi saat Lokasi AWS IoT Core Perangkat mencoba menyelesaikan lokasi. Untuk mencoba memperbaiki kesalahan ini, segarkan sesi dan coba lagi mengirimkan data pengukuran yang akan diselesaikan.



# Menyelesaikan lokasi perangkat menggunakan topik AWS IoT Core MQTT Lokasi Perangkat

Anda dapat menggunakan topik MQTT yang dipesan untuk mendapatkan informasi lokasi terbaru untuk perangkat Anda dengan fitur Lokasi AWS IoT Core Perangkat.

## Format topik MQTT lokasi perangkat

Topik cadangan untuk Lokasi AWS IoT Core Perangkat menggunakan awalan berikut:

```
$aws/device_location/{customer_device_id}/
```

Untuk membuat topik lengkap, pertama-tama ganti *customer\_device\_id* dengan ID unik yang Anda gunakan untuk mengidentifikasi perangkat Anda. Kami menyarankan Anda menentukan `WirelessDeviceId`, seperti untuk perangkat LoRa WAN dan Trotoar, dan *thingName*, jika perangkat Anda terdaftar sebagai AWS IoT sesuatu. Anda kemudian menambahkan topik dengan rintisan topik, seperti `get_position_estimate` atau `get_position_estimate/accepted` seperti yang ditunjukkan di bagian berikut.

### Note

Hanya *{customer\_device\_id}* dapat berisi huruf, angka, dan tanda hubung. Saat berlangganan topik lokasi perangkat, Anda hanya dapat menggunakan tanda plus (+) sebagai karakter wildcard. Misalnya, Anda dapat menggunakan + wildcard *{customer\_device\_id}* untuk mendapatkan informasi lokasi untuk perangkat Anda. Saat Anda berlangganan topik `$aws/device_location/+/get_position_estimate/accepted`, pesan akan dipublikasikan dengan informasi lokasi untuk perangkat yang cocok dengan ID perangkat apa pun jika berhasil diselesaikan.

Berikut ini adalah topik cadangan yang digunakan untuk berinteraksi dengan Lokasi AWS IoT Core Perangkat.

## Lokasi perangkat topik MQTT

| Topik                                                                                | Operasi yang diizinkan | Deskripsi                                                                                                                     |
|--------------------------------------------------------------------------------------|------------------------|-------------------------------------------------------------------------------------------------------------------------------|
| <code>\$aws/device_location/customer_device_id/get_position_estimate</code>          | Publikasikan           | Perangkat memublikasikan topik ini agar data pengukuran mentah yang dipindai diselesaikan oleh Lokasi AWS IoT Core Perangkat. |
| <code>\$aws/device_location/customer_device_id/get_position_estimate/diterima</code> | Langganan              | AWS IoT Core Lokasi Perangkat memublikasikan informasi lokasi ke topik ini saat berhasil menyelesaikan lokasi perangkat.      |
| <code>\$aws/device_location/customer_device_id/get_position_estimate/ditolak</code>  | Langganan              | AWS IoT Core Lokasi Perangkat memublikasikan informasi kesalahan ke topik ini jika gagal menyelesaikan lokasi perangkat.      |

## Kebijakan untuk topik MQTT lokasi perangkat

Untuk menerima pesan dari topik lokasi perangkat, perangkat Anda harus menggunakan kebijakan yang memungkinkannya terhubung ke gateway AWS IoT perangkat dan berlangganan topik MQTT.

Berikut ini adalah contoh kebijakan yang diperlukan untuk menerima pesan untuk berbagai topik.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "iot:Publish"
],
 "Resource": [
 "arn:aws:iot:region:account:topic/$aws/device_location/customer_device_id/get_position_estimate"
]
 }
]
}
```

```

]
 },
 {
 "Effect": "Allow",
 "Action": [
 "iot:Receive"
],
 "Resource": [
 "arn:aws:iot:region:account:topic/$aws/device_location/customer_device_id/get_position_estimate/accepted",
 "arn:aws:iot:region:account:topic/$aws/device_location/customer_device_id/get_position_estimate/rejected"
]
 },
 {
 "Effect": "Allow",
 "Action": [
 "iot:Subscribe"
],
 "Resource": [
 "arn:aws:iot:region:account:topicfilter/$aws/device_location/customer_device_id/get_position_estimate/accepted",
 "arn:aws:iot:region:account:topicfilter/$aws/device_location/customer_device_id/get_position_estimate/rejected"
]
 }
]
}

```

## Topik dan muatan lokasi perangkat

Berikut ini menampilkan topik Lokasi AWS IoT Core Perangkat, format payload pesannya, dan kebijakan contoh untuk setiap topik.

### Topik

- [/get\\_position\\_estimasi](#)
- [/get\\_position\\_estimate/diterima](#)
- [/get\\_position\\_estimate/ditolak](#)

## /get\_position\_estimasi

Publikasikan pesan ke topik ini untuk mendapatkan data pengukuran mentah dari perangkat yang akan diselesaikan oleh Lokasi AWS IoT Core Perangkat.

```
$aws/device_location/customer_device_id/get_position_estimate
```

AWS IoT Core Lokasi Perangkat merespons dengan memublikasikan salah satu [/get\\_position\\_estimate/diterima](#) atau [/get\\_position\\_estimate/ditolak](#).

### Note

Pesan yang dipublikasikan untuk topik ini harus berupa payload JSON yang valid. Jika pesan input tidak dalam format JSON yang valid, Anda tidak akan mendapatkan respons apa pun. Untuk informasi selengkapnya, lihat [Payload pesan](#).

## Muatan pesan

Format payload pesan mengikuti struktur yang mirip dengan badan permintaan operasi AWS IoT Wireless API, [GetPositionEstimate](#). Ini berisi:

- `TimestampString` opsional, yang sesuai dengan tanggal dan waktu lokasi diselesaikan. `TimestampString` dapat memiliki panjang minimum 1 dan panjang maksimum 10.
- `MessageIdString` opsional, yang dapat digunakan untuk memetakan permintaan ke respons. Jika Anda menentukan string ini, pesan yang dipublikasikan ke `get_position_estimate/rejected` topik `get_position_estimate/accepted` atau akan berisi `iniMessageId`. `MessageIDString` dapat memiliki panjang minimum 1 dan panjang maksimum 256.
- Data pengukuran dari perangkat yang berisi satu atau lebih jenis pengukuran berikut:
  - [WiFiAccessPoint](#)
  - [CellTowers](#)
  - [IpAddress](#)
  - [Gnss](#)

Berikut ini menunjukkan payload pesan sampel.

```
{
```

```

"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
}
}

```

## Contoh kebijakan

Berikut ini adalah contoh kebijakan yang diperlukan:

```

{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "iot:Publish"
],
 "Resource": [
 "arn:aws:iot:region:account:topic/$aws/device_location/customer_device_id/get_position_estimate"
]
 }
]
}

```

## /get\_position\_estimate/diterima

AWS IoT Core Lokasi Perangkat menerbitkan respons terhadap topik ini saat mengembalikan informasi lokasi yang telah diselesaikan untuk perangkat Anda. Informasi lokasi dikembalikan dalam format [GeoJSON](#).

```
$aws/device_location/customer_device_id/get_position_estimate/accepted
```

Berikut ini menunjukkan payload pesan dan kebijakan contoh.

### Muatan pesan

Berikut ini adalah contoh payload pesan dalam format GeoJSON. Jika Anda menetapkan data pengukuran mentah dan Lokasi AWS IoT Core Perangkat berhasil menyelesaikan informasi lokasi, maka payload pesan akan menampilkan MessageId informasi yang sama. MessageId

```
{
 "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"
 }
}
```

### Contoh kebijakan

Berikut ini adalah contoh kebijakan yang diperlukan:

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

```

 "Resource": [
 "arn:aws:iot:region:account:topicfilter/$aws/
device_location/customer_device_id/get_position_estimate/accepted"
]
 },
 {
 "Effect": "Allow",
 "Action": [
 "iot:Receive"
],
 "Resource": [
 "arn:aws:iot:region:account:topic/$aws/device_location/customer_device_id/
get_position_estimate/accepted"
]
 }
]
}

```

## /get\_position\_estimate/ditolak

AWS IoT Core Lokasi Perangkat memublikasikan respons kesalahan terhadap topik ini jika gagal menyelesaikan lokasi perangkat.

```
$aws/device_location/customer_device_id/get_position_estimate/rejected
```

Berikut ini menunjukkan payload pesan dan contoh kebijakan. Untuk informasi tentang kesalahan, lihat [Memecahkan masalah kesalahan saat menyelesaikan lokasi](#).

## Muatan pesan

Berikut ini adalah contoh payload pesan yang menyediakan kode kesalahan dan pesan, yang menunjukkan mengapa Lokasi AWS IoT Core Perangkat gagal menyelesaikan informasi lokasi. Jika Anda menentukan MessageId saat memberikan data pengukuran mentah dan Lokasi AWS IoT Core Perangkat gagal menyelesaikan informasi lokasi, maka MessageId informasi yang sama akan dikembalikan dalam muatan pesan.

```

{
 "errorCode": 500,
 "errorMessage": "Internal server error",
 "messageId": "ABCD1"
}

```

## Contoh kebijakan

Berikut ini adalah contoh kebijakan yang diperlukan:

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "iot:Subscribe"
],
 "Resource": [
 "arn:aws:iot:region:account:topicfilter/$aws/device_location/customer_device_id/get_position_estimate/rejected"
]
 },
 {
 "Action": [
 "iot:Receive"
],
 "Resource": [
 "arn:aws:iot:region:account:topic/$aws/device_location/customer_device_id/get_position_estimate/rejected"
]
 }
]
}
```

## Pemecah lokasi dan muatan perangkat

Pemecah lokasi adalah algoritma yang dapat digunakan untuk menyelesaikan lokasi perangkat IoT Anda. AWS IoT Core Lokasi Perangkat mendukung pemecah lokasi berikut. Anda akan melihat contoh format payload JSON untuk jenis pengukuran ini, perangkat yang didukung oleh pemecah, dan bagaimana lokasi diselesaikan.

Untuk menyelesaikan lokasi perangkat, tentukan satu atau beberapa tipe data pengukuran ini. Satu lokasi yang diselesaikan akan dikembalikan untuk semua data pengukuran yang digabungkan.

### Topik

- [Pemecah berbasis Wi-Fi](#)



- [Pemecah berbasis seluler](#)
- [Pemecah pencarian terbalik IP](#)
- [Pemecah GNSS](#)

## Pemecah berbasis Wi-Fi

Gunakan pemecah berbasis Wi-Fi untuk menyelesaikan lokasi menggunakan informasi pemindaian dari titik akses Wi-Fi. Pemecah ini mendukung teknologi WLAN, dan dapat digunakan untuk menghitung lokasi perangkat untuk perangkat IoT umum dan perangkat nirkabel WAN. LoRa

Perangkat LoRa WAN harus memiliki chipset LoRa Edge, yang dapat memecahkan kode informasi pemindaian Wi-Fi yang masuk. LoRa Edge adalah platform daya ultra-rendah yang mengintegrasikan LoRa transceiver jarak jauh, pemindai GNSS multi-konstelasi, dan pemindai MAC Wi-Fi pasif yang menargetkan aplikasi geolokasi. Ketika pesan uplink diterima dari perangkat, data pemindaian Wi-Fi dikirim ke Lokasi AWS IoT Core Perangkat, dan lokasi diperkirakan berdasarkan hasil pemindaian Wi-Fi. Informasi yang diterjemahkan kemudian diteruskan ke pemecah berbasis Wi-Fi untuk mengambil informasi lokasi.

Contoh muatan pemecah berbasis Wi-Fi

Kode berikut menunjukkan contoh payload JSON dari perangkat yang berisi data pengukuran. Ketika Lokasi AWS IoT Core Perangkat menerima data ini sebagai input, ia mengirimkan permintaan HTTP ke penyedia pemecah untuk menyelesaikan informasi lokasi. Untuk mengambil informasi, tentukan nilai untuk Alamat MAC dan RSS (kekuatan sinyal yang diterima). Untuk melakukan ini, berikan payload JSON menggunakan format ini, atau gunakan parameter [WiFiAccessPointsobjek](#) operasi [GetPositionEstimateAPI](#).

```
{
 "Timestamp": 1664313161, // optional
 "WiFiAccessPoints": [
 {
 "MacAddress": "A0:EC:F9:1E:32:C1", // required
 "Rss": -75 // required
 }
]
}
```

## Pemecah berbasis seluler

Anda dapat menggunakan pemecah berbasis seluler untuk menyelesaikan lokasi menggunakan data pengukuran yang diperoleh dari menara radio seluler. Pemecah mendukung teknologi berikut. Satu informasi lokasi yang diselesaikan diperoleh, bahkan jika Anda menyertakan data pengukuran dari salah satu atau semua teknologi ini.

- GSM
- CDMA
- WCDMA
- TD-SCDMA
- LTE

### Contoh muatan pemecah berbasis seluler

Kode berikut menunjukkan contoh muatan JSON dari perangkat yang berisi data pengukuran seluler. Ketika Lokasi AWS IoT Core Perangkat menerima data ini sebagai input, ia mengirimkan permintaan HTTP ke penyedia pemecah untuk menyelesaikan informasi lokasi. Untuk mengambil informasi, Anda dapat memberikan payload JSON menggunakan format ini di konsol, atau menentukan nilai untuk [CellTowers](#) parameter operasi API. [GetPositionEstimate](#) Anda dapat memberikan data pengukuran dengan menentukan nilai untuk parameter menggunakan salah satu atau semua teknologi seluler ini.

#### LTE (Evolusi jangka panjang)

Saat Anda menggunakan data pengukuran ini, Anda harus menentukan informasi seperti jaringan dan kode negara jaringan seluler, dan parameter tambahan opsional termasuk informasi tentang ID lokal. Kode berikut menunjukkan contoh format payload. Untuk informasi selengkapnya tentang parameter ini, lihat [objek 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 (Sistem Global untuk Komunikasi Seluler)

Saat Anda menggunakan data pengukuran ini, Anda harus menentukan informasi seperti jaringan dan kode negara jaringan seluler, informasi stasiun pangkalan, dan parameter tambahan opsional. Kode berikut menunjukkan contoh format payload. Untuk informasi selengkapnya tentang parameter ini, lihat [objek 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 (Akses ganda pembagian kode)

Saat Anda menggunakan data pengukuran ini, Anda harus menentukan informasi seperti daya sinyal dan informasi identifikasi, informasi stasiun pangkalan, dan parameter tambahan opsional. Kode berikut menunjukkan contoh format payload. Untuk informasi selengkapnya tentang parameter ini, lihat [objek CDMA](#).

```

{
 "Timestamp": 1664313161, // optional
 "CellTowers": {
 "Cdma": [
 {
 "SystemId": int, // required
 "NetworkId": int, // required
 "BaseStationId": int, // required
 "RegistrationZone": int, // optional
 "CdmaLocalId": {
 "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 (akses ganda pembagian kode Wideband)

Saat Anda menggunakan data pengukuran ini, Anda harus menentukan informasi seperti kode jaringan dan negara, daya sinyal dan informasi identifikasi, informasi stasiun pangkalan, dan parameter tambahan opsional. Kode berikut menunjukkan contoh format payload. Untuk informasi selengkapnya tentang parameter ini, lihat [objek 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 (Akses ganda pembagian kode sinkron pembagian waktu)

Saat Anda menggunakan data pengukuran ini, Anda harus menentukan informasi seperti kode jaringan dan negara, daya sinyal dan informasi identifikasi, informasi stasiun pangkalan, dan parameter tambahan opsional. Kode berikut menunjukkan contoh format payload. Untuk informasi selengkapnya tentang parameter ini, lihat [objek 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
 }
]
 }
]
 }
}

```

## Pemecah pencarian terbalik IP

Anda dapat menggunakan pemecah pencarian terbalik IP untuk menyelesaikan lokasi menggunakan alamat IP sebagai input. Pemecah dapat memperoleh informasi lokasi dari perangkat yang telah disediakan. AWS IoT Tentukan informasi alamat IP menggunakan format yang merupakan pola standar IPv4 atau IPv6, atau pola terkompresi hex IPv6. Anda kemudian mendapatkan perkiraan lokasi yang diselesaikan, termasuk informasi tambahan seperti kota dan negara tempat perangkat berada.

### Note

Dengan menggunakan pencarian terbalik IP, Anda setuju untuk tidak menggunakannya untuk tujuan mengidentifikasi atau menemukan alamat rumah tangga atau jalan tertentu.

### Contoh payload pemecah pencarian terbalik IP

Kode berikut menunjukkan contoh payload JSON dari perangkat yang berisi data pengukuran. Saat Lokasi AWS IoT Core Perangkat menerima informasi alamat IP dalam data pengukuran, lokasi akan mencari informasi ini di database penyedia pemecah, yang kemudian digunakan untuk menyelesaikan informasi lokasi. Untuk mengambil informasi, berikan payload JSON menggunakan format ini, atau tentukan nilai untuk parameter `ip` operasi API. [GetPositionEstimate](#)

### Note

Ketika pemecah ini digunakan, kota, negara bagian, negara, dan kode pos tempat perangkat berada juga dilaporkan selain koordinat. Sebagai contoh, lihat [Menyelesaikan lokasi perangkat \(konsol\)](#).

```
{
 "Timestamp": 1664313161,
 "Ip": {
 "IpAddress": "54.240.198.35"
 }
}
```

## Pemecah GNSS

Gunakan pemecah GNSS (Global Navigation Satellite System) untuk mengambil lokasi perangkat menggunakan informasi yang terdapat dalam pesan hasil pemindaian GNSS atau pesan NAV. Anda dapat secara opsional memberikan informasi bantuan GNSS tambahan, yang mengurangi jumlah variabel yang harus digunakan pemecah untuk mencari sinyal. Dengan memberikan informasi bantuan ini, yang mencakup posisi, ketinggian, dan waktu pengambilan serta informasi akurasi, pemecah dapat dengan mudah mengidentifikasi satelit dalam pandangan dan menghitung lokasi perangkat.

Pemecah ini dapat digunakan dengan perangkat LoRa WAN, dan perangkat lain yang telah disediakan. AWS IoT Untuk perangkat IoT umum, jika perangkat mendukung estimasi lokasi menggunakan GNSS, ketika informasi pemindaian GNSS diterima dari perangkat, transceiver menyelesaikan informasi lokasi. Untuk perangkat LoRa WAN, perangkat harus memiliki chipset LoRa Edge. Ketika pesan uplink diterima dari perangkat, data pemindaian GNSS dikirim ke AWS IoT Core for LoRaWAN, dan lokasi diperkirakan berdasarkan hasil pemindaian dari transceiver.

### Contoh muatan pemecah GNSS

Kode berikut menunjukkan contoh payload JSON dari perangkat yang berisi data pengukuran. Saat Lokasi AWS IoT Core Perangkat menerima informasi pemindaian GNSS yang berisi muatan dalam data pengukuran, ia menggunakan transceiver dan informasi bantuan tambahan apa pun yang disertakan untuk mencari sinyal dan menyelesaikan informasi lokasi. Untuk mengambil informasi, berikan payload JSON menggunakan format ini, atau tentukan nilai untuk parameter [Gnss](#) dari operasi API. [GetPositionEstimate](#)

#### Note

Sebelum Lokasi AWS IoT Core Perangkat dapat menyelesaikan lokasi perangkat, Anda harus menghapus byte tujuan dari payload.

```
{
 "Timestamp": 1664313161, // optional
 "Gnss": {
 "AssistAltitude": number, // optional
 "AssistPosition": [number], // optional
 "CaptureTime": number, // optional
 "CaptureTimeAccuracy": number, // optional
 }
}
```



```
 "Payload": "string", // required
 "Use2DSolver": boolean // optional
 }
}
```

# Pesan kejadian

Bagian ini berisi informasi tentang pesan yang diterbitkan oleh AWS IoT ketika sesuatu atau pekerjaan diperbarui atau diubah. Untuk informasi tentang AWS IoT Events layanan yang memungkinkan Anda membuat detektor untuk memantau perangkat Anda untuk kegagalan atau perubahan dalam operasi, dan untuk memicu tindakan ketika terjadi, lihat [AWS IoT Events](#).

## Bagaimana pesan acara dihasilkan

AWS IoT menerbitkan pesan acara ketika peristiwa tertentu terjadi. Misalnya, peristiwa dihasilkan oleh registri ketika hal-hal ditambahkan, diperbarui, atau dihapus. Setiap peristiwa menyebabkan satu pesan acara dikirim. Pesan acara dipublikasikan melalui MQTT dengan muatan JSON. Isi muatan tergantung pada jenis acara.

### Note

Pesan acara dijamin akan dipublikasikan satu kali. Dimungkinkan bagi mereka untuk diterbitkan lebih dari sekali. Pemesanan pesan acara tidak dijamin.

## Kebijakan untuk menerima pesan acara

Untuk menerima pesan acara, perangkat Anda harus menggunakan kebijakan yang sesuai yang memungkinkannya terhubung ke gateway AWS IoT perangkat dan berlangganan topik acara MQTT. Anda juga harus berlangganan filter topik yang sesuai.

Berikut ini adalah contoh kebijakan yang diperlukan untuk menerima peristiwa siklus hidup:

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

```
 "arn:aws:iot:region:account:/$aws/events/*"
]
 }
}
```

## Aktifkan acara untuk AWS IoT

Sebelum pelanggan topik yang dicadangkan dapat menerima pesan, Anda harus mengaktifkan pesan acara dari AWS Management Console atau dengan menggunakan API atau CLI. Untuk informasi tentang pesan peristiwa yang dikelola oleh berbagai opsi, lihat [Tabel setelah konfigurasi AWS IoT acara](#).

- Untuk mengaktifkan pesan peristiwa, buka tab [Pengaturan](#) AWS IoT konsol dan kemudian, di bagian Pesan berbasis peristiwa, pilih Kelola acara. Anda dapat menentukan acara yang ingin Anda kelola.
- Untuk mengontrol jenis acara mana yang diterbitkan dengan menggunakan API atau CLI, panggil [UpdateEventConfigurations](#) API atau gunakan perintah CLI `update-event-configurations`. Sebagai contoh:

```
aws iot update-event-configurations --event-configurations "{\"THING\":{\"Enabled\":true}}"
```

### Note

Semua tanda kutip (") diloloskan dengan garis miring terbalik (\).

Anda bisa mendapatkan konfigurasi acara saat ini dengan memanggil [DescribeEventConfigurations](#) API atau dengan menggunakan perintah `describe-event-configurations` CLI. Sebagai contoh:

```
aws iot describe-event-configurations
```

## Tabel pengaturan konfigurasi AWS IoT acara

| Kategori acara<br>(AWS IoT Konsol: Pengaturan:<br>Pesan berbasis acara) | <b>eventConfigurations</b> nilai kunci<br>(AWS CLI/API) | Topik pesan acara                                                        |
|-------------------------------------------------------------------------|---------------------------------------------------------|--------------------------------------------------------------------------|
| (Hanya dapat dikonfigurasi dengan menggunakan AWS CLI/API)              | CA_CERTIFICATE                                          | <code>\$aws/events/certificates/registered/<i>caCertificateId</i></code> |
| (Hanya dapat dikonfigurasi dengan menggunakan AWS CLI/API)              | CERTIFICATE                                             | <code>\$aws/events/presence/connected/<i>clientId</i></code>             |
| (Hanya dapat dikonfigurasi dengan menggunakan AWS CLI/API)              | CERTIFICATE                                             | <code>\$aws/events/presence/disconnected/<i>clientId</i></code>          |
| (Hanya dapat dikonfigurasi dengan menggunakan AWS CLI/API)              | CERTIFICATE                                             | <code>\$aws/events/subscriptions/subscribed/<i>clientId</i></code>       |
| (Hanya dapat dikonfigurasi dengan menggunakan AWS CLI/API)              | CERTIFICATE                                             | <code>\$aws/events/subscriptions/unsubscribed/<i>clientId</i></code>     |
| Job selesai, dibatalkan                                                 | JOB                                                     | <code>\$aws/events/job/<i>jobID</i>/canceled</code>                      |
| Job selesai, dibatalkan                                                 | JOB                                                     | <code>\$aws/events/job/<i>jobID</i>/cancellation_in_progress</code>      |
| Job selesai, dibatalkan                                                 | JOB                                                     | <code>\$aws/events/job/<i>jobID</i>/completed</code>                     |
| Job selesai, dibatalkan                                                 | JOB                                                     | <code>\$aws/events/job/<i>jobID</i>/deleted</code>                       |

| Kategori acara<br>(AWS IoT Konsol: Pengaturan:<br>Pesan berbasis acara) | <b>eventConfiguration</b><br>s nilai kunci<br>(AWS CLI/API) | Topik pesan acara                                                         |
|-------------------------------------------------------------------------|-------------------------------------------------------------|---------------------------------------------------------------------------|
| Job selesai, dibatalkan                                                 | JOB                                                         | <code>\$aws/events/<br/>job/<i>jobID</i>/deletion<br/>_in_progress</code> |
| Eksekusi Job: sukses, gagal,<br>ditolak, dibatalkan, dihapus            | JOB_EXECUTION                                               | <code>\$aws/events/jobExe<br/>cution/<i>jobID</i>/canceled</code>         |
| Eksekusi Job: sukses, gagal,<br>ditolak, dibatalkan, dihapus            | JOB_EXECUTION                                               | <code>\$aws/events/jobExe<br/>cution/<i>jobID</i>/deleted</code>          |
| Eksekusi Job: sukses, gagal,<br>ditolak, dibatalkan, dihapus            | JOB_EXECUTION                                               | <code>\$aws/events/jobExe<br/>cution/<i>jobID</i>/failed</code>           |
| Eksekusi Job: sukses, gagal,<br>ditolak, dibatalkan, dihapus            | JOB_EXECUTION                                               | <code>\$aws/events/jobExe<br/>cution/<i>jobID</i>/rejected</code>         |
| Eksekusi Job: sukses, gagal,<br>ditolak, dibatalkan, dihapus            | JOB_EXECUTION                                               | <code>\$aws/events/jobExe<br/>cution/<i>jobID</i>/removed</code>          |
| Eksekusi Job: sukses, gagal,<br>ditolak, dibatalkan, dihapus            | JOB_EXECUTION                                               | <code>\$aws/events/jobExe<br/>cution/<i>jobID</i>/succeede<br/>d</code>   |
| Eksekusi Job: sukses, gagal,<br>ditolak, dibatalkan, dihapus            | JOB_EXECUTION                                               | <code>\$aws/events/jobExe<br/>cution/<i>jobID</i>/timed_ou<br/>t</code>   |
| Hal: dibuat, diperbarui,<br>dihapus                                     | THING                                                       | <code>\$aws/events/thing/<br/><i>thingName</i> /created</code>            |
| Hal: dibuat, diperbarui,<br>dihapus                                     | THING                                                       | <code>\$aws/events/thing/<br/><i>thingName</i> /updated</code>            |

|                                                                      |                                                             |                                                                                                                                                                      |
|----------------------------------------------------------------------|-------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Kategori acara<br>(AWS IoT Konsol: Pengaturan: Pesan berbasis acara) | <b>eventConfiguration</b><br>s nilai kunci<br>(AWS CLI/API) | Topik pesan acara                                                                                                                                                    |
| Hal: dibuat, diperbarui, dihapus                                     | THING                                                       | <code>\$aws/events/thing/<br/><i>thingName</i> /deleted</code>                                                                                                       |
| Grup hal: ditambahkan, dihapus                                       | THING_GROUP                                                 | <code>\$aws/events/thingG<br/>roup/ <i>thingGroupName</i> /<br/>created</code>                                                                                       |
| Grup hal: ditambahkan, dihapus                                       | THING_GROUP                                                 | <code>\$aws/events/thingG<br/>roup/ <i>thingGroupName</i> /<br/>updated</code>                                                                                       |
| Grup hal: ditambahkan, dihapus                                       | THING_GROUP                                                 | <code>\$aws/events/thingG<br/>roup/ <i>thingGroupName</i> /<br/>deleted</code>                                                                                       |
| Hirarki grup hal: ditambahkan, dihapus                               | THING_GROUP_HIERAR<br>CHY                                   | <code>\$aws/events/thingG<br/>roupHierarchy/thin<br/>gGroup/ <i>parentThi<br/>ngGroupName</i> /childThi<br/>ngGroup/ <i>childThin<br/>gGroupName</i> /added</code>   |
| Hirarki grup hal: ditambahkan, dihapus                               | THING_GROUP_HIERAR<br>CHY                                   | <code>\$aws/events/thingG<br/>roupHierarchy/thin<br/>gGroup/ <i>parentThi<br/>ngGroupName</i> /childThi<br/>ngGroup/ <i>childThin<br/>gGroupName</i> /removed</code> |

|                                                                      |                                                             |                                                                                                                |
|----------------------------------------------------------------------|-------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------|
| Kategori acara<br>(AWS IoT Konsol: Pengaturan: Pesan berbasis acara) | <b>eventConfiguration</b><br>s nilai kunci<br>(AWS CLI/API) | Topik pesan acara                                                                                              |
| Keanggotaan grup hal: ditambahkan, dihapus                           | THING_GROUP_MEMBERSHIP                                      | <code>\$aws/events/thingGroupMembership/thingGroup/<i>thingGroupName</i>/thing/<i>thingName</i>/added</code>   |
| Keanggotaan grup hal: ditambahkan, dihapus                           | THING_GROUP_MEMBERSHIP                                      | <code>\$aws/events/thingGroupMembership/thingGroup/<i>thingGroupName</i>/thing/<i>thingName</i>/removed</code> |
| Jenis hal: dibuat, diperbarui, dihapus                               | THING_TYPE                                                  | <code>\$aws/events/thingType/<i>thingTypeName</i>/created</code>                                               |
| Jenis hal: dibuat, diperbarui, dihapus                               | THING_TYPE                                                  | <code>\$aws/events/thingType/<i>thingTypeName</i>/updated</code>                                               |
| Jenis hal: dibuat, diperbarui, dihapus                               | THING_TYPE                                                  | <code>\$aws/events/thingType/<i>thingTypeName</i>/deleted</code>                                               |

| Kategori acara<br>(AWS IoT Konsol: Pengaturan:<br>Pesan berbasis acara) | <b>eventConfiguration</b><br>s nilai kunci<br>(AWS CLI/API) | Topik pesan acara                                                                                                                                                                                                                       |
|-------------------------------------------------------------------------|-------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Asosiasi jenis benda:<br>ditambahkan, dihapus                           | THING_TYPE_ASSOCIA<br>TION                                  | <pre>\$aws/events/thingT ypeAssociation/ thing/ <i>thingName</i> / thingType/ <i>thingType</i> <i>Name</i> /added  \$aws/events/thingT ypeAssociation/ thing/ <i>thingName</i> / thingType/ <i>thingType</i> <i>Name</i> /removed</pre> |

## Acara registri

Registri dapat mempublikasikan pesan peristiwa ketika sesuatu, tipe benda, dan grup hal dibuat, diperbarui, atau dihapus. Peristiwa ini, bagaimanapun, tidak tersedia secara default. Untuk informasi tentang cara mengaktifkan acara ini, lihat [Aktifkan acara untuk AWS IoT](#).

Registri dapat menyediakan jenis acara berikut:

- [Peristiwa hal](#)
- [Acara jenis hal](#)
- [Acara kelompok hal](#)

## Peristiwa hal

Hal yang Dibuat/Diperbarui/Dihapus

Registri menerbitkan pesan peristiwa berikut ketika sesuatu dibuat, diperbarui, atau dihapus:

- `$aws/events/thing/thingName/created`
- `$aws/events/thing/thingName/updated`



- `$aws/events/thing/thingName/deleted`

Pesan berisi contoh payload berikut:

```
{
 "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"
 }
}
```

Muatan berisi atribut berikut:

`eventType`

Setel ke "THING\_EVENT".

`eventId`

ID peristiwa unik (string).

`timestamp`

Stempel waktu UNIX saat peristiwa terjadi.

`operation`

Operasi yang memicu acara tersebut. Nilai yang valid adalah:

- DIBUAT
- DIPERBARUI
- DELETED

## accountId

Akun AWS ID Anda.

## thingId

ID dari benda yang sedang dibuat, diperbarui, atau dihapus.

## thingName

Nama benda yang sedang dibuat, diperbarui, atau dihapus.

## versionNumber

Versi benda yang sedang dibuat, diperbarui, atau dihapus. Nilai ini diatur ke 1 ketika sesuatu dibuat. Ini bertambah 1 setiap kali hal diperbarui.

## hal TypeName

Jenis benda yang terkait dengan hal itu, jika ada. Atau, null.

## atribut

Kumpulan pasangan nama-nilai yang terkait dengan benda tersebut.

## Acara jenis hal

Jenis hal terkait peristiwa:

- [Tipe Benda yang Dibuat/Ulang/Tidak Diusang/Dihapus](#)
- [Tipe Hal yang Terkait atau Terputus dengan Sesuatu](#)

## Tipe Benda yang Dibuat/Ulang/Tidak Diusang/Dihapus

Registri menerbitkan pesan peristiwa berikut ketika jenis sesuatu dibuat, tidak digunakan lagi, tidak digunakan lagi, atau dihapus:

- \$aws/events/thingType/*thingTypeName*/created
- \$aws/events/thingType/*thingTypeName*/updated
- \$aws/events/thingType/*thingTypeName*/deleted

Pesan berisi contoh payload berikut:

```
{
 "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"],
 "description" : "My thing type"
}
```

Muatan berisi atribut berikut:

#### eventType

Setel ke "THING\_TYPE\_EVENT".

#### eventId

ID peristiwa unik (string).

#### timestamp

Stempel waktu UNIX saat peristiwa terjadi.

#### operation

Operasi yang memicu acara tersebut. Nilai yang valid adalah:

- DIBUAT
- DIPERBARUI
- DELETED

#### accountId

Akun AWS ID Anda.

#### thingTypeId

ID dari jenis benda yang dibuat, tidak digunakan lagi, atau dihapus.

#### thingTypeName

Nama jenis benda yang dibuat, tidak digunakan lagi, atau dihapus.

## isDeprecated

`true` jika tipe benda tidak digunakan lagi. Atau, `false`.

## deprecationDate

Stempel waktu UNIX untuk saat tipe benda tidak digunakan lagi.

## searchableAttributes

Kumpulan pasangan nama-nilai yang terkait dengan jenis benda yang dapat digunakan untuk pencarian.

## deskripsi

Deskripsi jenis benda.

## Tipe Hal yang Terkait atau Terputus dengan Sesuatu

Registri menerbitkan pesan peristiwa berikut ketika jenis sesuatu dikaitkan atau dipisahkan dengan sesuatu.

- `$aws/events/thingTypeAssociation/thing/thingName/thingType/typeName/added`
- `$aws/events/thingTypeAssociation/thing/thingName/thingType/typeName/removed`

Berikut ini adalah contoh added payload. Muatan untuk removed pesan serupa.

```
{
 "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,
}
```

Muatan berisi atribut berikut:

## eventId

ID peristiwa unik (string).

## eventType

Setel ke “THING\_TYPE\_ASSOCIATION\_EVENT”.

## operation

Operasi yang memicu acara tersebut. Nilai yang valid adalah:

- DITAMBAHKAN
- DIKELUARKAN

## thingId

ID dari benda yang asosiasi tipenya diubah.

## thingName

Nama benda yang asosiasi tipenya diubah.

## hal TypeName

Jenis benda yang terkait dengan, atau tidak lagi terkait dengan, benda itu.

## timestamp

Stempel waktu UNIX saat peristiwa terjadi.

## Acara kelompok hal

Acara terkait kelompok hal:

- [Thing Group Dibuat/Diperbarui/Dihapus](#)
- [Hal yang Ditambahkan atau Dihapus dari Thing Group](#)
- [Thing Group Ditambahkan atau Dihapus dari Thing Group](#)

## Thing Group Dibuat/Diperbarui/Dihapus

Registri menerbitkan pesan peristiwa berikut ketika grup sesuatu dibuat, diperbarui, atau dihapus.

- `$aws/events/thingGroup/groupName/created`
- `$aws/events/thingGroup/groupName/updated`
- `$aws/events/thingGroup/groupName/deleted`

Berikut ini adalah contoh updated payload. Muatan untuk created dan deleted pesan serupa.

```
{
 "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
}
```

Muatan berisi atribut berikut:

eventType

Setel ke "THING\_GROUP\_EVENT".

eventId

ID peristiwa unik (string).

timestamp

Stempel waktu UNIX saat peristiwa terjadi.

operation

Operasi yang memicu acara tersebut. Nilai yang valid adalah:

- DIBUAT
- DIPERBARUI
- DELETED

accountId

Akun AWS ID Anda.

halGroupId

ID grup hal yang sedang dibuat, diperbarui, atau dihapus.

halGroupName

Nama grup benda yang sedang dibuat, diperbarui, atau dihapus.

versionNumber

Versi dari grup benda. Nilai ini diatur ke 1 ketika grup benda dibuat. Ini bertambah 1 setiap kali grup hal diperbarui.

orang tua GroupName

Nama kelompok hal induk, jika ada.

orang tua GroupId

ID dari grup hal induk, jika ada.

deskripsi

Deskripsi dari kelompok benda.

## akar ToParent ThingGroups

Sebuah array informasi tentang kelompok hal induk. Ada satu elemen untuk setiap grup hal induk, mulai dari grup root thing dan berlanjut ke induk grup benda. Setiap entri berisi kelompok benda `groupArn` dan `groupId`.

### atribut

Kumpulan pasangan nama-nilai yang terkait dengan grup benda.

## Hal yang Ditambahkan atau Dihapus dari Thing Group

Registri menerbitkan pesan peristiwa berikut ketika sesuatu ditambahkan atau dihapus dari grup sesuatu.

- `$aws/events/thingGroupMembership/thingGroup/thingGroupName/thing/thingName/added`
- `$aws/events/thingGroupMembership/thingGroup/thingGroupName/thing/thingName/removed`

Pesan berisi contoh payload berikut:

```
{
 "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"
}
```

Muatan berisi atribut berikut:

### eventType

Setel ke "THING\_GROUP\_MEMBERSHIP\_EVENT".



**eventId**

ID acara.

**timestamp**

Stempel waktu UNIX untuk saat peristiwa terjadi.

**operation**

ADDEDketika sesuatu ditambahkan ke grup sesuatu. REMOVEDketika sesuatu dihapus dari kelompok sesuatu.

**accountId**

Akun AWS ID Anda.

**groupArn**

ARN dari kelompok benda.

**groupId**

ID grup.

**thingArn**

ARN dari hal yang ditambahkan atau dihapus dari kelompok benda.

**thingId**

ID dari hal yang ditambahkan atau dihapus dari grup benda.

**membershipId**

ID yang mewakili hubungan antara benda dan kelompok benda. Nilai ini dihasilkan ketika Anda menambahkan sesuatu ke grup benda.

## Thing Group Ditambahkan atau Dihapus dari Thing Group

Registri menerbitkan pesan peristiwa berikut ketika grup sesuatu ditambahkan atau dihapus dari grup hal lain.

- `$aws/events/thingGroupHierarchy/thingGroup/parentThingGroupName/childThingGroup/childThingGroupName/added`
- `$aws/events/thingGroupHierarchy/thingGroup/parentThingGroupName/childThingGroup/childThingGroupName/removed`

Pesan berisi contoh payload berikut:

```
{
 "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"
}
```

Muatan berisi atribut berikut:

eventType

Setel ke "THING\_GROUP\_HIERARCHY\_EVENT".

eventId

ID acara.

timestamp

Stempel waktu UNIX untuk saat peristiwa terjadi.

operation

ADDEDketika sesuatu ditambahkan ke grup sesuatu. REMOVEDketika sesuatu dihapus dari kelompok sesuatu.

accountId

Akun AWS ID Anda.

hal GroupId

ID dari grup hal induk.

hal GroupName

Nama kelompok hal induk.

anak GroupId

ID grup benda anak.

anak GroupName

Nama kelompok benda anak.

## Acara Lowongan

Layanan AWS IoT Pekerjaan memublikasikan topik yang dicadangkan pada protokol MQTT saat pekerjaan tertunda, diselesaikan, atau dibatalkan, dan saat perangkat melaporkan keberhasilan atau kegagalan saat menjalankan pekerjaan. Perangkat atau aplikasi manajemen dan pemantauan dapat melacak status pekerjaan dengan berlangganan topik ini.

Cara mengaktifkan acara pekerjaan

Pesan respons dari layanan AWS IoT Jobs tidak melewati broker pesan dan mereka tidak dapat berlangganan oleh klien atau aturan lain. Untuk berlangganan pesan terkait aktivitas pekerjaan, gunakan `notify` dan topik `notify-notify-next`. Untuk informasi tentang topik pekerjaan, lihat [Topik Job](#).

Untuk diberitahu tentang pembaruan lowongan kerja, aktifkan acara pekerjaan ini dengan menggunakan AWS Management Console, atau dengan menggunakan API atau CLI. Untuk informasi selengkapnya, lihat [Aktifkan acara untuk AWS IoT](#).

Bagaimana acara pekerjaan bekerja

Karena perlu waktu untuk membatalkan atau menghapus pekerjaan, dua pesan dikirim untuk menunjukkan awal dan akhir permintaan. Misalnya, saat permintaan pembatalan dimulai, pesan dikirim ke `$aws/events/job/jobID/cancellation_in_progress` topik. Ketika permintaan pembatalan selesai, pesan dikirim ke `$aws/events/job/jobID/canceled` topik.

Proses serupa terjadi untuk permintaan penghapusan pekerjaan. Aplikasi manajemen dan pemantauan dapat berlangganan topik ini untuk melacak status pekerjaan. Untuk informasi selengkapnya tentang menerbitkan dan berlangganan topik MQTT, lihat [the section called "Protokol komunikasi perangkat"](#)

Jenis acara Job

Berikut ini menunjukkan berbagai jenis acara pekerjaan:

Job Selesai/Dibatalkan/Dihapus

Layanan AWS IoT Pekerjaan memublikasikan pesan tentang topik MQTT saat pekerjaan selesai, dibatalkan, dihapus, atau saat pembatalan atau penghapusan sedang berlangsung:

- \$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

completedPesan berisi contoh payload berikut:

```
{
 "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
 }
}
```

canceledPesan berisi contoh payload berikut.

```
{
 "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
}

```

deletedPesan berisi contoh payload berikut.

```

{
 "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"
}

```

cancellation\_in\_progressPesan berisi contoh payload berikut:

```

{
 "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"
 }

```

deletion\_in\_progress Pesan berisi contoh payload berikut:

```

{
 "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"
}

```

## Status Terminal Eksekusi Job

Layanan AWS IoT Pekerjaan memublikasikan pesan saat perangkat memperbarui eksekusi pekerjaan ke status 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`

Pesan berisi contoh payload berikut:

```
{
 "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"
 }
}
```

## Peristiwa siklus hidup

AWS IoT dapat mempublikasikan peristiwa siklus hidup pada topik MQTT. Acara ini tersedia secara default dan tidak dapat dinonaktifkan.

### Note

Pesan siklus hidup mungkin dikirim rusak. Anda mungkin menerima pesan duplikat.

Dalam topik ini:

- [Hubungkan/Putuskan acara](#)
- [Acara Berlangganan/Berhenti Berlangganan](#)

## Hubungkan/Putuskan acara

### Note

Dengan pengindeksan armada Manajemen AWS IoT Perangkat, Anda dapat mencari berbagai hal, menjalankan kueri agregat, dan membuat grup dinamis berdasarkan peristiwa Connect/Disconnect. Untuk informasi selengkapnya, lihat [Pengindeksan armada](#).

AWS IoT menerbitkan pesan ke topik MQTT berikut saat klien menghubungkan atau memutuskan sambungan:

- `$aws/events/presence/connected/clientId`— Klien yang terhubung ke broker pesan.
- `$aws/events/presence/disconnected/clientId`— Klien terputus dari broker pesan.

Berikut ini adalah daftar elemen JSON yang terkandung dalam pesan koneksi/pemutusan yang dipublikasikan ke topik. `$aws/events/presence/connected/clientId`

`clientId`

ID klien dari klien yang menghubungkan atau memutuskan sambungan.

### Note

ID klien yang berisi # atau + tidak menerima peristiwa siklus hidup.

`klien InitiatedDisconnect`

Benar jika klien memulai pemutusan. Kalau tidak, salah. Hanya ditemukan di pesan putuskan sambungan.

`disconnectReason`

Alasan mengapa klien terputus. Hanya ditemukan di pesan putuskan sambungan. Tabel berikut berisi nilai yang valid dan apakah broker akan mengirim [pesan Last Will and Testament \(LWT\)](#) ketika pemutusan terjadi.



| Putuskan alasan             | Deskripsi                                                                                                                                                                                                                                               | Broker akan mengirim pesan LWT                                            |
|-----------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------|
| AUTH_ERROR                  | Klien gagal mengautentikasi atau otorisasi gagal.                                                                                                                                                                                                       | Ya. Jika perangkat memiliki koneksi aktif sebelum menerima kesalahan ini. |
| CLIENT_INITIATED_DISCONNECT | Klien menunjukkan bahwa itu akan terputus. Klien dapat melakukan ini dengan mengirimkan paket DISCONNECT kontrol MQTT atau Close frame jika klien menggunakan koneksi. WebSocket                                                                        | Tidak.                                                                    |
| CLIENT_ERROR                | Klien melakukan sesuatu yang salah yang menyebabkannya terputus. Misalnya, klien akan terputus karena mengirim lebih dari 1 CONNECT paket MQTT pada koneksi yang sama atau jika klien mencoba mempublikasikan dengan muatan yang melebihi batas muatan. | Ya.                                                                       |
| CONNECTION_LOST             | Koneksi client-server terputus. Hal ini dapat terjadi selama periode latensi jaringan tinggi atau ketika koneksi internet terputus.                                                                                                                     | Ya.                                                                       |
| DUPLICATE_CLIENTID          | Klien menggunakan ID klien yang sudah digunakan. Dalam hal ini, klien yang sudah terhubung akan terputus dengan alasan pemutusan ini.                                                                                                                   | Ya.                                                                       |
| FORBIDDEN_ACCESS            | Klien tidak diizinkan untuk terhubung. Misalnya, klien dengan alamat IP yang ditolak akan gagal terhubung.                                                                                                                                              | Ya. Jika perangkat memiliki koneksi aktif sebelum menerima kesalahan ini. |

| Putuskan alasan             | Deskripsi                                                                                       | Broker akan mengirim pesan LWT |
|-----------------------------|-------------------------------------------------------------------------------------------------|--------------------------------|
| MQTT_KEEP_ALIVE_TIMEOUT     | Jika tidak ada komunikasi client-server untuk 1.5x dari waktu keep-alive klien, klien terputus. | Ya.                            |
| SERVER_ERROR                | Terputus karena masalah server yang tidak terduga.                                              | Ya.                            |
| SERVER_INITIATED_DISCONNECT | Server sengaja memutus klien karena alasan operasional.                                         | Ya.                            |
| THROTTLED                   | Klien terputus karena melebihi batas pelambatan.                                                | Ya.                            |
| WEBSOCKET_TTL_EXPIRATION    | Klien terputus karena WebSocket telah terhubung lebih lama dari time-to-live nilainya.          | Ya.                            |
| CUSTOMAUTH_TTL_EXPIRATION   | Klien terputus karena telah terhubung lebih lama dari time-to-live nilai otorisasi kustomnya.   | Ya.                            |

### eventType

Jenis peristiwa. Nilai-nilai yang valid adalah `connected` atau `disconnected`.

### ipAddress

Alamat IP dari klien penghubung. Ini bisa dalam format IPv4 atau IPv6. Ditemukan dalam pesan koneksi saja.

### principalIdentifier

Kredensi yang digunakan untuk mengautentikasi. Untuk sertifikat otentikasi timbal balik TLS, ini adalah ID sertifikat. Untuk koneksi lain, ini adalah kredensi IAM.

## sessionIdentifier

Pengidentifikasi unik secara global AWS IoT yang ada selama masa sesi.

## timestamp

Perkiraan kapan peristiwa itu terjadi.

## versionNumber

Nomor versi untuk acara siklus hidup. Ini adalah nilai integer panjang yang meningkat secara monoton untuk setiap koneksi ID klien. Nomor versi dapat digunakan oleh pelanggan untuk menyimpulkan urutan peristiwa siklus hidup.

### Note

Pesan sambungkan dan putus sambungan untuk koneksi klien memiliki nomor versi yang sama.

Nomor versi mungkin melewati nilai dan tidak dijamin akan meningkat secara konsisten sebesar 1 untuk setiap acara.

Jika klien tidak terhubung selama kurang lebih satu jam, nomor versi diatur ulang ke 0.

Untuk sesi persisten, nomor versi disetel ulang ke 0 setelah klien terputus lebih lama dari yang dikonfigurasi time-to-live (TTL) untuk sesi persisten.

Pesan connect memiliki struktur berikut.

```
{
 "clientId": "186b5",
 "timestamp": 1573002230757,
 "eventType": "connected",
 "sessionIdentifier": "a4666d2a7d844ae4ac5d7b38c9cb7967",
 "principalIdentifier": "12345678901234567890123456789012",
 "ipAddress": "192.0.2.0",
 "versionNumber": 0
}
```

Pesan pemutusan memiliki struktur berikut.

```
{
 "clientId": "186b5",
 "timestamp": 1573002340451,
```

```
"eventType": "disconnected",
"sessionIdentifier": "a4666d2a7d844ae4ac5d7b38c9cb7967",
"principalIdentifier": "12345678901234567890123456789012",
"clientInitiatedDisconnect": true,
"disconnectReason": "CLIENT_INITIATED_DISCONNECT",
"versionNumber": 0
}
```

## Menangani pemutusan klien

Praktik terbaik adalah selalu menerapkan status tunggu untuk peristiwa siklus hidup, termasuk pesan [Kehendak Terakhir dan Perjanjian \(LWT\)](#). Ketika pesan pemutusan diterima, kode Anda harus menunggu jangka waktu tertentu dan memverifikasi perangkat masih offline sebelum mengambil tindakan. Salah satu cara untuk melakukannya adalah dengan menggunakan [SQS Delay Queues](#). Saat klien menerima LWT atau peristiwa siklus hidup, Anda dapat mengantrekan pesan (misalnya, selama 5 detik). Ketika pesan itu tersedia dan diproses (oleh Lambda atau layanan lain), Anda dapat terlebih dahulu memeriksa apakah perangkat masih offline sebelum mengambil tindakan lebih lanjut.

## Acara Berlangganan/Berhenti Berlangganan

AWS IoT menerbitkan pesan ke topik MQTT berikut ketika klien berlangganan atau berhenti berlangganan topik MQTT:

```
$aws/events/subscriptions/subscribed/clientId
```

atau

```
$aws/events/subscriptions/unsubscribed/clientId
```

Di `clientId` mana ID klien MQTT yang terhubung ke broker pesan. AWS IoT

Pesan yang dipublikasikan untuk topik ini memiliki struktur sebagai berikut:

```
{
 "clientId": "186b5",
 "timestamp": 1460065214626,
 "eventType": "subscribed" | "unsubscribed",
 "sessionIdentifier": "00000000-0000-0000-0000-000000000000",
 "principalIdentifier": "000000000000/ABCDEFGHIJKLMNOPQRSTU:some-user/
ABCDEFGHIJKLMNOPQRSTU:some-user",
}
```

```
"topics" : ["foo/bar", "device/data", "dog/cat"]
}
```

Berikut ini adalah daftar elemen JSON yang terkandung dalam pesan berlangganan dan berhenti berlangganan yang diterbitkan ke dan topik. `$aws/events/subscriptions/subscribed/clientId` `$aws/events/subscriptions/unsubscribed/clientId`

#### clientId

ID klien dari klien berlangganan atau berhenti berlangganan.

#### Note

ID klien yang berisi # atau + tidak menerima peristiwa siklus hidup.

#### eventType

Jenis peristiwa. Nilai-nilai yang valid adalah `subscribed` atau `unsubscribed`.

#### principalIdentifier

Kredensi yang digunakan untuk mengautentikasi. Untuk sertifikat otentikasi timbal balik TLS, ini adalah ID sertifikat. Untuk koneksi lain, ini adalah kredensi IAM.

#### sessionIdentifier

Pengidentifikasi unik secara global AWS IoT yang ada selama masa sesi.

#### timestamp

Perkiraan kapan peristiwa itu terjadi.


#### topik

Sebuah array dari topik MQTT yang klien telah berlangganan.

#### Note

Pesan siklus hidup mungkin dikirim rusak. Anda mungkin menerima pesan duplikat.

# Pemecahan masalah AWS IoT

 Bantu kami meningkatkan topik ini

[Beritahu kami apa yang akan membantu membuatnya lebih baik](#)

Informasi berikut dapat membantu Anda memecahkan masalah umum di AWS IoT.

## Tugas

- [AWS IoT Core panduan pemecahan masalah](#)
- [AWS IoT Panduan pemecahan masalah Device Advisor](#)
- [AWS IoT Device Management panduan pemecahan masalah](#)
- [AWS IoT kesalahan](#)

## AWS IoT Core panduan pemecahan masalah

 Bantu kami meningkatkan topik ini

[Beritahu kami apa yang akan membantu membuatnya lebih baik](#)

Ini adalah bagian pemecahan masalah untuk AWS IoT Core

## Topik

- [Mendiagnosis masalah konektivitas](#)
- [Mendiagnosis masalah aturan](#)
- [Mendiagnosis masalah dengan bayangan](#)
- [Mendiagnosis masalah tindakan aliran input IoT Salesforce](#)
- [Mendiagnosis Batas Stream](#)
- [Pemecahan masalah armada perangkat terputus](#)

## Mendiagnosis masalah konektivitas

 Bantu kami meningkatkan topik ini

[Beritahu kami apa yang akan membantu membuatnya lebih baik](#)

Koneksi yang sukses untuk AWS IoT membutuhkan:

- Koneksi yang valid
- Sertifikat yang valid dan aktif
- Kebijakan yang memungkinkan koneksi dan operasi yang diinginkan

### Koneksi

Bagaimana cara menemukan titik akhir yang benar?

- Yang `endpointAddress` dikembalikan oleh `aws iot describe-endpoint --endpoint-type iot:Data-ATS`

atau

- Yang `domainName` dikembalikan oleh `aws iot describe-domain-configuration --domain-configuration-name "domain_configuration_name"`

Bagaimana cara menemukan nilai Server Name Indication (SNI) yang benar?

Nilai SNI yang benar adalah `endpointAddress` dikembalikan oleh [describe-endpoint](#) atau `domainName` dikembalikan oleh [describe-domain-configuration](#) perintah. Ini alamat yang sama dengan titik akhir pada langkah sebelumnya. Saat menghubungkan perangkat ke AWS IoT Core, klien dapat mengirim [ekstensi Server Name Indication \(SNI\)](#), yang tidak diperlukan tetapi sangat disarankan. Untuk menggunakan fitur seperti [pendaftaran multi-akun](#), [domain khusus](#), dan [titik akhir VPC](#), Anda harus menggunakan ekstensi SNI. Untuk informasi selengkapnya, lihat [Keamanan Transportasi di AWS IoT](#).

Bagaimana cara mengatasi masalah konektivitas yang berlanjut?

Anda dapat menggunakan AWS Device Advisor untuk membantu memecahkan masalah. [Pengujian bawaan Device Advisor membantu Anda memvalidasi perangkat lunak perangkat Anda terhadap praktik terbaik untuk penggunaan TLS, MQTT, AWS IoT Device Shadow, dan Pekerjaan.AWS IoT](#)

Berikut ini tautan ke konten [Device Advisor](#) yang ada.

## Autentikasi

Perangkat harus [diautentikasi](#) untuk terhubung ke titik AWS IoT akhir. Untuk perangkat yang digunakan [Sertifikat klien X.509](#) untuk otentikasi, sertifikat harus terdaftar AWS IoT dan aktif.

Bagaimana cara perangkat saya mengautentikasi titik AWS IoT akhir?

Tambahkan sertifikat AWS IoT CA ke toko kepercayaan klien Anda. Lihat dokumentasi tentang [Otentikasi Server di AWS IoT Core](#) dan kemudian ikuti tautan untuk mengunduh sertifikat CA yang sesuai.

Apa yang diperiksa saat perangkat terhubung AWS IoT?

Ketika perangkat mencoba untuk terhubung ke AWS IoT:

1. AWS IoT memeriksa sertifikat yang valid dan nilai Server Name Indication (SNI).
2. AWS IoT memeriksa untuk melihat bahwa sertifikat yang digunakan terdaftar di AWS IoT Akun dan telah diaktifkan.
3. Saat perangkat mencoba melakukan tindakan apa pun AWS IoT, seperti berlangganan atau memublikasikan pesan, kebijakan yang dilampirkan pada sertifikat yang digunakan untuk menyambungkannya akan diperiksa untuk mengonfirmasi bahwa perangkat berwenang untuk melakukan tindakan tersebut.

Bagaimana saya bisa memvalidasi sertifikat yang dikonfigurasi dengan benar?

Gunakan perintah `s_client` OpenSSL untuk menguji koneksi ke titik akhir: AWS IoT

```
openssl s_client -connect custom_endpoint.iot.aws-region.amazonaws.com:8443 -
CAfile CA.pem -cert cert.pem -key privateKey.pem
```

Untuk informasi selengkapnya tentang penggunaan `openssl s_client`, lihat dokumentasi [s\\_client OpenSSL](#).

Bagaimana cara memeriksa status sertifikat?

- Buat daftar sertifikat

Jika Anda tidak mengetahui ID sertifikat, Anda dapat melihat status semua sertifikat Anda dengan menggunakan `aws iot list-certificates` perintah.

- Tampilkan detail sertifikat



Jika Anda mengetahui ID sertifikat, perintah ini menunjukkan informasi lebih rinci tentang sertifikat.

```
aws iot describe-certificate --certificate-id "certificateId"
```

- Tinjau sertifikat di AWS IoT Konsol

Di [AWS IoT konsol](#), di menu sebelah kiri, pilih Aman, lalu pilih Sertifikat.

Pilih sertifikat yang Anda gunakan untuk terhubung dari daftar untuk membuka halaman detailnya.

Di halaman detail sertifikat, Anda dapat melihat statusnya saat ini.

Status sertifikat dapat diubah dengan menggunakan menu Tindakan di sudut kanan atas halaman detail.

## Otorisasi

AWS IoT sumber daya digunakan [AWS IoT Core kebijakan](#) untuk mengotorisasi sumber daya tersebut untuk melakukan [tindakan](#). Agar suatu tindakan diotorisasi, AWS IoT sumber daya yang ditentukan harus memiliki dokumen kebijakan yang dilampirkan padanya yang memberikan izin untuk melakukan tindakan itu.

Saya menerima PUBNACK atau SUBNACK tanggapan dari broker. Apa yang harus saya lakukan?

Pastikan ada kebijakan yang dilampirkan pada sertifikat yang Anda gunakan untuk menelepon AWS IoT. Semua operasi terbitkan/berlangganan ditolak secara default.

Pastikan kebijakan terlampir mengotorisasi [tindakan](#) yang Anda coba lakukan.

Pastikan kebijakan terlampir mengotorisasi [sumber daya](#) yang mencoba melakukan tindakan resmi.

Saya memiliki entri AUTHORIZATION\_FAILURE di log saya.

Pastikan ada kebijakan yang dilampirkan pada sertifikat yang Anda gunakan untuk menelepon AWS IoT. Semua operasi terbitkan/berlangganan ditolak secara default.

Pastikan kebijakan terlampir mengotorisasi [tindakan](#) yang Anda coba lakukan.

Pastikan kebijakan terlampir mengotorisasi [sumber daya](#) yang mencoba melakukan tindakan resmi.

Bagaimana cara memeriksa apa yang diizinkan oleh kebijakan?

Di [AWS IoT konsol](#), di menu sebelah kiri, pilih Aman, lalu pilih Sertifikat.

Pilih sertifikat yang Anda gunakan untuk terhubung dari daftar untuk membuka halaman detailnya.

Di halaman detail sertifikat, Anda dapat melihat statusnya saat ini.

Di menu sebelah kiri halaman detail sertifikat, pilih Kebijakan untuk melihat kebijakan yang dilampirkan pada sertifikat.

Pilih kebijakan yang diinginkan untuk melihat halaman detailnya.

Di halaman detail kebijakan, tinjau dokumen Kebijakan kebijakan untuk melihat apa yang diotorisasi.

Pilih Edit dokumen kebijakan untuk membuat perubahan pada dokumen kebijakan.

## Keamanan dan identitas

Ketika Anda memberikan sertifikat server untuk konfigurasi domain AWS IoT kustom, sertifikat memiliki maksimal empat nama domain.

Untuk informasi lebih lanjut, lihat [AWS IoT Core kuota dan titik akhir](#).

## Mendiagnosis masalah aturan

 Bantu kami meningkatkan topik ini

[Beri tahu kami apa yang akan membantu membuatnya lebih baik](#)

Bagian ini menjelaskan beberapa hal yang harus diperiksa ketika Anda mengalami masalah dengan aturan.

## Mengkonfigurasi CloudWatch Log untuk pemecahan masalah

Cara terbaik untuk men-debug masalah yang Anda alami dengan aturan adalah dengan menggunakan CloudWatch Log. Saat Anda mengaktifkan CloudWatch Log for AWS IoT, Anda dapat

melihat aturan mana yang dipicu dan keberhasilan atau kegagalannya. Anda juga mendapatkan informasi tentang apakah kondisi klausa WHERE cocok. Untuk informasi selengkapnya, lihat [Monitor AWS IoT menggunakan CloudWatch Log](#).

Masalah aturan yang paling umum adalah otorisasi. Log menunjukkan jika peran Anda tidak diizinkan untuk dilakukan AssumeRole pada sumber daya. Berikut adalah contoh log yang dihasilkan oleh logging [berbutir halus](#):

```
{
 "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": "ABCDEFGH1234567ABCD890: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)"
}
```

Berikut adalah contoh log serupa yang dihasilkan oleh [pencatatan global](#):

```
2017-12-09 22:49:17.954 TRACEID:ff562535-6964-506a-e141-78d40375fc4e
PRINCIPALID:ABCDEFGH1234567ABCD890: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.
```

Untuk informasi selengkapnya, lihat [the section called “Melihat AWS IoT log di CloudWatch konsol”](#).

## Mendiagnosis layanan eksternal

Layanan eksternal dikendalikan oleh pengguna akhir. Sebelum eksekusi aturan, pastikan bahwa layanan eksternal yang telah Anda tautkan ke aturan sudah disiapkan dan memiliki unit throughput dan kapasitas yang cukup untuk aplikasi Anda.

## Mendiagnosis masalah SQL

Jika kueri SQL Anda tidak mengembalikan data yang Anda harapkan:


- Tinjau log untuk pesan kesalahan.
- Konfirmasikan bahwa sintaks SQL Anda cocok dengan dokumen JSON dalam pesan.

Tinjau nama objek dan properti yang digunakan dalam kueri dengan yang digunakan dalam dokumen JSON dari muatan pesan topik. Untuk informasi selengkapnya tentang pemformatan JSON dalam kueri SQL, lihat. [Ekstensi JSON](#)

- Periksa untuk melihat apakah objek JSON atau nama properti menyertakan karakter cadangan atau numerik.

Untuk informasi selengkapnya tentang karakter yang dicadangkan dalam referensi objek JSON dalam kueri SQL, lihat. [Ekstensi JSON](#)

## Mendiagnosis masalah dengan bayangan

 Bantu kami meningkatkan topik ini

[Beritahu kami apa yang akan membantu membuatnya lebih baik](#)

## Mendiagnosis bayangan

| Isu                                                                                                                    | Pedoman pemecahan masalah                                                                                                                                                                                                                                                                                                                      |
|------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Dokumen bayangan perangkat ditolak dengan <code>Invalid JSON document</code> .                                         | Jika Anda tidak terbiasa dengan JSON, ubah contoh yang disediakan dalam panduan ini untuk Anda gunakan sendiri. Untuk informasi selengkapnya, lihat <a href="#">Contoh dokumen bayangan</a> .                                                                                                                                                  |
| Saya mengirimkan JSON yang benar, tetapi tidak ada atau hanya sebagian yang disimpan dalam dokumen bayangan perangkat. | Pastikan Anda mengikuti pedoman pemformatan JSON. Hanya bidang JSON di <code>reported</code> bagian <code>desired</code> dan yang disimpan. Konten JSON (meskipun secara formal benar) di luar bagian tersebut diabaikan.                                                                                                                      |
| Saya menerima kesalahan bahwa bayangan perangkat melebihi ukuran yang diizinkan.                                       | Bayangan perangkat hanya mendukung 8 KB data. Coba perpendek nama bidang di dalam dokumen JSON Anda atau cukup buat lebih banyak bayangan dengan membuat lebih banyak hal. Perangkat dapat memiliki jumlah hal/bayangan yang tidak terbatas yang terkait dengannya. Satu-satunya persyaratan adalah bahwa setiap nama harus unik di akun Anda. |
| Ketika saya menerima bayangan perangkat, itu lebih besar dari 8 KB. Bagaimana ini bisa terjadi?                        | Setelah diterima, AWS IoT layanan menambahkan metadata ke bayangan perangkat. Layanan ini mencakup data ini dalam tanggapannya, tetapi tidak dihitung terhadap batas 8 KB. Hanya data untuk <code>desired</code> dan <code>reported</code> status di dalam dokumen status yang dikirim ke bayangan perangkat yang dihitung menuju batas.       |
| Permintaan saya ditolak karena versi yang salah. Apa yang harus saya lakukan?                                          | Lakukan operasi GET untuk menyinkronkan ke versi dokumen status terbaru. Saat menggunakan MQTT, berlangganan file <code>./update/diterima</code> topik untuk diberitahu tentang perubahan status                                                                                                                                               |

| Isu                                                                                                                                                                                         | Pedoman pemecahan masalah                                                                                                                                                                                                                                                                                                |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                                                                                                                                                                             | dan menerima versi terbaru dari dokumen JSON.                                                                                                                                                                                                                                                                            |
| Stempel waktu dimatikan beberapa detik.                                                                                                                                                     | Stempel waktu untuk bidang individual dan seluruh dokumen JSON diperbarui saat dokumen diterima oleh AWS IoT layanan atau saat dokumen negara dipublikasikan ke file. / update/diterima dan. /perbarui/pesan delta. Pesan dapat ditunda melalui jaringan, yang dapat menyebabkan stempel waktu dimatikan beberapa detik. |
| Perangkat saya dapat mempublikasikan dan berlangganan topik bayangan yang sesuai, tetapi ketika saya mencoba memperbarui dokumen bayangan melalui HTTP REST API, saya mendapatkan HTTP 403. | Pastikan Anda telah membuat kebijakan di IAM untuk mengizinkan akses ke topik ini dan untuk tindakan yang sesuai (UPDATE/GET/DELETE) untuk kredensial yang Anda gunakan. Kebijakan IAM dan kebijakan sertifikat bersifat independen.                                                                                     |
| Masalah lainnya.                                                                                                                                                                            | Layanan Device Shadow mencatat kesalahan ke CloudWatch Log. Untuk mengidentifikasi masalah perangkat dan konfigurasi, aktifkan CloudWatch Log dan lihat log untuk informasi debug.                                                                                                                                       |

## Mendiagnosis masalah tindakan aliran input IoT Salesforce

 Bantu kami meningkatkan topik ini

[Beritahu kami apa yang akan membantu membuatnya lebih baik](#)

## Jejak eksekusi

Bagaimana cara melihat jejak eksekusi tindakan Salesforce?

Lihat [Monitor AWS IoT menggunakan CloudWatch Log](#) bagian. Setelah Anda mengaktifkan log, Anda dapat melihat jejak eksekusi tindakan Salesforce.

## Keberhasilan dan kegagalan tindakan

Bagaimana cara memeriksa bahwa pesan telah berhasil dikirim ke aliran input IoT Salesforce?

Lihat log yang dihasilkan oleh eksekusi tindakan Salesforce di CloudWatch Log. Jika Anda melihat `Action executed successfully`, maka itu berarti bahwa mesin AWS IoT aturan menerima konfirmasi dari Salesforce IoT bahwa pesan berhasil didorong ke aliran input yang ditargetkan.

Jika Anda mengalami masalah dengan platform IoT Salesforce, hubungi dukungan IoT Salesforce.

Apa yang harus saya lakukan jika pesan belum berhasil dikirim ke aliran input IoT Salesforce?

Lihat log yang dihasilkan oleh eksekusi tindakan Salesforce di CloudWatch Log. Tergantung pada entri log, Anda dapat mencoba tindakan berikut:

`Failed to locate the host`

Periksa apakah `url` parameter tindakan sudah benar dan aliran input IoT Salesforce Anda ada.

`Received Internal Server Error from Salesforce`

Coba lagi. Jika masalah berlanjut, hubungi Salesforce IoT Support.

`Received Bad Request Exception from Salesforce`

Periksa muatan yang Anda kirim untuk kesalahan.

`Received Unsupported Media Type Exception from Salesforce`

Salesforce IoT tidak mendukung payload biner saat ini. Periksa apakah Anda mengirim muatan JSON.

`Received Unauthorized Exception from Salesforce`

Periksa apakah token parameter tindakan sudah benar dan token Anda masih valid.

## Received Not Found Exception from Salesforce

Periksa apakah url parameter tindakan sudah benar dan aliran input IoT Salesforce Anda ada.

Jika Anda menerima kesalahan yang tidak tercantum di sini, hubungi AWS IoT Support.

## Mendiagnosis Batas Stream

Pemecahan masalah “Batas streaming terlampaui untuk akun Anda” AWS

Jika Anda melihat "Error: You have exceeded the limit for the number of streams in your AWS account.", Anda dapat membersihkan aliran yang tidak terpakai di akun Anda alih-alih meminta kenaikan batas.

Untuk membersihkan aliran yang tidak digunakan yang Anda buat menggunakan AWS CLI atau SDK:

```
aws iot delete-stream --stream-id value
```

Untuk detail selengkapnya, lihat [delete-stream](#).

### Note

Anda dapat menggunakan `list-streams` perintah untuk menemukan ID aliran.

## Pemecahan masalah armada perangkat terputus

### Bantu kami meningkatkan topik ini

[Beri tahu kami apa yang akan membantu membuatnya lebih baik](#)

AWS IoT pemutusan armada perangkat dapat terjadi karena berbagai alasan. Artikel ini menjelaskan cara mendiagnosis alasan pemutusan sambungan dan cara menangani pemutusan yang disebabkan oleh pemeliharaan AWS IoT layanan rutin atau batas pelambatan.

Untuk mendiagnosis alasan pemutusan



Anda dapat memeriksa grup log [AWS IoT Logs V2 CloudWatch](#) untuk mengidentifikasi alasan pemutusan di `disconnectReason` bidang entri log.

Anda juga dapat menggunakan AWS IoT fitur [peristiwa siklus hidup](#) untuk mengidentifikasi alasan pemutusan sambungan. Jika Anda telah berlangganan [peristiwa pemutusan \(`\$aws/events/presence/disconnected/clientId`\) siklus hidup](#), Anda akan mendapatkan pemberitahuan AWS IoT saat pemutusan terjadi. Anda dapat mengidentifikasi alasan pemutusan di `disconnectReason` bidang notifikasi.

Untuk informasi selengkapnya, lihat [entri CloudWatch AWS IoT log](#) dan peristiwa [Siklus Hidup](#).

Untuk memecahkan masalah pemutusan karena pemeliharaan layanan AWS IoT

Pemutusan yang disebabkan oleh AWS IoT pemeliharaan layanan dicatat sebagai peristiwa siklus hidup `SERVER_INITIATED_DISCONNECT` dalam AWS IoT dan CloudWatch. Untuk menangani pemutusan ini, sesuaikan pengaturan sisi klien Anda untuk memastikan perangkat Anda dapat terhubung kembali secara otomatis ke platform. AWS IoT

Untuk memecahkan masalah pemutusan karena batas pelambatan

Pemutusan yang disebabkan oleh batas pelambatan dicatat seperti peristiwa siklus hidup `THROTTLED` dalam AWS IoT dan CloudWatch. Untuk menangani pemutusan ini, Anda dapat meminta [peningkatan batas broker pesan](#) seiring bertambahnya jumlah perangkat.

Untuk informasi selengkapnya, lihat [Broker Pesan AWS IoT Inti](#).

## AWS IoT Panduan pemecahan masalah Device Advisor

 Bantu kami meningkatkan topik ini

[Beri tahu kami apa yang akan membantu membuatnya lebih baik](#)

Umum

T: Dapatkah saya menjalankan beberapa suite pengujian secara paralel?

J: Ya. Device Advisor sekarang mendukung menjalankan beberapa rangkaian pengujian pada perangkat yang berbeda menggunakan titik akhir tingkat Perangkat. Jika Anda menggunakan titik

akhir tingkat Akun, Anda dapat menjalankan satu suite pada satu waktu karena satu titik akhir Device Advisor tersedia per akun. Untuk informasi selengkapnya, lihat [Mengonfigurasi perangkat Anda](#).

T: Saya melihat dari perangkat saya bahwa koneksi TLS ditolak oleh Device Advisor. Apakah ini diharapkan?

J: Ya. Device Advisor menyangkal koneksi TLS sebelum dan sesudah setiap pengujian dijalankan. Kami menyarankan agar pengguna menerapkan mekanisme coba ulang perangkat untuk memiliki pengalaman pengujian yang sepenuhnya otomatis dengan Device Advisor. Jika Anda menjalankan rangkaian pengujian dengan lebih dari satu kasus uji, misalnya TLS connect, MQTT connect, dan MQTT publish, maka sebaiknya Anda memiliki mekanisme yang dibuat untuk perangkat Anda. Mekanisme ini dapat mencoba terhubung ke titik akhir pengujian kami setiap 5 detik selama satu hingga dua menit. Dengan cara ini Anda dapat menjalankan beberapa kasus uji secara berurutan secara otomatis.

T: Dapatkah saya mendapatkan riwayat panggilan Device Advisor API yang dilakukan di akun saya untuk tujuan analisis keamanan dan pemecahan masalah operasional?

J: Ya. Untuk menerima riwayat panggilan Device Advisor API yang dilakukan di akun Anda, Anda cukup mengaktifkan Konsol AWS IoT Manajemen dan memfilter sumber acara yang akan menjadi `iotdeviceadvisor.amazonaws.com`. CloudTrail

T: Bagaimana cara melihat log Device Advisor? CloudWatch

J: Log yang dihasilkan selama rangkaian pengujian dijalankan akan diunggah CloudWatch jika Anda menambahkan kebijakan yang diperlukan (misalnya, `CloudWatchFullAccess`) ke peran layanan Anda (lihat [Menyiapkan](#)). Jika setidaknya ada satu kasus uji dalam rangkaian pengujian, grup log `aws/iot/deviceadvisor/$` dibuat `testSuiteId` dengan dua aliran log. Satu aliran adalah `$testRunId` dan menyertakan log tindakan yang diambil sebelum dan sesudah menjalankan kasus pengujian di rangkaian pengujian Anda, seperti langkah persiapan dan pembersihan. Aliran log lainnya adalah `$ suiteRunId _$testRunId,` yang khusus untuk menjalankan rangkaian pengujian. Peristiwa dikirim dari perangkat dan AWS IoT Core akan dicatat ke aliran log ini.

T: Apa tujuan dari peran izin perangkat?

A: Device Advisor berdiri di antara perangkat uji Anda dan AWS IoT Core untuk mensimulasikan skenario pengujian. Ini menerima koneksi dan pesan dari perangkat pengujian Anda dan meneruskannya ke AWS IoT Core dengan mengasumsikan peran izin perangkat Anda dan memulai koneksi atas nama Anda. Penting untuk memastikan izin peran perangkat sama dengan izin pada sertifikat yang Anda gunakan untuk menjalankan pengujian. AWS IoT Kebijakan

sertifikat tidak diberlakukan saat Device Advisor memulai koneksi ke AWS IoT Core atas nama Anda dengan menggunakan peran izin perangkat. Namun, izin dari peran izin perangkat yang Anda tetapkan diberlakukan.

T: Di Wilayah apa Device Advisor didukung?

J: Device Advisor didukung di Wilayah us-east-1, us-west-2, ap-northeast-1, dan eu-west-1.

T: Mengapa saya melihat hasil yang tidak konsisten?

J: Salah satu penyebab utama hasil yang tidak konsisten adalah menetapkan tes EXECUTION\_TIMEOUT ke nilai yang terlalu rendah. Untuk informasi selengkapnya tentang EXECUTION\_TIMEOUT nilai yang direkomendasikan dan default, lihat [Kasus pengujian Device Advisor](#).

T: Protokol MQTT apa yang didukung Device Advisor?

A: Device Advisor mendukung MQTT Versi 3.1.1 dengan sertifikat klien X509.

T: Bagaimana jika kasus pengujian saya gagal dengan pesan habis waktu eksekusi meskipun saya mencoba menghubungkan perangkat saya ke titik akhir pengujian?

J: Validasi semua langkah di bawah [Buat peran IAM untuk digunakan sebagai peran perangkat Anda](#). Jika pengujian masih gagal, bisa jadi perangkat tidak mengirimkan ekstensi Server Name Indication (SNI) yang benar, yang diperlukan agar Device Advisor berfungsi. Nilai SNI yang benar adalah alamat endpoint yang dikembalikan saat mengikuti bagian [Configure your device](#). AWS IoT juga mengharuskan perangkat untuk mengirim ekstensi Server Name Indication (SNI) ke protokol Transport Layer Security (TLS). Untuk informasi selengkapnya, lihat [Keamanan transportasi di AWS IoT](#).

T: Koneksi MQTT saya gagal dengan kesalahan "libaws-c-mqtt:

AWS\_ERROR\_MQTT\_UNEXPECTED\_HANGUP" (atau) koneksi MQTT perangkat saya secara otomatis terputus dari titik akhir Device Advisor. Bagaimana kesalahan ini bisa diatasi?


J: Kode kesalahan khusus ini dan pemutusan yang tidak terduga dapat disebabkan oleh banyak hal yang berbeda, tetapi kemungkinan besar terkait dengan [peran perangkat](#) yang terpasang pada perangkat. Pos pemeriksaan di bawah ini (dalam urutan prioritas) akan menyelesaikan masalah ini.

- Peran perangkat yang terpasang pada perangkat harus memiliki izin IAM minimum yang diperlukan untuk menjalankan pengujian. Device Advisor akan menggunakan peran perangkat terlampir untuk melakukan tindakan AWS IoT MQTT atas nama perangkat uji. Jika izin yang diperlukan tidak ada, maka AWS\_ERROR\_MQTT\_UNEXPECTED\_HANGUP kesalahan akan terlihat

atau pemutusan tak terduga akan terjadi saat perangkat mencoba terhubung ke titik akhir Device Advisor. Misalnya, jika Anda memilih untuk menjalankan kasus uji MQTT Publish, tindakan Connect dan Publish harus disertakan dalam peran dengan yang sesuai ClientId dan Topik (Anda dapat memberikan beberapa nilai dengan menggunakan koma untuk memisahkan nilai, dan Anda dapat memberikan nilai awalan menggunakan karakter wildcard (\*). Misalnya: Untuk memberikan izin untuk mempublikasikan topik apa pun yang dimulai TestTopic, Anda dapat memberikan "TestTopic\*" sebagai nilai sumber daya. Berikut adalah beberapa [contoh kebijakan](#).

- Ketidakcocokan antara nilai yang ditentukan dalam peran perangkat untuk jenis sumber daya Anda dan nilai aktual yang digunakan dalam kode. Misalnya: Ketidakcocokan dalam ClientId didefinisikan dalam peran dan aktual yang ClientId digunakan dalam kode perangkat Anda. Nilai seperti ClientId, Topik, dan TopicFilter harus identik dalam peran perangkat dan kode.
- Sertifikat perangkat yang dilampirkan ke perangkat Anda harus aktif dan memiliki [kebijakan](#) yang dilampirkan dengan [izin tindakan](#) yang diperlukan untuk [sumber daya](#). Perhatikan bahwa, kebijakan sertifikat perangkat memberikan atau menolak akses ke AWS IoT sumber daya dan operasi bidang AWS IoT Core data. Device Advisor mengharuskan Anda untuk memiliki sertifikat perangkat aktif yang dilampirkan ke perangkat Anda yang memberikan izin tindakan yang digunakan selama kasus uji.

## AWS IoT Device Management panduan pemecahan masalah

 Bantu kami meningkatkan topik ini

[Beritahu kami apa yang akan membantu membuatnya lebih baik](#)

Ini adalah bagian pemecahan masalah untuk. AWS IoT Device Management

Topik

- [AWS IoT Pemecahan Masalah Pekerjaan](#)
- [Panduan pemecahan masalah pengindeksan armada](#)

### AWS IoT Pemecahan Masalah Pekerjaan

Ini adalah bagian pemecahan masalah untuk AWS IoT Pekerjaan.

## Bagaimana cara menemukan titik akhir AWS IoT pekerjaan?

### Bagaimana cara menemukan titik akhir bidang kontrol AWS IoT pekerjaan?

AWS IoT Jobs mendukung kontrol operasi API pesawat menggunakan protokol HTTPS. Pastikan Anda telah terhubung ke titik akhir bidang kontrol yang benar menggunakan protokol HTTPS.

Untuk daftar titik akhir AWS khusus wilayah, lihat Titik akhir bidang [kontrol AWS IoT inti](#).

Untuk daftar titik akhir bidang kontrol AWS IoT Pekerjaan yang sesuai dengan FIPS, lihat Titik Akhir [FIPS](#) menurut Layanan

#### Note

AWS IoT Pekerjaan dan AWS IoT Core berbagi titik akhir AWS khusus Wilayah yang sama.

### Bagaimana cara menemukan titik akhir bidang data AWS IoT pekerjaan?

AWS IoT Jobs mendukung operasi API bidang data menggunakan protokol HTTPS dan MQTT. Pastikan Anda telah terhubung ke titik akhir bidang data yang benar menggunakan protokol HTTPS atau MQTT.

- Protokol HTTPS
  - Gunakan perintah [describe-endpoint](#) CLI berikut yang ditunjukkan di bawah ini atau [DescribeEndpoint](#) REST API. Untuk tipe titik akhir, gunakan `iot:Jobs`.

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

- Protokol MQTT
  - Gunakan perintah [describe-endpoint](#) CLI berikut yang ditunjukkan di bawah ini atau [DescribeEndpoint](#) REST API. Untuk tipe titik akhir, gunakan `iot>Data-ATS` (disarankan) atau `iot>Data`.

```
aws iot describe-endpoint --endpoint-type iot>Data-ATS (recommended)
```

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

Untuk daftar titik akhir pesawat data AWS IoT Pekerjaan yang sesuai dengan FIPS, lihat Titik Akhir [FIPS](#) menurut Layanan

## Bagaimana cara memantau aktivitas AWS IoT Lowongan Kerja dan memberikan metrik?

Memantau aktivitas AWS IoT Pekerjaan menggunakan Amazon CloudWatch memberikan visibilitas real-time ke dalam operasi AWS IoT Pekerjaan yang sedang berlangsung dan membantu mengontrol biaya dengan CloudWatch alarm melalui AWS IoT Aturan. Anda harus mengonfigurasi pencatatan sebelum dapat memantau aktivitas AWS IoT Pekerjaan dan mengatur CloudWatch alarm. Untuk informasi selengkapnya tentang pengaturan logging, lihat [Konfigurasi AWS IoT logging](#).

Untuk informasi selengkapnya tentang Amazon CloudWatch dan cara mengatur izin melalui peran pengguna IAM untuk menggunakan CloudWatch sumber daya, lihat [Identitas dan manajemen akses untuk Amazon CloudWatch](#).

## Bagaimana cara mengatur metrik dan pemantauan AWS IoT Pekerjaan menggunakan Amazon CloudWatch?

Untuk mengatur AWS IoT logging, ikuti langkah-langkah yang diuraikan dalam [Konfigurasi AWS IoT logging](#). AWS IoT pengaturan logging dapat dilakukan di AWS Management Console, AWS CLI, atau API. AWS IoT pengaturan logging untuk grup hal tertentu harus dilakukan di AWS CLI atau API saja.

Bagian [metrik AWS IoT Pekerjaan](#) berisi metrik AWS IoT Pekerjaan yang digunakan untuk memantau aktivitas AWS IoT Pekerjaan. Ini menjelaskan cara melihat metrik di AWS Management Console dan AWS CLI.

Selain itu, Anda dapat mengatur CloudWatch alarm untuk mengingatkan Anda tentang metrik tertentu yang ingin Anda pantau dengan cermat. Untuk panduan tentang pengaturan alarm, lihat [Menggunakan CloudWatch alarm Amazon](#).

## Armada perangkat dan pemecahan masalah perangkat tunggal

### Eksekusi pekerjaan mempertahankan status **QUEUED** tanpa batas waktu

Ketika eksekusi pekerjaan dengan status status QUEUED tidak melanjutkan ke status status logis berikutnya seperti IN\_PROGRESS, FAILED, atau TIMED\_OUT, salah satu skenario berikut mungkin menjadi penyebabnya:

- Tinjau aktivitas perangkat Anda di CloudWatch log yang terletak di [CloudWatch konsol](#). Untuk informasi selengkapnya, lihat [Monitor AWS IoT menggunakan CloudWatch Log](#).
- Peran IAM yang terkait dengan pekerjaan dan pelaksanaan pekerjaan berikutnya mungkin tidak memiliki izin yang benar yang tercantum dalam salah satu pernyataan kebijakan kebijakan IAM yang dilampirkan pada peran IAM tersebut. Gunakan [describe-job](#) API untuk mengidentifikasi peran IAM yang ditautkan ke pekerjaan tersebut dan pelaksanaan pekerjaan selanjutnya dan tinjau kebijakan IAM untuk mendapatkan izin yang benar. Setelah pernyataan izin kebijakan diperbarui, Anda harus dapat menjalankan perintah [AssumeRole](#) API pada sumber daya.

Eksekusi pekerjaan tidak dibuat untuk kelompok benda atau benda saya

Saat pekerjaan memperbarui status statusnya `IN_PROGRESS`, pekerjaan akan memulai peluncuran dokumen pekerjaan ke semua perangkat di grup target Anda. Pembaruan status status ini akan membuat eksekusi pekerjaan untuk setiap perangkat target. Jika eksekusi pekerjaan tidak dibuat untuk salah satu perangkat target, lihat panduan berikut:

- Apakah yang ditargetkan `thing` langsung oleh pekerjaan, pekerjaan memiliki status status `IN_PROGRESS`, dan pekerjaan itu bersamaan? Jika ketiga kondisi terpenuhi, maka pekerjaan tersebut masih mengirimkan eksekusi pekerjaan ke semua perangkat di grup target Anda dan yang spesifik `thing` belum menerima eksekusi pekerjaannya.
- Tinjau perangkat di grup target untuk pekerjaan dan status status pekerjaan di Konsol AWS Manajemen atau gunakan perintah [describe-job](#) API.
- Gunakan perintah [describe-job](#) API untuk meninjau apakah job memiliki `IsConcurrent` properti yang disetel ke `true` atau `false`. Untuk informasi selengkapnya, lihat [Batas pekerjaan](#).
- `thing` itu tidak langsung ditargetkan oleh pekerjaan.
- Jika `Thing` ditambahkan ke a `ThingGroup` dan pekerjaan ditargetkan `ThingGroup`, maka verifikasi `Thing` adalah bagian dari `ThingGroup`.
- Jika pekerjaan tersebut adalah pekerjaan snapshot dengan status status `IN_PROGRESS` dan bersamaan, maka pekerjaan tersebut masih mengirimkan eksekusi pekerjaan ke semua perangkat di grup target Anda dan yang spesifik `Thing` belum menerima eksekusi pekerjaannya.
- Jika pekerjaan tersebut adalah pekerjaan berkelanjutan dengan status status `IN_PROGRESS` dan bersamaan, maka pekerjaan tersebut masih mengirimkan eksekusi pekerjaan ke semua perangkat di grup target Anda dan yang spesifik `Thing` belum menerima eksekusi

pekerjaannya. Untuk pekerjaan berkelanjutan saja, Anda juga dapat menghapus Thing dari ThingGroup dan kemudian menambahkan Thing kembali keThingGroup.

- Jika pekerjaan adalah pekerjaan snapshot dengan status status IN\_PROGRESS dan tidak bersamaan, maka kemungkinan hubungan Thing atau ThingGroup keanggotaan tidak diakui oleh Jobs. AWS IoT Disarankan untuk menambahkan beberapa detik waktu tunggu setelah AddThingToThingGroup panggilan Anda sebelum Anda membuatJob. Atau, Anda dapat mengalihkan pemilihan target keContinuous, sehingga membuat layanan mengisi kembali acara lampiran tertunda Thing dan ThingGroup keanggotaan.

## Pekerjaan baru gagal karena **LimitedExceededException** kesalahan

Jika pembuatan pekerjaan Anda gagal dengan respons kesalahanLimitedExceededException, hubungi list-jobs API dan tinjau semua pekerjaan isConcurrent=true untuk menentukan apakah Anda berada pada batas konkurensi pekerjaan Anda. Lihat [batas Job](#) untuk informasi tambahan tentang pekerjaan bersamaan. Untuk melihat batas konkurensi pekerjaan Anda dan untuk meminta kenaikan batas, lihat [batas dan kuota AWS IoT Device Management pekerjaan](#).

## Batas ukuran dokumen Job

Ukuran dokumen pekerjaan dibatasi oleh ukuran muatan MQTT. Jika Anda memerlukan dokumen pekerjaan yang lebih besar dari 32 kB (kilobyte), 32.000 B (byte), kemudian buat dan simpan dokumen pekerjaan di Amazon S3 dan tambahkan URL objek Amazon S3 di bidang untuk API atau menggunakan file. documentSource CreateJob AWS CLI Untuk itu AWS Management Console, tambahkan URL objek Amazon S3 di kotak teks URL Amazon S3 saat membuat pekerjaan.

- AWS Management Console buat dokumentasi pekerjaan: [Buat dan kelola pekerjaan dengan menggunakan AWS Management Console](#)
- AWS CLI buat dokumentasi pekerjaan: [Buat dan kelola pekerjaan menggunakan AWS CLI](#)
- CreateJobDokumentasi API: [CreateJob](#)

## Pesan MQTT Sisi Perangkat meminta batas throttle

Jika Anda menerima kode kesalahan 400ThrottlingException, pesan MQTT sisi perangkat gagal karena mencapai batas permintaan sisi perangkat secara bersamaan. Lihat [batas AWS IoT](#)



[Device Management pekerjaan dan kuota](#) untuk informasi lebih lanjut tentang batas throttle dan jika dapat disesuaikan.

### Kesalahan batas waktu koneksi

Kode kesalahan 400 RequestExpired menunjukkan kegagalan koneksi karena latensi tinggi atau nilai batas waktu sisi klien yang rendah.

- Lihat [Menguji konektivitas dengan titik akhir data perangkat Anda](#) untuk informasi tentang pengujian koneksi antara sisi klien dan sisi server.

### Perintah API tidak valid

Konfirmasikan perintah API yang benar dimasukkan untuk menghindari pesan kesalahan yang menyatakan perintah API tidak valid. Lihat [Referensi AWS IoT API](#) untuk daftar lengkap semua perintah AWS IoT API.

### Kesalahan koneksi sisi layanan

Kode kesalahan 503 ServiceUnavailable menunjukkan kesalahan berasal dari sisi server.

- Lihat [AWS Health Dashboard \(semua AWS layanan\)](#) untuk status saat ini dari semua AWS layanan.
- Lihat [AWS Health Dashboard \(pribadi Akun AWS\)](#) untuk status pribadi Anda saat ini Akun AWS.

## Panduan pemecahan masalah pengindeksan armada

### Memecahkan masalah kueri agregasi untuk layanan pengindeksan armada

Jika Anda mengalami kesalahan jenis ketidakcocokan, Anda dapat menggunakan CloudWatch Log untuk memecahkan masalah. CloudWatch Log harus diaktifkan sebelum log ditulis oleh layanan Pengindeksan Armada. Untuk informasi selengkapnya, lihat [Monitor AWS IoT menggunakan CloudWatch Log](#).

Untuk membuat kueri agregasi pada bidang yang tidak dikelola, Anda harus menentukan bidang yang Anda tentukan dalam customFields argumen yang diteruskan ke atau UpdateIndexingConfiguration update-indexing-configuration. Jika nilai bidang tidak konsisten dengan tipe data bidang yang dikonfigurasi, nilai ini diabaikan saat Anda melakukan kueri agregasi.

Jika bidang tidak dapat diindeks karena jenis yang tidak cocok, layanan Pengindeksan Armada akan mengirimkan log kesalahan ke Log. CloudWatch Log kesalahan berisi nama bidang, nilai yang tidak dapat dikonversi, dan nama benda untuk perangkat. Berikut ini adalah contoh log kesalahan:

```
{
 "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"
 }
]
}
```

Jika perangkat telah terputus selama kurang lebih satu jam, `timestamp` nilai status konektivitas mungkin hilang. Untuk sesi persisten, nilainya mungkin hilang setelah klien terputus lebih lama dari yang dikonfigurasi `time-to-live (TTL)` untuk sesi persisten. Data status konektivitas diindeks hanya untuk koneksi di mana ID klien memiliki nama benda yang cocok. (ID klien adalah nilai yang digunakan untuk menghubungkan perangkat ke AWS IoT Core.)

## Memecahkan masalah konfigurasi pengindeksan armada

Tidak dapat menurunkan versi konfigurasi pengindeksan armada

Menurunkan konfigurasi pengindeksan armada tidak didukung saat Anda ingin menghapus sumber data yang terkait dengan metrik armada atau grup dinamis.

Misalnya, jika konfigurasi pengindeksan Anda memiliki data registri, data bayangan, dan data konektivitas, dan metrik armada ada dengan `kuerithingName:TempSensor* AND`

`shadow.desired.temperature>80`, memperbarui konfigurasi pengindeksan untuk menyertakan hanya data registri akan mengakibatkan kesalahan.

Memodifikasi bidang kustom yang digunakan oleh metrik armada yang ada tidak didukung.

Tidak dapat memperbarui konfigurasi pengindeksan karena metrik armada atau grup dinamis yang tidak kompatibel

Jika Anda tidak dapat memperbarui konfigurasi pengindeksan karena metrik armada atau grup dinamis yang tidak kompatibel, hapus metrik armada atau grup dinamis yang tidak kompatibel sebelum memperbarui konfigurasi pengindeksan.

## Memecahkan masalah pengindeksan lokasi dan geoquery

Untuk memecahkan masalah kesalahan jenis yang tidak cocok dalam pengindeksan lokasi dan geoquery, Anda dapat mengaktifkan log. CloudWatch Untuk informasi lebih lanjut tentang cara memantau AWS IoT penggunaan CloudWatch, ikuti [step-by-step panduan ini](#).

Saat Anda mengindeks data lokasi menggunakan geoquery, bidang lokasi yang Anda tentukan `geoLocations` harus cocok dengan bidang lokasi yang Anda berikan.

`UpdateIndexingConfiguration` Jika ada ketidakcocokan, pengindeksan armada mengirimkan kesalahan tipe yang tidak cocok ke. CloudWatch Log kesalahan berisi nama bidang, nilai yang tidak dapat dikonversi, dan nama benda untuk perangkat.

Berikut ini adalah contoh log kesalahan:

```
{
 "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."
}
```

Untuk informasi selengkapnya, lihat [???](#).

## Pemecahan masalah metrik armada

Tidak dapat melihat titik data di CloudWatch

Jika Anda dapat membuat metrik armada tetapi Anda tidak dapat melihat titik data CloudWatch, kemungkinan Anda tidak memiliki sesuatu yang memenuhi kriteria string kueri.

Lihat contoh perintah ini tentang cara membuat metrik armada:

```
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
```

Jika Anda tidak memiliki sesuatu yang memenuhi kriteria string kueri `--query-string "thingName:TempSensor* AND attributes.temperature>80"`:

- Dengan `values=count`, Anda akan dapat membuat metrik armada dan akan ada titik data untuk ditampilkan CloudWatch. Titik data dari nilai `count` selalu 0.
- Dengan `values` selain `count`, Anda akan dapat membuat metrik armada tetapi Anda tidak akan melihat metrik armada CloudWatch dan tidak akan ada titik data untuk ditampilkan CloudWatch.

## AWS IoT kesalahan

 Bantu kami meningkatkan topik ini

[Beritahu kami apa yang akan membantu membuatnya lebih baik](#)

Bagian ini mencantumkan kode kesalahan yang dikirim oleh AWS IoT.

## Kode kesalahan broker pesan

| Kode kesalahan | Deskripsi kesalahan     |
|----------------|-------------------------|
| 400            | Permintaan buruk.       |
| 401            | Tidak sah.              |
| 403            | Terlarang.              |
| 503            | Layanan tidak tersedia. |

## Kode kesalahan identitas dan keamanan

| Kode kesalahan | Deskripsi kesalahan |
|----------------|---------------------|
| 401            | Tidak sah.          |

## Kode kesalahan bayangan perangkat

| Kode kesalahan | Deskripsi kesalahan         |
|----------------|-----------------------------|
| 400            | Permintaan buruk.           |
| 401            | Tidak sah.                  |
| 403            | Terlarang.                  |
| 404            | Tidak ditemukan.            |
| 409            | Konflik.                    |
| 413            | Permintaan terlalu besar.   |
| 422            | Gagal memproses permintaan. |
| 429            | Terlalu banyak permintaan.  |
| 500            | Kesalahan internal.         |
| 503            | Layanan tidak tersedia.     |

# AWS IoT SDK Perangkat, SDK Seluler, dan AWS IoT Klien Perangkat

Halaman ini merangkum SDK AWS IoT Perangkat, pustaka sumber terbuka, panduan pengembang, contoh aplikasi, dan panduan porting untuk membantu Anda membangun solusi IoT inovatif dengan AWS IoT dan pilihan platform perangkat keras Anda.

SDK ini untuk digunakan pada perangkat IoT Anda. Jika Anda mengembangkan aplikasi IoT untuk digunakan di perangkat seluler, lihat. [AWS SDK Seluler](#) Jika Anda mengembangkan aplikasi IoT atau program sisi server, lihat. [AWS SDK](#)

## AWS IoT SDK perangkat

SDK AWS IoT Perangkat mencakup pustaka sumber terbuka, panduan pengembang dengan sampel, dan panduan porting sehingga Anda dapat membangun produk atau solusi IoT inovatif pada platform perangkat keras pilihan Anda.

### Note

AWS IoT Perangkat SDK telah merilis klien MQTT 5. SDK AWS IoT Perangkat tidak mendukung penggunaan TLS 1.3 di macOS.

SDK ini membantu Anda menghubungkan perangkat IoT Anda AWS IoT untuk menggunakan protokol MQTT dan WSS.

### C++

#### AWS IoT SDK Perangkat C ++

AWS IoT C++ Device SDK memungkinkan pengembang untuk membangun aplikasi yang terhubung menggunakan AWS dan API. AWS IoT Secara khusus, SDK ini dirancang untuk perangkat yang tidak dibatasi sumber daya dan memerlukan fitur-fitur canggih seperti antrian pesan, dukungan multi-threading, dan fitur bahasa terbaru. Untuk informasi selengkapnya, lihat hal berikut:

- [AWS IoT Perangkat SDK C++ v2 aktif GitHub](#)

- [AWS IoT Perangkat SDK C++ v2 Readme](#)
- [AWS IoT Sampel Perangkat SDK C++ v2](#)
- [AWS IoT Dokumentasi API SDK C++ v2 perangkat](#)

## Python

### AWS IoT Perangkat SDK untuk Python

AWS IoT Perangkat SDK untuk Python memungkinkan pengembang untuk menulis skrip Python untuk menggunakan perangkat mereka untuk mengakses platform melalui MQTT atau AWS IoT MQTT melalui protokol WebSocket. Dengan menghubungkan perangkat mereka ke AWS IoT, pengguna dapat bekerja dengan aman dengan broker pesan, aturan, dan bayangan yang disediakan oleh AWS IoT dan dengan AWS layanan lain seperti AWS Lambda, Kinesis, dan Amazon S3, dan banyak lagi.

- [AWS IoT Perangkat SDK untuk Python v2 aktif GitHub](#)
- [AWS IoT Perangkat SDK untuk Python v2 Readme](#)
- [AWS IoT SDK Perangkat untuk Sampel Python v2](#)
- [AWS IoT SDK perangkat untuk dokumentasi API Python v2](#)

## JavaScript

### AWS IoT Perangkat SDK untuk JavaScript

aws-iot-device-sdkPaket.js memungkinkan pengembang untuk menulis JavaScript aplikasi yang mengakses AWS IoT menggunakan MQTT atau MQTT melalui protokol WebSocket. Ini dapat digunakan di lingkungan Node.js dan aplikasi browser. Untuk informasi selengkapnya, lihat hal berikut:

- [AWS IoT SDK perangkat untuk JavaScript v2 aktif GitHub](#)
- [AWS IoT Perangkat SDK untuk JavaScript v2 Readme](#)
- [AWS IoT SDK Perangkat untuk Sampel JavaScript v2](#)
- [AWS IoT SDK perangkat untuk dokumentasi API JavaScript v2](#)

## Java

### AWS IoT Perangkat SDK for Java

AWS IoT Device SDK for Java memungkinkan pengembang Java untuk mengakses AWS IoT platform melalui MQTT atau MQTT melalui protokol. WebSocket SDK dibangun dengan dukungan bayangan. Anda dapat mengakses bayangan dengan menggunakan metode HTTP, termasuk GET, UPDATE, dan DELETE. SDK juga mendukung model akses bayangan yang disederhanakan, yang memungkinkan pengembang untuk bertukar data dengan bayangan hanya dengan menggunakan metode pengambil dan penyetel, tanpa harus membuat serial atau deserialisasi dokumen JSON apa pun.


 Note

AWS IoT Device SDK for Java v2 sekarang mendukung pengembangan Android. Untuk informasi selengkapnya, lihat [SDK AWS IoT perangkat untuk Android](#).

Untuk informasi selengkapnya, lihat hal berikut:

- [AWS IoT Perangkat SDK for Java v2 aktif GitHub](#)
- [AWS IoT Perangkat SDK for Java v2 Readme](#)
- [AWS IoT Perangkat SDK for Java v2 Sampel](#)
- [AWS IoT Dokumentasi SDK for Java v2 API](#)

## AWS IoT Perangkat SDK untuk Embedded C

 Note

SDK ini dimaksudkan untuk digunakan oleh pengembang perangkat lunak tertanam berpengalaman.

AWS IoT Device SDK for Embedded C C-SDK adalah kumpulan file sumber C di bawah lisensi sumber terbuka MIT yang dapat digunakan dalam aplikasi tertanam untuk menghubungkan perangkat IoT dengan aman. AWS IoT Core Ini termasuk klien MQTT, JSON Parser, dan Device Shadow AWS IoT, AWS IoT Jobs, AWS IoT Fleet Provisioning, dan perpustakaan. AWS IoT Device Defender SDK ini didistribusikan dalam bentuk sumber dan dapat dibangun ke dalam firmware pelanggan bersama dengan kode aplikasi, perpustakaan lain, dan sistem operasi (OS) pilihan Anda.



Umumnya AWS IoT Device SDK for Embedded C ditargetkan pada perangkat terbatas sumber daya yang memerlukan runtime bahasa C yang dioptimalkan. Anda dapat menggunakan SDK pada sistem operasi apa pun dan menghostingnya pada semua jenis prosesor (misalnya, MCU dan MPU).

Untuk informasi selengkapnya, lihat hal berikut:

- [AWS IoT Perangkat SDK untuk Embedded C aktif GitHub](#)
- [AWS IoT SDK Perangkat untuk Readme C Tertanam](#)
- [AWS IoT SDK Perangkat untuk Sampel C Tertanam](#)

## Versi SDK AWS IoT Perangkat Sebelumnya

Ini adalah versi SDK AWS IoT Perangkat sebelumnya yang telah digantikan oleh versi yang lebih baru yang tercantum di atas. SDK ini hanya menerima pembaruan pemeliharaan dan keamanan. Mereka tidak akan diperbarui untuk menyertakan fitur baru dan tidak boleh digunakan pada proyek baru.

- [AWS IoT SDK Perangkat C ++ aktif GitHub](#)
- [AWS IoT C++ Perangkat SDK Readme](#)
- [AWS IoT Perangkat SDK untuk Python v1 aktif GitHub](#)
- [AWS IoT Perangkat SDK untuk Python v1 Readme](#)
- [AWS IoT Perangkat SDK for Java aktif GitHub](#)
- [AWS IoT Perangkat SDK for Java Readme](#)
- [AWS IoT SDK perangkat untuk aktif JavaScript GitHub](#)
- [AWS IoT Perangkat SDK untuk Readme JavaScript](#)
- [Arduino Yún SDK aktif GitHub](#)
- [Arduino Yun SDK Readme](#)

## AWS SDK Seluler

AWS Mobile SDK menyediakan dukungan khusus platform pengembang aplikasi seluler untuk API layanan AWS IoT Core , komunikasi perangkat IoT menggunakan MQTT, dan API layanan lainnya.  
AWS

## Android

### AWS Mobile SDK for Android

AWS Mobile SDK for Android Ini berisi pustaka, sampel, dan dokumentasi bagi pengembang untuk membangun aplikasi seluler yang terhubung menggunakan AWS. SDK ini juga mencakup dukungan untuk komunikasi perangkat MQTT dan memanggil API layanan. AWS IoT Core Untuk informasi selengkapnya, lihat hal berikut:

- [AWS Mobile SDK for Android pada GitHub](#)
- [AWS Mobile SDK for Android Readme](#)
- [AWS Mobile SDK for Android Sampel](#)
- [AWS Mobile SDK for Android Referensi API](#)
- [AWSIoTClient Dokumentasi referensi kelas](#)

## iOS

### AWS Mobile SDK for iOS


AWS Mobile SDK for iOS Ini adalah kit pengembangan perangkat lunak open-source, didistribusikan di bawah lisensi Apache Open Source. AWS Mobile SDK for iOS Ini menyediakan pustaka, contoh kode, dan dokumentasi untuk membantu pengembang membangun aplikasi seluler yang terhubung menggunakan AWS. SDK ini juga mencakup dukungan untuk komunikasi perangkat MQTT dan memanggil API layanan. AWS IoT Core Untuk informasi selengkapnya, lihat hal berikut:

- [AWS Mobile SDK for iOS pada GitHub](#)
- [AWS Mobile SDK for iOS Readme](#)
- [AWS Mobile SDK for iOS Sampel](#)
- [AWSIoT Dokumen referensi kelas di AWS Mobile SDK for iOS](#)

## AWS IoT Klien Perangkat

Klien AWS IoT Perangkat menyediakan kode untuk membantu perangkat Anda terhubung AWS IoT, melakukan tugas penyediaan armada, mendukung kebijakan keamanan perangkat, terhubung menggunakan tunneling aman, dan memproses pekerjaan di perangkat Anda. Anda dapat

menginstal perangkat lunak ini di perangkat Anda untuk menangani tugas-tugas perangkat rutin ini sehingga Anda dapat fokus pada solusi spesifik Anda.

 Note

AWS IoT Device Client bekerja dengan perangkat IoT berbasis mikroprosesor dengan prosesor x86\_64 atau ARM dan sistem operasi Linux umum.

## C++

### AWS IoT Klien Perangkat

Untuk informasi selengkapnya tentang Klien AWS IoT Perangkat di C ++, lihat berikut ini:

- [AWS IoT Klien Perangkat dalam kode sumber C ++ pada GitHub](#)
- [AWS IoT Klien Perangkat di C ++ Readme](#)

# Contoh kode untuk AWS IoT menggunakan AWS SDK

Contoh kode berikut menunjukkan cara menggunakan AWS IoT kit pengembangan AWS perangkat lunak (SDK).

Tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks pada skenario terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Menggunakan AWS IoT dengan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Memulai

## Halo AWS IoT

Contoh kode berikut menunjukkan cara untuk mulai menggunakan AWS IoT.

C++

SDK untuk C++

Kode untuk file CMake MakeLists C.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})
```

Kode untuk file sumber 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;
}
```

- Untuk detail API, lihat [ListThings](#) di Referensi AWS SDK for C++ API.

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

## Java

### SDK untuk Java 2.x

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
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 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) {
 ListThingsRequest thingsRequest = ListThingsRequest.builder()
 .maxResults(10)
 .build();

 ListThingsResponse response = iotClient.listThings(thingsRequest) ;
 List<ThingAttribute> thingList = response.things();
 for (ThingAttribute attribute : thingList) {
 System.out.println("Thing name: "+attribute.thingName());
 System.out.println("Thing ARN: "+attribute.thingArn());
 }
 }
}
```

- Untuk detail API, lihat [ListThings](#) di Referensi AWS SDK for Java 2.x API.

## Kotlin

### SDK untuk Kotlin

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
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 { 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}")
 }
 }
 }
}
```

- Untuk detail API, lihat [ListThings](#) di AWS SDK untuk referensi API Kotlin.

#### Contoh kode

- [Tindakan untuk AWS IoT menggunakan AWS SDK](#)
  - [Gunakan AttachThingPrincipal dengan AWS SDK atau CLI](#)
  - [Gunakan CreateKeysAndCertificate dengan AWS SDK atau CLI](#)
  - [Gunakan CreateThing dengan AWS SDK atau CLI](#)
  - [Gunakan CreateTopicRule dengan AWS SDK atau CLI](#)
  - [Gunakan DeleteCertificate dengan AWS SDK atau CLI](#)
  - [Gunakan DeleteThing dengan AWS SDK atau CLI](#)
  - [Gunakan DeleteTopicRule dengan AWS SDK atau CLI](#)
  - [Gunakan DescribeEndpoint dengan AWS SDK atau CLI](#)
  - [Gunakan DescribeThing dengan AWS SDK atau CLI](#)
  - [Gunakan DetachThingPrincipal dengan AWS SDK atau CLI](#)
  - [Gunakan ListCertificates dengan AWS SDK atau CLI](#)
  - [Gunakan ListThings dengan AWS SDK atau CLI](#)
  - [Gunakan SearchIndex dengan AWS SDK atau CLI](#)

- [Gunakan UpdateIndexingConfiguration dengan AWS SDK atau CLI](#)
- [Gunakan UpdateThing dengan AWS SDK atau CLI](#)
- [Skenario untuk AWS IoT menggunakan AWS SDK](#)
- [Bekerja dengan AWS IoT perangkat, benda, dan bayangan menggunakan AWS IoT SDK](#)

## Tindakan untuk AWS IoT menggunakan AWS SDK

Contoh kode berikut menunjukkan cara melakukan AWS IoT tindakan individual dengan AWS SDK. Kutipan ini memanggil AWS IoT API dan merupakan kutipan kode dari program yang lebih besar yang harus dijalankan dalam konteks. Setiap contoh menyertakan tautan ke GitHub, di mana Anda dapat menemukan instruksi untuk mengatur dan menjalankan kode.

Contoh berikut hanya mencakup tindakan yang paling umum digunakan. Untuk daftar lengkapnya, lihat [Referensi AWS IoT API](#).

### Contoh

- [Gunakan AttachThingPrincipal dengan AWS SDK atau CLI](#)
- [Gunakan CreateKeysAndCertificate dengan AWS SDK atau CLI](#)
- [Gunakan CreateThing dengan AWS SDK atau CLI](#)
- [Gunakan CreateTopicRule dengan AWS SDK atau CLI](#)
- [Gunakan DeleteCertificate dengan AWS SDK atau CLI](#)
- [Gunakan DeleteThing dengan AWS SDK atau CLI](#)
- [Gunakan DeleteTopicRule dengan AWS SDK atau CLI](#)
- [Gunakan DescribeEndpoint dengan AWS SDK atau CLI](#)
- [Gunakan DescribeThing dengan AWS SDK atau CLI](#)
- [Gunakan DetachThingPrincipal dengan AWS SDK atau CLI](#)
- [Gunakan ListCertificates dengan AWS SDK atau CLI](#)
- [Gunakan ListThings dengan AWS SDK atau CLI](#)
- [Gunakan SearchIndex dengan AWS SDK atau CLI](#)
- [Gunakan UpdateIndexingConfiguration dengan AWS SDK atau CLI](#)
- [Gunakan UpdateThing dengan AWS SDK atau CLI](#)

## Gunakan **AttachThingPrincipal** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `AttachThingPrincipal`.

C++

SDK untuk C++

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
//! 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();
}
```

- Untuk detail API, lihat [AttachThingPrincipal](#) dalam Referensi AWS SDK for C++ API.

## CLI

### AWS CLI

Untuk melampirkan sertifikat ke barang Anda

`attach-thing-principal` Contoh berikut melampirkan sertifikat untuk `MyTemperatureSensor` benda itu. Sertifikat diidentifikasi oleh ARN. Anda dapat menemukan ARN untuk sertifikat di konsol IoT AWS .

```
aws iot attach-thing-principal \
 --thing-name MyTemperatureSensor \
 --principal arn:aws:iot:us-
west-2:123456789012:cert/2e1eb273792174ec2b9bf4e9b37e6c6c692345499506002a35159767055278e8
```

Perintah ini tidak menghasilkan output.

Untuk informasi selengkapnya, lihat [Cara Mengelola Sesuatu dengan Registri](#) di Panduan Pengembang AWS IoT.

- Untuk detail API, lihat [AttachThingPrincipal](#) dalam Referensi AWS CLI Perintah.

## Java

### SDK untuk Java 2.x

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
public static void attachCertificateToThing(IotClient iotClient, String
thingName, String certificateArn) {
 // Attach the certificate to the thing.
 AttachThingPrincipalRequest principalRequest =
 AttachThingPrincipalRequest.builder()
```

```
 .thingName(thingName)
 .principal(certificateArn)
 .build();

 AttachThingPrincipalResponse attachResponse =
 iotClient.attachThingPrincipal(principalRequest);

 // Verify the attachment was successful.
 if (attachResponse.sdkHttpResponse().isSuccessful()) {
 System.out.println("Certificate attached to Thing successfully.");

 // Print additional information about the Thing.
 describeThing(iotClient, thingName);
 } else {
 System.err.println("Failed to attach certificate to Thing. HTTP
 Status Code: " +
 attachResponse.sdkHttpResponse().statusCode());
 }
}
```

- Untuk detail API, lihat [AttachThingPrincipal](#) dalam Referensi AWS SDK for Java 2.x API.

## Kotlin

### SDK untuk Kotlin

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
suspend fun attachCertificateToThing(
 thingNameVal: String?,
 certificateArn: String?,
) {
 val principalRequest =
 AttachThingPrincipalRequest {
 thingName = thingNameVal
 principal = certificateArn
 }
}
```

```
IotClient { region = "us-east-1" }.use { iotClient ->
 iotClient.attachThingPrincipal(principalRequest)
 println("Certificate attached to $thingNameVal successfully.")
}
}
```

- Untuk detail API, lihat [AttachThingPrincipal](#) di AWS SDK untuk referensi API Kotlin.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Menggunakan AWS IoT dengan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

## Gunakan **CreateKeysAndCertificate** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `CreateKeysAndCertificate`.

C++

SDK untuk C++

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
//! 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();
}
```

- Untuk detail API, lihat [CreateKeysAndCertificate](#) di Referensi AWS SDK for C++ API.

## CLI

### AWS CLI

Untuk membuat key pair RSA dan mengeluarkan sertifikat X.509

Berikut ini `create-keys-and-certificate` membuat key pair RSA 2048-bit dan mengeluarkan sertifikat X.509 menggunakan kunci publik yang diterbitkan. Karena ini adalah satu-satunya waktu AWS IoT menyediakan kunci pribadi untuk sertifikat ini, pastikan untuk menyimpannya di lokasi yang aman.



```
aws iot create-keys-and-certificate \
 --certificate-pem-outfile "myTest.cert.pem" \
 --public-key-outfile "myTest.public.key" \
 --private-key-outfile "myTest.private.key"
```

### Output:

```
{
 "certificateArn": "arn:aws:iot:us-
west-2:123456789012:cert/9894ba17925e663f1d29c23af4582b8e3b7619c31f3fbd93adcb51ae54b83dc2",
 "certificateId":
 "9894ba17925e663f1d29c23af4582b8e3b7619c31f3fbd93adcb51ae54b83dc2",
 "certificatePem": "
-----BEGIN CERTIFICATE-----
MIICiTCCEXAMPLE6m7oRw0uX0jANBgkqhkiG9w0BAQUFADCBiDELMakGA1UEBhMC
VVMxZCZAJBgNVBAGEXAMPLEAwDgYDVQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6
b24xFDASBgNVBA5TC01BTSEXAMPLE2x1MRIwEAYDVQQDEw1UZXR0Q21sYW11eXhAd
BgkqhkiG9w0BCQEWEG5vb251QGFtYXV0YXV0YXV0YXV0YXV0YXV0YXV0YXV0YXV0
MTIwNDI0MjA0NTIxwjcBiDELMakGA1UEBhMCCEXAMPLEJBgNVBAGTAldBMRAdgYD
VQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBA5TC01BTSEBDb25z
b2x1MRIwEAYDVQQDEw1UZXR0Q21sYW11eXhAdBgkqhkiG9w0BCQEXAMPLE251QGFt
YXp1b251b251b251b251b251b251b251b251b251b251b251b251b251b251b251b251
EXAMPLEfEvySwTc2XADZ4nB+BLyGvIk60CpiwsZ3G93vUEI03IyNoH/f0wYK8m9T
rDHudUZEXAMPLEL6M43q7Wgc/MbQITx0USQv7c7ugFFDzQGBzZswY6786m86gpE
Ibb30hjZnzcVQEXAMPLEWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
nUhVVxYUntneD9+h8Mg9qEXAMPEyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb
FFBjvSfpJI1J0zbbNYS5f6GuoEDEXAMPLEBHjJnyp3780D8uTs7fLvJx79LjSTb
NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
-----END CERTIFICATE-----\n",
 "keyPair": {
 "PublicKey": "-----BEGIN PUBLIC KEY-----
\nMIIBIjANBgkqhkiG9w0BAQEFAA0CAQ8AMIIBCgKCAQEAEXAMPLE1nnyJwKSMHw4h\nnMMEXAMPLEuuN/
dMAS3fyce8DW/4+EXAMPLEYjmoF/YVF/gHr99VEEXAMPLE5VF13\nn59VK7cEXAMPLE67GK+y
+jikqX0gHh/xJTtwo
+sGpwEXAMPLEDz18x0d2ka4tCzuWEXAMPLEahJbYkCPUBSU8opVkr7qkEXAMPLE1DR6sx2Hocli00Ltu6Fkw91swQ
\GB3ZPrNh0PzQYvjUStZecyNCx2EXAMPLEvp9mQ0UXP6p1fgxwKRX2fEXAMPLEDa
\nhJLXkX3rHU2xbxJSq7D+XEXAMPLEcw+LyFhI5mgFR188eGdsAEXAMPLE1nI9EesG\nnFQIDAQAB
\n-----END PUBLIC KEY-----\n",
 "PrivateKey": "-----BEGIN RSA PRIVATE KEY-----\nkey omitted for security
reasons\n-----END RSA PRIVATE KEY-----\n"
 }
}
```

Untuk informasi selengkapnya, lihat [Membuat dan Mendaftarkan Sertifikat Perangkat AWS IoT di Panduan Pengembang AWS IoT](#).

- Untuk detail API, lihat [CreateKeysAndCertificate](#) di Referensi AWS CLI Perintah.

## Java

### SDK untuk Java 2.x

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
public static String createCertificate(IotClient iotClient) {
 try {
 CreateKeysAndCertificateResponse response =
iotClient.createKeysAndCertificate();
 String certificatePem = response.certificatePem();
 String certificateArn = response.certificateArn();

 // Print the details.
 System.out.println("\nCertificate:");
 System.out.println(certificatePem);
 System.out.println("\nCertificate ARN:");
 System.out.println(certificateArn);
 return certificateArn;

 } catch (IotException e) {
 System.err.println(e.awsErrorDetails().errorMessage());
 System.exit(1);
 }

 return "";
}
```

- Untuk detail API, lihat [CreateKeysAndCertificate](#) di Referensi AWS SDK for Java 2.x API.

## Kotlin

### SDK untuk Kotlin

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
public static String createCertificate(IotClient iotClient) {
 try {
 CreateKeysAndCertificateResponse response =
iotClient.createKeysAndCertificate();
 String certificatePem = response.certificatePem();
 String certificateArn = response.certificateArn();

 // Print the details.
 System.out.println("\nCertificate:");
 System.out.println(certificatePem);
 System.out.println("\nCertificate ARN:");
 System.out.println(certificateArn);
 return certificateArn;

 } catch (IotException e) {
 System.err.println(e.awsErrorDetails().errorMessage());
 System.exit(1);
 }

 return "";
}
```

- Untuk detail API, lihat [CreateKeysAndCertificate](#) di AWS SDK untuk referensi API Kotlin.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Menggunakan AWS IoT dengan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

## Gunakan **CreateThing** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `CreateThing`.

C++

SDK untuk C++

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
#!/ 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();
}
```

- Untuk detail API, lihat [CreateThing](#) di Referensi AWS SDK for C++ API.

## CLI

### AWS CLI

Contoh 1: Untuk membuat catatan sesuatu di registri

`create-thing` Contoh berikut membuat entri untuk perangkat di registri hal AWS IoT.

```
aws iot create-thing \
 --thing-name SampleIoTThing
```

Output:

```
{
 "thingName": "SampleIoTThing",
 "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/SampleIoTThing",
 "thingId": "EXAMPLE1-90ab-cdef-fedc-ba987EXAMPLE "
}
```

Contoh 2: Untuk mendefinisikan sesuatu yang terkait dengan tipe benda

`create-thing` Contoh berikut membuat sesuatu yang memiliki jenis hal yang ditentukan dan atributnya.

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

Output:

```
{
 "thingName": "MyLightBulb",
 "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/MyLightBulb",
 "thingId": "40da2e73-c6af-406e-b415-15acae538797"
}
```

Untuk informasi selengkapnya, lihat [Cara Mengelola Sesuatu dengan Registri](#) dan [Jenis Hal](#) di Panduan Pengembang AWS IoT.

- Untuk detail API, lihat [CreateThing](#) di Referensi AWS CLI Perintah.

## Java

### SDK untuk Java 2.x

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
public static void createIoTThing(IotClient iotClient, String thingName) {
 try {
 CreateThingRequest createThingRequest = CreateThingRequest.builder()
 .thingName(thingName)
 .build();

 CreateThingResponse createThingResponse =
 iotClient.createThing(createThingRequest);
 System.out.println(thingName + " was successfully created. The ARN
 value is " + createThingResponse.thingArn());

 } catch (IotException e) {
 System.err.println(e.awsErrorDetails().errorMessage());
 System.exit(1);
 }
}
```

- Untuk detail API, lihat [CreateThing](#) di Referensi AWS SDK for Java 2.x API.

## Kotlin

### SDK untuk Kotlin

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
suspend fun createIoTThing(thingNameVal: String) {
 val createThingRequest =
 CreateThingRequest {
 thingName = thingNameVal
 }

 IotClient { region = "us-east-1" }.use { iotClient ->
 iotClient.createThing(createThingRequest)
 println("Created $thingNameVal}")
 }
}
```

- Untuk detail API, lihat [CreateThing](#) di AWS SDK untuk referensi API Kotlin.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Menggunakan AWS IoT dengan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

## Gunakan **CreateTopicRule** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `CreateTopicRule`.

C++

SDK untuk C++

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
//! 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();
}
```

- Untuk detail API, lihat [CreateTopicAturan](#) dalam Referensi AWS SDK for C++ API.



## CLI

### AWS CLI

Untuk membuat aturan yang mengirimkan peringatan Amazon SNS

`create-topic-rule` Contoh berikut membuat aturan yang mengirimkan pesan Amazon SNS saat pembacaan tingkat kelembaban tanah, seperti yang ditemukan dalam bayangan perangkat, rendah.

```
aws iot create-topic-rule \
 --rule-name "LowMoistureRule" \
 --topic-rule-payload file://plant-rule.json
```

Contoh ini memerlukan kode JSON berikut untuk disimpan ke file bernama `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"
 }
 }
}]
}
```

Perintah ini tidak menghasilkan output.

Untuk informasi selengkapnya, lihat [Membuat Aturan AWS IoT di Panduan](#) Pengembang AWS IoT.

- Untuk detail API, lihat [CreateTopicAturan](#) dalam Referensi AWS CLI Perintah.

## Java

### SDK untuk Java 2.x

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
public static void createIoTRule(IotClient iotClient, String roleARN, String
ruleName, String action) {
 try {
 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();

 // Create the rule.
 iotClient.createTopicRule(topicRuleRequest);
 System.out.println("IoT Rule created successfully.");

 } catch (IotException e) {
```

```
 System.err.println(e.getMessage());
 System.exit(1);
 }
}
```

- Untuk detail API, lihat [CreateTopicAturan](#) dalam Referensi AWS SDK for Java 2.x API.

## Kotlin

### SDK untuk Kotlin

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
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 { region = "us-east-1" }.use { iotClient ->
 iotClient.createTopicRule(topicRuleRequest)
 println("IoT rule created successfully.")
 }
 }
```

- Untuk detail API, lihat [CreateTopicAturan](#) dalam AWS SDK untuk referensi API Kotlin.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Menggunakan AWS IoT dengan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

## Gunakan **DeleteCertificate** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `DeleteCertificate`.

C++

SDK untuk C++

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
#!/ 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();
}
```

- Untuk detail API, lihat [DeleteCertificate](#) di Referensi AWS SDK for C++ API.

## CLI

### AWS CLI

Untuk menghapus sertifikat perangkat

`delete-certificate` Contoh berikut menghapus sertifikat perangkat dengan ID yang ditentukan.

```
aws iot delete-certificate \
 --certificate-id
 c0c57bbc8baaf4631a9a0345c957657f5e710473e3ddbee1428d216d54d53ac9
```

Perintah ini tidak menghasilkan output.

Untuk informasi selengkapnya, lihat [DeleteCertificate](#) di Referensi API AWS IoT.

- Untuk detail API, lihat [DeleteCertificate](#) di Referensi AWS CLI Perintah.

## Java

### SDK untuk Java 2.x

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
public static void deleteCertificate(IotClient iotClient, String
certificateArn) {
 DeleteCertificateRequest certificateProviderRequest =
DeleteCertificateRequest.builder()
 .certificateId(extractCertificateId(certificateArn))
 .build();

 iotClient.deleteCertificate(certificateProviderRequest);
 System.out.println(certificateArn +" was successfully deleted.");
}
```

- Untuk detail API, lihat [DeleteCertificate](#) di Referensi AWS SDK for Java 2.x API.

## Kotlin

### SDK untuk Kotlin

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
suspend fun deleteCertificate(certificateArn: String) {
 val certificateProviderRequest =
 DeleteCertificateRequest {
 certificateId = extractCertificateId(certificateArn)
 }
 IotClient { region = "us-east-1" }.use { iotClient ->
```

```

 iotClient.deleteCertificate(certificateProviderRequest)
 println("$certificateArn was successfully deleted.")
 }
}

```

- Untuk detail API, lihat [DeleteCertificate](#) di AWS SDK untuk referensi API Kotlin.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Menggunakan AWS IoT dengan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

## Gunakan **DeleteThing** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `DeleteThing`.

C++

SDK untuk C++

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```

//! 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();
}
```

- Untuk detail API, lihat [DeleteThing](#) di Referensi AWS SDK for C++ API.

## CLI

### AWS CLI

Untuk menampilkan informasi rinci tentang suatu hal

`delete-thing` Contoh berikut menghapus sesuatu dari registri AWS IoT untuk AWS akun Anda.

```
aws iot delete-thing --thing-name "" FourthBulb
```

Perintah ini tidak menghasilkan output.

Untuk informasi selengkapnya, lihat [Cara Mengelola Sesuatu dengan Registri](#) di Panduan Pengembang AWS IoT.

- Untuk detail API, lihat [DeleteThing](#) di Referensi AWS CLI Perintah.

## Java

### SDK untuk Java 2.x

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
public static void deleteIoTThing(IotClient iotClient, String thingName) {
```



```
try {
 DeleteThingRequest deleteThingRequest = DeleteThingRequest.builder()
 .thingName(thingName)
 .build();

 iotClient.deleteThing(deleteThingRequest);
 System.out.println("Deleted Thing " + thingName);

} catch (IotException e) {
 System.err.println(e.awsErrorDetails().errorMessage());
 System.exit(1);
}
}
```

- Untuk detail API, lihat [DeleteThing](#) di Referensi AWS SDK for Java 2.x API.

## Kotlin

### SDK untuk Kotlin

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
suspend fun deleteIoTThing(thingNameVal: String) {
 val deleteThingRequest =
 DeleteThingRequest {
 thingName = thingNameVal
 }

 IotClient { region = "us-east-1" }.use { iotClient ->
 iotClient.deleteThing(deleteThingRequest)
 println("Deleted $thingNameVal")
 }
}
```

- Untuk detail API, lihat [DeleteThing](#) di AWS SDK untuk referensi API Kotlin.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Menggunakan AWS IoT dengan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

## Gunakan **DeleteTopicRule** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `DeleteTopicRule`.

C++

SDK untuk C++

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
#!/ 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();
}
```

```
}
```

- Untuk detail API, lihat [DeleteTopicAturan](#) dalam Referensi AWS SDK for C++ API.

## CLI

### AWS CLI

Untuk menghapus aturan

`delete-topic-rule` Contoh berikut menghapus aturan yang ditentukan.

```
aws iot delete-topic-rule \
 --rule-name "LowMoistureRule"
```

Perintah ini tidak menghasilkan output.

Untuk informasi selengkapnya, lihat [Menghapus Aturan](#) di Panduan Pengembang AWS IoT.

- Untuk detail API, lihat [DeleteTopicAturan](#) dalam Referensi AWS CLI Perintah.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Menggunakan AWS IoT dengan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

## Gunakan **DescribeEndpoint** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `DescribeEndpoint`.

### C++

#### SDK untuk C++

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
///
//! 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();
}

```

- Untuk detail API, lihat [DescribeEndpoint](#) di Referensi AWS SDK for C++ API.

## CLI

### AWS CLI

Contoh 1: Untuk mendapatkan titik AWS akhir Anda saat ini

`describe-endpoint` Contoh berikut mengambil AWS endpoint default yang semua perintah diterapkan.

```
aws iot describe-endpoint
```

Output:

```
{
 "endpointAddress": "abc123defghijk.iot.us-west-2.amazonaws.com"
}
```

Untuk informasi selengkapnya, lihat [DescribeEndpoint](#) di Panduan Pengembang AWS IoT.

Contoh 2: Untuk mendapatkan titik akhir ATS Anda

`describe-endpoint` Contoh berikut mengambil titik akhir Amazon Trust Services (ATS).

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

Output:

```
{
 "endpointAddress": "abc123defghijk-ats.iot.us-west-2.amazonaws.com"
}
```

Untuk informasi selengkapnya, lihat [Sertifikat X.509 dan IoT di Panduan Pengembang AWS IoT](#).

- Untuk detail API, lihat [DescribeEndpoint](#) di Referensi AWS CLI Perintah.

## Java

### SDK untuk Java 2.x

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
public static String describeEndpoint(IotClient iotClient) {
 try {
```

```
DescribeEndpointResponse endpointResponse =
iotClient.describeEndpoint(DescribeEndpointRequest.builder().build());

// Get the endpoint URL.
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);
return fullEndpoint;

} catch (IotException e) {
 System.err.println(e.awsErrorDetails().errorMessage());
 System.exit(1);
}
return "" ;
}
```

- Untuk detail API, lihat [DescribeEndpoint](#) di Referensi AWS SDK for Java 2.x API.

## Kotlin

### SDK untuk Kotlin

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
suspend fun describeEndpoint(): String? {
 val request = DescribeEndpointRequest {}
 IotClient { 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
 }
}
```

```
}
```

- Untuk detail API, lihat [DescribeEndpoint](#) di AWS SDK untuk referensi API Kotlin.

## Rust

### SDK untuk Rust

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
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(())
}
```

- Untuk detail API, lihat [DescribeEndpoint](#) referensi AWS SDK for Rust API.


Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Menggunakan AWS IoT dengan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

## Gunakan **DescribeThing** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `DescribeThing`.

## C++

## SDK untuk C++

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
#!/ 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();
}
```

- Untuk detail API, lihat [DescribeThing](#) di Referensi AWS SDK for C++ API.

## CLI

### AWS CLI

Untuk menampilkan informasi rinci tentang suatu hal

`describe-thing` Contoh berikut menampilkan informasi tentang sesuatu (perangkat) yang didefinisikan dalam registri AWS IoT untuk akun Anda AWS .

```
aws iot mendeskripsikan-hal --thing-name "Bulb" MyLight
```

Output:

```
{
 "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
}
```

Untuk informasi selengkapnya, lihat [Cara Mengelola Sesuatu dengan Registri](#) di Panduan Pengembang AWS IoT.

- Untuk detail API, lihat [DescribeThing](#) di Referensi AWS CLI Perintah.

## Java

### SDK untuk Java 2.x

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
private static void describeThing(IotClient iotClient, String thingName) {
 try {
 DescribeThingRequest thingRequest = DescribeThingRequest.builder()
 .thingName(thingName)
 .build() ;

 // Print Thing details.
 DescribeThingResponse describeResponse =
iotClient.describeThing(thingRequest);
 System.out.println("Thing Details:");
 System.out.println("Thing Name: " + describeResponse.thingName());
 System.out.println("Thing ARN: " + describeResponse.thingArn());

 } catch (IotException e) {
 System.err.println(e.awsErrorDetails().errorMessage());
 System.exit(1);
 }
}
```

- Untuk detail API, lihat [DescribeThing](#) di Referensi AWS SDK for Java 2.x API.

## Kotlin

### SDK untuk Kotlin

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
suspend fun describeThing(thingNameVal: String) {
 val thingRequest =
 DescribeThingRequest {
 thingName = thingNameVal
 }

 // Print Thing details.
 IotClient { region = "us-east-1" }.use { iotClient ->
 val describeResponse = iotClient.describeThing(thingRequest)
 println("Thing details:")
 println("Thing name: ${describeResponse.thingName}")
 println("Thing ARN: ${describeResponse.thingArn}")
 }
}
```

- Untuk detail API, lihat [DescribeThing](#) di AWS SDK untuk referensi API Kotlin.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Menggunakan AWS IoT dengan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

## Gunakan **DetachThingPrincipal** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `DetachThingPrincipal`.

C++

SDK untuk C++

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
//! 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();
}

```

- Untuk detail API, lihat [DetachThingPrincipal](#) dalam Referensi AWS SDK for C++ API.

## CLI

### AWS CLI

Untuk melepaskan sertifikat/prinsipal dari suatu hal

`detach-thing-principal` Contoh berikut menghapus sertifikat yang mewakili prinsipal dari hal yang ditentukan.

```
aws iot detach-thing-principal \
 --thing-name "MyLightBulb" \
 --principal "arn:aws:iot:us-
west-2:123456789012:cert/604c48437a57b7d5fc5d137c5be75011c6ee67c9a6943683a1acb4b1626bac36"
```

Perintah ini tidak menghasilkan output.

Untuk informasi selengkapnya, lihat [Cara Mengelola Sesuatu dengan Registri](#) di Panduan Pengembang AWS IoT.

- Untuk detail API, lihat [DetachThingPrincipal](#) dalam Referensi AWS CLI Perintah.

## Java

### SDK untuk Java 2.x

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
public static void detachThingPrincipal(IotClient iotClient, String
thingName, String certificateArn){
 try {
 DetachThingPrincipalRequest thingPrincipalRequest =
 DetachThingPrincipalRequest.builder()
 .principal(certificateArn)
 .thingName(thingName)
 .build();

 iotClient.detachThingPrincipal(thingPrincipalRequest);
 System.out.println(certificateArn + " was successfully removed from "
+thingName);

 } catch (IotException e) {
 System.err.println(e.awsErrorDetails().errorMessage());
 System.exit(1);
 }
}
```

- Untuk detail API, lihat [DetachThingPrincipal](#) dalam Referensi AWS SDK for Java 2.x API.

## Kotlin

### SDK untuk Kotlin

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
suspend fun detachThingPrincipal(
 thingNameVal: String,
 certificateArn: String,
) {
 val thingPrincipalRequest =
 DetachThingPrincipalRequest {
 principal = certificateArn
 thingName = thingNameVal
 }

 IotClient { region = "us-east-1" }.use { iotClient ->
 iotClient.detachThingPrincipal(thingPrincipalRequest)
 println("$certificateArn was successfully removed from $thingNameVal")
 }
}
```

- Untuk detail API, lihat [DetachThingPrincipal](#) di AWS SDK untuk referensi API Kotlin.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Menggunakan AWS IoT dengan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

## Gunakan **ListCertificates** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `ListCertificates`.

## C++

## SDK untuk C++

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
//! 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;
}

```

- Untuk detail API, lihat [ListCertificates](#) di Referensi AWS SDK for C++ API.

## CLI

### AWS CLI

Contoh 1: Untuk mencantumkan sertifikat yang terdaftar di AWS akun Anda

`list-certificates` Contoh berikut mencantumkan semua sertifikat yang terdaftar di akun Anda. Jika Anda memiliki lebih dari batas paging default 25, Anda dapat menggunakan nilai `nextMarker` respons dari perintah ini dan memasoknya ke perintah berikutnya untuk mendapatkan batch hasil berikutnya. Ulangi sampai `nextMarker` kembali tanpa nilai.

```
aws iot list-certificates
```

Output:

```

{
 "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
 }
}
]
```

```
}
```

- Untuk detail API, lihat [ListCertificates](#) di Referensi AWS CLI Perintah.

## Java

### SDK untuk Java 2.x

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
public static void listCertificates(IotClient iotClient) {
 ListCertificatesResponse response = iotClient.listCertificates();
 List<Certificate> certList = response.certificates();
 for (Certificate cert : certList) {
 System.out.println("Cert id: " + cert.certificateId());
 System.out.println("Cert Arn: " + cert.certificateArn());
 }
}
```

- Untuk detail API, lihat [ListCertificates](#) di Referensi AWS SDK for Java 2.x API.

## Kotlin

### SDK untuk Kotlin

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
suspend fun listCertificates() {
 IotClient { 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}")
}
}
```

- Untuk detail API, lihat [ListCertificates](#) di AWS SDK untuk referensi API Kotlin.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Menggunakan AWS IoT dengan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

## Gunakan **ListThings** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `ListThings`.

### CLI

#### AWS CLI

Contoh 1: Untuk membuat daftar semua hal dalam registri

`list-things` Contoh berikut mencantumkan hal-hal (perangkat) yang didefinisikan dalam registri AWS IoT untuk akun Anda AWS .

```
aws iot list-things
```

Output:

```
{
 "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
}
]
}

```

Contoh 2: Untuk daftar hal-hal yang ditentukan yang memiliki atribut tertentu

`list-things` Contoh berikut menampilkan daftar hal-hal yang memiliki atribut bernama `wattage`.

```
aws iot list-things \
 --attribute-name wattage
```

Output:

```
{
```

```
"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
 }
]
}
```

Untuk informasi selengkapnya, lihat [Cara Mengelola Sesuatu dengan Registri](#) di Panduan Pengembang AWS IoT.

- Untuk detail API, lihat [ListThings](#) di Referensi AWS CLI Perintah.

## Rust

### SDK untuk Rust

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
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(())
}
```

- Untuk detail API, lihat [ListThings](#) referensi AWS SDK for Rust API.


Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Menggunakan AWS IoT dengan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

## Gunakan **SearchIndex** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `SearchIndex`.

## C++

## SDK untuk C++

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
#!/ Query the AWS IoT fleet index.
#!/ For query information, see https://docs.aws.amazon.com/iot/latest/
develoerguide/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());
 }
 } while (outcome.IsSuccess() && !nextToken.empty());
}
```

```
 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;
}
```

- Untuk detail API, lihat [SearchIndex](#) di Referensi AWS SDK for C++ API.

## CLI

### AWS CLI

Untuk menanyakan indeks benda

search-index Contoh berikut query AWS\_Things indeks untuk hal-hal yang memiliki jenis.  
LightBulb

```
aws iot search-index \
 --index-name "AWS_Things" \
 --query-string "thingTypeName:LightBulb"
```

Output:

```
{
 "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
 }
 }
]
```

Untuk informasi selengkapnya, lihat [Mengelola Pengindeksan Hal](#) di Panduan Pengembang AWS IoT.

- Untuk detail API, lihat [SearchIndex](#) di Referensi AWS CLI Perintah.

## Java

### SDK untuk Java 2.x

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
public static void searchThings(IotClient iotClient, String queryString){
 SearchIndexRequest searchIndexRequest = SearchIndexRequest.builder()
 .queryString(queryString)
 .build();

 try {
 // Perform the search and get the result.
 SearchIndexResponse searchIndexResponse =
iotClient.searchIndex(searchIndexRequest);

 // 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()));
 }
 } catch (IotException e) {
 System.err.println(e.awsErrorDetails().errorMessage());
 System.exit(1);
 }
}
```

- Untuk detail API, lihat [SearchIndex](#) di Referensi AWS SDK for Java 2.x API.

## Kotlin

### SDK untuk Kotlin

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
suspend fun searchThings(queryStringVal: String?) {
 val searchIndexRequest =
 SearchIndexRequest {
 queryString = queryStringVal
 }

 IotClient { 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}") }
 }
 }
}
```

- Untuk detail API, lihat [SearchIndex](#) di AWS SDK untuk referensi API Kotlin.


Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Menggunakan AWS IoT dengan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

## Gunakan **UpdateIndexingConfiguration** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `UpdateIndexingConfiguration`.

## C++

## SDK untuk C++

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
//! 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();
}
```

- Untuk detail API, lihat [UpdateIndexingKonfigurasi](#) di Referensi AWS SDK for C++ API.

## CLI

### AWS CLI

Untuk mengaktifkan pengindeksan hal

`update-indexing-configuration`Contoh berikut memungkinkan pengindeksan hal untuk mendukung pencarian data registri, data bayangan, dan status konektivitas benda menggunakan indeks `AWS_Things`.

```
aws iot update-indexing-configuration
 --thing-indexing-configuration
 thingIndexingMode=REGISTRY_AND_SHADOW,thingConnectivityIndexingMode=STATUS
```

Perintah ini tidak menghasilkan output.

Untuk informasi selengkapnya, lihat [Mengelola Pengindeksan Hal](#) di Panduan Pengembang AWS IoT.

- Untuk detail API, lihat [UpdateIndexingKonfigurasi](#) di Referensi AWS CLI Perintah.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Menggunakan AWS IoT dengan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

## Gunakan **UpdateThing** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `UpdateThing`.

C++

SDK untuk C++

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
#!/ 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();
}
```

- Untuk detail API, lihat [UpdateThing](#) di Referensi AWS SDK for C++ API.

## CLI

### AWS CLI

Untuk mengasosiasikan sesuatu dengan tipe benda

`update-thing` Contoh berikut mengaitkan sesuatu dalam registri AWS IoT dengan tipe benda. Saat Anda membuat asosiasi, Anda memberikan nilai untuk atribut yang ditentukan oleh tipe benda.

```
aws iot update-thing \
 --thing-name "MyOtherLightBulb" \
 --thing-type-name "LightBulb" \
 --attribute-payload '{"attributes": {"wattage": "75", "model": "123"}}'
```

Perintah ini tidak menghasilkan output. Gunakan `describe-thing` perintah untuk melihat hasilnya.

Untuk informasi selengkapnya, lihat [Tipe Hal](#) di Panduan Pengembang AWS IoT.

- Untuk detail API, lihat [UpdateThing](#) di Referensi AWS CLI Perintah.

## Java

### SDK untuk Java 2.x

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
public static void updateThing(IotClient iotClient, String thingName) {
```

```
// Specify the new attribute values.
String newLocation = "Office";
String newFirmwareVersion = "v2.0";

Map<String, String> attMap = new HashMap<>();
attMap.put("location", newLocation);
attMap.put("firmwareVersion", newFirmwareVersion);

AttributePayload attributePayload = AttributePayload.builder()
 .attributes(attMap)
 .build();

UpdateThingRequest updateThingRequest = UpdateThingRequest.builder()
 .thingName(thingName)
 .attributePayload(attributePayload)
 .build();

try {
 // Update the IoT Thing attributes.
 iotClient.updateThing(updateThingRequest);
 System.out.println("Thing attributes updated successfully.");
} catch (IotException e) {
 System.err.println(e.awsErrorDetails().errorMessage());
 System.exit(1);
}
}
```

- Untuk detail API, lihat [UpdateThing](#) di Referensi AWS SDK for Java 2.x API.

## Kotlin

### SDK untuk Kotlin

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
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 { region = "us-east-1" }.use { iotClient ->
 // Update the IoT thing attributes.
 iotClient.updateThing(updateThingRequest)
 println("$thingNameVal attributes updated successfully.")
}
}
```

- Untuk detail API, lihat [UpdateThing](#) di AWS SDK untuk referensi API Kotlin.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Menggunakan AWS IoT dengan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

## Skenario untuk AWS IoT menggunakan AWS SDK

Contoh kode berikut menunjukkan cara menerapkan skenario umum AWS IoT dengan AWS SDK. Skenario ini menunjukkan kepada Anda bagaimana menyelesaikan tugas tertentu dengan memanggil beberapa fungsi di dalamnya AWS IoT. Setiap skenario menyertakan tautan ke GitHub, di mana Anda dapat menemukan petunjuk tentang cara mengatur dan menjalankan kode.

### Contoh

- [Bekerja dengan AWS IoT perangkat, benda, dan bayangan menggunakan AWS IoT SDK](#)

## Bekerja dengan AWS IoT perangkat, benda, dan bayangan menggunakan AWS IoT SDK

Contoh kode berikut menunjukkan cara bekerja dengan kasus penggunaan manajemen AWS IoT perangkat menggunakan AWS IoT SDK

C++

SDK untuk C++

### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

Ciptakan AWS IoT sesuatu.

```
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();
}

```

Buat dan lampirkan sertifikat perangkat.

```

 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();
}

```

Lakukan berbagai operasi pada AWS IoT benda itu.

```
 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\\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::ThingConnectivityIndexingMode::REGI
 // 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;
}
```

## Pembersihan sumber daya

```
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 untuk Java 2.x

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
import software.amazon.awssdk.core.SdkBytes;
```

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iot.IotClient;
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.AttributePayload;
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.CreateTopicRuleRequest;
import software.amazon.awssdk.services.iot.model.DeleteCertificateRequest;
import software.amazon.awssdk.services.iot.model.CreateThingResponse;
import software.amazon.awssdk.services.iot.model.DeleteThingRequest;
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.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.SnsAction;
import software.amazon.awssdk.services.iot.model.TopicRuleListItem;
import software.amazon.awssdk.services.iot.model.TopicRulePayload;
import software.amazon.awssdk.services.iot.model.UpdateThingRequest;
import software.amazon.awssdk.services.iotdataplane.IotDataPlaneClient;
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 java.net.URI;
import java.nio.charset.StandardCharsets;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

/**
```

```

* 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 and delete the certificate.
* 12. Delete Thing.
*/
public class IotScenario {
 public static final String DASHES = new String(new char[80]).replace("\0",
"-");
 private static final String TOPIC = "your-iot-topic";
 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);
 }
 }
}

```



```
String thingName;
String ruleName;
String roleARN = args[0];
String snsAction = args[1];
Scanner scanner = new Scanner(System.in);
IotClient iotClient = IotClient.builder()
 .region(Region.US_EAST_1)
 .build();

System.out.println(DASHES);
System.out.println("Welcome to the AWS IoT example workflow.");
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.

 """);
System.out.print("Press Enter to continue...");
scanner.nextLine();
System.out.println(DASHES);

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();
createIoTThing(iotClient, 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 = createCertificate(iotClient);
 System.out.println("Attach the certificate to the AWS IoT Thing.");
 attachCertificateToThing(iotClient, 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.
 """);
 System.out.print("Press Enter to continue...");
 scanner.nextLine();
 updateThing(iotClient, thingName);
 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.
 """);
 System.out.print("Press Enter to continue...");
 scanner.nextLine();
 String endpointUrl = describeEndpoint(iotClient);
 System.out.println(DASHES);

 System.out.println(DASHES);
 System.out.println("5. List your AWS IoT certificates");
```

```
System.out.print("Press Enter to continue...");
scanner.nextLine();
if (certificateArn.length() > 0) {
 listCertificates(iotClient);
} else {
 System.out.println("You did not create a certificates. Skipping this
step.");
}
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.
 """);
System.out.print("Press Enter to continue...");
scanner.nextLine();
IotDataPlaneClient iotPlaneClient = IotDataPlaneClient.builder()
 .region(Region.US_EAST_1)
 .endpointOverride(URI.create(endpointUrl))
 .build();

updateShadowThing(iotPlaneClient, thingName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Write out the state information, in JSON
format.");
System.out.print("Press Enter to continue...");
scanner.nextLine();
getPayload(iotPlaneClient, thingName);
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();
createIoTRule(iotClient, roleARN, ruleName, snsAction);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("9. List your rules.");
System.out.print("Press Enter to continue...");
scanner.nextLine();
listIoTRules(iotClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("10. Search things using the Thing name.");
System.out.print("Press Enter to continue...");
scanner.nextLine();
String queryString = "thingName:"+thingName ;
searchThings(iotClient, queryString);
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.");
 System.out.print("Press Enter to continue...");
 scanner.nextLine();
 detachThingPrincipal(iotClient, thingName, certificateArn);
 deleteCertificate(iotClient, certificateArn);
 } 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")) {
 deleteIoTThing(iotClient, 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);
}

public static void listCertificates(IotClient iotClient) {
 ListCertificatesResponse response = iotClient.listCertificates();
 List<Certificate> certList = response.certificates();
 for (Certificate cert : certList) {
 System.out.println("Cert id: " + cert.certificateId());
 System.out.println("Cert Arn: " + cert.certificateArn());
 }
}

public static void listIoTRules(IotClient iotClient) {
 try {
 ListTopicRulesRequest listTopicRulesRequest =
ListTopicRulesRequest.builder().build();
 ListTopicRulesResponse listTopicRulesResponse =
iotClient.listTopicRules(listTopicRulesRequest);
 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("-----");
 }
 } catch (IotException e) {
 System.err.println(e.awsErrorDetails().errorMessage());
 System.exit(1);
 }
}
```

```
 }

 public static void createIoTRule(IotClient iotClient, String roleARN, String
ruleName, String action) {
 try {
 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();

 // Create the rule.
 iotClient.createTopicRule(topicRuleRequest);
 System.out.println("IoT Rule created successfully.");

 } catch (IotException e) {
 System.err.println(e.getMessage());
 System.exit(1);
 }
 }

 public static void getPayload(IotDataPlaneClient iotPlaneClient, String
thingName) {
 try {
 GetThingShadowRequest getThingShadowRequest =
GetThingShadowRequest.builder()
```

```
 .thingName(thingName)
 .build();

 GetThingShadowResponse getThingShadowResponse =
iotPlaneClient.getThingShadow(getThingShadowRequest);

 // Extracting payload from response.
 SdkBytes payload = getThingShadowResponse.payload();
 String payloadString = payload.asUtf8String();
 System.out.println("Received Shadow Data: " + payloadString);

 } catch (IotException e) {
 System.err.println(e.getMessage());
 System.exit(1);
 }
}

public static void updateShadowThing(IotDataPlaneClient iotPlaneClient,
String thingName) {
 try {
 // 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();

 // Update Thing Shadow.
 iotPlaneClient.updateThingShadow(updateThingShadowRequest);
 System.out.println("Thing Shadow updated successfully.");

 } catch (IotException e) {
 System.err.println(e.awsErrorDetails().errorMessage());
 System.exit(1);
 }
}

public static void updateThing(IotClient iotClient, String thingName) {
 // Specify the new attribute values.
 String newLocation = "Office";
```

```
String newFirmwareVersion = "v2.0";

Map<String, String> attMap = new HashMap<>();
attMap.put("location", newLocation);
attMap.put("firmwareVersion", newFirmwareVersion);

AttributePayload attributePayload = AttributePayload.builder()
 .attributes(attMap)
 .build();

UpdateThingRequest updateThingRequest = UpdateThingRequest.builder()
 .thingName(thingName)
 .attributePayload(attributePayload)
 .build();

try {
 // Update the IoT Thing attributes.
 iotClient.updateThing(updateThingRequest);
 System.out.println("Thing attributes updated successfully.");
} catch (IotException e) {
 System.err.println(e.awsErrorDetails().errorMessage());
 System.exit(1);
}

}

public static String describeEndpoint(IotClient iotClient) {
 try {
 DescribeEndpointResponse endpointResponse =
iotClient.describeEndpoint(DescribeEndpointRequest.builder().build());

 // Get the endpoint URL.
 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);
 return fullEndpoint;
 } catch (IotException e) {
 System.err.println(e.awsErrorDetails().errorMessage());
 System.exit(1);
 }
}
```



```
 return "" ;
 }

 public static void detachThingPrincipal(IotClient iotClient, String
thingName, String certificateArn){
 try {
 DetachThingPrincipalRequest thingPrincipalRequest =
DetachThingPrincipalRequest.builder()
 .principal(certificateArn)
 .thingName(thingName)
 .build();

 iotClient.detachThingPrincipal(thingPrincipalRequest);
 System.out.println(certificateArn +" was successfully removed from "
+thingName);

 } catch (IotException e) {
 System.err.println(e.awsErrorDetails().errorMessage());
 System.exit(1);
 }
 }

 public static void deleteCertificate(IotClient iotClient, String
certificateArn) {
 DeleteCertificateRequest certificateProviderRequest =
DeleteCertificateRequest.builder()
 .certificateId(extractCertificateId(certificateArn))
 .build();

 iotClient.deleteCertificate(certificateProviderRequest);
 System.out.println(certificateArn +" was successfully deleted.");
 }

 // Get the cert Id from the Cert ARN value.
 private static 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);
 }

 public static String createCertificate(IotClient iotClient) {
 try {
```

```
 CreateKeysAndCertificateResponse response =
iotClient.createKeysAndCertificate();
 String certificatePem = response.certificatePem();
 String certificateArn = response.certificateArn();

 // Print the details.
 System.out.println("\nCertificate:");
 System.out.println(certificatePem);
 System.out.println("\nCertificate ARN:");
 System.out.println(certificateArn);
 return certificateArn;

 } catch (IotException e) {
 System.err.println(e.awsErrorDetails().errorMessage());
 System.exit(1);
 }

 return "";
}

public static void attachCertificateToThing(IotClient iotClient, String
thingName, String certificateArn) {
 // Attach the certificate to the thing.
 AttachThingPrincipalRequest principalRequest =
AttachThingPrincipalRequest.builder()
 .thingName(thingName)
 .principal(certificateArn)
 .build();

 AttachThingPrincipalResponse attachResponse =
iotClient.attachThingPrincipal(principalRequest);

 // Verify the attachment was successful.
 if (attachResponse.sdkHttpResponse().isSuccessful()) {
 System.out.println("Certificate attached to Thing successfully.");

 // Print additional information about the Thing.
 describeThing(iotClient, thingName);
 } else {
 System.err.println("Failed to attach certificate to Thing. HTTP
Status Code: " +
 attachResponse.sdkHttpResponse().statusCode());
 }
}
```

```
private static void describeThing(IotClient iotClient, String thingName) {
 try {
 DescribeThingRequest thingRequest = DescribeThingRequest.builder()
 .thingName(thingName)
 .build() ;

 // Print Thing details.
 DescribeThingResponse describeResponse =
iotClient.describeThing(thingRequest);
 System.out.println("Thing Details:");
 System.out.println("Thing Name: " + describeResponse.thingName());
 System.out.println("Thing ARN: " + describeResponse.thingArn());

 } catch (IotException e) {
 System.err.println(e.awsErrorDetails().errorMessage());
 System.exit(1);
 }
}

public static void deleteIoTThing(IotClient iotClient, String thingName) {
 try {
 DeleteThingRequest deleteThingRequest = DeleteThingRequest.builder()
 .thingName(thingName)
 .build();

 iotClient.deleteThing(deleteThingRequest);
 System.out.println("Deleted Thing " + thingName);

 } catch (IotException e) {
 System.err.println(e.awsErrorDetails().errorMessage());
 System.exit(1);
 }
}

public static void createIoTThing(IotClient iotClient, String thingName) {
 try {
 CreateThingRequest createThingRequest = CreateThingRequest.builder()
 .thingName(thingName)
 .build();

 CreateThingResponse createThingResponse =
iotClient.createThing(createThingRequest);
```

```
 System.out.println(thingName + " was successfully created. The ARN
value is " + createThingResponse.thingArn());

 } catch (IotException e) {
 System.err.println(e.awsErrorDetails().errorMessage());
 System.exit(1);
 }
}

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 "" ;
}

public static void searchThings(IotClient iotClient, String queryString){
 SearchIndexRequest searchIndexRequest = SearchIndexRequest.builder()
 .queryString(queryString)
 .build();

 try {
 // Perform the search and get the result.
 SearchIndexResponse searchIndexResponse =
iotClient.searchIndex(searchIndexRequest);

 // 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()));
 }
 } catch (IotException e) {
 System.err.println(e.awsErrorDetails().errorMessage());
 System.exit(1);
 }
}
}
```

## Kotlin

### SDK untuk Kotlin

#### Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
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.ByteString
```

```
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 { region = "us-east-1" }.use { iotClient ->
 iotClient.deleteThing(deleteThingRequest)
 println("Deleted $thingNameVal")
 }
}

suspend fun deleteCertificate(certificateArn: String) {
 val certificateProviderRequest =
 DeleteCertificateRequest {
 certificateId = extractCertificateId(certificateArn)
 }
 IotClient { 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 { 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 { 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 { 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 { 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 { 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 { 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 { 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 { 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 { 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 { 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 { 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 { 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 { region = "us-east-1" }.use { iotClient ->
 iotClient.createThing(createThingRequest)
 println("Created $thingNameVal")
 }
}
```

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Menggunakan AWS IoT dengan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

# Kuota AWS IoT

Anda dapat menemukan informasi tentang AWS IoT Kuota di [AWS Referensi Umum](#).

- Untuk AWS IoT Core informasi kuota, lihat [AWS IoT Core Endpoint dan Kuota](#).
- Untuk AWS IoT Device Management informasi kuota, lihat [AWS IoT Device Management Endpoint dan Kuota](#).
- Untuk AWS IoT Device Defender informasi kuota, lihat [AWS IoT Device Defender Endpoint dan Kuota](#).

# Harga AWS IoT Core

Anda dapat menemukan informasi tentang AWS IoT Core harga di halaman AWS Pemasaran dan [Kalkulator AWS Harga](#).

- Untuk memeriksa informasi AWS IoT Core harga, lihat [AWS IoT Core Harga](#).
- Untuk memperkirakan biaya solusi arsitek Anda, lihat [Kalkulator AWS Harga](#).

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